

# ***J721E DRA829/TDA4VM/AM68P Processors Silicon Revision 1.1 Texas Instruments Families of Products***

*Technical Reference Manual*

---



Literature Number: SPRUIL1D  
MAY 2019 – REVISED DECEMBER 2024





# Table of Contents



<b>1 Introduction</b>	<b>79</b>
1.1 Device Overview	80
1.2 Device Block Diagram	82
1.3 Device Main Domain	85
1.3.1 Arm Cortex-A72 Subsystem	87
1.3.2 Arm Cortex-R5F Processor	87
1.3.3 C66x DSP Subsystem	87
1.3.4 C71x DSP Subsystem	88
1.3.5 Graphics Processing Unit	88
1.3.6 Multi-Standard HD Video Decoder	89
1.3.7 Multi-Standard HD Video Encoder	89
1.3.8 Vision Pre-processing Accelerator	90
1.3.9 Depth and Motion Perception Accelerator	90
1.3.10 Navigator Subsystem	91
1.3.11 Region-based Address Translation Module	93
1.3.12 Data Routing Unit	93
1.3.13 Display Subsystem	93
1.3.14 Camera Subsystem	96
1.3.15 Shared D-PHY Transmitter	97
1.3.16 Video Processing Front End	97
1.3.17 Multicore Shared Memory Controller	97
1.3.18 DDR Subsystem	98
1.3.19 Region-based Address Translation Module	98
1.3.20 General Purpose Input/Output Interface	98
1.3.21 Inter-Integrated Circuit Interface	98
1.3.22 Improved Inter-Integrated Circuit Interface	99
1.3.23 Multi-channel Serial Peripheral Interface	99
1.3.24 Universal Asynchronous Receiver/Transmitter	99
1.3.25 Gigabit Ethernet Switch	99
1.3.26 Peripheral Component Interconnect Express Subsystem	100
1.3.27 Universal Serial Bus (USB) Subsystem	100
1.3.28 SerDes	101
1.3.29 General Purpose Memory Controller with Error Location Module	102
1.3.30 Multimedia Card/Secure Digital Interface	103
1.3.31 Universal Flash Storage Interface	103
1.3.32 Enhanced Capture Module	104
1.3.33 Enhanced Pulse-Width Modulation Module	104
1.3.34 Enhanced Quadrature Encoder Pulse Module	104
1.3.35 Controller Area Network	104
1.3.36 Audio Tracking Logic	105
1.3.37 Multi-channel Audio Serial Port	105
1.3.38 Timers	105
1.3.39 Internal Diagnostics Modules	106
1.4 Device MCU Domain	107
1.4.1 MCU Arm Cortex-R5F Processor	108
1.4.2 MCU Region-based Address Translation Module	109
1.4.3 MCU Navigator Subsystem	109
1.4.4 MCU Analog-to-Digital Converter	110
1.4.5 MCU Inter-Integrated Circuit Interface	110

1.4.6 MCU Improved Inter-Integrated Circuit Interface.....	111
1.4.7 MCU Multi-channel Serial Peripheral Interface.....	111
1.4.8 MCU Universal Asynchronous Receiver/Transmitter.....	111
1.4.9 MCU Gigabit Ethernet Switch.....	111
1.4.10 MCU Octal Serial Peripheral Interface and HyperBus Memory Controller as a Flash Subsystem.....	112
1.4.11 MCU Controller Area Network.....	113
1.4.12 MCU Timers.....	113
1.4.13 MCU Internal Diagnostics Modules.....	113
1.5 Device WKUP Domain.....	114
1.5.1 WKUP Device Management and Security Controller.....	115
1.5.2 WKUP General Purpose Input/Output Interface.....	115
1.5.3 WKUP Inter-Integrated Circuit Interface.....	115
1.5.4 WKUP Universal Asynchronous Receiver/Transmitter.....	116
1.5.5 WKUP Internal Diagnostics Modules.....	116
1.6 Device Identification.....	117
<b>2 Memory Map.....</b>	<b>118</b>
2.1 MAIN Domain Memory Map.....	119
2.2 MCU Domain Memory Map.....	173
2.3 WKUP Domain Memory Map.....	177
2.4 Processors View Memory Map.....	178
2.5 Region-based Address Translation.....	179
<b>3 System Interconnect.....</b>	<b>181</b>
3.1 System Interconnect Overview.....	182
3.2 System Interconnect Integration.....	184
3.2.1 Interconnect Integration in WKUP Domain.....	184
3.2.2 Interconnect Integration in MCU Domain.....	185
3.2.3 Interconnect Integration in MAIN Domain.....	186
3.3 System Interconnect Functional Description.....	189
3.3.1 Master-Slave Connections.....	189
3.3.2 Quality of Service (QoS).....	221
3.3.3 Route ID.....	226
3.3.4 Initiator-Side Security Controls and Firewalls.....	227
3.3.5 VBUSM_TIMEOUT_GASKET (MCU_TIMEOUT_64B2).....	251
<b>4 Initialization.....</b>	<b>269</b>
4.1 Initialization Overview.....	270
4.1.1 ROM Code Overview.....	270
4.1.2 Bootloader Modes.....	271
4.1.3 Terminology.....	271
4.2 Boot Process.....	273
4.2.1 MCU ROM Code Architecture.....	273
4.2.2 DMSC ROM Description.....	274
4.2.3 Boot Process Flow.....	274
4.2.4 MCU Only vs Normal Boot.....	277
4.3 Boot Mode Pins.....	279
4.3.1 MCU_BOOTMODE Pin Mapping.....	280
4.3.2 BOOTMODE Pin Mapping.....	282
4.3.3 No-boot/Dev-boot Configuration.....	284
4.3.4 OSPI Boot Device Configuration.....	285
4.3.5 xSPI Boot Device Configuration.....	285
4.3.6 QSPI Boot Device Configuration.....	290
4.3.7 SPI Boot Device Configuration.....	291
4.3.8 I2C Boot Device Configuration.....	292
4.3.9 MMC/SD Card Boot Device Configuration.....	293
4.3.10 Ethernet Boot Device Configuration.....	294
4.3.11 USB Boot Device Configuration.....	296
4.3.12 PCIe Boot Device Configuration.....	297
4.3.13 UART Boot Device Configuration.....	298
4.3.14 GPMC NOR Boot Device Configuration.....	299
4.3.15 eMMC Boot Device Configuration.....	303
4.3.16 PLL Configuration.....	304
4.4 Boot Parameter Tables.....	306

4.4.1 Common Header.....	306
4.4.2 PLL Setup.....	307
4.4.3 PCIe Boot Parameter Table.....	308
4.4.4 I2C Boot Parameter Table.....	308
4.4.5 OSPI/QSPI/SPI/xSPI Boot Parameter Table.....	309
4.4.6 GPMC NOR Boot Parameter Table.....	311
4.4.7 Ethernet Boot Parameter Table.....	311
4.4.8 USB Boot Parameter Table.....	313
4.4.9 MMCSD Boot Parameter Table.....	313
4.4.10 UART Boot Parameter Table.....	314
4.5 Boot Image Format.....	315
4.5.1 Overall Structure.....	315
4.5.2 X.509 Certificate.....	315
4.5.3 Organizational Identifier (OID).....	315
4.5.4 X.509 Extensions Specific to Boot.....	316
4.5.5 Generating X.509 Certificates.....	316
4.5.6 Image Data.....	318
4.6 Boot Modes.....	319
4.6.1 I2C Bootloader Operation.....	319
4.6.2 SPI Bootloader Operation.....	319
4.6.3 QSPI Bootloader Operation.....	320
4.6.4 OSPI Bootloader Operation.....	320
4.6.5 PCIe Bootloader Operation.....	320
4.6.6 GPMC NOR Bootloader Operation.....	320
4.6.7 Ethernet Bootloader Operation.....	321
4.6.8 USB Bootloader Operation.....	322
4.6.9 MMCSD Bootloader Operation.....	323
4.6.10 UART Bootloader Operation.....	323
4.7 Boot Memory Maps.....	325
4.7.1 Memory Layout/MPU.....	325
4.7.2 Global Memory Addresses Used by ROM Code.....	325
4.7.3 Memory Reserved by ROM Code.....	326
<b>5 Device Configuration.....</b>	<b>327</b>
5.1 Control Module (CTRL_MMR).....	328
5.1.1 WKUP_CTRL_MMR0.....	329
5.1.2 MCU_CTRL_MMR0.....	337
5.1.3 CTRL_MMR0.....	344
5.2 Power.....	360
5.2.1 Power Management Overview.....	360
5.2.2 Power Management Subsystems.....	361
5.2.3 Device Power States.....	441
5.2.4 Dynamic Power Management.....	459
5.2.5 Thermal Management.....	461
5.3 Reset.....	462
5.3.1 Reset Overview.....	462
5.3.2 Reset Sources.....	463
5.3.3 Reset Status.....	464
5.3.4 Reset Control.....	464
5.3.5 BOOTMODE Pins.....	465
5.3.6 Reset Sequences.....	466
5.3.7 PLL Behavior on Reset.....	470
5.4 Clocking.....	472
5.4.1 Overview.....	472
5.4.2 Clock Inputs.....	474
5.4.3 Clock Outputs.....	476
5.4.4 Device Oscillators.....	479
5.4.5 PLLs.....	482
<b>6 Processors and Accelerators.....</b>	<b>508</b>
6.1 Compute Cluster.....	509
6.1.1 Compute Cluster Overview.....	509
6.1.2 Compute Cluster Functional Description.....	510

6.2 Dual-A72 MPU Subsystem.....	517
6.2.1 A72SS Overview.....	517
6.2.2 A72SS Integration.....	519
6.2.3 A72SS Functional Description.....	521
6.3 Dual-R5F MCU Subsystem.....	531
6.3.1 R5FSS Overview.....	531
6.3.2 R5FSS Integration.....	534
6.3.3 R5FSS Functional Description.....	543
6.4 C66x DSP Subsystem.....	563
6.5 C71x DSP Subsystem.....	598
6.5.1 C71SS Overview.....	598
6.5.2 C71SS Integration.....	601
6.5.3 C71SS Functional Description.....	602
6.6 Graphics Accelerator (GPU).....	609
6.6.1 GPU Overview.....	609
6.6.2 GPU Integration.....	611
6.6.3 GPU Functional Description.....	614
6.7 Multi-Standard HD Video Decoder (DECODER).....	616
6.7.1 DECODER Overview.....	616
6.7.2 DECODER Integration.....	618
6.7.3 DECODER Functional Description.....	620
6.8 Multi-Standard HD Video Encoder (ENCODER).....	621
6.8.1 ENCODER Overview.....	621
6.8.2 ENCODER Integration.....	623
6.8.3 ENCODER Functional Description.....	625
6.9 Vision Pre-processing Accelerator (VPAC).....	626
6.9.1 VPAC Overview.....	626
6.9.2 VPAC Integration.....	630
6.9.3 VPAC Subsystem Level.....	633
6.9.4 VPAC Vision Imaging Subsystem (VISS).....	644
6.9.5 VPAC Lens Distortion Correction (LDC) Module.....	719
6.9.6 VPAC Multi-Scaler (MSC).....	743
6.9.7 VPAC Noise Filter (NF).....	765
6.10 Depth and Motion Perception Accelerator (DMPAC).....	779
6.10.1 DMPAC Overview.....	779
6.10.2 DMPAC Integration.....	782
6.10.3 DMPAC Functional Description.....	785
6.10.4 DMPAC Programming Guide.....	805
<b>7 Interprocessor Communication.....</b>	<b>813</b>
7.1 Mailbox.....	814
7.1.1 Mailbox Overview.....	814
7.1.2 Mailbox Integration.....	815
7.1.3 Mailbox Functional Description.....	819
7.1.4 Mailbox Programming Guide.....	823
7.2 Spinlock.....	825
7.2.1 Spinlock Overview.....	825
7.2.2 Spinlock Integration.....	826
7.2.3 Spinlock Functional Description.....	827
7.2.4 Spinlock Programming Guide.....	829
<b>8 Memory Controllers.....</b>	<b>831</b>
8.1 Multicore Shared Memory Controller (MSMC).....	832
8.1.1 MSMC Overview.....	832
8.1.2 MSMC Integration.....	834
8.1.3 MSMC Functional Description.....	837
8.2 DDR Subsystem (DDRSS).....	848
8.2.1 DDRSS Overview.....	848
8.2.2 DDRSS Environment.....	851
8.2.3 DDRSS Integration.....	856
8.2.4 DDRSS Functional Description.....	859
8.3 Virtualization Subsystem (VirtSS).....	902
8.3.1 VirtSS Overview.....	903

8.3.2 Peripheral Virtualization Unit (PVU).....	916
8.3.3 Page Based Address Translation Unit (PAT).....	924
8.4 Region-based Address Translation (RAT) Module.....	930
8.4.1 RAT Functional Description.....	930
<b>9 Interrupts</b> .....	933
9.1 Interrupt Architecture.....	934
9.2 Interrupt Controllers.....	936
9.2.1 Generic Interrupt Controller (GIC).....	936
9.2.2 Cluster Level Event Controller (CLEC).....	944
9.2.3 Other Interrupt Controllers.....	953
9.3 Interrupt Routers.....	954
9.3.1 INTRTR Overview.....	954
9.3.2 INTRTR Integration.....	955
9.4 Interrupt Sources.....	966
9.4.1 WKUP Domain Interrupt Maps.....	966
9.4.2 MCU Domain Interrupt Maps.....	973
9.4.3 MAIN Domain Interrupt Maps.....	991
<b>10 Data Movement Architecture (DMA)</b> .....	1119
10.1 DMA Architecture.....	1120
10.1.1 Overview.....	1120
10.1.2 DMA Hardware/Software Interface.....	1129
10.1.3 Operational Description.....	1165
10.2 Navigator Subsystem (NAVSS).....	1186
10.2.1 Main Navigator Subsystem (NAVSS).....	1187
10.2.2 MCU Navigator Subsystem (MCU NAVSS).....	1203
10.2.3 Unified DMA Controller (UDMA).....	1212
10.2.4 Ring Accelerator (RINGACC).....	1228
10.2.5 Proxy.....	1241
10.2.6 Secure Proxy.....	1247
10.2.7 Interrupt Aggregator (INTR_AGGR).....	1254
10.2.8 Packet Streaming Interface Link (PSI-L).....	1262
10.2.9 PSIL Subsystem (PSILSS).....	1267
10.2.10 NAVSS North Bridge (NB).....	1271
10.3 Peripheral DMA (PDMA).....	1276
10.3.1 PDMA Controller.....	1276
10.3.2 PDMA Sources.....	1297
10.4 Data Routing Unit (DRU).....	1307
10.4.1 DRU Overview.....	1307
10.4.2 DRU Integration.....	1309
10.4.3 DRU Functional Description.....	1312
<b>11 Time Sync</b> .....	1320
11.1 Time Sync Module (CPTS).....	1321
11.1.1 CPTS Overview.....	1321
11.1.2 CPTS Integration.....	1323
11.1.3 CPTS Functional Description.....	1325
11.2 Timer Manager.....	1332
11.2.1 Timer Manager Overview.....	1332
11.2.2 Timer Manager Integration.....	1334
11.2.3 Timer Manager Functional Description.....	1335
11.2.4 Timer Manager Programming Guide.....	1338
11.3 Time Sync and Compare Events.....	1340
11.3.1 Time Sync Architecture.....	1340
11.3.2 Time Sync Routers.....	1342
11.3.3 Time Sync Event Sources.....	1356
<b>12 Peripherals</b> .....	1366
12.1 General Connectivity Peripherals.....	1367
12.1.1 Analog-to-Digital Converter (ADC).....	1367
12.1.2 General-Purpose Interface (GPIO).....	1386
12.1.3 Inter-Integrated Circuit (I2C) Interface.....	1408
12.1.4 Improved Inter-Integrated Circuit (I3C) Interface.....	1445
12.1.5 Multichannel Serial Peripheral Interface (MCSPI).....	1472

12.1.6 Universal Asynchronous Receiver/Transmitter (UART).....	1534
12.2 High-speed Serial Interfaces.....	1614
12.2.1 Gigabit Ethernet MAC (MCU_CPSW0).....	1615
12.2.2 Gigabit Ethernet Switch (CPSW0).....	1716
12.2.3 Peripheral Component Interconnect Express (PCIe) Subsystem.....	1829
12.2.4 Universal Serial Bus (USB) Subsystem.....	1874
12.2.5 2-L Serializer/Deserializer (SerDes).....	1890
12.2.6 4-L Serializer/Deserializer (SerDes).....	1910
12.3 Memory Interfaces.....	1920
12.3.1 Flash Subsystem (FSS).....	1921
12.3.2 Octal Serial Peripheral Interface (OSPI).....	1933
12.3.3 HyperBus Interface.....	1966
12.3.4 General-Purpose Memory Controller (GPMC).....	1978
12.3.5 Error Location Module (ELM).....	2088
12.3.6 Multimedia Card Secure Digital (MMCSD) Interface.....	2102
12.3.7 Universal Flash Storage (UFS) Interface.....	2189
12.4 Industrial and Control Interfaces.....	2212
12.4.1 Enhanced Capture (ECAP) Module.....	2212
12.4.2 Enhanced Pulse Width Modulation (EPWM) Module.....	2245
12.4.3 Enhanced Quadrature Encoder Pulse (EQEP) Module.....	2337
12.4.4 Controller Area Network (MCAN).....	2369
12.5 Audio Interfaces.....	2435
12.5.1 Audio Tracking Logic (ATL).....	2436
12.5.2 Multichannel Audio Serial Port (MCASP).....	2438
12.6 Display Subsystem (DSS) and Peripherals.....	2513
12.6.1 DSS Overview.....	2513
12.6.2 DSS Environment.....	2520
12.6.3 DSS Integration.....	2533
12.6.4 Display Subsystem Controller (DISPC) with Frame Buffer Decompression Core (FBDC).....	2545
12.6.5 MIPI Display Serial Interface (DSI) Controller.....	3162
12.6.6 Embedded DisplayPort (eDP) Transmitter.....	3225
12.7 Camera Subsystem.....	3254
12.7.1 Camera Streaming Interface Receiver (CSI_RX_IF).....	3255
12.7.2 MIPI D-PHY Receiver (DPHY_RX).....	3283
12.7.3 Camera Streaming Interface Transmitter (CSI_TX_IF).....	3297
12.8 Shared MIPI D-PHY Transmitter (DPHY_TX).....	3317
12.8.1 DPHY_TX Subsystem Overview.....	3317
12.8.2 DPHY_TX Environment.....	3319
12.8.3 DPHY_TX Integration.....	3322
12.9 Video Processing Front End (VPFE).....	3324
12.9.1 VPFE Overview.....	3324
12.9.2 VPFE Environment.....	3326
12.9.3 VPFE Integration.....	3328
12.9.4 VPFE Functional Description.....	3331
12.10 Timer Modules.....	3354
12.10.1 Global Timebase Counter (GTC).....	3354
12.10.2 Windowed Watchdog Timer (WWDT).....	3359
12.10.3 Timers.....	3378
12.11 Internal Diagnostics Modules.....	3430
12.11.1 Dual Clock Comparator (DCC).....	3430
12.11.2 Error Signaling Module (ESM).....	3481
12.11.3 Memory Cyclic Redundancy Check (MCRC) Controller.....	3505
12.11.4 ECC Aggregator.....	3521
<b>13 On-Chip Debug.....</b>	<b>3529</b>
13.1 Introduction to SoC Debug Framework.....	3530
<b>Revision History.....</b>	<b>3532</b>

## List of Figures

Figure 1-1. Device Top-level Block Diagram.....	83
Figure 1-2. Device Main Domain Block Diagram.....	86



Figure 1-3. Device MCU Domain Block Diagram.....	108
Figure 1-4. Device WKUP Domain Block Diagram.....	114
Figure 3-1. Device System Interconnect Overview.....	183
Figure 3-2. ISC and Firewall in SoC.....	227
Figure 3-3. ISC Config Registers per Region.....	229
Figure 3-4. Peripheral Firewall.....	234
Figure 3-5. Peripheral Firewall Config Registers per Regions.....	235
Figure 3-6. Peripherals Firewall Regions.....	236
Figure 3-7. Memory/Data Firewall Config Registers with 3 Priv-ID slot per Region.....	245
Figure 3-8. Memory/Data Firewall Config Registers with 1 Priv-ID slot per Region.....	246
Figure 3-9. Channelized Firewall Config Registers with 1 Priv-ID slot per Region.....	250
Figure 3-10. VBUSM Block Diagram.....	252
Figure 4-1. Initialization Process.....	270
Figure 4-2. MCU ROM Code Architecture.....	273
Figure 4-3. Boot Process.....	275
Figure 4-4. External Bootloader Tasks.....	277
Figure 4-5. xSPI Flow Chart.....	288
Figure 4-6. ROM SFDP Parser Flow.....	289
Figure 5-1. WKUP_CTRL_MMR0 Integration.....	330
Figure 5-2. MCU_CTRL_MMR0 and MCU_SEC_MMR0 Integration.....	338
Figure 5-3. MCU Domain TIMER I/O Multiplexing Scheme.....	342
Figure 5-4. CTRL_MMR0 and SEC_MMR0 Integration.....	346
Figure 5-5. MAIN Domain TIMER I/O Multiplexing Scheme.....	356
Figure 5-6. Overview of Common Power Management.....	360
Figure 5-7. POK Block Diagram.....	362
Figure 5-8. POK_SA Block Diagram.....	363
Figure 5-9. POK Programming Model.....	365
Figure 5-10. POR Overview.....	366
Figure 5-11. POR Integration.....	367
Figure 5-12. POR Block Diagram.....	368
Figure 5-13. POKHV Wrapper.....	370
Figure 5-14. HHV masking and trimming.....	371
Figure 5-15. POKs in POR Module Programming.....	373
Figure 5-16. PRG Overview.....	375
Figure 5-17. PRG Integration.....	376
Figure 5-18. PGD Block Diagram.....	379
Figure 5-19. VTM Overview.....	381
Figure 5-20. WKUP_VTM0 Integration.....	383
Figure 5-21. Top-level IPs for AVS-Class0 and Thermal Management.....	386
Figure 5-22. VTM Alert and Interrupt Generation.....	388
Figure 5-23. ARMi_COREn Power Transitions.....	434
Figure 5-24. A72SS Power Transitions.....	435
Figure 5-25. Overview of Device Management and Security Control (WKUP_DMSC0).....	439
Figure 5-26. Power Modes.....	441
Figure 5-27. Transition between Low Power Modes.....	444
Figure 5-28. Reset Architecture.....	462
Figure 5-29. Top-Level Clock Diagram.....	473
Figure 5-30. MCU_OBSCLK Muxes Diagram.....	476
Figure 5-31. OBSCLK0 Muxes Diagram.....	477
Figure 5-32. WKUP_HFOSC0 Integration Diagram.....	479
Figure 5-33. WKUP_LFOSC0 Integration Diagram.....	480
Figure 5-34. HFOSC1 Integration Diagram.....	480
Figure 5-35. WKUP_HFOSC0 Clock Loss Detection Integration Diagram.....	481
Figure 5-36. WKUP/MCU Domain PLLs Integration.....	483
Figure 5-37. MAIN Domain PLLs Integration - Part 1.....	485
Figure 5-38. MAIN Domain PLLs Integration - Part 2.....	486
Figure 5-39. Generic PLL Functional Diagram for PLLTS16FFCLAFRAC2 Type.....	488
Figure 5-40. Generic PLL Functional Diagram for PLLTS16FFCLAFRACF Type.....	488
Figure 5-41. Generic PLL Functional Diagram for PLLTS16FFCLVDESKEWC Type.....	488
Figure 5-42. PLLTS16FFCLAFRAC2 / PLLTS16FFCLAFRACF and HSDIV Connection.....	492
Figure 5-43. PLLTS16FFCLVDESKEWC and HSDIV Connection.....	493

Figure 5-44. Spreading Generation Block Diagram.....	494
Figure 5-45. PLL and SSMOD Connection.....	496
Figure 5-46. PLL and PLL Controller.....	502
Figure 6-1. COMPUTE_CLUSTER0 Overview.....	509
Figure 6-2. A72SS Integration.....	519
Figure 6-3. A72SS Block Diagram.....	521
Figure 6-4. MCU_R5FSS0 Integration.....	534
Figure 6-5. R5FSS0 Integration.....	538
Figure 6-6. R5FSS1 Integration.....	539
Figure 6-7. R5FSS Block Diagram.....	543
Figure 6-8. VIM RAM Interrupt Vector Map.....	556
Figure 6-9. C66SS Overview.....	564
Figure 6-10. C66SS0 Integration.....	566
Figure 6-11. C66SS1 Integration.....	566
Figure 6-12. L1P Memory Configurations.....	569
Figure 6-13. L1D Memory Configurations.....	570
Figure 6-14. L2 Memory Configurations.....	570
Figure 6-15. C66_RAT_PID Register.....	579
Figure 6-16. C66_RAT_CONFIG Register.....	580
Figure 6-17. C66_RAT_CTRL_j Register.....	581
Figure 6-18. C66_RAT_BASE_j Register.....	582
Figure 6-19. C66_RAT_TRANS_L_j Register.....	583
Figure 6-20. C66_RAT_TRANS_U_j Register.....	584
Figure 6-21. C66_RAT_DESTINATION_ID Register.....	585
Figure 6-22. C66_RAT_EXCEPTION_LOGGING_CONTROL Register.....	586
Figure 6-23. C66_RAT_EXCEPTION_LOGGING_HEADER0 Register.....	587
Figure 6-24. C66_RAT_EXCEPTION_LOGGING_HEADER1 Register.....	588
Figure 6-25. C66_RAT_EXCEPTION_LOGGING_DATA0 Register.....	589
Figure 6-26. C66_RAT_EXCEPTION_LOGGING_DATA1 Register.....	590
Figure 6-27. C66_RAT_EXCEPTION_LOGGING_DATA2 Register.....	591
Figure 6-28. C66_RAT_EXCEPTION_LOGGING_DATA3 Register.....	592
Figure 6-29. C66_RAT_EXCEPTION_PEND_SET Register.....	593
Figure 6-30. C66_RAT_EXCEPTION_PEND_CLEAR Register.....	594
Figure 6-31. C66_RAT_EXCEPTION_ENABLE_SET Register.....	595
Figure 6-32. C66_RAT_EXCEPTION_ENABLE_CLEAR Register.....	596
Figure 6-33. C66_RAT_EOI_REG Register.....	597
Figure 6-34. C71SS Overview.....	599
Figure 6-35. C71SS0 Integration.....	601
Figure 6-36. GPU Module Overview.....	609
Figure 6-37. GPU0 Integration.....	611
Figure 6-38. GPU Block Diagram.....	614
Figure 6-39. DECODER Module Overview.....	616
Figure 6-40. DECODER Integration.....	618
Figure 6-41. DECODER Functional Block Diagram.....	620
Figure 6-42. ENCODER Module Overview.....	621
Figure 6-43. ENCODER Integration.....	623
Figure 6-44. ENCODER Functional Block Diagram.....	625
Figure 6-45. VPAC Overview.....	627
Figure 6-46. VPAC Integration.....	630
Figure 6-47. VPAC Subsystem FW/ISC Configuration.....	643
Figure 6-48. VISS Block Diagram.....	644
Figure 6-49. VISS On-the-fly Operation.....	646
Figure 6-50. VISS Non-WDR/Companded: High Level.....	647
Figure 6-51. VISS Memory to Memory Operation.....	648
Figure 6-52. VISS LSE Channel Integration.....	649
Figure 6-53. RAWFE Block Diagram.....	657
Figure 6-54. RAWFE Decompanding Block Diagram.....	659
Figure 6-55. RAWFE PWL Curve implementation.....	659
Figure 6-56. RAWFE PWL Block Diagram.....	660
Figure 6-57. RAWFE WDR Motion Adaptive Merge.....	662
Figure 6-58. RAWFE DPC Block Diagram.....	662



Figure 6-59. RAWFE DPC LUT Memory Organization.....	663
Figure 6-60. RAWFE DPC Pixel numbering convention.....	664
Figure 6-61. RAWFE OTF DPC.....	664
Figure 6-62. RAWFE Before and After LSC.....	665
Figure 6-63. RAWFE LSC block diagram.....	666
Figure 6-64. RAWFE LSC active region with respect to gain map samples.....	667
Figure 6-65. RAWFE H3A Top-Level Block Diagram.....	668
Figure 6-66. RAWFE H3A Frame Format Settings.....	669
Figure 6-67. RAWFE H3A Red, Green, and Blue Pixel Extraction Examples.....	671
Figure 6-68. RAWFE H3A Horizontal/Vertical FV Pixel Configuration.....	672
Figure 6-69. RAWFE H3A AE/AWB Window Configurations.....	675
Figure 6-70. RAWFE H3A Black Row of Windows Before Regular Rows of Windows.....	676
Figure 6-71. NSF4V Kernel Representation for Decomposition Filters.....	687
Figure 6-72. NSF4V Kernel Representation Filter Alignment.....	688
Figure 6-73. NSF4V Example F1 Kernel Result.....	689
Figure 6-74. NSF4V Local Image Intensity to Noise Threshold Piece-wise Linear Function.....	690
Figure 6-75. GLBCE Iridix Strength Formula.....	692
Figure 6-76. GLBCE Iridix f(Analog Gain) Function.....	693
Figure 6-77. FCP Block Diagram.....	698
Figure 6-78. Gradient Bit Masks.....	699
Figure 6-79. Intensity Bit Mask.....	699
Figure 6-80. Software Controlled Direction Selection.....	700
Figure 6-81. Block Diagram of Edge Enhancer Module Wrapper.....	701
Figure 6-82. Flexible CC Block Diagram (Logical View).....	702
Figure 6-83. Flexible CC Interface Mux.....	703
Figure 6-84. CCM-1.....	703
Figure 6-85. S & V Generation.....	704
Figure 6-86. Gray Scale Computation.....	704
Figure 6-87. Weighted Average Block.....	704
Figure 6-88. Division LUT.....	706
Figure 6-89. LUT Based 12-8 Compression.....	706
Figure 6-90. Linear Interpolation on LUT.....	707
Figure 6-91. Histogram Module.....	708
Figure 6-92. Contrast Enhancement Module.....	709
Figure 6-93. Contrast Enhancement Linera Interpolation.....	709
Figure 6-94. RGB-to-YUV Conversion.....	710
Figure 6-95. RGB to YUV.....	710
Figure 6-96. 444 to 420 Module.....	711
Figure 6-97. Edge Enhancer Block Diagram.....	713
Figure 6-98. Edge Enhancer Linear Filter.....	714
Figure 6-99. Edge Enhancer Indexing.....	714
Figure 6-100. Edge Intensity LUT Formula.....	714
Figure 6-101. 5 × 5 HPF Filter.....	715
Figure 6-102. Edge Sharpener Function.....	715
Figure 6-103. Edge Sharpener Details.....	716
Figure 6-104. Edge Intensity Clipping Formula.....	716
Figure 6-105. Edge Threshold Value Formula.....	716
Figure 6-106. Edge Enhancer and Sharpener Merger Formula.....	717
Figure 6-107. LDC Block Diagram.....	720
Figure 6-108. LDC Affine Transformation.....	722
Figure 6-109. LDC Lens Distortion Back Mapping Offset Calculation.....	725
Figure 6-110. LDC Back Mapping Procedure Using Offset Table.....	726
Figure 6-111. LDC Bi-cubic Interpolation for Y.....	727
Figure 6-112. LDC Bi-linear Interpolation for Y.....	728
Figure 6-113. LDC Bilinear Interpolation for CB/Cr in UYVY and NV12 Format.....	728
Figure 6-114. LDC Input Block Bound.....	729
Figure 6-115. LDC Variable Block Size with Multiple Regions.....	731
Figure 6-116. LDC Region Skipping Example.....	732
Figure 6-117. LDC 2x2 Region Partitioning Example.....	733
Figure 6-118. LDC Multiple-Pass Correction Example.....	734
Figure 6-119. LDC Sample Grids for UYVY.....	735

Figure 6-120. LDC Sample Grids for NV12.....	736
Figure 6-121. LDC YUV420 (2-plane 12-bit Fully Packed Format).....	736
Figure 6-122. LDC YUV422 to YUV420 Conversion.....	737
Figure 6-123. Block Diagram of Multi-Scaler Filter Core.....	744
Figure 6-124. MSC_LSC Block Diagram.....	746
Figure 6-125. MSC_CORE Block Diagram.....	748
Figure 6-126. Polyphase Filter Micro-Architecture.....	749
Figure 6-127. Coefficient Selection.....	753
Figure 6-128. Multiple Scale Generation.....	754
Figure 6-129. Interpolation Principle.....	755
Figure 6-130. Interpolation Process.....	755
Figure 6-131. Polyphase Filtering.....	756
Figure 6-132. Polyphase Filter Unit.....	756
Figure 6-133. Horizontal Pixel Replication.....	757
Figure 6-134. Multiple-ROI Support Illustration.....	758
Figure 6-135. Multiple-ROI From a Common Source ROI.....	758
Figure 6-136. Bilateral Noise Filter Block Diagram.....	765
Figure 6-137. Bilateral Filtering Equation.....	766
Figure 6-138. Range and Space Parameters.....	766
Figure 6-139. NF Hardware Partition.....	767
Figure 6-140. NF_CFG Block Diagram.....	768
Figure 6-141. NF-CORE Block Diagram.....	770
Figure 6-142. Space Distance For 5×5 Pixels.....	771
Figure 6-143. LUT For Combined Weight For Space And Range.....	771
Figure 6-144. LUT Organization For Combined Weights.....	772
Figure 6-145. Weight LUT Logic.....	772
Figure 6-146. Border Handling.....	773
Figure 6-147. 5×5 Weight (Generic 2D Filtering).....	773
Figure 6-148. Weight LUT Usage In Generic Mode.....	774
Figure 6-149. Interleaved Entries In Table.....	775
Figure 6-150. Adaptive Bilateral Filtering.....	776
Figure 6-151. Programming Steps.....	777
Figure 6-152. DMPAC Overview.....	780
Figure 6-153. DMPAC Integration.....	782
Figure 6-154. DMPAC Block Diagram.....	785
Figure 6-155. DMPAC SoC Level Dataflow for Stereo Pre-Processing.....	786
Figure 6-156. DMPAC SoC Level Dataflow for Stereo Processing.....	787
Figure 6-157. DMPAC SoC Level Dataflow for Optical Flow Pre-Processing.....	787
Figure 6-158. DMPAC SoC Level Dataflow for Optical Flow Processing.....	787
Figure 6-159. DMPAC Functional View of Stereo Processing Dataflow.....	788
Figure 6-160. DMPAC Functional View of Optical Flow Processing Dataflow.....	790
Figure 6-161. DMPAC with Format Conversion Engine (FOCO).....	793
Figure 6-162. DMPAC Data Flow with FOCO.....	793
Figure 6-163. DMPAC FOCO Logical Block Diagram.....	794
Figure 6-164. DMPAC Hardware Security.....	802
Figure 6-165. DMPAC 12bpp Optical Flow Processing Initialization Sequence.....	806
Figure 6-166. DMPAC 12bpp Stereo Disparity Processing Initialization Sequence.....	810
Figure 7-1. Mailbox Integration.....	815
Figure 7-2. Mailbox Block Diagram.....	819
Figure 7-3. Example of Communication.....	822
Figure 7-4. Spinlock Overview.....	825
Figure 7-5. Spinlock Integration.....	826
Figure 7-6. Lock Register State Diagram.....	828
Figure 7-7. Take and Release Spinlock.....	830
Figure 8-1. MSMC Overview.....	832
Figure 8-2. MSMC Integration.....	834
Figure 8-3. MSMC Functional Block Diagram.....	837
Figure 8-4. MSMC Memory Organization.....	837
Figure 8-5. Way Partition Allocation Pointer Generation.....	838
Figure 8-6. DDRSS Overview.....	848
Figure 8-7. LPDDR4 Connection.....	851

Figure 8-8. DDRSS0 Integration.....	856
Figure 8-9. DDRSS CoS Mapping Diagram.....	860
Figure 8-10. DDR Controller Functional Blocks.....	867
Figure 8-11. VirtSS Block Diagram.....	904
Figure 8-12. Functional Diagram.....	910
Figure 8-13. Functional Example.....	911
Figure 8-14. TBU Functional Diagram.....	912
Figure 8-15. DTI Functional Diagram.....	913
Figure 8-16. TCU Functional Diagram.....	913
Figure 8-17. PAT Functional Diagram.....	914
Figure 8-18. PVU Functional Diagram.....	915
Figure 8-19. PVU Overview.....	916
Figure 8-20. PVU Integration.....	918
Figure 8-21. PAT Overview.....	924
Figure 8-22. PAT Integration.....	926
Figure 9-1. Compute Cluster Interrupt Architecture.....	936
Figure 9-2. GIC Integration.....	939
Figure 9-3. GIC Block Diagram.....	941
Figure 9-4. CLEC Integration.....	945
Figure 9-5. WKUP_GPIOMUX_INTRTR0 Integration.....	956
Figure 9-6. GPIOMUX_INTRTR0 Integration.....	958
Figure 9-7. MAIN2MCU_LVL_INTRTR0 Integration (TBD).....	960
Figure 9-8. MAIN2MCU_PLS_INTRTR0 Integration.....	961
Figure 9-9. C66SS0_INTRTR0 Integration.....	962
Figure 9-10. C66SS1_INTRTR0 Integration.....	963
Figure 9-11. R5FSS0_INTRTR0 Integration.....	964
Figure 9-12. R5FSS1_INTRTR0 Integration.....	965
Figure 10-1. NAVSS Overview.....	1121
Figure 10-2. UDMA-C Block Diagram.....	1124
Figure 10-3. UDMA-P Block Diagram.....	1125
Figure 10-4. Example of Multiple Buffer Interleaving Read with 4 Offsets.....	1157
Figure 10-5. Example of Interleaved Read with 4 Offsets and Transpose Enabled.....	1158
Figure 10-6. Ring Accelerator Based Queue Structure.....	1164
Figure 10-7. Ring Accelerator Based Direct Transfer Request Queue Structure.....	1165
Figure 10-8. Host Packet Tx Operation – Complete Packet Return.....	1170
Figure 10-9. Host Packet Tx Operation – Automatic Buffer Recycling Packet Return.....	1171
Figure 10-10. Monolithic Packet Tx Operation.....	1172
Figure 10-11. Host Packet Receive Operation.....	1177
Figure 10-12. Monolithic Packet Receive Operation.....	1179
Figure 10-13. NAVSS0 Top-Level Block Diagram.....	1188
Figure 10-14. NAVSS0 Clocks, Resets, and Interrupts.....	1189
Figure 10-15. NAVSS0_CPTS0 Integration.....	1190
Figure 10-16. NAVSS0 Interconnects.....	1191
Figure 10-17. NAVSS Interrupt Flow.....	1200
Figure 10-18. MCU NAVSS Top-Level Block Diagram.....	1203
Figure 10-19. MCU NAVSS Clocks, Resets, and Interrupts.....	1205
Figure 10-20. MCU NAVSS Interconnects.....	1206
Figure 10-21. UDMA Overview.....	1212
Figure 10-22. UDMA Integration.....	1216
Figure 10-23. UDMA Block-Diagram.....	1218
Figure 10-24. RINGACC Overview.....	1228
Figure 10-25. NAVSS0_RINGACC0 Integration.....	1230
Figure 10-26. Ring Accelerator Block-Diagram.....	1232
Figure 10-27. Proxy Overview.....	1241
Figure 10-28. Proxy Integration.....	1243
Figure 10-29. Secure Proxy Integration.....	1248
Figure 10-30. Secure Proxy Block Diagram.....	1249
Figure 10-31. NAVSS0_INTR_AGGR0 Integration.....	1256
Figure 10-32. PSI-L Overview.....	1263
Figure 10-33. PSI-L Connection.....	1264
Figure 10-34. NB Overview.....	1271

Figure 10-35. NB Block Diagram.....	1273
Figure 10-36. MCU Domain PDMA Integration.....	1284
Figure 10-37. MAIN Domain PDMA Integration.....	1285
Figure 10-38. PDMA Block Diagram.....	1287
Figure 10-39. DRU Overview.....	1307
Figure 10-40. DRU Integration.....	1310
Figure 10-41. DRU Functional Diagram.....	1312
Figure 10-42. DRU Software Managed Flow Diagram.....	1314
Figure 11-1. NAVSS0_CPTS0 Overview.....	1321
Figure 11-2. NAVSS0_CPTS0 Integration.....	1323
Figure 11-3. CPTS Block Diagram.....	1326
Figure 11-4. Event FIFO Misalignment Condition.....	1330
Figure 11-5. Timer Manager Overview.....	1332
Figure 11-6. Timer Manager Integration.....	1334
Figure 11-7. Timer Manager Block Diagram.....	1335
Figure 11-8. Timer State Machine.....	1336
Figure 11-9. SoC Time Sync Architecture.....	1341
Figure 11-10. TIMESYNC_INTRTR0 Integration.....	1343
Figure 11-11. CMPEVT_INTRTR0 Integration.....	1346
Figure 11-12. TIMESYNC_INTRTR0_PID Register.....	1351
Figure 11-13. TIMESYNC_INTRTR0_MUXCNTL_y Register.....	1352
Figure 11-14. CMPEVT_INTRTR0_PID Register.....	1354
Figure 11-15. CMPEVT_INTRTR0_MUXCNTL_y Register.....	1355
Figure 12-1. ADC Modules Overview.....	1367
Figure 12-2. ADC Signals.....	1369
Figure 12-3. MCU_ADC[0-1] Integration.....	1373
Figure 12-4. Sequencer FSM.....	1377
Figure 12-5. Example Timing Diagram for Sequencer.....	1381
Figure 12-6. Functional Block Diagram.....	1381
Figure 12-7. GPIO Modules Overview.....	1387
Figure 12-8. GPIO Typical Application.....	1389
Figure 12-9. GPIO Interface Signals.....	1390
Figure 12-10. WKUP_GPIO[0-1] Integration.....	1394
Figure 12-11. GPIO <sub>n</sub> Integration.....	1396
Figure 12-12. GPIO <sub>m</sub> Integration.....	1397
Figure 12-13. GPIO Block Diagram.....	1401
Figure 12-14. WKUP_GPIO Interrupt Router Connectivity.....	1404
Figure 12-15. GPIO Interrupt Router Connectivity.....	1404
Figure 12-16. I2C Modules Overview.....	1409
Figure 12-17. I2C and Typical Connections to I2C Devices.....	1411
Figure 12-18. I2C Interface Signals.....	1411
Figure 12-19. I2C Data Transfer.....	1412
Figure 12-20. I2C Bit Transfer on the I2C Bus.....	1413
Figure 12-21. I2C S and P Condition Events.....	1413
Figure 12-22. I2C Data Transfer Formats in F/S Mode.....	1414
Figure 12-23. HS I2C Data Transfer in HS Mode.....	1414
Figure 12-24. I2C Arbitration Between Controller Transmitters.....	1415
Figure 12-25. I2C Clock Generators Synchronization.....	1416
Figure 12-26. WKUP_I2C0 Integration.....	1417
Figure 12-27. MCU_I2C[0-1] Integration.....	1419
Figure 12-28. I2C[0-6] Integration.....	1422
Figure 12-29. I2C Block Diagram.....	1426
Figure 12-30. I2C Clock Generation.....	1427
Figure 12-31. I2C Receive FIFO Interrupt Request Generation.....	1431
Figure 12-32. I2C Transmit FIFO Interrupt Request Generation.....	1432
Figure 12-33. I2C Setup Procedure.....	1437
Figure 12-34. I2C Controller Transmitter Mode, Polling Method, in F/S and HS Modes.....	1438
Figure 12-35. I2C Controller Receiver Mode, Polling Method, in F/S and HS Modes.....	1439
Figure 12-36. I2C Controller Transmitter Mode, Interrupt Method, in F/S and HS Modes.....	1440
Figure 12-37. I2C Controller Receiver Mode, Interrupt Method, in F/S and HS Modes.....	1442
Figure 12-38. I2C Target Transmitter/Receiver Mode, Polling.....	1443

Figure 12-39. I2C Target Transmitter/Receiver Mode, Interrupt.....	1443
Figure 12-40. I3C Modules Overview.....	1445
Figure 12-41. I3C and Typical Connections to I3C Devices.....	1447
Figure 12-42. I3C Interface Signals.....	1447
Figure 12-43. MCU_I3C[0-1] Integration.....	1449
Figure 12-44. I3C Integration.....	1452
Figure 12-45. I3C Block Diagram.....	1454
Figure 12-46. I3C Base Frequency.....	1455
Figure 12-47. SCL Asymmetric Push-Pull.....	1455
Figure 12-48. SCL Asymmetric Push-Pull Actual Low Period.....	1455
Figure 12-49. SCL Open-Drain Low Period.....	1456
Figure 12-50. SCL Open-Drain Actual Low Period.....	1456
Figure 12-51. Power-up I3C Bus Initialization.....	1459
Figure 12-52. Flow Diagram for DAA Procedure.....	1460
Figure 12-53. Master Write Transaction for Single and Multi Address Sequences.....	1465
Figure 12-54. Master Read Transaction for Single and Multi Address Sequences.....	1465
Figure 12-55. MCSPI Overview.....	1473
Figure 12-56. MCSPI Interface Signals in Master Mode <sup>(4)</sup> .....	1475
Figure 12-57. MCSPI Interface Signals in Slave Mode <sup>(3)</sup> .....	1477
Figure 12-58. MCSPI3 and MCU_MCSPI1 Connectivity Details.....	1479
Figure 12-59. MCU_MCSPI2 to MCSPI4 Internal Connectivity.....	1480
Figure 12-60. Phase and Polarity Combinations.....	1482
Figure 12-61. Full-Duplex Transfer Format With PHA = 0.....	1483
Figure 12-62. Extended MCSPI Transfer With a Start-Bit (SBE = 1).....	1484
Figure 12-63. MCSPI Controller Mode (Full Duplex).....	1484
Figure 12-64. MCSPI Controller Single Mode (Receive Only).....	1485
Figure 12-65. MCSPI Peripheral Mode (Full Duplex).....	1485
Figure 12-66. MCSPI Peripheral Single Mode (Transmit Only).....	1486
Figure 12-67. MCU_MCSPI Integration.....	1488
Figure 12-68. MCSPI Integration.....	1492
Figure 12-69. SPI Block Diagram.....	1499
Figure 12-70. MCSPI Full-Duplex Transmission (Example).....	1501
Figure 12-71. Continuous Transfers With SPIEN[i] Maintained Active (Single-Data-Pin Interface Mode).....	1503
Figure 12-72. Continuous Transfers With SPIEN[i] Maintained Active (Dual-Data-Pin Interface Mode).....	1503
Figure 12-73. CS SPIEN Timing Controls.....	1504
Figure 12-74. Example of MCSPI Peripheral With One Controller and Multiple Peripheral Devices on Channel 0.....	1507
Figure 12-75. MCSPI Half-Duplex Transmission (Transmit-Only peripheral).....	1509
Figure 12-76. MCSPI Half-Duplex Transmission (Receive-Only peripheral).....	1510
Figure 12-77. Buffer Used in Transmit Direction Only.....	1511
Figure 12-78. Buffer Used in Receive Direction Only.....	1511
Figure 12-79. Buffer Used for Transmit and Receive Directions.....	1511
Figure 12-80. Buffer Almost Full Level (AFL).....	1512
Figure 12-81. Buffer Almost Empty Level (AEL).....	1513
Figure 12-82. Controller Single Channel Initial Delay.....	1514
Figure 12-83. FIFO Mode Transmit-and-Receive With Word Count (Controller).....	1528
Figure 12-84. FIFO Mode Transmit-and-Receive Without Word Count (Controller).....	1529
Figure 12-85. FIFO Mode Transmit-Only (Controller).....	1530
Figure 12-86. FIFO Mode Receive-Only With Word Count (Controller).....	1531
Figure 12-87. FIFO Mode Receive-Only Without Word Count (Controller).....	1532
Figure 12-88. UART Modules Overview.....	1535
Figure 12-89. UART Mode Interface Signals.....	1537
Figure 12-90. UART Frame Data Format.....	1539
Figure 12-91. RS-485 Mode Interface Signals.....	1540
Figure 12-92. IrDA Mode Interface Signals.....	1541
Figure 12-93. IrDA SIR Frame Format.....	1543
Figure 12-94. IrDA SIR Encoding Mechanism.....	1544
Figure 12-95. IrDA SIR Decoding Mechanism.....	1545
Figure 12-96. SIR FF Mode.....	1546
Figure 12-97. MIR Transmit Frame Format.....	1546
Figure 12-98. MIR Baud Rate Adjustment Mechanism.....	1547
Figure 12-99. SIP.....	1548



Figure 12-100. FIR Transmit Frame Format.....	1548
Figure 12-101. CIR Mode Interface Signals.....	1550
Figure 12-102. CIR Pulse Modulation.....	1551
Figure 12-103. CIR Modulation Duty Cycle.....	1552
Figure 12-104. UART RC-5 Bit Encoding.....	1553
Figure 12-105. UART SIRC Bit Encoding.....	1553
Figure 12-106. UART RC-5 Standard Packet Format.....	1554
Figure 12-107. UART SIRC Packet Format.....	1554
Figure 12-108. UART SIRC Bit Transmission Example.....	1554
Figure 12-109. WKUP_UART0 Integration.....	1556
Figure 12-110. MCU_UART0 Integration.....	1558
Figure 12-111. UART[0-2] Integration.....	1561
Figure 12-112. UART[3-9] Integration.....	1562
Figure 12-113. UART Functional Block Diagram.....	1568
Figure 12-114. UART FIFO Management Registers.....	1574
Figure 12-115. UART RX FIFO Interrupt Request Generation.....	1576
Figure 12-116. UART TX FIFO Interrupt Request Generation.....	1576
Figure 12-117. UART Receive FIFO DMA Request Generation (32 Characters).....	1579
Figure 12-118. UART Transmit FIFO DMA Request Generation (56 Spaces).....	1580
Figure 12-119. UART Transmit FIFO DMA Request Generation (8 Spaces).....	1581
Figure 12-120. UART Transmit FIFO DMA Request Generation (1 Space).....	1582
Figure 12-121. UART Transmit FIFO DMA Request Generation Using Direct TX DMA Threshold Programming. (Threshold = 3; Spaces = 8).....	1583
Figure 12-122. DMA Transmission.....	1583
Figure 12-123. DMA Reception.....	1584
Figure 12-124. UART Baud Rate Generation.....	1590
Figure 12-125. RS-485 External Transceiver Direction Control.....	1596
Figure 12-126. IrDA Baud Rate Generator.....	1597
Figure 12-127. CIR Mode Block Components.....	1603
Figure 12-128. MCU_CPSW0 Overview.....	1615
Figure 12-129. RMII Interface Typical Application (Internal Clock Source).....	1618
Figure 12-130. RMII Interface Typical Application (External Clock Source).....	1619
Figure 12-131. RGMII Interface Typical Application.....	1620
Figure 12-132. MCU_CPSW0 Integration.....	1622
Figure 12-133. CPSW Top Level Block Diagram.....	1627
Figure 12-134. CPSW_2G Block Diagram.....	1630
Figure 12-135. Gigabit Ethernet Switch Priority Mapping and Transmit VLAN Processing.....	1644
Figure 12-136. The Network Static with AVB.....	1654
Figure 12-137. AVB Network & PTP Clock Entities.....	1656
Figure 12-138. IEEE 1722 Packets.....	1656
Figure 12-139. Cross Time Stamping and Presentation Timestamps.....	1657
Figure 12-140. AV Stream Queuing/Policing.....	1659
Figure 12-141. MCU_CPSW0 CPTS Integration.....	1688
Figure 12-142. CPTS Block Diagram.....	1689
Figure 12-143. CPTS_COMP Output in Toggle and Non-Toggle Mode.....	1692
Figure 12-144. CPTS_GENFn Output Signal Diagram.....	1693
Figure 12-145. Event FIFO Misalignment Condition.....	1695
Figure 12-146. Partial Ethernet-II Frames Showing Register Mapping of EtherTypes for a Simple Frame (1), a Single 1Q Tag Added (2), and Two 1Q Tags Added (3).....	1696
Figure 12-147. TX INFO Word 0 Format.....	1705
Figure 12-148. TX INFO Word 1 Format.....	1706
Figure 12-149. TX INFO Word 2 Format.....	1706
Figure 12-150. TX INFO Word 3 Format.....	1706
Figure 12-151. TX Status Data Word 0 Format.....	1707
Figure 12-152. TX Status Data Word 1 Format.....	1707
Figure 12-153. TX Status Data Word 2 Format.....	1707
Figure 12-154. TX Status Data Word 3 Format.....	1708
Figure 12-155. RX INFO Word 0 Format.....	1708
Figure 12-156. RX INFO Word 2 Format.....	1709
Figure 12-157. RX Control Data Word 1 Format.....	1709
Figure 12-158. RX Control Data Word 2 Format.....	1709

Figure 12-159. CPSW0 Overview.....	1716
Figure 12-160. RMI Interface Typical Application (Internal Clock Source).....	1720
Figure 12-161. RMI Interface Typical Application (External Clock Source).....	1721
Figure 12-162. RGMII Interface Typical Application.....	1722
Figure 12-163. CPSW0 Integration.....	1725
Figure 12-164. CPSW Top Level Block Diagram.....	1730
Figure 12-165. CPSW_9G Block Diagram.....	1733
Figure 12-166. Gigabit Ethernet Switch Priority Mapping and Transmit VLAN Processing.....	1754
Figure 12-167. The Network Static with AVB.....	1764
Figure 12-168. AVB Network & PTP Clock Entities.....	1765
Figure 12-169. IEEE 1722 Packets.....	1766
Figure 12-170. Cross Time Stamping and Presentation Timestamps.....	1767
Figure 12-171. AV Stream Queuing/Policing.....	1769
Figure 12-172. CPSW0 CPTS Integration.....	1801
Figure 12-173. CPTS Block Diagram.....	1802
Figure 12-174. CPTS_COMP Output in Toggle and Non-Toggle Mode.....	1805
Figure 12-175. CPTS_GENFn Output Signal Diagram.....	1806
Figure 12-176. Event FIFO Misalignment Condition.....	1808
Figure 12-177. Partial Ethernet-II Frames Showing Register Mapping of EtherTypes for a Simple Frame (1), a Single 1Q Tag Added (2), and Two 1Q Tags Added (3).....	1809
Figure 12-178. TX INFO Word 0 Format.....	1818
Figure 12-179. TX INFO Word 1 Format.....	1818
Figure 12-180. TX INFO Word 2 Format.....	1819
Figure 12-181. TX INFO Word 3 Format.....	1819
Figure 12-182. TX Status Data Word 0 Format.....	1819
Figure 12-183. TX Status Data Word 1 Format.....	1820
Figure 12-184. TX Status Data Word 2 Format.....	1820
Figure 12-185. TX Status Data Word 3 Format.....	1820
Figure 12-186. RX INFO Word 0 Format.....	1821
Figure 12-187. RX INFO Word 2 Format.....	1821
Figure 12-188. RX Control Data Word 1 Format.....	1822
Figure 12-189. RX Control Data Word 2 Format.....	1822
Figure 12-190. PCIe Subsystems Overview.....	1829
Figure 12-191. PCIe Subsystem Enviroment.....	1831
Figure 12-192. PCIe Subsystem Integration.....	1834
Figure 12-193. PCIe Subsystem Block Diagram.....	1850
Figure 12-194. PCIE Root Port Inbound PCIe to AXI Address Translation.....	1863
Figure 12-195. PCIE End Point Inbound PCIe to AXI Address Translation.....	1864
Figure 12-196. PCIe Subsystem VirtID Mapper.....	1866
Figure 12-197. Parity Logic on AXI AWADDR Input.....	1872
Figure 12-198. USB Subsystem Overview.....	1875
Figure 12-199. USB3SS0 Subsystem Environment.....	1878
Figure 12-200. USB3SS1 Subsystem Environment.....	1879
Figure 12-201. USB3SS0 Integration.....	1883
Figure 12-202. USB3SS1 Integration.....	1884
Figure 12-203. SERDES0 Overview.....	1891
Figure 12-204. SERDES1 Overview.....	1892
Figure 12-205. SERDES2 Overview.....	1893
Figure 12-206. SERDES3 Overview.....	1894
Figure 12-207. SERDES0 Environment.....	1896
Figure 12-208. SERDES1 Environment.....	1898
Figure 12-209. SERDES2 Environment.....	1899
Figure 12-210. SERDES3 Environment.....	1901
Figure 12-211. 2-L SerDes Integration.....	1903
Figure 12-212. SerDes and WIZ Block Diagram.....	1909
Figure 12-213. SERDES4 Overview.....	1911
Figure 12-214. 4-L SerDes Environment.....	1913
Figure 12-215. 4-L SerDes Integration.....	1916
Figure 12-216. 4-L SerDes and WIZ Block Diagram.....	1919
Figure 12-217. FSS Overview.....	1921
Figure 12-218. MCU_FSS0 Typical Application - HyperBus Interface and OSPI1 .....	1923

Figure 12-219. MCU_FSS0 Typical Application - OSPI0 and OSPI1 .....	1923
Figure 12-220. MCU_FSS0 Integration.....	1925
Figure 12-221. FSS Block Diagram.....	1927
Figure 12-222. OSPI Overview.....	1934
Figure 12-223. OSPI Connected to an External Octal-SPI Flash Memory.....	1936
Figure 12-224. FSS0_OSPI Integration.....	1939
Figure 12-225. OSPI Block Diagram.....	1940
Figure 12-226. Read Data Capture Logic.....	1942
Figure 12-227. HyperBus Module Overview.....	1966
Figure 12-228. HyperBus Connected to HyperRAM or HyperFlash.....	1968
Figure 12-229. MCU_FSS0_HPBO Integration.....	1970
Figure 12-230. HyperBus Module Block Diagram.....	1972
Figure 12-231. GPMC0 Overview.....	1978
Figure 12-232. GPMC to 16-Bit Address/Data-Multiplexed Memory.....	1980
Figure 12-233. GPMC to 16-Bit Non-Multiplexed Memory.....	1981
Figure 12-234. GPMC to 8-Bit Non-Multiplexed Memory.....	1982
Figure 12-235. GPMC to 8-Bit NAND Device.....	1982
Figure 12-236. GPMC0 Integration.....	1987
Figure 12-237. GPMC Block Diagram.....	1990
Figure 12-238. Chip-Select Address Mapping and Decoding Mask.....	1994
Figure 12-239. Wait Behavior During an Asynchronous Single Read Access (GPMCFCLKDIVIDER = 1).....	1997
Figure 12-240. Wait Behavior During a Synchronous Read Burst Access.....	1999
Figure 12-241. Read-to-Read for an Address-Data Multiplexed Device, on Different Chip-Select, Without Bus Turnaround (nCS Attached to a Fast Device).....	2001
Figure 12-242. Read- to-Read/Write for an Address-Data Multiplexed Device, on Different Chip-Select, With Bus Turnaround.....	2001
Figure 12-243. Read-to-Read/Write for a Address-Data or AAD-Multiplexed Device, on Same Chip-Select, With Bus Turnaround.....	2002
Figure 12-244. Asynchronous Single Read on an Address/Data-Multiplexed Device.....	2011
Figure 12-245. Two Asynchronous Single-Read Accesses on an Address/Data-Multiplexed Device (32-Bit Read Split Into 2 x 16-Bit Read).....	2012
Figure 12-246. Asynchronous Single-Write on an Address/Data-Multiplexed Device.....	2013
Figure 12-247. Asynchronous Single Read on an AAD-Multiplexed Device.....	2014
Figure 12-248. Asynchronous Single Write on an AAD-Multiplexed Device.....	2016
Figure 12-249. Synchronous Single Read (GPMCFCLKDIVIDER = 0).....	2018
Figure 12-250. Synchronous Single Read (GPMCFCLKDIVIDER = 1).....	2019
Figure 12-251. Synchronous Multiple (Burst) Read (GPMCFCLKDIVIDER = 0).....	2021
Figure 12-252. Synchronous Multiple (Burst) Read (GPMCFCLKDIVIDER = 1).....	2022
Figure 12-253. Synchronous Single Write on an Address/Data-Multiplexed Device.....	2023
Figure 12-254. Synchronous Multiple Write (Burst Write) in Address/Data-Multiplexed Mode.....	2024
Figure 12-255. Synchronous Multiple Write (Burst Write) in Address/Address/Data-Multiplexed Mode.....	2025
Figure 12-256. Asynchronous Single Read on an Address/Data-non-multiplexed Device.....	2027
Figure 12-257. Asynchronous Single Write on an Address/Data-non-multiplexed Device.....	2028
Figure 12-258. Asynchronous Multiple (Page Mode) Read.....	2029
Figure 12-259. NAND Command Latch Cycle.....	2034
Figure 12-260. NAND Address Latch Cycle.....	2035
Figure 12-261. NAND Data Read Cycle.....	2036
Figure 12-262. NAND Data Write Cycle.....	2036
Figure 12-263. Hamming Code Accumulation Algorithm (1/2).....	2041
Figure 12-264. Hamming Code Accumulation Algorithm (2/2).....	2042
Figure 12-265. ECC Computation for a 256-Byte Data Stream (Read or Write).....	2042
Figure 12-266. ECC Computation for a 512-Byte Data Stream (Read or Write).....	2043
Figure 12-267. 128 Word16 ECC Computation.....	2044
Figure 12-268. 256 Word16 ECC Computation.....	2044
Figure 12-269. Manual Mode Sequence and Mapping.....	2049
Figure 12-270. NAND Page Mapping and ECC: Per-Sector Schemes.....	2053
Figure 12-271. NAND Page Mapping and ECC: Pooled Spare Schemes.....	2054
Figure 12-272. NAND Page Mapping and ECC: Per-Sector Schemes, With Separate ECC.....	2055
Figure 12-273. NAND Read Cycle Optimization Timing Description.....	2062
Figure 12-274. GPMC Connection to an External NOR Flash Memory.....	2064
Figure 12-275. Synchronous Burst Read Access (Timing Parameters in Clock Cycles).....	2066



Figure 12-276. Asynchronous Single Read Access (Timing Parameters in Clock Cycles).....	2067
Figure 12-277. Asynchronous Single Write Access (Timing Parameters in Clock Cycles).....	2068
Figure 12-278. Programming Model Top-Level Diagram.....	2073
Figure 12-279. NOR Interfacing Timing Parameters Diagram.....	2078
Figure 12-280. NAND Command Latch Cycle Timing Simplified Example.....	2081
Figure 12-281. Synchronous NOR Single Read Simplified Example.....	2085
Figure 12-282. Asynchronous NOR Single Write Simplified Example.....	2087
Figure 12-283. ELM0 Overview.....	2089
Figure 12-284. ELM0 Integration.....	2091
Figure 12-285. MMCSDi Module Overview.....	2102
Figure 12-286. MMCSDi Connected to MMC, SD, or SDIO Device.....	2105
Figure 12-287. Sequential Read Operation (MMCs Only).....	2107
Figure 12-288. Sequential Write Operation (MMCs Only).....	2108
Figure 12-289. Multiple Block Read Operation.....	2108
Figure 12-290. Multiple Block Write Operation With Card Busy Signal.....	2109
Figure 12-291. Command Token Format.....	2109
Figure 12-292. Response Token Format (R1/R1b, R3, R4, R5/R5b, R6, R7).....	2110
Figure 12-293. Response Token Format (R2).....	2110
Figure 12-294. Data Token Format for 1-Bit Transfers.....	2110
Figure 12-295. Data Token Format for 4-Bit Transfers.....	2111
Figure 12-296. Data Token Format for 8-Bit Transfers.....	2111
Figure 12-297. MMCSD Integration.....	2113
Figure 12-298. MMCSDi Module Block Diagram.....	2117
Figure 12-299. ECC Aggregator Block Diagram.....	2119
Figure 12-300. 128-byte Test Pattern.....	2120
Figure 12-301. Double Box Notation.....	2129
Figure 12-302. SD Card Detect Sequence.....	2129
Figure 12-303. Internal Clock Setup Sequence.....	2131
Figure 12-304. SD Clock Supply and Stop Sequence.....	2132
Figure 12-305. SD Clock Change Sequence.....	2133
Figure 12-306. SD Bus Power Control Sequence.....	2134
Figure 12-307. Change Bus Width Sequence.....	2135
Figure 12-308. Timeout Setting Sequence.....	2136
Figure 12-309. Card Initialization and Identification (1).....	2137
Figure 12-310. Card Initialization and Identification (2).....	2138
Figure 12-311. Signal Voltage Switch Procedure.....	2141
Figure 12-312. SD Command Issue Sequence.....	2143
Figure 12-313. Command Complete Sequence.....	2145
Figure 12-314. Transaction Control with Data Transfer Using DAT Line Sequence (Not using DMA).....	2147
Figure 12-315. Transaction Control with Data Transfer Using DAT Line Sequence (Using SDMA).....	2149
Figure 12-316. Transaction Control with Data Transfer Using DAT Line Sequence (Using ADMA).....	2151
Figure 12-317. Asynchronous Abort Sequence.....	2153
Figure 12-318. Synchronous Abort Sequence.....	2154
Figure 12-319. Changing Bus Speed Mode.....	2155
Figure 12-320. Error Report and Recovery.....	2156
Figure 12-321. Return Status of Auto CMD12 Error Recovery.....	2157
Figure 12-322. Error Interrupt Recovery Sequence (1).....	2158
Figure 12-323. Error Interrupt Recovery Sequence (2).....	2159
Figure 12-324. Auto CMD12 Error Recovery.....	2161
Figure 12-325. Wakeup Control before Standby Mode.....	2163
Figure 12-326. Wakeup from Standby.....	2163
Figure 12-327. The Sequence for Suspend.....	2165
Figure 12-328. Suspend/Resume Condition.....	2166
Figure 12-329. The Sequence for Resume.....	2167
Figure 12-330. Wait Read Transfer by Stop At Block Gap Request.....	2168
Figure 12-331. Stop At Block Gap Request is Not Accepted at the Last Block of the Read Transfer.....	2169
Figure 12-332. Continue Read Transfer by Continue Request.....	2169
Figure 12-333. Wait Write Transfer by Stop At Block Gap Request.....	2170
Figure 12-334. Stop At Block Gap Request is Not Accepted at the Last Block of the Write Transfer.....	2170
Figure 12-335. Continue Write Transfer by Continue Request.....	2171
Figure 12-336. Host Controller Setup Sequence.....	2172

Figure 12-337. Card Interface Detection Sequence.....	2174
Figure 12-338. Boot Operation Sequence.....	2176
Figure 12-339. Boot Code Access Flow Diagram (1).....	2179
Figure 12-340. Boot Code Access Flow Diagram (2).....	2180
Figure 12-341. MMCSD Clock Tuning.....	2181
Figure 12-342. Command Queuing Initialization Sequence.....	2182
Figure 12-343. Task Issuance Sequence.....	2183
Figure 12-344. Task Execution and Completion Sequence.....	2185
Figure 12-345. Task Discard and Clear Sequence.....	2186
Figure 12-346. Error Detect and Recovery Sequence.....	2187
Figure 12-347. UFS Module Overview.....	2190
Figure 12-348. UFS I/O Interface Signals.....	2191
Figure 12-349. UFS0 Integration.....	2192
Figure 12-350. UFS Subsystem Block Diagram.....	2194
Figure 12-351. UFS IPS Subsystem.....	2195
Figure 12-352. UFS Controller Block Diagram.....	2196
Figure 12-353. UFS System Level Block Diagram.....	2197
Figure 12-354. UFS System Model.....	2198
Figure 12-355. UFS Host Controller Reset Waveform.....	2203
Figure 12-356. General UPIU Structure.....	2205
Figure 12-357. ECAP Overview.....	2212
Figure 12-358. ECAP External Interface I/Os.....	2214
Figure 12-359. ECAP Integration.....	2216
Figure 12-360. ECAP Daisy-Chain Connectivity.....	2219
Figure 12-361. Multiple ECAP Modules.....	2220
Figure 12-362. Capture and APWM Modes of Operation.....	2221
Figure 12-363. Capture Function Diagram.....	2222
Figure 12-364. Event Prescale Control.....	2223
Figure 12-365. Prescale Function Waveforms.....	2223
Figure 12-366. ECAP Continuous/One-shot Block Diagram.....	2224
Figure 12-367. ECAP Counter and Synchronization Block Diagram.....	2225
Figure 12-368. Interrupts in ECAP Module.....	2227
Figure 12-369. PWM Waveform Details of ECAP APWM Mode Operation.....	2228
Figure 12-370. Capture Sequence for Absolute Time-Stamp, Rising Edge Detect.....	2231
Figure 12-371. Capture Sequence for Absolute Time-Stamp, Rising and Falling Edge Detect.....	2233
Figure 12-372. Capture Sequence for Delta Mode Time-Stamp, Rising Edge Detect.....	2235
Figure 12-373. Capture Sequence for Delta Mode Time-Stamp, Rising and Falling Edge Detect.....	2237
Figure 12-374. PWM Waveform Details of APWM Mode Operation.....	2239
Figure 12-375. Multichannel PWM Example Using 4 ECAP Modules.....	2241
Figure 12-376. Multiphase (channel) Interleaved PWM Example Using 3 ECAP Modules.....	2244
Figure 12-377. EPWM Overview.....	2247
Figure 12-378. Multiple EPWM Modules.....	2248
Figure 12-379. Submodules and Signal Connections for an EPWM Module.....	2249
Figure 12-380. EPWM Submodules and Critical Internal Signal Interconnects.....	2250
Figure 12-381. Multiple EPWM Modules.....	2251
Figure 12-382. Submodules and Signal Connections for an EPWM Module.....	2252
Figure 12-383. EPWM Submodules and Critical Internal Signal Interconnects.....	2253
Figure 12-384. EPWM External Interface I/Os.....	2255
Figure 12-385. EPWM Integration.....	2259
Figure 12-386. EPWM Tripzone Connectivity Detail.....	2267
Figure 12-387. Daisy-Chain Connectivity between EPWM Modules.....	2270
Figure 12-388. Interconnectivity of ADC start of conversion.....	2273
Figure 12-389. EPWM Time-Base Submodule.....	2278
Figure 12-390. EPWM Time-Base Submodule Signals and Registers.....	2279
Figure 12-391. EPWM Time-Base Frequency and Period.....	2281
Figure 12-392. EPWM Time-Base Counter Synchronization Scheme 1.....	2282
Figure 12-393. EPWM Time-Base Up-Count Mode Waveforms.....	2283
Figure 12-394. EPWM Time-Base Down-Count Mode Waveforms.....	2284
Figure 12-395. EPWM Time-Base Up-Down-Count Waveforms, EPWM_TBCTL[13] PHSDIR = 0 Count Down on Synchronization Event.....	2284

Figure 12-396. EPWM Time-Base Up-Down Count Waveforms, EPWM_TBCTL[13] PHSDIR = 1 Count Up on Synchronization Event.....	2285
Figure 12-397. EPWM Counter-Compare Submodule.....	2286
Figure 12-398. EPWM Counter-Compare Submodule Signals and Registers.....	2287
Figure 12-399. Compare A Dual Shadow register.....	2289
Figure 12-400. Compare B Dual Shadow register.....	2289
Figure 12-401. EPWM Counter-Compare Event Waveforms in Up-Count Mode.....	2290
Figure 12-402. EPWM Counter-Compare Events in Down-Count Mode.....	2290
Figure 12-403. EPWM Counter-Compare Events in Up-Down-Count Mode, EPWM_TBCTL[13] PHSDIR = 0 Count Down on Synchronization Event.....	2291
Figure 12-404. EPWM Counter-Compare Events in Up-Down-Count Mode, EPWM_TBCTL[13] PHSDIR = 1 Count Up on Synchronization Event.....	2291
Figure 12-405. EPWM Action-Qualifier Submodule.....	2292
Figure 12-406. EPWM Action-Qualifier Submodule Inputs and Outputs.....	2293
Figure 12-407. Possible Action-Qualifier Actions for EPWMxA and EPWMxB Outputs.....	2294
Figure 12-408. EPWM Up-Down-Count Mode Symmetrical Waveform.....	2297
Figure 12-409. Up, Single Edge Asymmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Active High.....	2298
Figure 12-410. Up, Single Edge Asymmetric Waveform With Independent Modulation on EPWMxA and EPWMxB — Active Low.....	2300
Figure 12-411. Up-Count, Pulse Placement Asymmetric Waveform With Independent Modulation on EPWMxA.....	2302
Figure 12-412. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Active Low.....	2304
Figure 12-413. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Complementary.....	2306
Figure 12-414. Up-Down-Count, Dual Edge Asymmetric Waveform, With Independent Modulation on EPWMxA — Active Low.....	2308
Figure 12-415. Dead-Band Generator Submodule.....	2310
Figure 12-416. Configuration Options for the EPWM Dead-Band Generator Submodule.....	2311
Figure 12-417. Dead-Band Waveforms for Typical Cases (0% < Duty < 100%).....	2313
Figure 12-418. PWM-Chopper Submodule.....	2314
Figure 12-419. PWM-Chopper Submodule Signals and Registers.....	2315
Figure 12-420. Simple EPWM-Chopper Submodule Waveforms Showing Chopping Action Only.....	2316
Figure 12-421. EPWM-Chopper Submodule Waveforms Showing the First Pulse and Subsequent Sustaining Pulses.....	2316
Figure 12-422. EPWM-Chopper Submodule Waveforms Showing the Pulse Width (Duty Cycle) Control of Sustaining Pulses.....	2317
Figure 12-423. EPWM Trip-Zone Submodule.....	2318
Figure 12-424. EPWM Trip-Zone Submodule Mode Control Logic.....	2321
Figure 12-425. EPWM Trip-Zone Submodule Interrupt Logic.....	2322
Figure 12-426. EPWM Event-Trigger Submodule.....	2322
Figure 12-427. EPWM Event-Trigger Submodule Inter-Connectivity to Interrupt Controller.....	2323
Figure 12-428. EPWM Event-Trigger Submodule Showing Event Inputs and Prescaled Outputs.....	2324
Figure 12-429. EPWM Event-Trigger Interrupt Generator.....	2326
Figure 12-430. EPWM Event-Trigger SOCA Pulse Generator.....	2326
Figure 12-431. EPWM Event-Trigger SOCB Pulse Generator.....	2327
Figure 12-432. HRPWM System Interface.....	2328
Figure 12-433. Resolution Calculations for Conventionally Generated PWM.....	2329
Figure 12-434. Operating Logic Using MEP.....	2330
Figure 12-435. Required PWM Waveform for a Requested Duty = 40.5%.....	2332
Figure 12-436. Low % Duty Cycle Range Limitation Example When PWM Frequency = 1 MHz.....	2334
Figure 12-437. High % Duty Cycle Range Limitation Example when PWM Frequency = 1 MHz.....	2334
Figure 12-438. Optical Encoder Disk.....	2337
Figure 12-439. QEP Encoder Output Signal for Forward/Reverse Movement.....	2338
Figure 12-440. Index Pulse Example.....	2338
Figure 12-441. EQEP Overview.....	2340
Figure 12-442. EQEP External Interface I/Os.....	2342
Figure 12-443. EQEP Integration.....	2345
Figure 12-444. Functional Block Diagram of the EQEP Peripheral.....	2349
Figure 12-445. Functional Block Diagram of Decoder Unit.....	2351
Figure 12-446. Quadrature Decoder State Machine.....	2353
Figure 12-447. Quadrature-clock and Direction Decoding.....	2353

Figure 12-448. Position Counter Reset by Index Pulse for 1000 Line Encoder (QPOS MAX = 3999 or F9Fh).....	2355
Figure 12-449. Position Counter Underflow/Overflow (QPOS MAX = 4).....	2356
Figure 12-450. Software Index Marker for 1000-line Encoder (EQEP_QDEC_QEP_CTL[21-20] IEL = 0b01).....	2358
Figure 12-451. EQEP Strobe Event Latch (EQEP_QDEC_QEP_CTL[22] SEL = 0b1).....	2359
Figure 12-452. EQEP Position-compare Unit.....	2361
Figure 12-453. EQEP Position-compare Event Generation Points.....	2362
Figure 12-454. EQEP Position-compare Sync Output Pulse Stretcher.....	2362
Figure 12-455. EQEP Edge Capture Unit.....	2364
Figure 12-456. Unit Position Event for Low Speed Measurement (EQEP_QCAP_QPOS_CTL[UPPS] = 0010).....	2364
Figure 12-457. EQEP Edge Capture Unit - Timing Details.....	2365
Figure 12-458. EQEP Watchdog Timer.....	2366
Figure 12-459. EQEP Unit Time Base.....	2367
Figure 12-460. EQEP Interrupt Generation.....	2367
Figure 12-461. MCAN Modules Overview.....	2370
Figure 12-462. MCAN Typical Application.....	2372
Figure 12-463. MCU_MCAN0 Integration.....	2375
Figure 12-464. MCU_MCAN1 Integration.....	2376
Figure 12-465. MCAN[0-13] Integration.....	2379
Figure 12-466. MCAN Block Diagram.....	2400
Figure 12-467. CAN FD Frame.....	2404
Figure 12-468. CAN Bit Timing.....	2405
Figure 12-469. Transmitter Delay Measurement.....	2406
Figure 12-470. Connection of Signals in Bus Monitoring Mode.....	2407
Figure 12-471. Internal Loopback Mode.....	2410
Figure 12-472. External Timestamp Counter Interrupt.....	2411
Figure 12-473. Standard Message ID Filter Path.....	2415
Figure 12-474. Extended Message ID Filter Path.....	2416
Figure 12-475. Rx FIFO Status.....	2417
Figure 12-476. Rx FIFO Overflow Handling.....	2418
Figure 12-477. Mixed Dedicated Tx Buffers/Tx FIFO (example).....	2422
Figure 12-478. Mixed Dedicated Tx Buffers/Tx Queue (example).....	2422
Figure 12-479. Message RAM Configuration.....	2425
Figure 12-480. Rx Buffer/Rx FIFO Element Structure.....	2425
Figure 12-481. Tx Buffer Element Structure.....	2428
Figure 12-482. Tx Event FIFO Element Structure.....	2430
Figure 12-483. Standard Message ID Filter Element Structure.....	2431
Figure 12-484. Extended Message ID Filter Element Structure.....	2433
Figure 12-485. ATL Module Overview.....	2436
Figure 12-486. MCASP Modules Overview.....	2438
Figure 12-487. MCASP Interface Signals.....	2440
Figure 12-488. Definition of Bit, Word, and Slot.....	2443
Figure 12-489. Bit Order and Word Alignment Within a Slot Examples.....	2444
Figure 12-490. Definition of Frame and Frame-Sync Width.....	2445
Figure 12-491. TDM Format - 6 channel example.....	2446
Figure 12-492. I2S Format Overview.....	2447
Figure 12-493. Biphase-Mark Code.....	2448
Figure 12-494. S/PDIF Subframe Format.....	2449
Figure 12-495. S/PDIF Frame Format.....	2449
Figure 12-496. MCASP Integration.....	2450
Figure 12-497. MCASP Module Block Diagram.....	2452
Figure 12-498. Transmit Clock Generator Block Diagram.....	2454
Figure 12-499. Receive Clock Generator Block Diagram.....	2456
Figure 12-500. Frame Sync Generator Block Diagram.....	2457
Figure 12-501. Individual Serializer and Connections Within MCASP.....	2459
Figure 12-502. Transmit Format Unit.....	2461
Figure 12-503. Receive Format Unit.....	2464
Figure 12-504. Burst Frame Sync Mode.....	2467
Figure 12-505. Transmit DMA Event (XINT) Generation in TDM Time Slots.....	2469
Figure 12-506. Service Time Upon Transmit DMA Event (XINT).....	2474
Figure 12-507. CPU Service Time Upon Receive Event (RINT).....	2475
Figure 12-508. DMA Transmit and Receive Event in an Audio Example – One Event.....	2478



Figure 12-509. MCASP Audio FIFO (AFIFO) Block Diagram.....	2479
Figure 12-510. MCASP Serializers Operation in Loopback Mode.....	2484
Figure 12-511. Transmit Clock Failure Detection Circuit Block Diagram.....	2488
Figure 12-512. Receive Clock Failure Detection Circuit Block Diagram.....	2489
Figure 12-513. MCASP DIT- /TDM- Transmission Polling Method.....	2500
Figure 12-514. Subsequence – DIT-/TDM- Transmission Startup Procedure.....	2502
Figure 12-515. MCASP Polling Reception Method.....	2504
Figure 12-516. Subsequence – TDM - Reception Startup Procedure.....	2506
Figure 12-517. MCASP Transmit Interrupt Events Servicing.....	2508
Figure 12-518. MCASP Receive Interrupt Events Servicing.....	2509
Figure 12-519. MCASP Transmit Error Handling.....	2510
Figure 12-520. MCASP Receive Error Handling.....	2511
Figure 12-521. DSS and Display Peripherals Overview.....	2514
Figure 12-522. DSS Pixel Mapping on the DISPC Video Port Outputs.....	2520
Figure 12-523. DSS DPI Parallel Interface Signals.....	2521
Figure 12-524. DISPC Video Port Pixel Data - 12-bit RGB Active Matrix.....	2523
Figure 12-525. DISPC Video Port Pixel Data - 16-bit RGB Active Matrix.....	2524
Figure 12-526. DISPC Video Port Pixel Data - 18-bit RGB Active Matrix.....	2524
Figure 12-527. DISPC Video Port Pixel Data - 24-bit RGB Active Matrix.....	2525
Figure 12-528. DISPC Video Port Data Mapping for BT.656 Mode.....	2525
Figure 12-529. DISPC Video Port Data Mapping for BT.1120 Mode.....	2525
Figure 12-530. DISPC Display Timing Diagram of Configuration 1 (Start of Frame).....	2526
Figure 12-531. DISPC Display Timing Diagram of Configuration 1 (Between Lines).....	2527
Figure 12-532. DISPC Display Timing Diagram of Configuration 1 (Between Frames).....	2527
Figure 12-533. DISPC Display Timing Diagram of Configuration 1 (End of Frame).....	2527
Figure 12-534. DISPC Display Timing Diagram of Configuration 2 (Start of Frame).....	2528
Figure 12-535. DISPC Display Timing Diagram of Configuration 2 (Between Lines).....	2528
Figure 12-536. DISPC Display Timing Diagram of Configuration 2 (Between Frames).....	2528
Figure 12-537. DISPC Display Timing Diagram of Configuration 2 (End of Frame).....	2528
Figure 12-538. DISPC Display Timing Diagram of Configuration 3 (Start of Frame).....	2529
Figure 12-539. DISPC Display Timing Diagram of Configuration 3 (Between Lines).....	2529
Figure 12-540. DISPC Display Timing Diagram of Configuration 3 (Between Frames).....	2529
Figure 12-541. DISPC Display Timing Diagram of Configuration 3 (End of Frame).....	2529
Figure 12-542. DSI Interface Signals.....	2531
Figure 12-543. DP/eDP Interface Signals.....	2532
Figure 12-544. DISPC Integration.....	2533
Figure 12-545. DSI Integration.....	2539
Figure 12-546. EDP Integration.....	2541
Figure 12-547. DISPC Architecture Overview.....	2545
Figure 12-548. DISPC Interrupts Generation.....	2548
Figure 12-549. DISPC YUV422 Predecimation.....	2563
Figure 12-550. DISPC DMA Channel Memory Allocation at Reset.....	2564
Figure 12-551. DISPC Dynamic Buffer Repartition between DMA Read Channels.....	2564
Figure 12-552. DISPC DMA High/Low Priority Arbitration.....	2567
Figure 12-553. DISPC FBDC Tile Request (16x4 Tile).....	2568
Figure 12-554. DISPC FBDC Example of a Frame Buffer Source Cropping (Decompression).....	2570
Figure 12-555. DISPC Bitmap Pixel Formats.....	2573
Figure 12-556. DISPC RGB 16-bit Pixel Formats 1.....	2574
Figure 12-557. DISPC RGB 16-bit Pixel Formats 2.....	2575
Figure 12-558. DISPC RGB 24-bit Pixel Formats.....	2575
Figure 12-559. DISPC RGB 32-bit Pixel Formats 1.....	2576
Figure 12-560. DISPC RGB 32-bit Pixel Formats 2.....	2577
Figure 12-561. DISPC RGB 64-bit Pixel Formats.....	2578
Figure 12-562. DISPC YUV 8-bit Pixel Formats 1.....	2578
Figure 12-563. DISPC YUV 8-bit Pixel Formats 2.....	2579
Figure 12-564. DISPC YUV 10-bit Pixel Packed Formats.....	2580
Figure 12-565. DISPC YUV 12-bit Packed Pixel Formats 1.....	2581
Figure 12-566. DISPC YUV 12-bit Packed Pixel Formats 2.....	2582
Figure 12-567. DISPC YUV 10-bit Unpacked Pixel Formats 1.....	2583
Figure 12-568. DISPC YUV 12-bit Unpacked Pixel Formats 2.....	2584
Figure 12-569. DISPC YUV 10-bit/12-bit Unpacked Pixel Formats 3.....	2585

Figure 12-570. DISPC Video Pipeline Configuration.....	2586
Figure 12-571. DISPC Video Lite Pipeline Configuration.....	2586
Figure 12-572. DISPC VID CLUT/Gamma Data Memory Organization.....	2588
Figure 12-573. DISPC VID YUV420 to ARGB48 Using Scaler Unit for Resampling Chrominance.....	2589
Figure 12-574. DISPC VID YUV422 to ARGB48 Using Scaler Unit for Resampling Chrominance.....	2589
Figure 12-575. DISPC Video Window Attributes.....	2590
Figure 12-576. DISPC VID Macro-Architecture of the Horizontal Scaling for A, R, G, and B Components (5-tap Restriction).....	2593
Figure 12-577. DISPC VID Macro-Architecture of the Vertical Scaling for A, R, G, and B Components (5 and 3 taps).....	2593
Figure 12-578. DISPC VID Macro-Architecture of the Horizontal Scaling for Y, Cr, and Cb Components (5-tap Restriction).....	2593
Figure 12-579. DISPC VID Macro-Architecture of the Vertical Scaling for Y, Cr, and Cb Components (5 and 3 taps).....	2594
Figure 12-580. DISPC VID CSC YCbCr to RGB Equation (Limited Data Range), 12-Bit Outputs.....	2598
Figure 12-581. DISPC VID CSC YCbCr to RGB Equation (Full Data Range), 12-Bit Outputs.....	2598
Figure 12-582. DISPC VID Brightness/Contrast/Saturation Equation for YUV to RGB.....	2599
Figure 12-583. DISPC VID Brightness/Contrast/Saturation/Hue Equation for YUV to RGB.....	2599
Figure 12-584. DISPC VID Brightness/Contrast/Saturation/Hue Equation for RGB Input.....	2600
Figure 12-585. DISPC Write-Back Pipeline.....	2601
Figure 12-586. DISPC WB CSC RGB to YCbCr Equation (Limited Data Range), 12-bit Outputs.....	2602
Figure 12-587. DISPC WB CSC RGB to YCbCr Equation (Full Data Range), 12-bit Outputs.....	2602
Figure 12-588. DISPC Overlay Manager and Input Selector.....	2607
Figure 12-589. DISPC Overlay Example and Display Attributes.....	2609
Figure 12-590. DISPC Overlay Source Transparency Color Key Example.....	2612
Figure 12-591. DISPC Overlay Destination Transparency Color Key Example.....	2613
Figure 12-592. DISPC Overlay Internal Color Bar.....	2613
Figure 12-593. DISPC VP Output Architecture.....	2614
Figure 12-594. DISPC VP Output Gamma Table Write Data Format.....	2615
Figure 12-595. DISPC VP CPR Matrix.....	2615
Figure 12-596. DISPC VP CSC RGB to YUV Equation (FULLRANGE=0).....	2616
Figure 12-597. DISPC VP CSC RGB to YUV Equation (FULLRANGE=1).....	2616
Figure 12-598. DISPC BT Mode Bit-Assignment for the Fourth Byte of EAV/SAV Codes.....	2618
Figure 12-599. DISPC VP TDM 8-Bit Interface Settings.....	2620
Figure 12-600. DISPC VP TDM 9-Bit Interface Settings.....	2621
Figure 12-601. DISPC VP TDM 12-Bit Interface Settings.....	2622
Figure 12-602. DISPC VP TDM 16-Bit Interface Settings.....	2623
Figure 12-603. DISPC VP Display Timing Parameters.....	2624
Figure 12-604. DISPC VP Merge-Split Scheme.....	2626
Figure 12-605. DISPC VP Clocking Scheme for MSS BYPASS and SYNC Modes.....	2627
Figure 12-606. DISPC VP Clocking Scheme for MSS MERGE and SPLIT Modes.....	2628
Figure 12-607. DISPC Internal Diagnostic Check Regions.....	2629
Figure 12-608. DISPC Internal Diagnostic Check Region Examples.....	2631
Figure 12-609. DISPC Internal Diagnostic Check Locations.....	2632
Figure 12-610. DISPC Internal Diagnostic 32-bit MISR Implementation.....	2633
Figure 12-611. DISPC Secure Bit Setting and Illegal Connection Block.....	2636
Figure 12-612. DSS0_COMMON_DSS_REVISION Register.....	2639
Figure 12-613. DSS0_COMMON_DSS_SYSCONFIG Register.....	2640
Figure 12-614. DSS0_COMMON_DSS_SYSSTATUS Register.....	2642
Figure 12-615. DSS0_COMMON_DISPC_IRQSTATUS_RAW Register.....	2644
Figure 12-616. DSS0_COMMON_DISPC_IRQSTATUS Register.....	2646
Figure 12-617. DSS0_COMMON_DISPC_IRQENABLE_SET Register.....	2648
Figure 12-618. DSS0_COMMON_DISPC_IRQENABLE_CLR Register.....	2650
Figure 12-619. DSS0_COMMON_VID_IRQENABLE_0 Register.....	2652
Figure 12-620. DSS0_COMMON_VID_IRQENABLE_1 Register.....	2654
Figure 12-621. DSS0_COMMON_VID_IRQENABLE_2 Register.....	2656
Figure 12-622. DSS0_COMMON_VID_IRQENABLE_3 Register.....	2658
Figure 12-623. DSS0_COMMON_VID_IRQSTATUS_0 Register.....	2660
Figure 12-624. DSS0_COMMON_VID_IRQSTATUS_1 Register.....	2662
Figure 12-625. DSS0_COMMON_VID_IRQSTATUS_2 Register.....	2664
Figure 12-626. DSS0_COMMON_VID_IRQSTATUS_3 Register.....	2666
Figure 12-627. DSS0_COMMON_VP_IRQENABLE_0 Register.....	2668
Figure 12-628. DSS0_COMMON_VP_IRQENABLE_1 Register.....	2670
Figure 12-629. DSS0_COMMON_VP_IRQENABLE_2 Register.....	2672

Figure 12-630. DSS0_COMMON_VP_IRQENABLE_3 Register.....	2674
Figure 12-631. DSS0_COMMON_VP_IRQSTATUS_0 Register.....	2676
Figure 12-632. DSS0_COMMON_VP_IRQSTATUS_1 Register.....	2678
Figure 12-633. DSS0_COMMON_VP_IRQSTATUS_2 Register.....	2680
Figure 12-634. DSS0_COMMON_VP_IRQSTATUS_3 Register.....	2682
Figure 12-635. DSS0_COMMON_WB_IRQENABLE Register.....	2684
Figure 12-636. DSS0_COMMON_WB_IRQSTATUS Register.....	2686
Figure 12-637. DSS0_COMMON_DISPC_IRQ_EOI_FUNC Register.....	2688
Figure 12-638. DSS0_COMMON_DISPC_IRQ_EOI_SAFETY Register.....	2689
Figure 12-639. DSS0_COMMON_DISPC_IRQ_EOI_SECURITY Register.....	2690
Figure 12-640. DSS0_COMMON_DISPC_SECURE_DISABLE Register.....	2691
Figure 12-641. DSS0_COMMON_DISPC_GLOBAL_MFLAG_ATTRIBUTE Register.....	2692
Figure 12-642. DSS0_COMMON_DISPC_GLOBAL_OUTPUT_ENABLE Register.....	2694
Figure 12-643. DSS0_COMMON_DISPC_GLOBAL_BUFFER Register.....	2695
Figure 12-644. DSS0_COMMON_DSS_CBA_CFG Register.....	2697
Figure 12-645. DSS0_COMMON_DISPC_DBG_CONTROL Register.....	2698
Figure 12-646. DSS0_COMMON_DISPC_DBG_STATUS Register.....	2700
Figure 12-647. DSS0_COMMON_DISPC_CLKGATING_DISABLE Register.....	2701
Figure 12-648. DSS0_COMMON_FBDC_REVISION_1 Register.....	2703
Figure 12-649. DSS0_COMMON_FBDC_REVISION_2 Register.....	2704
Figure 12-650. DSS0_COMMON_FBDC_REVISION_3 Register.....	2705
Figure 12-651. DSS0_COMMON_FBDC_REVISION_4 Register.....	2706
Figure 12-652. DSS0_COMMON_FBDC_REVISION_5 Register.....	2707
Figure 12-653. DSS0_COMMON_FBDC_REVISION_6 Register.....	2708
Figure 12-654. DSS0_COMMON_FBDC_COMMON_CONTROL Register.....	2709
Figure 12-655. DSS0_COMMON_FBDC_CONSTANT_COLOR_0 Register.....	2710
Figure 12-656. DSS0_COMMON_FBDC_CONSTANT_COLOR_1 Register.....	2711
Figure 12-657. DSS0_COMMON_DISPC_CONNECTIONS Register.....	2712
Figure 12-658. DSS0_COMMON_DISPC_MSS_VP1 Register.....	2714
Figure 12-659. DSS0_COMMON_DISPC_MSS_VP3 Register.....	2715
Figure 12-660. DSS0_COMMON_GLOBAL_DMA_THREADSIZE Register.....	2716
Figure 12-661. DSS0_COMMON_GLOBAL_DMA_THREADSIZESTATUS Register.....	2717
Figure 12-662. DSS0_COMMON_GLOBAL_GOBITMODE Register.....	2718
Figure 12-663. DSS0_VID_ACCUH_0 Register.....	2724
Figure 12-664. DSS0_VID_ACCUH_1 Register.....	2725
Figure 12-665. DSS0_VID_ACCUH2_0 Register.....	2726
Figure 12-666. DSS0_VID_ACCUH2_1 Register.....	2727
Figure 12-667. DSS0_VID_ACCUV_0 Register.....	2728
Figure 12-668. DSS0_VID_ACCUV_1 Register.....	2729
Figure 12-669. DSS0_VID_ACCUV2_0 Register.....	2730
Figure 12-670. DSS0_VID_ACCUV2_1 Register.....	2731
Figure 12-671. DSS0_VID_ATTRIBUTES Register.....	2732
Figure 12-672. DSS0_VID_ATTRIBUTES2 Register.....	2737
Figure 12-673. DSS0_VID_BA_0 Register.....	2739
Figure 12-674. DSS0_VID_BA_1 Register.....	2740
Figure 12-675. DSS0_VID_BA_UV_0 Register.....	2741
Figure 12-676. DSS0_VID_BA_UV_1 Register.....	2742
Figure 12-677. DSS0_VID_BUF_SIZE_STATUS Register.....	2743
Figure 12-678. DSS0_VID_BUF_THRESHOLD Register.....	2744
Figure 12-679. DSS0_VID_CSC_COEF0 Register.....	2745
Figure 12-680. DSS0_VID_CSC_COEF1 Register.....	2746
Figure 12-681. DSS0_VID_CSC_COEF2 Register.....	2747
Figure 12-682. DSS0_VID_CSC_COEF3 Register.....	2748
Figure 12-683. DSS0_VID_CSC_COEF4 Register.....	2749
Figure 12-684. DSS0_VID_CSC_COEF5 Register.....	2750
Figure 12-685. DSS0_VID_CSC_COEF6 Register.....	2751
Figure 12-686. DSS0_VID_FIRH Register.....	2752
Figure 12-687. DSS0_VID_FIRH2 Register.....	2753
Figure 12-688. DSS0_VID_FIRV Register.....	2754
Figure 12-689. DSS0_VID_FIRV2 Register.....	2755
Figure 12-690. DSS0_VID_FIR_COEF_H0_0 Register.....	2756

Figure 12-691. DSS0_VID_FIR_COEF_H0_1 Register.....	2757
Figure 12-692. DSS0_VID_FIR_COEF_H0_2 Register.....	2758
Figure 12-693. DSS0_VID_FIR_COEF_H0_3 Register.....	2759
Figure 12-694. DSS0_VID_FIR_COEF_H0_4 Register.....	2760
Figure 12-695. DSS0_VID_FIR_COEF_H0_5 Register.....	2761
Figure 12-696. DSS0_VID_FIR_COEF_H0_6 Register.....	2762
Figure 12-697. DSS0_VID_FIR_COEF_H0_7 Register.....	2763
Figure 12-698. DSS0_VID_FIR_COEF_H0_8 Register.....	2764
Figure 12-699. DSS0_VID_FIR_COEF_H0_C_0 Register.....	2765
Figure 12-700. DSS0_VID_FIR_COEF_H0_C_1 Register.....	2766
Figure 12-701. DSS0_VID_FIR_COEF_H0_C_2 Register.....	2767
Figure 12-702. DSS0_VID_FIR_COEF_H0_C_3 Register.....	2768
Figure 12-703. DSS0_VID_FIR_COEF_H0_C_4 Register.....	2769
Figure 12-704. DSS0_VID_FIR_COEF_H0_C_5 Register.....	2770
Figure 12-705. DSS0_VID_FIR_COEF_H0_C_6 Register.....	2771
Figure 12-706. DSS0_VID_FIR_COEF_H0_C_7 Register.....	2772
Figure 12-707. DSS0_VID_FIR_COEF_H0_C_8 Register.....	2773
Figure 12-708. DSS0_VID_FIR_COEF_H12_0 Register.....	2774
Figure 12-709. DSS0_VID_FIR_COEF_H12_1 Register.....	2775
Figure 12-710. DSS0_VID_FIR_COEF_H12_2 Register.....	2776
Figure 12-711. DSS0_VID_FIR_COEF_H12_3 Register.....	2777
Figure 12-712. DSS0_VID_FIR_COEF_H12_4 Register.....	2778
Figure 12-713. DSS0_VID_FIR_COEF_H12_5 Register.....	2779
Figure 12-714. DSS0_VID_FIR_COEF_H12_6 Register.....	2780
Figure 12-715. DSS0_VID_FIR_COEF_H12_7 Register.....	2781
Figure 12-716. DSS0_VID_FIR_COEF_H12_8 Register.....	2782
Figure 12-717. DSS0_VID_FIR_COEF_H12_9 Register.....	2783
Figure 12-718. DSS0_VID_FIR_COEF_H12_10 Register.....	2784
Figure 12-719. DSS0_VID_FIR_COEF_H12_11 Register.....	2785
Figure 12-720. DSS0_VID_FIR_COEF_H12_12 Register.....	2786
Figure 12-721. DSS0_VID_FIR_COEF_H12_13 Register.....	2787
Figure 12-722. DSS0_VID_FIR_COEF_H12_14 Register.....	2788
Figure 12-723. DSS0_VID_FIR_COEF_H12_15 Register.....	2789
Figure 12-724. DSS0_VID_FIR_COEF_H12_C_0 Register.....	2790
Figure 12-725. DSS0_VID_FIR_COEF_H12_C_1 Register.....	2791
Figure 12-726. DSS0_VID_FIR_COEF_H12_C_2 Register.....	2792
Figure 12-727. DSS0_VID_FIR_COEF_H12_C_3 Register.....	2793
Figure 12-728. DSS0_VID_FIR_COEF_H12_C_4 Register.....	2794
Figure 12-729. DSS0_VID_FIR_COEF_H12_C_5 Register.....	2795
Figure 12-730. DSS0_VID_FIR_COEF_H12_C_6 Register.....	2796
Figure 12-731. DSS0_VID_FIR_COEF_H12_C_7 Register.....	2797
Figure 12-732. DSS0_VID_FIR_COEF_H12_C_8 Register.....	2798
Figure 12-733. DSS0_VID_FIR_COEF_H12_C_9 Register.....	2799
Figure 12-734. DSS0_VID_FIR_COEF_H12_C_10 Register.....	2800
Figure 12-735. DSS0_VID_FIR_COEF_H12_C_11 Register.....	2801
Figure 12-736. DSS0_VID_FIR_COEF_H12_C_12 Register.....	2802
Figure 12-737. DSS0_VID_FIR_COEF_H12_C_13 Register.....	2803
Figure 12-738. DSS0_VID_FIR_COEF_H12_C_14 Register.....	2804
Figure 12-739. DSS0_VID_FIR_COEF_H12_C_15 Register.....	2805
Figure 12-740. DSS0_VID_FIR_COEF_V0_0 Register.....	2806
Figure 12-741. DSS0_VID_FIR_COEF_V0_1 Register.....	2807
Figure 12-742. DSS0_VID_FIR_COEF_V0_2 Register.....	2808
Figure 12-743. DSS0_VID_FIR_COEF_V0_3 Register.....	2809
Figure 12-744. DSS0_VID_FIR_COEF_V0_4 Register.....	2810
Figure 12-745. DSS0_VID_FIR_COEF_V0_5 Register.....	2811
Figure 12-746. DSS0_VID_FIR_COEF_V0_6 Register.....	2812
Figure 12-747. DSS0_VID_FIR_COEF_V0_7 Register.....	2813
Figure 12-748. DSS0_VID_FIR_COEF_V0_8 Register.....	2814
Figure 12-749. DSS0_VID_FIR_COEF_V0_C_0 Register.....	2815
Figure 12-750. DSS0_VID_FIR_COEF_V0_C_1 Register.....	2816
Figure 12-751. DSS0_VID_FIR_COEF_V0_C_2 Register.....	2817



Figure 12-752. DSS0_VID_FIR_COEF_V0_C_3 Register.....	2818
Figure 12-753. DSS0_VID_FIR_COEF_V0_C_4 Register.....	2819
Figure 12-754. DSS0_VID_FIR_COEF_V0_C_5 Register.....	2820
Figure 12-755. DSS0_VID_FIR_COEF_V0_C_6 Register.....	2821
Figure 12-756. DSS0_VID_FIR_COEF_V0_C_7 Register.....	2822
Figure 12-757. DSS0_VID_FIR_COEF_V0_C_8 Register.....	2823
Figure 12-758. DSS0_VID_FIR_COEF_V12_0 Register.....	2824
Figure 12-759. DSS0_VID_FIR_COEF_V12_1 Register.....	2825
Figure 12-760. DSS0_VID_FIR_COEF_V12_2 Register.....	2826
Figure 12-761. DSS0_VID_FIR_COEF_V12_3 Register.....	2827
Figure 12-762. DSS0_VID_FIR_COEF_V12_4 Register.....	2828
Figure 12-763. DSS0_VID_FIR_COEF_V12_5 Register.....	2829
Figure 12-764. DSS0_VID_FIR_COEF_V12_6 Register.....	2830
Figure 12-765. DSS0_VID_FIR_COEF_V12_7 Register.....	2831
Figure 12-766. DSS0_VID_FIR_COEF_V12_8 Register.....	2832
Figure 12-767. DSS0_VID_FIR_COEF_V12_9 Register.....	2833
Figure 12-768. DSS0_VID_FIR_COEF_V12_10 Register.....	2834
Figure 12-769. DSS0_VID_FIR_COEF_V12_11 Register.....	2835
Figure 12-770. DSS0_VID_FIR_COEF_V12_12 Register.....	2836
Figure 12-771. DSS0_VID_FIR_COEF_V12_13 Register.....	2837
Figure 12-772. DSS0_VID_FIR_COEF_V12_14 Register.....	2838
Figure 12-773. DSS0_VID_FIR_COEF_V12_15 Register.....	2839
Figure 12-774. DSS0_VID_FIR_COEF_V12_C_0 Register.....	2840
Figure 12-775. DSS0_VID_FIR_COEF_V12_C_1 Register.....	2841
Figure 12-776. DSS0_VID_FIR_COEF_V12_C_2 Register.....	2842
Figure 12-777. DSS0_VID_FIR_COEF_V12_C_3 Register.....	2843
Figure 12-778. DSS0_VID_FIR_COEF_V12_C_4 Register.....	2844
Figure 12-779. DSS0_VID_FIR_COEF_V12_C_5 Register.....	2845
Figure 12-780. DSS0_VID_FIR_COEF_V12_C_6 Register.....	2846
Figure 12-781. DSS0_VID_FIR_COEF_V12_C_7 Register.....	2847
Figure 12-782. DSS0_VID_FIR_COEF_V12_C_8 Register.....	2848
Figure 12-783. DSS0_VID_FIR_COEF_V12_C_9 Register.....	2849
Figure 12-784. DSS0_VID_FIR_COEF_V12_C_10 Register.....	2850
Figure 12-785. DSS0_VID_FIR_COEF_V12_C_11 Register.....	2851
Figure 12-786. DSS0_VID_FIR_COEF_V12_C_12 Register.....	2852
Figure 12-787. DSS0_VID_FIR_COEF_V12_C_13 Register.....	2853
Figure 12-788. DSS0_VID_FIR_COEF_V12_C_14 Register.....	2854
Figure 12-789. DSS0_VID_FIR_COEF_V12_C_15 Register.....	2855
Figure 12-790. DSS0_VID_GLOBAL_ALPHA Register.....	2856
Figure 12-791. DSS0_VID_MFLAG_THRESHOLD Register.....	2857
Figure 12-792. DSS0_VID_PICTURE_SIZE Register.....	2858
Figure 12-793. DSS0_VID_PIXEL_INC Register.....	2860
Figure 12-794. DSS0_VID_PRELOAD Register.....	2861
Figure 12-795. DSS0_VID_ROW_INC Register.....	2862
Figure 12-796. DSS0_VID_SIZE Register.....	2863
Figure 12-797. DSS0_VID_BA_EXT_0 Register.....	2864
Figure 12-798. DSS0_VID_BA_EXT_1 Register.....	2865
Figure 12-799. DSS0_VID_BA_UV_EXT_0 Register.....	2866
Figure 12-800. DSS0_VID_BA_UV_EXT_1 Register.....	2867
Figure 12-801. DSS0_VID_CSC_COEF7 Register.....	2868
Figure 12-802. DSS0_VID_ROW_INC_UV Register.....	2869
Figure 12-803. DSS0_VID_TILE Register.....	2870
Figure 12-804. DSS0_VID_TILE2 Register.....	2871
Figure 12-805. DSS0_VID_FBDC_ATTRIBUTES Register.....	2872
Figure 12-806. DSS0_VID_FBDC_CLEAR_COLOR Register.....	2873
Figure 12-807. DSS0_VID_CLUT_0 Register.....	2874
Figure 12-808. DSS0_VID_CLUT_1 Register.....	2875
Figure 12-809. DSS0_VID_CLUT_2 Register.....	2876
Figure 12-810. DSS0_VID_CLUT_3 Register.....	2877
Figure 12-811. DSS0_VID_CLUT_4 Register.....	2878
Figure 12-812. DSS0_VID_CLUT_5 Register.....	2879

Figure 12-813. DSS0_VID_CLUT_6 Register.....	2880
Figure 12-814. DSS0_VID_CLUT_7 Register.....	2881
Figure 12-815. DSS0_VID_CLUT_8 Register.....	2882
Figure 12-816. DSS0_VID_CLUT_9 Register.....	2883
Figure 12-817. DSS0_VID_CLUT_10 Register.....	2884
Figure 12-818. DSS0_VID_CLUT_11 Register.....	2885
Figure 12-819. DSS0_VID_CLUT_12 Register.....	2886
Figure 12-820. DSS0_VID_CLUT_13 Register.....	2887
Figure 12-821. DSS0_VID_CLUT_14 Register.....	2888
Figure 12-822. DSS0_VID_CLUT_15 Register.....	2889
Figure 12-823. DSS0_VID_SAFETY_ATTRIBUTES Register.....	2890
Figure 12-824. DSS0_VID_SAFETY_CAPT_SIGNATURE Register.....	2892
Figure 12-825. DSS0_VID_SAFETY_POSITION Register.....	2893
Figure 12-826. DSS0_VID_SAFETY_REF_SIGNATURE Register.....	2894
Figure 12-827. DSS0_VID_SAFETY_SIZE Register.....	2895
Figure 12-828. DSS0_VID_SAFETY_LFSR_SEED Register.....	2896
Figure 12-829. DSS0_VID_LUMAKEY Register.....	2897
Figure 12-830. DSS0_VID_DMA_BUFSIZE Register.....	2898
Figure 12-831. DSS0_VID_CROP Register.....	2899
Figure 12-832. DSS0_VID_SECURE Register.....	2900
Figure 12-833. DSS0_VID_PIPE_GO Register.....	2901
Figure 12-834. DSS0_OVR_CONFIG Register.....	2903
Figure 12-835. DSS0_OVR_VIRTUALVP Register.....	2905
Figure 12-836. DSS0_OVR_DEFAULT_COLOR Register.....	2906
Figure 12-837. DSS0_OVR_DEFAULT_COLOR2 Register.....	2907
Figure 12-838. DSS0_OVR_TRANS_COLOR_MAX Register.....	2908
Figure 12-839. DSS0_OVR_TRANS_COLOR_MAX2 Register.....	2909
Figure 12-840. DSS0_OVR_TRANS_COLOR_MIN Register.....	2910
Figure 12-841. DSS0_OVR_TRANS_COLOR_MIN2 Register.....	2911
Figure 12-842. DSS0_OVR_ATTRIBUTES_0 Register.....	2912
Figure 12-843. DSS0_OVR_ATTRIBUTES_1 Register.....	2913
Figure 12-844. DSS0_OVR_ATTRIBUTES_2 Register.....	2914
Figure 12-845. DSS0_OVR_ATTRIBUTES_3 Register.....	2915
Figure 12-846. DSS0_OVR_ATTRIBUTES_4 Register.....	2916
Figure 12-847. DSS0_OVR_ATTRIBUTES2_0 Register.....	2917
Figure 12-848. DSS0_OVR_ATTRIBUTES2_1 Register.....	2918
Figure 12-849. DSS0_OVR_ATTRIBUTES2_2 Register.....	2919
Figure 12-850. DSS0_OVR_ATTRIBUTES2_3 Register.....	2920
Figure 12-851. DSS0_OVR_ATTRIBUTES2_4 Register.....	2921
Figure 12-852. DSS0_OVR_SECURE Register.....	2922
Figure 12-853. DSS0_VP_CONFIG Register.....	2926
Figure 12-854. DSS0_VP_CONTROL Register.....	2929
Figure 12-855. DSS0_VP_CSC_COEF0 Register.....	2932
Figure 12-856. DSS0_VP_CSC_COEF1 Register.....	2933
Figure 12-857. DSS0_VP_CSC_COEF2 Register.....	2934
Figure 12-858. DSS0_VP_DATA_CYCLE_0 Register.....	2935
Figure 12-859. DSS0_VP_DATA_CYCLE_1 Register.....	2936
Figure 12-860. DSS0_VP_DATA_CYCLE_2 Register.....	2937
Figure 12-861. DSS0_VP_LINE_NUMBER Register.....	2938
Figure 12-862. DSS0_VP_POL_FREQ Register.....	2939
Figure 12-863. DSS0_VP_SIZE_SCREEN Register.....	2941
Figure 12-864. DSS0_VP_TIMING_H Register.....	2942
Figure 12-865. DSS0_VP_TIMING_V Register.....	2943
Figure 12-866. DSS0_VP_CSC_COEF3 Register.....	2944
Figure 12-867. DSS0_VP_CSC_COEF4 Register.....	2945
Figure 12-868. DSS0_VP_CSC_COEF5 Register.....	2946
Figure 12-869. DSS0_VP_CSC_COEF6 Register.....	2947
Figure 12-870. DSS0_VP_CSC_COEF7 Register.....	2948
Figure 12-871. DSS0_VP_SAFETY_ATTRIBUTES_0 Register.....	2949
Figure 12-872. DSS0_VP_SAFETY_ATTRIBUTES_1 Register.....	2951
Figure 12-873. DSS0_VP_SAFETY_ATTRIBUTES_2 Register.....	2953

Figure 12-874. DSS0_VP_SAFETY_ATTRIBUTES_3 Register.....	2955
Figure 12-875. DSS0_VP_SAFETY_ATTRIBUTES_4 Register.....	2957
Figure 12-876. DSS0_VP_SAFETY_ATTRIBUTES_5 Register.....	2959
Figure 12-877. DSS0_VP_SAFETY_ATTRIBUTES_6 Register.....	2961
Figure 12-878. DSS0_VP_SAFETY_ATTRIBUTES_7 Register.....	2963
Figure 12-879. DSS0_VP_SAFETY_CAPT_SIGNATURE_0 Register.....	2965
Figure 12-880. DSS0_VP_SAFETY_CAPT_SIGNATURE_1 Register.....	2966
Figure 12-881. DSS0_VP_SAFETY_CAPT_SIGNATURE_2 Register.....	2967
Figure 12-882. DSS0_VP_SAFETY_CAPT_SIGNATURE_3 Register.....	2968
Figure 12-883. DSS0_VP_SAFETY_CAPT_SIGNATURE_4 Register.....	2969
Figure 12-884. DSS0_VP_SAFETY_CAPT_SIGNATURE_5 Register.....	2970
Figure 12-885. DSS0_VP_SAFETY_CAPT_SIGNATURE_6 Register.....	2971
Figure 12-886. DSS0_VP_SAFETY_CAPT_SIGNATURE_7 Register.....	2972
Figure 12-887. DSS0_VP_SAFETY_POSITION_0 Register.....	2973
Figure 12-888. DSS0_VP_SAFETY_POSITION_1 Register.....	2974
Figure 12-889. DSS0_VP_SAFETY_POSITION_2 Register.....	2975
Figure 12-890. DSS0_VP_SAFETY_POSITION_3 Register.....	2976
Figure 12-891. DSS0_VP_SAFETY_POSITION_4 Register.....	2977
Figure 12-892. DSS0_VP_SAFETY_POSITION_5 Register.....	2978
Figure 12-893. DSS0_VP_SAFETY_POSITION_6 Register.....	2979
Figure 12-894. DSS0_VP_SAFETY_POSITION_7 Register.....	2980
Figure 12-895. DSS0_VP_SAFETY_REF_SIGNATURE_0 Register.....	2981
Figure 12-896. DSS0_VP_SAFETY_REF_SIGNATURE_1 Register.....	2982
Figure 12-897. DSS0_VP_SAFETY_REF_SIGNATURE_2 Register.....	2983
Figure 12-898. DSS0_VP_SAFETY_REF_SIGNATURE_3 Register.....	2984
Figure 12-899. DSS0_VP_SAFETY_REF_SIGNATURE_4 Register.....	2985
Figure 12-900. DSS0_VP_SAFETY_REF_SIGNATURE_5 Register.....	2986
Figure 12-901. DSS0_VP_SAFETY_REF_SIGNATURE_6 Register.....	2987
Figure 12-902. DSS0_VP_SAFETY_REF_SIGNATURE_7 Register.....	2988
Figure 12-903. DSS0_VP_SAFETY_SIZE_0 Register.....	2989
Figure 12-904. DSS0_VP_SAFETY_SIZE_1 Register.....	2990
Figure 12-905. DSS0_VP_SAFETY_SIZE_2 Register.....	2991
Figure 12-906. DSS0_VP_SAFETY_SIZE_3 Register.....	2992
Figure 12-907. DSS0_VP_SAFETY_SIZE_4 Register.....	2993
Figure 12-908. DSS0_VP_SAFETY_SIZE_5 Register.....	2994
Figure 12-909. DSS0_VP_SAFETY_SIZE_6 Register.....	2995
Figure 12-910. DSS0_VP_SAFETY_SIZE_7 Register.....	2996
Figure 12-911. DSS0_VP_SAFETY_LFSR_SEED Register.....	2997
Figure 12-912. DSS0_VP_GAMMA_TABLE_0 Register.....	2998
Figure 12-913. DSS0_VP_GAMMA_TABLE_1 Register.....	2999
Figure 12-914. DSS0_VP_GAMMA_TABLE_2 Register.....	3000
Figure 12-915. DSS0_VP_GAMMA_TABLE_3 Register.....	3001
Figure 12-916. DSS0_VP_GAMMA_TABLE_4 Register.....	3002
Figure 12-917. DSS0_VP_GAMMA_TABLE_5 Register.....	3003
Figure 12-918. DSS0_VP_GAMMA_TABLE_6 Register.....	3004
Figure 12-919. DSS0_VP_GAMMA_TABLE_7 Register.....	3005
Figure 12-920. DSS0_VP_GAMMA_TABLE_8 Register.....	3006
Figure 12-921. DSS0_VP_GAMMA_TABLE_9 Register.....	3007
Figure 12-922. DSS0_VP_GAMMA_TABLE_10 Register.....	3008
Figure 12-923. DSS0_VP_GAMMA_TABLE_11 Register.....	3009
Figure 12-924. DSS0_VP_GAMMA_TABLE_12 Register.....	3010
Figure 12-925. DSS0_VP_GAMMA_TABLE_13 Register.....	3011
Figure 12-926. DSS0_VP_GAMMA_TABLE_14 Register.....	3012
Figure 12-927. DSS0_VP_GAMMA_TABLE_15 Register.....	3013
Figure 12-928. DSS0_VP_SECURE Register.....	3014
Figure 12-929. DSS0_WB_ACCUH_0 Register.....	3019
Figure 12-930. DSS0_WB_ACCUH_1 Register.....	3020
Figure 12-931. DSS0_WB_ACCUH2_0 Register.....	3021
Figure 12-932. DSS0_WB_ACCUH2_1 Register.....	3022
Figure 12-933. DSS0_WB_ACCUV_0 Register.....	3023
Figure 12-934. DSS0_WB_ACCUV_1 Register.....	3024

Figure 12-935. DSS0_WB_ACCUV2_0 Register.....	3025
Figure 12-936. DSS0_WB_ACCUV2_1 Register.....	3026
Figure 12-937. DSS0_WB_ATTRIBUTES Register.....	3027
Figure 12-938. DSS0_WB_ATTRIBUTES2 Register.....	3031
Figure 12-939. DSS0_WB_BA_0 Register.....	3033
Figure 12-940. DSS0_WB_BA_1 Register.....	3034
Figure 12-941. DSS0_WB_BA_UV_0 Register.....	3035
Figure 12-942. DSS0_WB_BA_UV_1 Register.....	3036
Figure 12-943. DSS0_WB_BUF_SIZE_STATUS Register.....	3037
Figure 12-944. DSS0_WB_BUF_THRESHOLD Register.....	3038
Figure 12-945. DSS0_WB_CSC_COEF0 Register.....	3039
Figure 12-946. DSS0_WB_CSC_COEF1 Register.....	3040
Figure 12-947. DSS0_WB_CSC_COEF2 Register.....	3041
Figure 12-948. DSS0_WB_CSC_COEF3 Register.....	3042
Figure 12-949. DSS0_WB_CSC_COEF4 Register.....	3043
Figure 12-950. DSS0_WB_CSC_COEF5 Register.....	3044
Figure 12-951. DSS0_WB_CSC_COEF6 Register.....	3045
Figure 12-952. DSS0_WB_FIRH Register.....	3046
Figure 12-953. DSS0_WB_FIRH2 Register.....	3047
Figure 12-954. DSS0_WB_FIRV Register.....	3048
Figure 12-955. DSS0_WB_FIRV2 Register.....	3049
Figure 12-956. DSS0_WB_FIR_COEF_H0_0 Register.....	3050
Figure 12-957. DSS0_WB_FIR_COEF_H0_1 Register.....	3051
Figure 12-958. DSS0_WB_FIR_COEF_H0_2 Register.....	3052
Figure 12-959. DSS0_WB_FIR_COEF_H0_3 Register.....	3053
Figure 12-960. DSS0_WB_FIR_COEF_H0_4 Register.....	3054
Figure 12-961. DSS0_WB_FIR_COEF_H0_5 Register.....	3055
Figure 12-962. DSS0_WB_FIR_COEF_H0_6 Register.....	3056
Figure 12-963. DSS0_WB_FIR_COEF_H0_7 Register.....	3057
Figure 12-964. DSS0_WB_FIR_COEF_H0_8 Register.....	3058
Figure 12-965. DSS0_WB_FIR_COEF_H0_C_0 Register.....	3059
Figure 12-966. DSS0_WB_FIR_COEF_H0_C_1 Register.....	3060
Figure 12-967. DSS0_WB_FIR_COEF_H0_C_2 Register.....	3061
Figure 12-968. DSS0_WB_FIR_COEF_H0_C_3 Register.....	3062
Figure 12-969. DSS0_WB_FIR_COEF_H0_C_4 Register.....	3063
Figure 12-970. DSS0_WB_FIR_COEF_H0_C_5 Register.....	3064
Figure 12-971. DSS0_WB_FIR_COEF_H0_C_6 Register.....	3065
Figure 12-972. DSS0_WB_FIR_COEF_H0_C_7 Register.....	3066
Figure 12-973. DSS0_WB_FIR_COEF_H0_C_8 Register.....	3067
Figure 12-974. DSS0_WB_FIR_COEF_H12_0 Register.....	3068
Figure 12-975. DSS0_WB_FIR_COEF_H12_1 Register.....	3069
Figure 12-976. DSS0_WB_FIR_COEF_H12_2 Register.....	3070
Figure 12-977. DSS0_WB_FIR_COEF_H12_3 Register.....	3071
Figure 12-978. DSS0_WB_FIR_COEF_H12_4 Register.....	3072
Figure 12-979. DSS0_WB_FIR_COEF_H12_5 Register.....	3073
Figure 12-980. DSS0_WB_FIR_COEF_H12_6 Register.....	3074
Figure 12-981. DSS0_WB_FIR_COEF_H12_7 Register.....	3075
Figure 12-982. DSS0_WB_FIR_COEF_H12_8 Register.....	3076
Figure 12-983. DSS0_WB_FIR_COEF_H12_9 Register.....	3077
Figure 12-984. DSS0_WB_FIR_COEF_H12_10 Register.....	3078
Figure 12-985. DSS0_WB_FIR_COEF_H12_11 Register.....	3079
Figure 12-986. DSS0_WB_FIR_COEF_H12_12 Register.....	3080
Figure 12-987. DSS0_WB_FIR_COEF_H12_13 Register.....	3081
Figure 12-988. DSS0_WB_FIR_COEF_H12_14 Register.....	3082
Figure 12-989. DSS0_WB_FIR_COEF_H12_15 Register.....	3083
Figure 12-990. DSS0_WB_FIR_COEF_H12_C_0 Register.....	3084
Figure 12-991. DSS0_WB_FIR_COEF_H12_C_1 Register.....	3085
Figure 12-992. DSS0_WB_FIR_COEF_H12_C_2 Register.....	3086
Figure 12-993. DSS0_WB_FIR_COEF_H12_C_3 Register.....	3087
Figure 12-994. DSS0_WB_FIR_COEF_H12_C_4 Register.....	3088
Figure 12-995. DSS0_WB_FIR_COEF_H12_C_5 Register.....	3089



Figure 12-996. DSS0_WB_FIR_COEF_H12_C_6 Register.....	3090
Figure 12-997. DSS0_WB_FIR_COEF_H12_C_7 Register.....	3091
Figure 12-998. DSS0_WB_FIR_COEF_H12_C_8 Register.....	3092
Figure 12-999. DSS0_WB_FIR_COEF_H12_C_9 Register.....	3093
Figure 12-1000. DSS0_WB_FIR_COEF_H12_C_10 Register.....	3094
Figure 12-1001. DSS0_WB_FIR_COEF_H12_C_11 Register.....	3095
Figure 12-1002. DSS0_WB_FIR_COEF_H12_C_12 Register.....	3096
Figure 12-1003. DSS0_WB_FIR_COEF_H12_C_13 Register.....	3097
Figure 12-1004. DSS0_WB_FIR_COEF_H12_C_14 Register.....	3098
Figure 12-1005. DSS0_WB_FIR_COEF_H12_C_15 Register.....	3099
Figure 12-1006. DSS0_WB_FIR_COEF_V0_0 Register.....	3100
Figure 12-1007. DSS0_WB_FIR_COEF_V0_1 Register.....	3101
Figure 12-1008. DSS0_WB_FIR_COEF_V0_2 Register.....	3102
Figure 12-1009. DSS0_WB_FIR_COEF_V0_3 Register.....	3103
Figure 12-1010. DSS0_WB_FIR_COEF_V0_4 Register.....	3104
Figure 12-1011. DSS0_WB_FIR_COEF_V0_5 Register.....	3105
Figure 12-1012. DSS0_WB_FIR_COEF_V0_6 Register.....	3106
Figure 12-1013. DSS0_WB_FIR_COEF_V0_7 Register.....	3107
Figure 12-1014. DSS0_WB_FIR_COEF_V0_8 Register.....	3108
Figure 12-1015. DSS0_WB_FIR_COEF_V0_C_0 Register.....	3109
Figure 12-1016. DSS0_WB_FIR_COEF_V0_C_1 Register.....	3110
Figure 12-1017. DSS0_WB_FIR_COEF_V0_C_2 Register.....	3111
Figure 12-1018. DSS0_WB_FIR_COEF_V0_C_3 Register.....	3112
Figure 12-1019. DSS0_WB_FIR_COEF_V0_C_4 Register.....	3113
Figure 12-1020. DSS0_WB_FIR_COEF_V0_C_5 Register.....	3114
Figure 12-1021. DSS0_WB_FIR_COEF_V0_C_6 Register.....	3115
Figure 12-1022. DSS0_WB_FIR_COEF_V0_C_7 Register.....	3116
Figure 12-1023. DSS0_WB_FIR_COEF_V0_C_8 Register.....	3117
Figure 12-1024. DSS0_WB_FIR_COEF_V12_0 Register.....	3118
Figure 12-1025. DSS0_WB_FIR_COEF_V12_1 Register.....	3119
Figure 12-1026. DSS0_WB_FIR_COEF_V12_2 Register.....	3120
Figure 12-1027. DSS0_WB_FIR_COEF_V12_3 Register.....	3121
Figure 12-1028. DSS0_WB_FIR_COEF_V12_4 Register.....	3122
Figure 12-1029. DSS0_WB_FIR_COEF_V12_5 Register.....	3123
Figure 12-1030. DSS0_WB_FIR_COEF_V12_6 Register.....	3124
Figure 12-1031. DSS0_WB_FIR_COEF_V12_7 Register.....	3125
Figure 12-1032. DSS0_WB_FIR_COEF_V12_8 Register.....	3126
Figure 12-1033. DSS0_WB_FIR_COEF_V12_9 Register.....	3127
Figure 12-1034. DSS0_WB_FIR_COEF_V12_10 Register.....	3128
Figure 12-1035. DSS0_WB_FIR_COEF_V12_11 Register.....	3129
Figure 12-1036. DSS0_WB_FIR_COEF_V12_12 Register.....	3130
Figure 12-1037. DSS0_WB_FIR_COEF_V12_13 Register.....	3131
Figure 12-1038. DSS0_WB_FIR_COEF_V12_14 Register.....	3132
Figure 12-1039. DSS0_WB_FIR_COEF_V12_15 Register.....	3133
Figure 12-1040. DSS0_WB_FIR_COEF_V12_C_0 Register.....	3134
Figure 12-1041. DSS0_WB_FIR_COEF_V12_C_1 Register.....	3135
Figure 12-1042. DSS0_WB_FIR_COEF_V12_C_2 Register.....	3136
Figure 12-1043. DSS0_WB_FIR_COEF_V12_C_3 Register.....	3137
Figure 12-1044. DSS0_WB_FIR_COEF_V12_C_4 Register.....	3138
Figure 12-1045. DSS0_WB_FIR_COEF_V12_C_5 Register.....	3139
Figure 12-1046. DSS0_WB_FIR_COEF_V12_C_6 Register.....	3140
Figure 12-1047. DSS0_WB_FIR_COEF_V12_C_7 Register.....	3141
Figure 12-1048. DSS0_WB_FIR_COEF_V12_C_8 Register.....	3142
Figure 12-1049. DSS0_WB_FIR_COEF_V12_C_9 Register.....	3143
Figure 12-1050. DSS0_WB_FIR_COEF_V12_C_10 Register.....	3144
Figure 12-1051. DSS0_WB_FIR_COEF_V12_C_11 Register.....	3145
Figure 12-1052. DSS0_WB_FIR_COEF_V12_C_12 Register.....	3146
Figure 12-1053. DSS0_WB_FIR_COEF_V12_C_13 Register.....	3147
Figure 12-1054. DSS0_WB_FIR_COEF_V12_C_14 Register.....	3148
Figure 12-1055. DSS0_WB_FIR_COEF_V12_C_15 Register.....	3149
Figure 12-1056. DSS0_WB_MFLAG_THRESHOLD Register.....	3150

Figure 12-1057. DSS0_WB_PICTURE_SIZE Register.....	3151
Figure 12-1058. DSS0_WB_SIZE Register.....	3152
Figure 12-1059. DSS0_WB_POSITION Register.....	3153
Figure 12-1060. DSS0_WB_CSC_COEF7 Register.....	3154
Figure 12-1061. DSS0_WB_ROW_INC Register.....	3155
Figure 12-1062. DSS0_WB_ROW_INC_UV Register.....	3156
Figure 12-1063. DSS0_WB_BA_EXT_0 Register.....	3157
Figure 12-1064. DSS0_WB_BA_EXT_1 Register.....	3158
Figure 12-1065. DSS0_WB_BA_UV_EXT_0 Register.....	3159
Figure 12-1066. DSS0_WB_BA_UV_EXT_1 Register.....	3160
Figure 12-1067. DSS0_WB_SECURE Register.....	3161
Figure 12-1068. DSITX Controller Block Architecture.....	3163
Figure 12-1069. DSITX Clock Scheme and Domains.....	3165
Figure 12-1070. DSI Clock Gate / Power Off Procedure.....	3166
Figure 12-1071. DSI Pixel Mapping On The DSI-DPI Interface.....	3167
Figure 12-1072. Example Display Subsystem Showing Controller and PHY Blocks.....	3169
Figure 12-1073. TVG MODE Patterns.....	3172
Figure 12-1074. Start-up Procedure Summary.....	3176
Figure 12-1075. Direct Command Management.....	3179
Figure 12-1076. Bus Turnaround Timing Sequence from Controller to Panel and Panel to Controller.....	3182
Figure 12-1077. get_scan_line Command.....	3183
Figure 12-1078. set_tear_on Command - 1.....	3184
Figure 12-1079. set_tear_on Command - 2.....	3185
Figure 12-1080. set_tear_on Command - 3.....	3185
Figure 12-1081. set_tear_scanline Command - 1.....	3186
Figure 12-1082. set_tear_scanline Command - 2.....	3186
Figure 12-1083. set_tear_off Command.....	3187
Figure 12-1084. LP Transmission Timing Diagram for Host Read and Trigger ACK.....	3190
Figure 12-1085. Panel Read Response Sequences.....	3190
Figure 12-1086. DSI Packet Terminology.....	3191
Figure 12-1087. Video Pulse Mode (Non-Burst) Timing Diagram.....	3192
Figure 12-1088. Video Pulse Mode (Non-Burst).....	3193
Figure 12-1089. Video Event Mode (Non-Burst).....	3194
Figure 12-1090. Video Sync Event in Burst Mode.....	3195
Figure 12-1091. Timing Diagram for Clock Lane Activation/Deactivation.....	3198
Figure 12-1092. Timing Diagram for Data Lane Activation/Deactivation.....	3198
Figure 12-1093. Wakeup Time Timing Diagram.....	3199
Figure 12-1094. TVG MODE Patterns.....	3201
Figure 12-1095. Timing Diagram.....	3202
Figure 12-1096. DPI Interface to DSI DPHY Timing Diagram.....	3203
Figure 12-1097. Horizontal Line Packet Timing Diagram.....	3204
Figure 12-1098. Horizontal Line Packet - Shorter HBP - Longer HFP Timing Diagram.....	3204
Figure 12-1099. Packet Alignment Control Using HFP Byte Increase - tx_byte_clk Faster than Ideal Rate.....	3205
Figure 12-1100. Low Power Operation with Four and Two Active Lanes.....	3206
Figure 12-1101. REG_LINE_DURATION Timing Example for LP Operation.....	3206
Figure 12-1102. DPI Operation with Event Burst Mode.....	3207
Figure 12-1103. Vertical Timing.....	3208
Figure 12-1104. Vertical Timing.....	3209
Figure 12-1105. Non-Burst Mode With Sync Pulse Line Structures.....	3209
Figure 12-1106. Horizontal Timing - 1.....	3211
Figure 12-1107. Horizontal Timing - 1.....	3212
Figure 12-1108. Burst Event Mode Horizontal Timing - 1.....	3213
Figure 12-1109. Burst Event Mode Horizontal Timing - 2.....	3214
Figure 12-1110. DSITX Controller Video Frames for Burst Mode Compared with DPI Driver Side.....	3216
Figure 12-1111. Burst Mode Operation.....	3217
Figure 12-1112. Burst Mode Command Locations During Video Frame.....	3218
Figure 12-1113. Example DPI Image Format for SDF Combining Left And Right Pixels.....	3223
Figure 12-1114. 3D Image format for 3DVSYNCR splitting first and second images.....	3223
Figure 12-1115. EDP System Integration Block Diagram.....	3225
Figure 12-1116. DP Wrapper DPI Data Muxing and VIF Mapping.....	3226
Figure 12-1117. EDP Wrapper Supported Input DSS_DPI_DATA Pixel Formats.....	3227

Figure 12-1118. EDP Wrapper Internal Pixel Bus Remapping (First Stage).....	3228
Figure 12-1119. EDP Wrapper Internal Pixel Remapping (Additional for DSC Bound Pixel Data Bus).....	3228
Figure 12-1120. EDP Audio I2S Timing - Bit Allocation (Right Justification).....	3230
Figure 12-1121. EDP Audio I2S Timing - TDM Time Slot Allocation (M = 8).....	3230
Figure 12-1122. EDP DSC Single Input/1-Transport Link Mode and Split Panel/Dual Panel Mode.....	3232
Figure 12-1123. EDP Display Port Transmitter Controller Functional Block Diagram.....	3233
Figure 12-1124. EDP Interface to AUX PHY.....	3234
Figure 12-1125. EDP Clock Diagram.....	3235
Figure 12-1126. EDP PHY Clock Connections.....	3236
Figure 12-1127. EDP ECC_WRAP/ECC_AGGR and PARITY_INV Logic Connection.....	3241
Figure 12-1128. EDP MHDPTX Controller Boot Sequence.....	3244
Figure 12-1129. EDP Operation Sequence – DisplayPort.....	3246
Figure 12-1130. EDP Operation Sequence – DisplayPort Video Format Change.....	3247
Figure 12-1131. EDP Operation Sequence – DisplayPort Audio Format Change.....	3248
Figure 12-1132. EDP Operation Sequence – Embedded HDCP 2.2.....	3249
Figure 12-1133. EDP Operation Sequence – Embedded HDCP 1.4.....	3250
Figure 12-1134. CSI_RX_IF Module Overview.....	3255
Figure 12-1135. CSI_RX_IF Integration.....	3257
Figure 12-1136. CSI_RX_IF Block Diagram.....	3263
Figure 12-1137. CSI_RX_IF YUV422-8 Memory Data Organization.....	3265
Figure 12-1138. CSI_RX_IF YUV422-10 memory data organization.....	3266
Figure 12-1139. CSI_RX_IF YUV420-8 memory data organization.....	3267
Figure 12-1140. CSI_RX_IF YUV420-10 memory data organization.....	3267
Figure 12-1141. CSI_RX_IF RAW (UNPACKED) memory data organization.....	3268
Figure 12-1142. CSI_RX_IF RGB memory data organization.....	3269
Figure 12-1143. CSI_RX_IF RAW12 (PACKED) memory data organization.....	3270
Figure 12-1144. Minimal Stream Control - Start and Stop.....	3272
Figure 12-1145. Stream Reconfiguration Using Start Stop Start Flow.....	3273
Figure 12-1146. Stream Start and Stop Using Monitor Control.....	3274
Figure 12-1147. Stream Error Detection Causing Stop.....	3275
Figure 12-1148. Stream Error Bypass - CRC Error During Payload.....	3275
Figure 12-1149. Stream Soft Reset After ECC Non-Recoverable Error.....	3276
Figure 12-1150. Stream Start and Stop Using Monitor Control.....	3276
Figure 12-1151. Stream Frame Capture Control Flow Diagram.....	3278
Figure 12-1152. Timer Interrupt Flow Diagram.....	3279
Figure 12-1153. Line/Byte Counters Interrupt Flow Diagram.....	3280
Figure 12-1154. Basic Single Stream Use Case Illustration.....	3280
Figure 12-1155. DPHY_RX Module Overview.....	3283
Figure 12-1156. DPHY_RX Environment.....	3285
Figure 12-1157. DPHY_RX Integration.....	3287
Figure 12-1158. Common Power Up and Initialization Timing Diagram.....	3289
Figure 12-1159. Lane Power Up and Initialization Timing Diagram.....	3290
Figure 12-1160. PHY Disable Timing Diagram.....	3291
Figure 12-1161. CSI_TX_IF Module Overview.....	3297
Figure 12-1162. CSI_TX_IF Integration.....	3299
Figure 12-1163. CSI_TX_IF Block Diagram.....	3302
Figure 12-1164. CSI_TX_IF PSI_L thread mapping.....	3302
Figure 12-1165. CSI_TX_IF PSI_L Color Bar sample output.....	3303
Figure 12-1166. CSI_TX_IF YUV422-8 Memory Data Organization.....	3306
Figure 12-1167. CSI_TX_IF YUV422-10 memory data organization.....	3307
Figure 12-1168. CSI_TX_IF YUV420-8 memory data organization.....	3308
Figure 12-1169. CSI_TX_IF YUV420-10 memory data organization.....	3308
Figure 12-1170. CSI_TX_IF RAW (UNPACKED) memory data organization.....	3309
Figure 12-1171. CSI_TX_IF RGB memory data organization.....	3310
Figure 12-1172. CSI_TX_IF RAW12 (PACKED) memory data organization.....	3311
Figure 12-1173. Clock Lane Control FSMs Timing Diagram.....	3314
Figure 12-1174. DPHY_TX Overview.....	3317
Figure 12-1175. DPHY_TX Environment.....	3319
Figure 12-1176. DPHY_TX Integration.....	3322
Figure 12-1177. VPFE Overview.....	3324
Figure 12-1178. VPFE External System Interface.....	3326

Figure 12-1179. VPFE Integration.....	3329
Figure 12-1180. VPFE Block Diagram.....	3331
Figure 12-1181. CCD/CMOS Processing.....	3332
Figure 12-1182. CCDC_VD0_INT/CCDC_VD1_INT Interrupt Behavior when VDPOL = 0.....	3333
Figure 12-1183. CCDC_VD0_INT/CCDC_VD1_INT Interrupt Behavior when VDPOL = 1.....	3333
Figure 12-1184. CCDC_VD2_INT Interrupt Behavior.....	3333
Figure 12-1185. CCD Controller Frame Settings and Format.....	3337
Figure 12-1186. BT.656 Signal Interface.....	3338
Figure 12-1187. Data Processing for Raw Data Mode.....	3340
Figure 12-1188. Color Patterns.....	3340
Figure 12-1189. Input Sampling Block Diagram for Raw Data Mode.....	3341
Figure 12-1190. Optical Black Averaging and Application for Raw Data Mode.....	3342
Figure 12-1191. Black Clamping and Black Level Compensation for Raw Data Mode.....	3343
Figure 12-1192. Output Formatter for Raw Data Mode.....	3344
Figure 12-1193. A-Law Table.....	3345
Figure 12-1194. Input and Output Image in Non-Inversed vs Inversed Formats.....	3348
Figure 12-1195. Data Processing for YCbCr/BT.656 Modes.....	3351
Figure 12-1196. Input Sampling Block Diagram for YCbCr/BT.656 Modes.....	3352
Figure 12-1197. Black Clamping for YCbCr/BT.656 Modes.....	3352
Figure 12-1198. Output Formatter for YCbCr/BT.656 Modes.....	3353
Figure 12-1199. GTC Integration.....	3355
Figure 12-1200. GTC Block Diagram.....	3357
Figure 12-1201. RTI Overview.....	3360
Figure 12-1202. MCU_RTI Integration.....	3363
Figure 12-1203. RTI Integration.....	3366
Figure 12-1204. RTI Counters Block Diagram.....	3371
Figure 12-1205. RTI Compare Block Diagram.....	3372
Figure 12-1206. RTI Digital Watchdog Functional Block Diagram.....	3373
Figure 12-1207. RTI Digital Watchdog Operation.....	3374
Figure 12-1208. RTI Digital Windowed Watchdog Timing Example.....	3375
Figure 12-1209. RTI Digital Windowed Watchdog Operation Block Diagram.....	3376
Figure 12-1210. Timers Overview.....	3379
Figure 12-1211. Timers External System Interface.....	3381
Figure 12-1212. MCU_TIMER Integration.....	3385
Figure 12-1213. TIMER Integration.....	3393
Figure 12-1214. Timer Block Diagram.....	3413
Figure 12-1215. Wake-Up Request Generation.....	3414
Figure 12-1216. TIMER_TCCR Timing Value.....	3416
Figure 12-1217. Block Diagram of the 1-ms Tick Module.....	3417
Figure 12-1218. Capture Wave Example for TIMER_TCLR[13] CAPT_MODE = 0.....	3419
Figure 12-1219. Capture Wave Example for TIMER_TCLR[13] CAPT_MODE = 1.....	3420
Figure 12-1220. Timing Diagram of PWM With TIMER_TCLR[7] SCPWM Bit = 0.....	3421
Figure 12-1221. Timing Diagram of PWM With TIMER_TCLR[7] SCPWM Bit = 1.....	3422
Figure 12-1222. DCC Modules Overview.....	3431
Figure 12-1223. MCU_DCC Integration.....	3433
Figure 12-1224. DCC Integration.....	3438
Figure 12-1225. DCC Functional Block Diagram.....	3452
Figure 12-1226. DCC Clock0 and Clock1 With no Error.....	3453
Figure 12-1227. DCC Clock1 slower than Clock0 results in an error and stops counting.....	3453
Figure 12-1228. DCC Clock1 faster than Clock0 results in an error and stops counting.....	3454
Figure 12-1229. DCC Clock1 not present results in an error and stops counting.....	3454
Figure 12-1230. DCC Clock0 not present results in an error and stops counting.....	3454
Figure 12-1231. DCC_GCTRL Register.....	3460
Figure 12-1232. DCC_REV Register.....	3462
Figure 12-1233. DCC_CNTSEED0 Register.....	3464
Figure 12-1234. DCC_VALIDSEED0 Register.....	3465
Figure 12-1235. DCC_CNTSEED1 Register.....	3466
Figure 12-1236. DCC_STATUS Register.....	3467
Figure 12-1237. DCC_CNT0 Register.....	3469
Figure 12-1238. DCC_VALID0 Register.....	3470
Figure 12-1239. DCC_CNT1 Register.....	3471



Figure 12-1240. DCC_CLKSRC1 Register.....	3472
Figure 12-1241. DCC_CLKSRC0 Register.....	3474
Figure 12-1242. DCC_GCTRL2 Register.....	3476
Figure 12-1243. DCC_STATUS2 Register.....	3478
Figure 12-1244. DCC_ERRCNT Register.....	3480
Figure 12-1245. ESM Modules Overview.....	3482
Figure 12-1246. ESM Modules Environment.....	3483
Figure 12-1247. WKUP_ESM Integration.....	3485
Figure 12-1248. MCU_ESM Integration.....	3488
Figure 12-1249. ESM Integration.....	3491
Figure 12-1250. ESM Block Diagram.....	3495
Figure 12-1251. ESM Error Pin State Flowchart.....	3500
Figure 12-1252. ESM Error Pin Assertion.....	3501
Figure 12-1253. ESM Error Pin Assertion with CLEAR during Minimum Interval.....	3501
Figure 12-1254. ESM Error Pin Asserting with CLEAR after Minimum Interval.....	3501
Figure 12-1255. ESM Error Pin Asserting with Interval Reset by Additional Error Event(s).....	3502
Figure 12-1256. ESM Error Pin Asserting with Single CLEAR for Multiple Events.....	3502
Figure 12-1257. ESM Error Pin Asserting with New Error During Minimum Time Interval.....	3502
Figure 12-1258. MCRC Integration.....	3506
Figure 12-1259. MCRC Block Diagram.....	3508
Figure 12-1260. AUTO Mode Using Hardware Timer Trigger.....	3512
Figure 12-1261. AUTO Mode With Software CPU Trigger.....	3513
Figure 12-1262. Semi-CPU Mode With Hardware Timer Trigger.....	3513
Figure 12-1263. ECC Aggregator Integration.....	3524
Figure 12-1264. ECC Aggregator Block Diagram.....	3526

## List of Tables

Table 1-1. Modules Allocation and Instances within Device Domains.....	83
Table 1-2. Device JTAG ID.....	117
Table 1-3. Device JTAG ID Values.....	117
Table 2-1. MAIN Domain Memory Map.....	119
Table 2-2. NAVSS0 Alias Memory Map.....	142
Table 2-3. MCU Domain Memory Map.....	173
Table 2-4. WKUP Domain Memory Map.....	177
Table 2-5. R5FSS0/1 Memory Map.....	178
Table 2-6. C66SS0/1 Memory Map.....	178
Table 2-7. PRU_ICSSG0/1 Memory Map.....	178
Table 2-8. MCU_R5FSS0 Memory Map.....	179
Table 3-1. WKUP_CBASS0 Integration Attributes.....	184
Table 3-2. WKUP_CBASS0 Clocks and Resets.....	184
Table 3-3. WKUP_CBASS0 Hardware Requests.....	184
Table 3-4. MCU_CBASS0 Integration Attributes.....	185
Table 3-5. MCU_CBASS0 Clocks and Resets.....	185
Table 3-6. MCU_CBASS0 Hardware Requests.....	186
Table 3-7. CBASS0 Integration Attributes.....	186
Table 3-8. CBASS0 Clocks and Resets.....	187
Table 3-9. CBASS0 Hardware Requests.....	187
Table 3-10. CBASS0 Connectivity Matrix (Part 1).....	189
Table 3-11. CBASS0 Connectivity Matrix (Part 2).....	191
Table 3-12. CBASS0 Connectivity Matrix (Part 3).....	194
Table 3-13. CBASS0 Connectivity Matrix (Part 4).....	196
Table 3-14. CBASS0 Connectivity Matrix (Part 5).....	199
Table 3-15. CBASS0 Connectivity Matrix (Part 6).....	201
Table 3-16. CBASS0 Connectivity Matrix (Part 7).....	203
Table 3-17. CBASS0 Connectivity Matrix (Part 8).....	206
Table 3-18. CBASS0 Connectivity Matrix (Part 9).....	208
Table 3-19. CBASS0 Connectivity Matrix (Part 10).....	211
Table 3-20. MCU_CBASS0 Connectivity Matrix (Part 1).....	213
Table 3-21. MCU_CBASS0 Connectivity Matrix (Part 2).....	216
Table 3-22. WKUP_CBASS0 Connectivity Matrix.....	218

Table 3-23. Limitations of Programming.....	220
Table 3-24. CBASS_AASRC0 Master Modules Attributes.....	221
Table 3-25. CBASS_AC0 Master Modules Attributes.....	222
Table 3-26. CBASS_CSI0 Master Modules Attributes.....	222
Table 3-27. CBASS_DATADEBUG0 Master Modules Attributes.....	222
Table 3-28. CBASS_HC_CFG0 Master Modules Attributes.....	222
Table 3-29. CBASS_HC0 Master Modules Attributes.....	222
Table 3-30. CBASS_HC2_0 Master Modules Attributes.....	223
Table 3-31. CBASS_IPPHY0 Master Modules Attributes.....	223
Table 3-32. CBASS_MCASP_G0_0 Master Modules Attributes.....	223
Table 3-33. CBASS_MCASP_G1_0 Master Modules Attributes.....	223
Table 3-34. CBASS_RC_CFG0 Master Modules Attributes.....	224
Table 3-35. CBASS_RC0 Master Modules Attributes.....	224
Table 3-36. MCU_CBASS0 Master Modules Attributes.....	225
Table 3-37. WKUP_CBASS0 Master Modules Attributes.....	225
Table 3-38. Special Priv-IDs.....	229
Table 3-39. Priv ID and ISC Assignment for DRA829/TDA4VM .....	229
Table 3-40. CBASS_AASRC0 Slave Modules Attributes.....	236
Table 3-41. CBASS_AC0 Slave Modules Attributes.....	237
Table 3-42. CBASS_CSI0 Slave Modules Attributes.....	237
Table 3-43. CBASS_DATADEBUG0 Slave Modules Attributes.....	237
Table 3-44. CBASS_HC_CFG0 Slave Modules Attributes.....	237
Table 3-45. CBASS_HC0 Slave Modules Attributes.....	238
Table 3-46. CBASS_HC2_0 Slave Modules Attributes.....	238
Table 3-47. CBASS_IPPHY0 Slave Modules Attributes.....	238
Table 3-48. CBASS_MCASP_G0_0 Slave Modules Attributes.....	240
Table 3-49. CBASS_MCASP_G1_0 Slave Modules Attributes.....	240
Table 3-50. CBASS_RC_CFG0 Slave Modules Attributes.....	241
Table 3-51. CBASS_RC0 Slave Modules Attributes.....	241
Table 3-52. INFRA_CBASS0 Slave Modules Attributes.....	242
Table 3-53. MCU_CBASS0 Slave Modules Attributes.....	243
Table 3-54. WKUP_CBASS0 Slave Modules Attributes.....	244
Table 3-55. Firewall Violation Parameters.....	248
Table 3-56. Interrupts.....	257
Table 3-57. Module Memory Map.....	259
Table 3-58. Module Parameters.....	267
Table 3-59. Interrupt Requests.....	268
Table 3-60. EDC Controller Interface.....	268
Table 4-1. ROM Code Boot Modes.....	271
Table 4-2. MCU_BOOTMODE Pin Mapping.....	280
Table 4-3. PLL Reference Clock Selection.....	280
Table 4-4. MCU Only Configuration.....	280
Table 4-5. Boot Mode Selection When MCU Only = 1.....	280
Table 4-6. POST Selection.....	281
Table 4-7. BOOTMODE Pin Mapping.....	282
Table 4-8. Primary Boot Mode Selection When MCU Only = 0.....	282
Table 4-9. Backup Mode Selection (MCU Only=0).....	282
Table 4-10. Primary Boot Mode Configuration.....	283
Table 4-11. Backup Boot Mode Configuration.....	283
Table 4-12. No-boot/Dev-boot Configuration Fields.....	284
Table 4-13. OSPI Configuration Fields.....	285
Table 4-14. OSPI Pin Usage.....	285
Table 4-15. MCU_BOOTMODE Pin Map OSPI.....	286
Table 4-16. BOOTMODE Pin Map OSPI.....	286
Table 4-17. xSPI Boot Configuration Fields.....	286
Table 4-18. xSPI Pin Usage.....	286
Table 4-19. QSPI Boot Configuration Fields.....	290
Table 4-20. QSPI Port 0 Pin Usage.....	290
Table 4-21. QSPI Port 1 Pin Usage.....	290
Table 4-22. SPI Boot Configuration Fields.....	291
Table 4-23. SPI Port 0 Pin Usage.....	291

Table 4-24. SPI Port 1 Pin Usage.....	291
Table 4-25. I2C Boot Configuration Fields.....	292
Table 4-26. I2C Pin Usage.....	292
Table 4-27. MMC/SD Card Boot Configuration Fields.....	293
Table 4-28. MMC/SD Card Port 1 Pin Usage.....	293
Table 4-29. Ethernet RGMII Boot Configuration Fields.....	294
Table 4-30. Ethernet RMII Boot Configuration Fields.....	294
Table 4-31. Ethernet Backup Boot Configuration Field .....	294
Table 4-32. RGMII Pin Usage.....	294
Table 4-33. RMII Pin Usage.....	295
Table 4-34. USB Boot Configuration Fields.....	296
Table 4-35. USB Port 0 Pin Usage.....	296
Table 4-36. USB Port 1 Pin Usage.....	296
Table 4-37. PCIe Boot Configuration Fields.....	297
Table 4-38. UART Boot Configuration Fields.....	298
Table 4-39. UART Port 0 Pin Usage.....	298
Table 4-40. UART Port 1 Pin Usage.....	298
Table 4-41. GPMC NOR Boot Configuration Fields.....	299
Table 4-42. GPMC non-muxed Pin Usage.....	299
Table 4-43. GPMC A/D muxed Pin Usage.....	300
Table 4-44. GPMC A/A/D muxed Pin Usage.....	301
Table 4-45. eMMC Boot Configuration Fields.....	303
Table 4-46. eMMC Port 1 Pin Usage.....	303
Table 4-47. ext_csd[162] Register.....	303
Table 4-48. PLL Configuration by Boot Mode.....	304
Table 4-49. PLL Configuration for MCU_PLL0, MCU_PLL2, Main PLL0, and Main PLL3.....	304
Table 4-50. PLL Configuration for MCU_PLL1.....	304
Table 4-51. PLL Configuration for Main PLL1.....	305
Table 4-52. PLL Configuration for Main PLL 2.....	305
Table 4-53. Boot Parameter Table Common Header.....	306
Table 4-54. Boot Peripheral Selection.....	306
Table 4-55. Boot Parameter Table PLL Configuration.....	307
Table 4-56. PLL Domain and Enable Configuration.....	307
Table 4-57. PLL Domain and Enable Field Description.....	307
Table 4-58. PLL Reference Source Bit Fields.....	308
Table 4-59. PLL Reference Source Field Description.....	308
Table 4-60. PCIe Boot Parameter Table.....	308
Table 4-61. I2C Boot Parameter Table.....	309
Table 4-62. OSPI/QSPI/SPI Boot Parameter Table.....	309
Table 4-63. xSPI Parameter Table.....	310
Table 4-64. GPMC NOR Boot Parameter Table.....	311
Table 4-65. Ethernet Boot Parameter Table.....	311
Table 4-66. USB Boot Parameter Table.....	313
Table 4-67. MMCSD Boot Parameter Table.....	313
Table 4-68. UART Boot Parameter Table.....	314
Table 4-69. Certificate Type Values.....	316
Table 4-70. Boot Core Values.....	316
Table 4-71. Core Options Bit Fields.....	316
Table 4-72. Core Options Field Description.....	316
Table 4-73. GPMC NOR Timing Configuration.....	320
Table 4-74. XMODEM 1024- and 128-byte Data Frames.....	323
Table 4-75. XMODEM Data Frame Fields.....	323
Table 4-76. Example of XMODEM Transfer protocol.....	324
Table 4-77. Memory Layout/MPU.....	325
Table 4-78. Global Memory Addresses.....	325
Table 4-79. ROM Code Version.....	326
Table 4-80. Memory Reserved By ROM Code.....	326
Table 5-1. WKUP_CTRL_MMR0 Integration Attributes.....	330
Table 5-2. WKUP_CTRL_MMR0 Clocks and Resets.....	330
Table 5-3. WKUP_CTRL_MMR0 Hardware Requests.....	331
Table 5-4. WKUP_CTRL_MMR0 Partition Unlock Values.....	332

Table 5-5. Summary of the WKUP_CTRL_MMR0 Clock Selection Registers.....	333
Table 5-6. Summary of the POK Registers.....	333
Table 5-7. Summary of the Voltage Glitch Detect Registers.....	334
Table 5-8. Debounce Period Values.....	336
Table 5-9. MCU_CTRL_MMR0 and MCU_SEC_MMR0 Integration Attributes.....	339
Table 5-10. MCU_CTRL_MMR0 and MCU_SEC_MMR0 Clocks and Resets.....	339
Table 5-11. MCU_CTRL_MMR0 Hardware Requests.....	339
Table 5-12. MCU_CTRL_MMR0 Partition Unlock Values.....	341
Table 5-13. Summary of the MCU_CTRL_MMR0 IPC Registers.....	341
Table 5-14. Summary of the MCU_CTRL_MMR0 Clock Muxing and Division Registers.....	343
Table 5-15. MCU_CPSW0 MAC Address Registers.....	343
Table 5-16. CTRL_MMR0 and SEC_MMR0 Integration Attributes.....	347
Table 5-17. CTRL_MMR0 and SEC_MMR0 Clocks and Resets.....	347
Table 5-18. CTRL_MMR0 Hardware Requests.....	347
Table 5-19. Description Of The Pad Configuration Register Bits.....	349
Table 5-20. Partition Unlock Values.....	351
Table 5-21. CTRL_MMR0 Events.....	353
Table 5-22. Summary of the IPC Registers.....	353
Table 5-23. Summary of the EHRPWM/EQEP Control and Status Registers.....	356
Table 5-24. PRU_ICSSG Control Registers.....	356
Table 5-25. Summary of the Clock Muxing and Division Registers.....	357
Table 5-26. Ethernet Port Operation Control Registers.....	358
Table 5-27. PCIe Operation Control Registers.....	358
Table 5-28. SERDES Lane Function Control Registers.....	358
Table 5-29. DDRSS Dynamic Frequency Change Registers.....	359
Table 5-30. POK Allocation Within Device Domains.....	361
Table 5-31. POK Modules Type and Monitored Voltage.....	362
Table 5-32. POK Programmable Features.....	363
Table 5-33. POK_SA Programmable Features.....	363
Table 5-34. POK Control Registers.....	363
Table 5-35. Values of the Possible Monitored Voltages.....	365
Table 5-36. POR Modules Allocation within Device Domains.....	366
Table 5-37. POR Clocks and Resets.....	367
Table 5-38. POR Operational Mode.....	370
Table 5-39. POK Operational Mode.....	371
Table 5-40. PRG Modules Allocation within Device Domains.....	374
Table 5-41. Dedicated Domains.....	375
Table 5-42. PRG Integration Attributes.....	376
Table 5-43. PRG Clocks and Resets.....	376
Table 5-44. PRG Hardware Requests.....	377
Table 5-45. PGD Modules Allocation within Device Domains.....	379
Table 5-46. PGD Integration Summary.....	379
Table 5-47. PGD Hardware Requests.....	380
Table 5-48. VTM Module Allocation within Device Domains.....	381
Table 5-49. WKUP_VTM0 Integration Attributes.....	384
Table 5-50. WKUP_VTM0 Clocks and Resets.....	384
Table 5-51. WKUP_VTM0 Hardware Requests.....	384
Table 5-52. Temperature Translation Table.....	387
Table 5-53. Equation Method to Calculate Code or Temperature.....	387
Table 5-54. VTM TEMPSENSOR Register Groups Mapping to Voltage Domains.....	388
Table 5-55. VTM VD Register Groups Mapping to Voltage Domains.....	389
Table 5-56. VID Bit-field Values and Their Corresponding Voltage.....	389
Table 5-57. RAM_SLEEPMODE Settings for Memory Power States.....	392
Table 5-58. WKUP_PSC0 Power Management Device-Level Layout.....	395
Table 5-59. WKUP_PSC0 Power Domain Features.....	396
Table 5-60. WKUP_PSC0 LPSC Features.....	396
Table 5-61. PSC0 Power Management Device-Level Layout.....	397
Table 5-62. PSC0 Power Domain Features.....	401
Table 5-63. PSC0 LPSC Features.....	401
Table 5-64. LPSC Dependences (Part 1).....	405
Table 5-65. LPSC Dependences (Part 2).....	409

Table 5-66. LPSC Dependences (Part 3).....	414
Table 5-67. LPSC Dependences (Part 4).....	418
Table 5-68. LPSC DEPENDENCES (PART 5).....	422
Table 5-69. Module States.....	427
Table 5-70. Local Reset of Modules Controlled by WKUP_PSC0.....	427
Table 5-71. Local Reset of Modules Controlled by PSC0.....	428
Table 5-72. A72 Core Power States.....	433
Table 5-73. Power Management Support by C7x.....	435
Table 5-74. MCU Cortex-R5F Core Power States.....	437
Table 5-75. Voltage Domains State in Low Power Modes.....	442
Table 5-76. Power Domains State in Low Power Modes.....	442
Table 5-77. Power Domains State in Low Power Modes.....	442
Table 5-78. Global Clock States in Low Power Modes.....	442
Table 5-79. Wake-up Sources in Low Power Modes.....	443
Table 5-80. LPM Entry Sequence.....	445
Table 5-81. ACTIVE to Standby Mode.....	450
Table 5-82. ACTIVE to CPD-OFF to DeepSleep.....	450
Table 5-83. ACTIVE to MCU-ONLY to OFF.....	451
Table 5-84. ACTIVE to SuspendToRAM.....	452
Table 5-85. MCU-ONLY to SuspendToRAM.....	453
Table 5-86. LPM Exit Step Labels.....	453
Table 5-87. SuspendToRAM to ACTIVE.....	456
Table 5-88. SuspendToRAM to MCU-ONLY.....	457
Table 5-89. MCU-ONLY to ACTIVE.....	457
Table 5-90. DeepSleep to CPD-OFF to ACTIVE.....	458
Table 5-91. Standby to ACTIVE.....	458
Table 5-92. MCU-ONLY-MAIN-OFF to MCU-ONLY-DEBUG.....	458
Table 5-93. Device Family OPP.....	459
Table 5-94. Valid OPP Combination.....	459
Table 5-95. Reset Supported in the Device.....	463
Table 5-96. Reset Status.....	464
Table 5-97. Reset Control.....	464
Table 5-98. PLL Behavior on Reset.....	470
Table 5-99. PLL Behavior on Warm Resets, Except VTM Event.....	471
Table 5-100. Mapping for Input Sources.....	474
Table 5-101. Internal RC Oscillator Input Sources.....	475
Table 5-102. MCU_OBSCLK0_MUX1 Output Clock Selection.....	476
Table 5-103. OBSCLK0_MUX1_CLKOUT.....	478
Table 5-104. OBSCLK0, OBSCLK1, and OBSCLK2 Clock Selection.....	478
Table 5-105. PLL MCU Domain Reference Clock Selection - WKUP_HFOSC0_CLKOUT Selection.....	486
Table 5-106. PLL MAIN Domain Reference Clock Selection.....	487
Table 5-107. PLL4 (AUDIO0 PLL) Clock Selection.....	487
Table 5-108. PLL15 (AUDIO1 PLL) Clock Selection.....	487
Table 5-109. PLLTS16FFCLAFRAC2 Output Clocks.....	489
Table 5-110. PLLTS16FFCLAFRACF Output Clocks.....	489
Table 5-111. PLLTS16FFCLVDESKEWC Output Clocks.....	490
Table 5-112. PLL Lock Signal Summary.....	491
Table 5-113. HSDIV_CLKOUT Frequency With PLL State.....	491
Table 5-114. SSMOD related bitfields.....	495
Table 5-115. Basic Information of MCU_PLL0, MCU_PLL1, and MCU_PLL2.....	497
Table 5-116. Basic Information of MAIN Domain PLLs.....	497
Table 5-117. Clock synthesis Parameters for PLLTS16FFCLAFRAC2 and PLLTS16FFCLAFRACF Types.....	500
Table 5-118. Clock synthesis Parameters for PLLTS16FFCLVDESKEWC Type.....	500
Table 5-119. Clock Output Parameter for PLLTS16FFCLAFRAC2 and PLLTS16FFCLAFRACF Types.....	500
Table 5-120. Clock Output Parameter for PLLTS16FFCLVDESKEWC Type.....	501
Table 5-121. Recalibration Feature Parameters.....	501
Table 5-122. System Clocks Operating Frequency Range.....	502
Table 5-123. Programming Sequence of PLLCTRL, HSDIV, and PLL.....	506
Table 6-1. COMPUTE_CLUSTER0 Memory Regions.....	510
Table 6-2. COMPUTE_CLUSTER0_MSMC_ECC_AGGR0 RAM ID Mapping.....	513
Table 6-3. COMPUTE_CLUSTER0_MSMC_ECC_AGGR1 RAM ID Mapping.....	515



Table 6-4. COMPUTE_CLUSTER0_MSMC_ECC_AGGR2 RAM ID Mapping.....	515
Table 6-5. C71SS0 ECC Aggregator RAM ID Mapping.....	515
Table 6-6. A72SS Modules Allocation within Device Domains.....	517
Table 6-7. A72SS Integration Attributes.....	519
Table 6-8. A72SS Clocks and Resets.....	520
Table 6-9. A72SS Hardware Requests.....	520
Table 6-10. A72 Cluster Configuration.....	521
Table 6-11. A72SS Interrupt Outputs.....	523
Table 6-12. A72SS0_CORE0_ECC_AGGR RAM ID Mapping for Interface Diagnostics.....	525
Table 6-13. A72SS0_CORE1_ECC_AGGR RAM ID Mapping for Interface Diagnostics.....	525
Table 6-14. A72SS0_CLUSTER_ECC_AGGR RAM ID Mapping for Interface Diagnostics.....	525
Table 6-15. A72SS SRAM ECC Support.....	525
Table 6-16. A72SS0_CORE0/1_ECC_AGGR Mapping.....	526
Table 6-17. A72SS0_CLUSTER_ECC_AGGR Mapping.....	527
Table 6-18. A72 Boot Mode Specifics.....	529
Table 6-19. R5FSS Allocation Across Device Domains.....	531
Table 6-20. MCU_R5FSS0 Integration Attributes.....	535
Table 6-21. MCU_R5FSS0 Clocks and Resets.....	535
Table 6-22. MCU_R5FSS0 Hardware Requests.....	535
Table 6-23. R5FSS0/1 Integration Attributes.....	539
Table 6-24. R5FSS0/1 Clocks and Resets.....	539
Table 6-25. R5FSS0/1 Hardware Requests.....	540
Table 6-26. MCU_R5FSS0 Special Features.....	545
Table 6-27. R5FSS0 Special Features.....	545
Table 6-28. R5FSS1 Special Features.....	546
Table 6-29. CPU Compare Block Operating Modes.....	549
Table 6-30. Compare Match Test Sequence.....	551
Table 6-31. Compare Mismatch Test Sequence.....	551
Table 6-32. Inactivity Monitor Block Signals.....	552
Table 6-33. Inactivity Monitor Block Operating Modes.....	553
Table 6-34. Inactivity Monitor Compare Mismatch Patterns.....	554
Table 6-35. RAM ID Map for ECC Aggregator (Per Core).....	561
Table 6-36. C66SS Modules Allocation within Device Domains.....	563
Table 6-37. C66SS Integration Attributes.....	567
Table 6-38. C66SS Clocks and Resets.....	567
Table 6-39. C66SS Hardware Requests.....	567
Table 6-40. C66SS ECC/Parity Support.....	571
Table 6-41. C66x CorePac Instances.....	574
Table 6-42. C66x CorePac Registers.....	574
Table 6-43. C66x RAT Instances.....	578
Table 6-44. C66x RAT Registers.....	578
Table 6-45. C66_RAT_PID Instances.....	579
Table 6-46. C66_RAT_PID Register Field Descriptions.....	579
Table 6-47. C66_RAT_CONFIG Instances.....	580
Table 6-48. C66_RAT_CONFIG Register Field Descriptions.....	580
Table 6-49. C66_RAT_CTRL_j Instances.....	581
Table 6-50. C66_RAT_CTRL_j Register Field Descriptions.....	581
Table 6-51. C66_RAT_BASE_j Instances.....	582
Table 6-52. C66_RAT_BASE_j Register Field Descriptions.....	582
Table 6-53. C66_RAT_TRANS_L_j Instances.....	583
Table 6-54. C66_RAT_TRANS_L_j Register Field Descriptions.....	583
Table 6-55. C66_RAT_TRANS_U_j Instances.....	584
Table 6-56. C66_RAT_TRANS_U_j Register Field Descriptions.....	584
Table 6-57. C66_RAT_DESTINATION_ID Instances.....	585
Table 6-58. C66_RAT_DESTINATION_ID Register Field Descriptions.....	585
Table 6-59. C66_RAT_EXCEPTION_LOGGING_CONTROL Instances.....	586
Table 6-60. C66_RAT_EXCEPTION_LOGGING_CONTROL Register Field Descriptions.....	586
Table 6-61. C66_RAT_EXCEPTION_LOGGING_HEADER0 Instances.....	587
Table 6-62. C66_RAT_EXCEPTION_LOGGING_HEADER0 Register Field Descriptions.....	587
Table 6-63. C66_RAT_EXCEPTION_LOGGING_HEADER1 Instances.....	588
Table 6-64. C66_RAT_EXCEPTION_LOGGING_HEADER1 Register Field Descriptions.....	588

Table 6-65. C66_RAT_EXCEPTION_LOGGING_DATA0 Instances.....	589
Table 6-66. C66_RAT_EXCEPTION_LOGGING_DATA0 Register Field Descriptions.....	589
Table 6-67. C66_RAT_EXCEPTION_LOGGING_DATA1 Instances.....	590
Table 6-68. C66_RAT_EXCEPTION_LOGGING_DATA1 Register Field Descriptions.....	590
Table 6-69. C66_RAT_EXCEPTION_LOGGING_DATA2 Instances.....	591
Table 6-70. C66_RAT_EXCEPTION_LOGGING_DATA2 Register Field Descriptions.....	591
Table 6-71. C66_RAT_EXCEPTION_LOGGING_DATA3 Instances.....	592
Table 6-72. C66_RAT_EXCEPTION_LOGGING_DATA3 Register Field Descriptions.....	592
Table 6-73. C66_RAT_EXCEPTION_PEND_SET Instances.....	593
Table 6-74. C66_RAT_EXCEPTION_PEND_SET Register Field Descriptions.....	593
Table 6-75. C66_RAT_EXCEPTION_PEND_CLEAR Instances.....	594
Table 6-76. C66_RAT_EXCEPTION_PEND_CLEAR Register Field Descriptions.....	594
Table 6-77. C66_RAT_EXCEPTION_ENABLE_SET Instances.....	595
Table 6-78. C66_RAT_EXCEPTION_ENABLE_SET Register Field Descriptions.....	595
Table 6-79. C66_RAT_EXCEPTION_ENABLE_CLEAR Instances.....	596
Table 6-80. C66_RAT_EXCEPTION_ENABLE_CLEAR Register Field Descriptions.....	596
Table 6-81. C66_RAT_EOI_REG Instances.....	597
Table 6-82. C66_RAT_EOI_REG Register Field Descriptions.....	597
Table 6-83. C71SS Allocation within Device Domains.....	598
Table 6-84. C71SS Integration Attributes.....	601
Table 6-85. C71SS Clocks and Resets.....	601
Table 6-86. RAM ID Map for C71x DSP ECC Aggregator.....	607
Table 6-87. GPU0 Integration Attributes.....	611
Table 6-88. GPU0 Clocks and Resets.....	611
Table 6-89. GPU0 Hardware Requests.....	612
Table 6-90. DECODER Modules Allocation within Device Domains.....	616
Table 6-91. DECODER Integration Attributes.....	619
Table 6-92. DECODER Clocks and Resets.....	619
Table 6-93. DECODER Hardware Requests.....	619
Table 6-94. ENCODER Modules Allocation within Device Domains.....	621
Table 6-95. ENCODER Integration Attributes.....	624
Table 6-96. ENCODER Clocks and Resets.....	624
Table 6-97. ENCODER Hardware Requests.....	624
Table 6-98. VPAC Allocation Across Device Domains.....	626
Table 6-99. VPAC Integration Attributes.....	630
Table 6-100. VPAC Clocks and Resets.....	631
Table 6-101. VPAC Hardware Requests.....	631
Table 6-102. VPAC Subsystem Interrupt Control Registers.....	634
Table 6-103. VPAC Subsystem VISS Interrupt Events.....	635
Table 6-104. VPAC Subsystem LDC Interrupt Events.....	636
Table 6-105. VPAC Subsystem MSC Interrupt Events.....	637
Table 6-106. VPAC Subsystem NF Interrupt Events.....	637
Table 6-107. VPAC Subsystem HTS Interrupt Events.....	637
Table 6-108. VPAC Subsystem UTC TR Complete Interrupt Events.....	638
Table 6-109. VPAC Subsystem UTC Error Interrupt Events.....	638
Table 6-110. VPAC Subsystem CTSET Interrupt Events.....	638
Table 6-111. VPAC Subsystem Data Formats Support.....	639
Table 6-112. VPAC Subsystem CTSET Event List.....	639
Table 6-113. VISS Data Formats.....	649
Table 6-114. VISS Configuration Error Interrupts Handling.....	651
Table 6-115. Table 3: RAM ECC capabilities.....	657
Table 6-116. RAWFE Interrupts.....	657
Table 6-117. RAWFE Debug events.....	658
Table 6-118. Figure 7:RAWFE Different merge modes.....	661
Table 6-119. Figure 14: RAWFE Sample DPC Table.....	663
Table 6-120. Figure 15:RAWFE LUT DPC Method(s).....	663
Table 6-121. RAWFE H3A Poxel Register Field Descriptions.....	672
Table 6-122. RAWFE H3A AE/AWB Window Register Field Descriptions.....	675
Table 6-123. RAWFE H3A AE/AWB Window With Additional Black Row Register Field Descriptions.....	676
Table 6-124. RAWFE H3A AF Packet Format With Vertical AF Enabled.....	677
Table 6-125. RAWFE H3A AE/AWB Packet Format for Sum of Square Mode.....	678

Table 6-126. RAWFE H3A AE/AWB Packet Format for Minimum-Maximum Mode.....	680
Table 6-127. RAWFE H3A AE/AWB Packet Format for Sum-Only Mode.....	682
Table 6-128. GLBCE Core Key Parameters.....	691
Table 6-129. GLBCE Memories.....	696
Table 6-130. Summary of HSX Spaces.....	700
Table 6-131. Summary of HSX Spaces.....	705
Table 6-132. Grey Scale and Saturation Calculation.....	705
Table 6-133. Edge Enhancer Programming Parameters.....	718
Table 6-134. LDC Interrupt Events.....	720
Table 6-135. LDC OBW Requirements.....	729
Table 6-136. LDC Supported Data Format Combinations.....	735
Table 6-137. LDC UYVY 8-bit Per Color Components.....	735
Table 6-138. LDC YCbCr 420 data stored 8-bit per color components.....	736
Table 6-139. Parameters of the Resizing (Downsampling) Vertical/Horizontal Filters.....	750
Table 6-140. Filter Mode and Coefficient Selection via VPAC_MSC_CORE_CFG_j Registers.....	750
Table 6-141. MSC Input/Output Data Formats.....	759
Table 6-142. YUV420 (2-plane 12-bit Fully Packed Format), First Plane.....	759
Table 6-143. YUV420 (2-plane 12-bit Fully Packed Format), Second Plane.....	759
Table 6-144. YUV420 (2-plane 8-bit Fully Packed Format), YUV 4:2:0 – NV12, First Plane.....	759
Table 6-145. YUV420 (2-plane 8-bit Fully Packed format), YUV 4:2:0 – NV12, Second Plane.....	760
Table 6-146. YUV420 (2-plane 12-bit Unpacked Format – LSB aligned), YUV 4:2:0 – NV12 (12-bit) (0x3D), First Plane....	760
Table 6-147. YUV420 (2-plane 12-bit Unpacked Format – LSB aligned), YUV 4:2:0 – NV12 (12-bit) (0x3D), Second Plane.....	760
Table 6-148. YUV420 (2-plane 12-bit Unpacked Format – MSB aligned), YUV 4:2:0 – NV12 (12-bit) (0x3D), First Plane....	760
Table 6-149. YUV420 (2-plane 12-bit Unpacked Format – MSB aligned), YUV 4:2:0 – NV12 (12-bit) (0x3D), Second Plane.....	761
Table 6-150. Interrupts.....	761
Table 6-151. Encoding for RSTATUS.....	761
Table 6-152. Encoding for WSTATUS.....	762
Table 6-153. Interrupts.....	766
Table 6-154. DMPAC Allocation Across Device Domains.....	779
Table 6-155. DMPAC Integration Attributes.....	782
Table 6-156. DMPAC Clocks and Resets.....	783
Table 6-157. DMPAC Hardware Requests.....	783
Table 6-158. DMPAC Input Resolution and Frame Rate Support for Stereo Processing.....	785
Table 6-159. DMPAC Input Resolution and Frame Rate Support for Optical Flow Processing.....	786
Table 6-160. DMPAC Input Pixel Data Format.....	786
Table 6-161. DMPAC Stereo Output Data Format.....	789
Table 6-162. DMPAC Optical Flow Output Data Format for Base Layer.....	792
Table 6-163. Flow Vector Map Output Packing Format for Non Base Layers.....	792
Table 6-164. DMPAC Interrupts Mapping for DMPAC0_INTD_0_SYSTEM_INTR_LEVEL_0.....	795
Table 6-165. DMPAC Interrupts Mapping for DMPAC0_INTD_0_SYSTEM_INTR_LEVEL_1.....	797
Table 6-166. DMPAC Configuration Interconnect FW Details.....	803
Table 6-167. DMPAC SL2 Interconnect FW and ISC Details.....	803
Table 6-168. DMPAC CTSET Events List.....	804
Table 6-169. DMPAC Programming Guide Terminology.....	806
Table 7-1. Mailbox Allocation Across Device Domains.....	814
Table 7-2. Mailbox Configuration Parameters.....	814
Table 7-3. Mailbox Integration Attributes.....	815
Table 7-4. Mailbox Clocks and Resets.....	815
Table 7-5. Mailbox Hardware Requests.....	815
Table 7-6. Interrupt Events.....	820
Table 7-7. Global Initialization of Surrounding Modules for MAILBOX.....	823
Table 7-8. Mailbox Global Initialization.....	823
Table 7-9. Sending a Message (Polling Method).....	823
Table 7-10. Sending a Message (Interrupt Method).....	823
Table 7-11. Receiving a Message (Polling Method).....	824
Table 7-12. Receiving a Message (Interrupt Method).....	824
Table 7-13. Events Servicing in Sending Mode.....	824
Table 7-14. Events Servicing in Receiving Mode.....	824
Table 7-15. Spinlock Allocation Across Device Domains.....	825
Table 7-16. Spinlock Integration Attributes.....	826

Table 7-17. Spinlock Clocks and Resets.....	826
Table 7-18. Spinlock Hardware Requests.....	826
Table 7-19. Global Initialization of Surrounding Modules.....	829
Table 7-20. Spinlock System Bug Recovery.....	829
Table 7-21. Register Call Summary.....	830
Table 7-22. Subprocess Call Summary.....	830
Table 8-1. MSMC Modules Allocation within Device Domains.....	832
Table 8-2. MSMC Integration Attributes.....	835
Table 8-3. MSMC Clocks and Resets.....	835
Table 8-4. MSMC Hardware Requests.....	835
Table 8-5. AND-OR Mask Way Selection.....	839
Table 8-6. Cache Way Grouping.....	839
Table 8-7. Cache Size Field Encodings.....	839
Table 8-8. Null Slave Response Status.....	840
Table 8-9. Starvation Bound Definitions.....	841
Table 8-10. MSMC Memory Regions.....	844
Table 8-11. QoS Threading.....	846
Table 8-12. DDRSS Modules Allocation within Device Domains.....	848
Table 8-13. DDRSS0 I/O signals.....	852
Table 8-14. DDRSS0 Integration Attributes.....	857
Table 8-15. DDRSS0 Clocks and Resets.....	857
Table 8-16. DDRSS0 Hardware Requests.....	857
Table 8-17. REGION_IDX and SDRAM_IDX Scenarios.....	862
Table 8-18. MSMC2DDR Bridge Events.....	864
Table 8-19. DDRSS0 Memory Regions.....	865
Table 8-20. Frequency Based Register Settings.....	880
Table 8-21. Chip Select Based and Frequency Based Register Settings.....	889
Table 8-22. PHY Write Level Delay Values.....	896
Table 8-23. Port Table of Accesses.....	904
Table 8-24. Port Routing Rules.....	906
Table 8-25. Blocked Paths.....	906
Table 8-26. VirtSS Global Parameters.....	907
Table 8-27. PAT Parameters.....	907
Table 8-28. PVU Parameters.....	907
Table 8-29. TBU Parameters.....	908
Table 8-30. TCU Parameters.....	908
Table 8-31. ECC Aggregator Parameters.....	908
Table 8-32. Transaction Options.....	909
Table 8-33. Address Translation Resource Summary.....	911
Table 8-34. PVU Configuration Parameters.....	916
Table 8-35. PVU Integration Attributes.....	918
Table 8-36. PVU Clocks and Resets.....	918
Table 8-37. PVU Hardware Requests.....	919
Table 8-38. Page Attributes to Bus Memory Attributes Mapping.....	922
Table 8-39. PAT Allocation Within Device Domains.....	924
Table 8-40. PAT Configuration Parameters.....	925
Table 8-41. PAT Integration Attributes.....	926
Table 8-42. PAT Clocks and Resets.....	926
Table 8-43. PAT Hardware Requests.....	927
Table 8-44. Index Bits and Final Address vs Page Size.....	928
Table 8-45. Device Modules and Subsystems with RAT Module.....	930
Table 8-46. RAT Clocks and Resets.....	930
Table 8-47. RAT Hardware Requests.....	930
Table 8-48. RAT Source ID Mapping.....	931
Table 9-1. GIC Module Allocation within Device Domains.....	936
Table 9-2. GIC Configuration Summary.....	937
Table 9-3. GIC Integration Attributes.....	940
Table 9-4. GIC Clocks and Resets.....	940
Table 9-5. GIC Hardware Requests.....	940
Table 9-6. GIC Interrupt Outputs.....	942
Table 9-7. CLEC Integration Attributes.....	946

Table 9-8. CLEC Clocks and Resets.....	946
Table 9-9. CLEC Hardware Requests.....	946
Table 9-10. CLEC Output Event Map.....	950
Table 9-11. CLEC Input Event Map.....	950
Table 9-12. CLEC ESM Event Output Map.....	951
Table 9-13. CLEC C7x DSP Input Event Map.....	952
Table 9-14. INTRTR Modules Configuration.....	954
Table 9-15. WKUP_GPIOMUX_INTRTR0 Integration Attributes.....	957
Table 9-16. WKUP_GPIOMUX_INTRTR0 Clocks and Resets.....	957
Table 9-17. WKUP_GPIOMUX_INTRTR0 Hardware Requests.....	957
Table 9-18. GPIOMUX_INTRTR0 Integration Attributes.....	958
Table 9-19. GPIOMUX_INTRTR0 Clocks and Resets.....	958
Table 9-20. GPIOMUX_INTRTR0 Hardware Requests.....	958
Table 9-21. MAIN2MCU_LVL_INTRTR0 Integration Attributes.....	960
Table 9-22. MAIN2MCU_LVL_INTRTR0 Clocks and Resets.....	960
Table 9-23. MAIN2MCU_LVL_INTRTR0 Hardware Requests.....	960
Table 9-24. MAIN2MCU_PLS_INTRTR0 Integration Attributes.....	961
Table 9-25. MAIN2MCU_PLS_INTRTR0 Clocks and Resets.....	961
Table 9-26. MAIN2MCU_PLS_INTRTR0 Hardware Requests.....	961
Table 9-27. C66SS0_INTRTR0 Integration Attributes.....	962
Table 9-28. C66SS0_INTRTR0 Clocks and Resets.....	962
Table 9-29. C66SS0_INTRTR0 Hardware Requests.....	962
Table 9-30. C66SS1_INTRTR0 Integration Attributes.....	963
Table 9-31. C66SS1_INTRTR0 Clocks and Resets.....	963
Table 9-32. C66SS1_INTRTR0 Hardware Requests.....	963
Table 9-33. R5FSS0_INTRTR0 Integration Attributes.....	964
Table 9-34. R5FSS0_INTRTR0 Clocks and Resets.....	964
Table 9-35. R5FSS0_INTRTR0 Hardware Requests.....	964
Table 9-36. R5FSS1_INTRTR0 Integration Attributes.....	965
Table 9-37. R5FSS1_INTRTR0 Clocks and Resets.....	965
Table 9-38. R5FSS1_INTRTR0 Hardware Requests.....	965
Table 9-39. WKUP_DMSC0 Interrupt Map.....	966
Table 9-40. WKUP_GPIOMUX_INTRTR0 Interrupt Map.....	968
Table 9-41. WKUP_ESM0 Interrupt Map.....	971
Table 9-42. MCU_R5FSS0_CORE0 Interrupt Map.....	973
Table 9-43. MCU_R5FSS0_CORE1 Interrupt Map.....	981
Table 9-44. MCU_ESM0 Interrupt Map.....	989
Table 9-45. GIC500 PPI Interrupt Map.....	991
Table 9-46. GIC500 SPI Interrupt Map.....	992
Table 9-47. R5FSS0_CORE0 Interrupt Map.....	1009
Table 9-48. R5FSS0_CORE1 Interrupt Map.....	1019
Table 9-49. R5FSS1_CORE0 Interrupt Map.....	1029
Table 9-50. R5FSS1_CORE1 Interrupt Map.....	1039
Table 9-51. R5FSS0_INTRTR0 Interrupt Map.....	1049
Table 9-52. R5FSS1_INTRTR0 Interrupt Map.....	1057
Table 9-53. C66SS0 Interrupt Map.....	1065
Table 9-54. C66SS1 Interrupt Map.....	1068
Table 9-55. C66SS0_INTRTR0 Interrupt Map.....	1071
Table 9-56. C66SS1_INTRTR0 Interrupt Map.....	1080
Table 9-57. PRU-ICSSG0 Interrupt Map.....	1089
Table 9-58. PRU-ICSSG1 Interrupt Map.....	1092
Table 9-59. MAIN2MCU_LVL_INTRTR0 Interrupt Map.....	1095
Table 9-60. MAIN2MCU_PLS_INTRTR0 Interrupt Map.....	1101
Table 9-61. GPIOMUX_INTRTR0 Interrupt Map.....	1104
Table 9-62. ESM0 Interrupt Map.....	1109
Table 10-1. Channel Classes.....	1126
Table 10-2. UDMA Descriptor Types and Attributes.....	1129
Table 10-3. Host Packet Descriptor Packet Information Word 0 (PD Word 0).....	1131
Table 10-4. Host Packet Descriptor Packet Information Word 1 (PD Word 1).....	1132
Table 10-5. Host Packet Descriptor Packet Information Word 2 (PD Word 2).....	1132
Table 10-6. Host Packet Descriptor Packet Information Word 3 (PD Word 3).....	1132



Table 10-7. Host Packet Descriptor Linking Word 0 (PD Word 4).....	1133
Table 10-8. Host Packet Descriptor Linking Word 1 (PD Word 5).....	1133
Table 10-9. Host Packet Descriptor Buffer 0 Info Word 0 (PD Word 6).....	1133
Table 10-10. Host Packet Descriptor Buffer 0 Info Word 1 (PD Word 7).....	1133
Table 10-11. Host Packet Descriptor Buffer 0 Info Word 2 (PD Word 8).....	1133
Table 10-12. Host Packet Descriptor Original Buffer Info Word 0 (PD Word 9).....	1133
Table 10-13. Host Packet Descriptor Original Buffer Info Word 1 (PD Word 10).....	1134
Table 10-14. Host Packet Descriptor Original Buffer Info Word 2 (PD Word 11).....	1134
Table 10-15. Host Packet Descriptor Extended Packet Info Block Word 0 (Optional).....	1134
Table 10-16. Host Packet Descriptor Extended Packet Info Block Word 1 (Optional).....	1134
Table 10-17. Host Packet Descriptor Extended Packet Info Block Word 2 (Optional).....	1134
Table 10-18. Host Packet Descriptor Extended Packet Info Block Word 3 (Optional).....	1134
Table 10-19. Host Packet Descriptor Protocol Specific Word N (Optional).....	1135
Table 10-20. Host Buffer Descriptor Reserved Word 0 (BD Word 0).....	1135
Table 10-21. Host Buffer Descriptor Reserved Word 1 (BD Word 1).....	1135
Table 10-22. Host Buffer Descriptor Buffer Reclamation Info (BD Word 2).....	1135
Table 10-23. Host Buffer Descriptor Reserved Word 3 (BD Word 3).....	1136
Table 10-24. Host Buffer Descriptor Linking Word 0 (BD Word 4).....	1136
Table 10-25. Host Buffer Descriptor Linking Word 1 (BD Word 5).....	1136
Table 10-26. Host Buffer Descriptor Buffer N Info Word 1 (BD Word 6).....	1136
Table 10-27. Host Buffer Descriptor Buffer N Info Word 1 (BD Word 7).....	1136
Table 10-28. Host Buffer Descriptor Buffer N Info Word 2 (BD Word 8).....	1137
Table 10-29. Host Buffer Descriptor Original Buffer Info Word 0 (BD Word 9).....	1137
Table 10-30. Host Buffer Descriptor Original Buffer Info Word 1 (BD Word 10).....	1137
Table 10-31. Host Buffer Descriptor Original Buffer Info Word 2 (BD Word 11).....	1137
Table 10-32. Monolithic Packet Descriptor Word 0.....	1138
Table 10-33. Monolithic Packet Descriptor Word 1.....	1138
Table 10-34. Monolithic Packet Descriptor Word 2.....	1139
Table 10-35. Monolithic Packet Descriptor Word 3.....	1139
Table 10-36. Monolithic Extended Packet Info Word 0 (Optional).....	1140
Table 10-37. Monolithic Extended Packet Info Word 1 (Optional).....	1140
Table 10-38. Monolithic Extended Packet Info Word 2 (Optional).....	1140
Table 10-39. Monolithic Extended Packet Info Word 3 (Optional).....	1140
Table 10-40. Monolithic Packet Descriptor Protocol Specific Word M (Optional).....	1140
Table 10-41. Monolithic Packet Descriptor Payload Data Words 0-N.....	1140
Table 10-42. Transfer Request Packet Descriptor Word 0.....	1141
Table 10-43. Transfer Request Packet Descriptor Word 1.....	1141
Table 10-44. Transfer Request Packet Descriptor Word 2.....	1142
Table 10-45. Transfer Request Packet Descriptor Word 3.....	1142
Table 10-46. Transfer Request Packet Descriptor Payload Words 0-N.....	1143
Table 10-47. Terms and Definitions.....	1143
Table 10-48. Addressing Parameters for a Basic Stream.....	1144
Table 10-49. Transfer Request Template.....	1145
Table 10-50. Transfer Request Fields.....	1145
Table 10-51. Transfer Request Template FLAGS Field.....	1146
Table 10-52. FLAGS Field Descriptions.....	1146
Table 10-53. Transfer Request Minimum Size Type 0 One Dimensional Transfer.....	1148
Table 10-54. Transfer Request Minimum Size Type 1 Two Dimensional Transfer.....	1148
Table 10-55. Transfer Request Minimum Size Type 2 Three Dimensional Transfer.....	1148
Table 10-56. Transfer Request Minimum Size Type 3 Four Dimensional Transfer.....	1148
Table 10-57. Transfer Request Minimum Size Type 4 Four Dimensional Transfer with Formatting.....	1148
Table 10-58. Transfer Request Minimum Size Type 5 Cache Warm.....	1148
Table 10-59. Transfer Request Minimum Size Type 8 Four Dimensional Block Copy.....	1148
Table 10-60. Transfer Request Minimum Size Type 9 Four Dimensional Block Copy with Repacking.....	1148
Table 10-61. Transfer Request Minimum Size Type 10 Two Dimensional Block Copy.....	1148
Table 10-62. Transfer Request Minimum Size Type 11 Two Dimensional Block Copy with Repacking.....	1148
Table 10-63. Transfer Request Minimum Size Type 15 Four Dimensional Block Copy with Repacking and Indirection Support.....	1149
Table 10-64. Event Size Encoding.....	1149
Table 10-65. Configuration Specific Flags.....	1150
Table 10-66. ADDR Field Format.....	1152

Table 10-67. DADDR Field Format.....	1152
Table 10-68. Transfer Request Template FMTFLAGS Field.....	1153
Table 10-69. FMTFLAGS Field Descriptions.....	1153
Table 10-70. ELTYPE Encoding.....	1155
Table 10-71. DFMT Encoding.....	1155
Table 10-72. Secondary Transfer Request Format.....	1156
Table 10-73. Secondary Transfer Request Template for 64-Byte Secondary TR.....	1156
Table 10-74. SECONDARY TR FLAGS Field Definition.....	1156
Table 10-75. FLAGS Field Descriptions.....	1157
Table 10-76. SECONDARY TR FLAGS Field Definition.....	1158
Table 10-77. Multiple Buffer Interleave FLAGS Field Descriptions.....	1158
Table 10-78. Multiple Buffer Interleave Secondary Transfer Request Template 128 Byte Secondary TR.....	1158
Table 10-79. Transfer Request Template FMTFLAGS AMODE SPECIFC Circular Addressing Field.....	1159
Table 10-80. Circular Address Mode Specific Flags Field Descriptions.....	1160
Table 10-81. Circular Block Size Decoding.....	1160
Table 10-82. AMX Address Mode Selection Encoding.....	1160
Table 10-83. Transfer Request Template CACHEFLAGS Field.....	1161
Table 10-84. FMTFLAGS Field Descriptions.....	1161
Table 10-85. Transfer Request Response Template STATUS FLAGS Field.....	1161
Table 10-86. STATUS FLAGS Field Descriptions.....	1161
Table 10-87. STATUS_TYPE Encoded Values.....	1162
Table 10-88. Submission Error STATUS_INFO Encodings.....	1162
Table 10-89. Unsupported Feature STATUS_INFO Encodings.....	1162
Table 10-90. Transfer Exception STATUS_INFO Encodings.....	1163
Table 10-91. External Transmit Channel Setup Sequence.....	1167
Table 10-92. External Transmit Channel Teardown Sequence.....	1167
Table 10-93. UDMA Tx Error/Exception Handling.....	1174
Table 10-94. UDMA Rx Error/Exception Handling.....	1181
Table 10-95. UTC Rx Error/Exception Handling.....	1184
Table 10-96. NAVSS0 Integration Attributes.....	1191
Table 10-97. NAVSS0 Clocks and Resets.....	1192
Table 10-98. NAVSS0 Hardware Requests.....	1192
Table 10-99. NAVSS Interrupt Router Input Mapping.....	1195
Table 10-100. Global Event Map.....	1196
Table 10-101. System Thread Map for All PSI-L Threads.....	1197
Table 10-102. VBUSM Route IDs.....	1197
Table 10-103. NAVSS Parameters.....	1200
Table 10-104. NAVSS Software Variables.....	1200
Table 10-105. NAVSS Parameters for the Application Example.....	1201
Table 10-106. NAVSS Software Variables for the Application Example.....	1202
Table 10-107. NAVSS Register Writes for the Application Example.....	1202
Table 10-108. MCU NAVSS Integration Attributes.....	1206
Table 10-109. MCU NAVSS Clocks and Resets.....	1206
Table 10-110. MCU NAVSS Hardware Requests.....	1207
Table 10-111. Interrupt Router Input Mapping.....	1207
Table 10-112. Interrupt Aggregator Parameters.....	1207
Table 10-113. UDMA Configuration Parameters.....	1208
Table 10-114. RINGACC Configuration Parameters.....	1208
Table 10-115. RINGACC Ring Mapping.....	1208
Table 10-116. MSRAM Configuration Parameters.....	1209
Table 10-117. Proxy Configuration Parameters.....	1209
Table 10-118. Secure Proxy Configuration Parameters.....	1209
Table 10-119. MCU NAVSS VBUSM Route IDs.....	1210
Table 10-120. UDMA Allocation Across Device Domains.....	1212
Table 10-121. UDMA Configuration Parameters.....	1215
Table 10-122. UDMA Integration Attributes.....	1216
Table 10-123. UDMA Clocks and Resets.....	1216
Table 10-124. UDMA Hardware Requests.....	1217
Table 10-125. Operational States.....	1221
Table 10-126. Internal Tx Channel Allocation.....	1222
Table 10-127. Rx Channel Allocation.....	1222

Table 10-128. Inbound Event Map.....	1223
Table 10-129. Emulation Behavior Control.....	1224
Table 10-130. RINGACC Allocation Across Device Domains.....	1228
Table 10-131. RINGACC Configuration Parameters.....	1229
Table 10-132. MSRAM Configuration Parameters.....	1229
Table 10-133. RINGACC Integration Attributes.....	1230
Table 10-134. RINGACC Ring Mapping.....	1230
Table 10-135. RINGACC Clocks and Resets.....	1230
Table 10-136. RINGACC Hardware Requests.....	1231
Table 10-137. Ring Memory Partition.....	1235
Table 10-138. Trace Message for a Push.....	1239
Table 10-139. Trace Message for a Pop.....	1239
Table 10-140. Trace Message for a Peek.....	1239
Table 10-141. Proxy Allocation Across Device Domains.....	1241
Table 10-142. Proxy Configuration Parameters.....	1241
Table 10-143. Proxy Integration Attributes.....	1243
Table 10-144. Proxy Clocks and Resets.....	1243
Table 10-145. Proxy Hardware Requests.....	1243
Table 10-146. RINGACC Offsets per Ring/Queue.....	1244
Table 10-147. Secure Proxy Allocation Across Device Domains.....	1247
Table 10-148. Secure Proxy Configuration Parameters.....	1247
Table 10-149. Secure Proxy Integration Attributes.....	1248
Table 10-150. Proxy Clocks and Resets.....	1248
Table 10-151. Proxy Hardware Requests.....	1248
Table 10-152. RING_ACC Offsets per Ring/Queue.....	1250
Table 10-153. INTR_AGGR Allocation Across Device Domains.....	1254
Table 10-154. Interrupt Aggregators Parameters.....	1255
Table 10-155. INTR_AGGR Integration Attributes.....	1257
Table 10-156. INTR_AGGR Clocks and Resets.....	1257
Table 10-157. INTR_AGGR Hardware Requests.....	1257
Table 10-158. NAVSS0_UDMASS_INTR_AGGR0 LEVI and L2G Local Interrupts.....	1258
Table 10-159. Types of PSI-L Threads.....	1264
Table 10-160. PSILSS Modules Allocation within Device Domains.....	1267
Table 10-161. PSILSS Endpoint Thread Map.....	1268
Table 10-162. PDMA_USART_PSILSS0 Configuration Parameters.....	1269
Table 10-163. PDMA_MISC_PSILSS0 Configuration Parameters.....	1269
Table 10-164. PDMA_DEBUG_PSILSS0 Configuration Parameters.....	1269
Table 10-165. PDMA_AASRC_PSILSS0 Configuration Parameters.....	1269
Table 10-166. CSI_PSILSS0 Configuration Parameters.....	1269
Table 10-167. NB0 Configuration.....	1272
Table 10-168. NB1 Configuration.....	1272
Table 10-169. PDMA Allocation Across Device Domains.....	1276
Table 10-170. MCU Domain PDMA Integration Attributes.....	1284
Table 10-171. MCU Domain PDMA Clocks and Resets.....	1284
Table 10-172. MAIN Domain PDMA Integration Attributes.....	1285
Table 10-173. MAIN Domain PDMA Clocks and Resets.....	1285
Table 10-174. MAIN Domain PDMA Hardware Requests.....	1286
Table 10-175. Tx Channel Out-of-Band Signals.....	1290
Table 10-176. Mapping of MCAN Filter Events to MCAN Channels.....	1295
Table 10-177. Ping-Pong Buffer Example in MCAN Mode.....	1295
Table 10-178. MCU_PDMA_MISC_G0 Tx Channel Allocation.....	1297
Table 10-179. MCU_PDMA_MISC_G0 Rx Channel Allocation.....	1297
Table 10-180. MCU_PDMA_MISC_G1 Tx Channel Allocation.....	1297
Table 10-181. MCU_PDMA_MISC_G1 Rx Channel Allocation.....	1297
Table 10-182. MCU_PDMA_MISC_G2 Tx Channel Allocation.....	1298
Table 10-183. MCU_PDMA_MISC_G2 Rx Channel Allocation.....	1298
Table 10-184. MCU_PDMA_ADC Rx Channel Allocation.....	1298
Table 10-185. PDMA_AASRC Tx Channel Allocation.....	1299
Table 10-186. PDMA_AASRC Tx Events.....	1299
Table 10-187. PDMA_AASRC Rx Channel Allocation.....	1299
Table 10-188. PDMA_AASRC Rx Events.....	1299

Table 10-189. PDMA_DEBUG_CCMCU Rx Channel Allocation.....	1299
Table 10-190. PDMA_DEBUG_C66 Rx Channel Allocation.....	1300
Table 10-191. PDMA_MCAN Tx Channel Allocation.....	1300
Table 10-192. PDMA_MCAN Rx Channel Allocation.....	1300
Table 10-193. PDMA_MCASP_G0 Tx Channel Allocation.....	1301
Table 10-194. PDMA_MCASP_G0 Rx Channel Allocation.....	1301
Table 10-195. PDMA_MCASP_G1 Tx Channel Allocation.....	1302
Table 10-196. PDMA_MCASP_G1 Rx Channel Allocation.....	1302
Table 10-197. PDMA_MISC_G0 Tx Channel Allocation.....	1302
Table 10-198. PDMA_MISC_G0 Rx Channel Allocation.....	1302
Table 10-199. PDMA_MISC_G1 Tx Channel Allocation.....	1303
Table 10-200. PDMA_MISC_G1 Rx Channel Allocation.....	1303
Table 10-201. PDMA_MISC_G2 Tx Channel Allocation.....	1303
Table 10-202. PDMA_MISC_G2 Rx Channel Allocation.....	1304
Table 10-203. PDMA_MISC_G3 Tx Channel Allocation.....	1304
Table 10-204. PDMA_MISC_G3 Rx Channel Allocation.....	1304
Table 10-205. PDMA_USART_G0 Tx Channel Allocation.....	1305
Table 10-206. PDMA_USART_G0 Rx Channel Allocation.....	1305
Table 10-207. PDMA_USART_G1 Tx Channel Allocation.....	1305
Table 10-208. PDMA_USART_G1 Rx Channel Allocation.....	1305
Table 10-209. PDMA_USART_G2 Tx Channel Allocation.....	1305
Table 10-210. PDMA_USART_G2 Rx Channel Allocation.....	1306
Table 10-211. DRU Modules Allocation within Device Domains.....	1307
Table 10-212. DRU Integration Attributes.....	1310
Table 10-213. DRU Clocks and Resets.....	1311
Table 10-214. DRU Hardware Requests.....	1311
Table 10-215. EVENT_SIZE Encoding.....	1317
Table 10-216. Differences between DRU Configurations.....	1319
Table 11-1. CPTS Allocation Across Device Domains.....	1321
Table 11-2. CPTS Integration Attributes.....	1324
Table 11-3. CPTS Clocks and Resets.....	1324
Table 11-4. CPTS Hardware Requests.....	1324
Table 11-5. TS_ADD_VAL.....	1327
Table 11-6. TIMER_MGR Allocation Across Device Domains.....	1332
Table 11-7. Timer Manager Integration Attributes.....	1334
Table 11-8. Timer Manager Clocks and Resets.....	1334
Table 11-9. Timer Manager Hardware Requests.....	1335
Table 11-10. Global Initialization of Surrounding Modules.....	1338
Table 11-11. Network Time Synchronization Functions in the SoC.....	1340
Table 11-12. TIMESYNC_INTRTR0 and CMPEVT_INTRTR0 Configuration.....	1342
Table 11-13. TIMESYNC_INTRTR0 Integration Attributes.....	1343
Table 11-14. TIMESYNC_INTRTR0 Clocks and Resets.....	1343
Table 11-15. TIMESYNC_INTRTR0 Hardware Requests.....	1344
Table 11-16. CMPEVT_INTRTR0 Integration Attributes.....	1346
Table 11-17. CMPEVT_INTRTR0 Clocks and Resets.....	1346
Table 11-18. CMPEVT_INTRTR0 Hardware Requests.....	1346
Table 11-19. TIMESYNC_INTRTR0 Instances.....	1350
Table 11-20. TIMESYNC_INTRTR0 Registers.....	1350
Table 11-21. TIMESYNC_INTRTR0_PID Instances.....	1351
Table 11-22. TIMESYNC_INTRTR0_PID Register Field Descriptions.....	1351
Table 11-23. TIMESYNC_INTRTR0_MUXCNTL_y Instances.....	1352
Table 11-24. TIMESYNC_INTRTR0_MUXCNTL_y Register Field Descriptions.....	1352
Table 11-25. CMPEVT_INTRTR0 Instances.....	1353
Table 11-26. CMPEVT_INTRTR0 Registers.....	1353
Table 11-27. CMPEVT_INTRTR0_PID Instances.....	1354
Table 11-28. CMPEVT_INTRTR0_PID Register Field Descriptions.....	1354
Table 11-29. CMPEVT_INTRTR0_MUXCNTL_y Instances.....	1355
Table 11-30. CMPEVT_INTRTR0_MUXCNTL_y Register Field Descriptions.....	1355
Table 11-31. CMPEVT_INTRTR0 Event Map.....	1356
Table 11-32. TIMESYNC_INTRTR0 Event Map.....	1359
Table 11-33. PRU_ICSSG0 Sync Event Map.....	1361

Table 11-34. PRU_ICSSG1 Sync Event Map.....	1361
Table 11-35. NAVSS0 Sync Event Map.....	1362
Table 11-36. PCIE0 Sync Event Map.....	1363
Table 11-37. PCIE1 Sync Event Map.....	1363
Table 11-38. PCIE2 Sync Event Map.....	1363
Table 11-39. PCIE3 Sync Event Map.....	1363
Table 11-40. MCU_CPSW0 Sync Event Map.....	1364
Table 11-41. CPSW0 Sync Event Map.....	1364
Table 11-42. I/O Sync Event Map.....	1365
Table 12-1. ADC Allocation Across Device Domains.....	1367
Table 12-2. ADC I/O Signals.....	1369
Table 12-3. MCU_ADC[0-1] Integration Attributes.....	1374
Table 12-4. MCU_ADC[0-1] Clocks and Resets.....	1374
Table 12-5. MCU_ADC[0-1] Hardware Requests.....	1374
Table 12-6. Global Initialization of Surrounding Modules.....	1384
Table 12-7. General Programming Model.....	1384
Table 12-8. Turn-Off.....	1385
Table 12-9. Turn-On After Turn-Off.....	1385
Table 12-10. GPIO Allocation Across Device Domains.....	1386
Table 12-11. GPIO Instance Multiplexing.....	1391
Table 12-12. GPIO I/O Signals.....	1392
Table 12-13. WKUP_GPIO[0-1] Integration Attributes.....	1394
Table 12-14. WKUP_GPIO[0-1] Clocks and Resets.....	1394
Table 12-15. WKUP_GPIO[0-1] Hardware Requests.....	1395
Table 12-16. GPIO[0-7] Integration Attributes <sup>(1)</sup> .....	1397
Table 12-17. GPIO[0-7] Clocks and Resets <sup>(1)</sup> .....	1398
Table 12-18. GPIO[0-7] Hardware Requests.....	1399
Table 12-19. GPIO Interrupt and DMA Event Configuration Options.....	1403
Table 12-20. Global Initialization of Surrounding Modules.....	1406
Table 12-21. GPIO Global Initialization.....	1406
Table 12-22. GPIO Read Input Register.....	1407
Table 12-23. GPIO Set Bit Function.....	1407
Table 12-24. GPIO Clear Bit Function.....	1407
Table 12-25. I2C Allocation Across Device Domains.....	1408
Table 12-26. I2C I/O Signals.....	1412
Table 12-27. WKUP_I2C0 Integration Attributes.....	1418
Table 12-28. WKUP_I2C0 Clocks and Resets <sup>(1)</sup> .....	1418
Table 12-29. WKUP_I2C0 Hardware Requests.....	1418
Table 12-30. MCU_I2C[0-1] Integration Attributes.....	1419
Table 12-31. MCU_I2C[0-1] Clocks and Resets <sup>(1)</sup> .....	1420
Table 12-32. MCU_I2C[0-1] Hardware Requests.....	1420
Table 12-33. I2C[0-6] Integration Attributes.....	1422
Table 12-34. I2C[0-6] Clocks and Resets.....	1423
Table 12-35. I2C[0-6] Hardware Requests.....	1424
Table 12-36. I <sup>2</sup> C Operation Mode Selection.....	1426
Table 12-37. I <sup>2</sup> C t <sub>LOW</sub> and t <sub>high</sub> Values of the I <sup>2</sup> C Clock.....	1427
Table 12-38. I2C Register Values for Maximum I2C Bit Rates in I2C F/S, I2C HS Modes.....	1428
Table 12-39. I2C Local Power-Management Features.....	1429
Table 12-40. I2C Clock Activity Settings.....	1429
Table 12-41. I2C Events.....	1430
Table 12-42. I2C List of Tests.....	1434
Table 12-43. I3C Allocation Across Device Domains.....	1445
Table 12-44. I3C I/O Signals.....	1448
Table 12-45. MCU_I3C[0-1] Integration Attributes.....	1450
Table 12-46. MCU_I3C[0-1] Clocks and Resets.....	1450
Table 12-47. MCU_I3C[0-1] Hardware Requests.....	1450
Table 12-48. I3C0 Integration Attributes.....	1452
Table 12-49. I3C0 Clocks and Resets.....	1452
Table 12-50. I3C0 Hardware Requests.....	1453
Table 12-51. I <sup>3</sup> C Operation Mode Selection.....	1454
Table 12-52. Achievable SCL Frequencies.....	1455



Table 12-53. Retaining Registers Map Organization.....	1458
Table 12-54. CCC Commands.....	1461
Table 12-55. SIR Map Organization.....	1462
Table 12-56. Payload Setting in SDR Mode.....	1466
Table 12-57. MCSPI Allocation Across Device Domains.....	1472
Table 12-58. MCSPI I/O Signals (Master Mode).....	1476
Table 12-59. MCSPI I/O Signals (Slave Mode).....	1477
Table 12-60. MCSPI Controller Clock Rates.....	1481
Table 12-61. Phase and Polarity Combinations.....	1481
Table 12-62. MCU_MCSPI Integration Attributes.....	1488
Table 12-63. MCU_MCSPI Clocks and Resets.....	1488
Table 12-64. MCU_MCSPI Hardware Requests.....	1489
Table 12-65. MCSPI Integration Attributes.....	1492
Table 12-66. MCSPI Clocks and Resets.....	1493
Table 12-67. MCSPI Hardware Requests.....	1494
Table 12-68. MCSPI Controller Clock Rates.....	1505
Table 12-69. CLKSPIO High/Low Time Computation.....	1505
Table 12-70. Clock Granularity Examples.....	1506
Table 12-71. FIFO Writes, Word Length Relationship.....	1511
Table 12-72. Global Initialization of Surrounding Modules.....	1520
Table 12-73. MCSPI Global Initialization.....	1520
Table 12-74. MCSPI Receive Mode Initialization.....	1520
Table 12-75. MCSPI Transmit Mode Initialization.....	1521
Table 12-76. MCSPI Transmit-and-Receive Mode Initialization.....	1521
Table 12-77. Common Transfer Sequence (Main Process).....	1521
Table 12-78. End of Transfer Sequences.....	1522
Table 12-79. Transmit-and-Receive (Controller and Peripheral) (Main Process).....	1522
Table 12-80. Transmit-and-Receive (Controller and Peripheral) (Interrupt Routine).....	1522
Table 12-81. Transmit-Only With Interrupts (Controller and Peripheral) (Main Process).....	1523
Table 12-82. Transmit-Only With Interrupts (Controller and Peripheral) (Interrupt Routine).....	1523
Table 12-83. Transmit-Only With DMA (Controller and Peripheral) (Main Process).....	1523
Table 12-84. Transmit-Only With DMA (Controller and Peripheral) (Interrupt Routine).....	1523
Table 12-85. Receive-Only With Interrupt (Controller Normal) (Main Process).....	1524
Table 12-86. Receive-Only With Interrupt (Controller Normal) (Interrupt Routine).....	1524
Table 12-87. Receive-Only With DMA (Controller Normal) (Main Process).....	1524
Table 12-88. Receive-Only With DMA (Controller Normal) (Interrupt Routine).....	1524
Table 12-89. Receive-Only With Interrupt (Controller Turbo) (Main Process).....	1524
Table 12-90. Receive-Only With Interrupt (Controller Turbo) (Interrupt Routine).....	1525
Table 12-91. Receive-Only With DMA (Controller Turbo) (Main Process).....	1525
Table 12-92. Receive-Only With DMA (Controller Turbo) (Interrupt Routine).....	1525
Table 12-93. Receive-Only (Peripheral) (Main Process).....	1526
Table 12-94. Receive-Only (Peripheral) (Interrupt Routine).....	1526
Table 12-95. FIFO Mode Common Sequence (Controller) (Main Process).....	1526
Table 12-96. End of Transfer Sequences in FIFO Mode.....	1527
Table 12-97. Receive-Only Procedure – Polling Method.....	1532
Table 12-98. Receive-Only Procedure – Interrupt Method.....	1533
Table 12-99. Transmit-Only Procedure – Polling Method.....	1533
Table 12-100. Transmit-and-Receive Procedure – Polling Method.....	1533
Table 12-101. UART Allocation Across Device Domains.....	1534
Table 12-102. UART I/O Signals.....	1538
Table 12-103. UART I/O Signals (RS-485 Mode).....	1540
Table 12-104. UART I/O Signals (IrDA Mode).....	1542
Table 12-105. UART_EFR[1-0] IR Address Checking Options.....	1545
Table 12-106. 4-PPM Format.....	1548
Table 12-107. FIR Preamble, Start Flag, and Stop Flag.....	1548
Table 12-108. FIR Data Byte Transmission Order Example.....	1549
Table 12-109. UART I/O Signals (CIR Mode).....	1550
Table 12-110. WKUP_UART0 Integration Attributes.....	1557
Table 12-111. WKUP_UART0 Clocks and Resets.....	1557
Table 12-112. WKUP_UART0 Hardware Requests.....	1557
Table 12-113. MCU_UART0 Integration Attributes.....	1559

Table 12-114. MCU_UART0 Clocks and Resets.....	1559
Table 12-115. MCU_UART0 Hardware Requests.....	1559
Table 12-116. UART Integration Attributes.....	1563
Table 12-117. UART Clocks and Resets.....	1563
Table 12-118. UART Hardware Requests.....	1564
Table 12-119. UART Local Power-Management Features.....	1570
Table 12-120. UART Mode Interrupts.....	1570
Table 12-121. IrDA Mode Interrupts.....	1571
Table 12-122. CIR Mode Interrupts.....	1572
Table 12-123. UART TX FIFO Trigger Level Setting Summary.....	1574
Table 12-124. UART RX FIFO Trigger-Level Setting Summary.....	1575
Table 12-125. UART Register Access Mode Programming (Using UART_LCR).....	1584
Table 12-126. UART Subconfiguration Mode A Summary.....	1585
Table 12-127. UART Subconfiguration Mode B Summary.....	1585
Table 12-128. UART Suboperational Mode Summary.....	1585
Table 12-129. UART Register Access Mode Overview.....	1585
Table 12-130. UART Mode Selection.....	1586
Table 12-131. UART Mode Register Overview.....	1586
Table 12-132. IrDA Mode Register Overview.....	1588
Table 12-133. CIR Mode Register Overview.....	1589
Table 12-134. UART Baud Rate Settings (48-MHz Clock).....	1590
Table 12-135. UART Parity Bit Encoding.....	1591
Table 12-136. UART_EFR[3:0] Software Flow Control Options.....	1592
Table 12-137. IrDA Baud Rate Settings.....	1597
Table 12-138. Details of address matching.....	1600
Table 12-139. CIR Duty Cycle.....	1604
Table 12-140. UART Global Initialization of Surrounding Modules.....	1606
Table 12-141. UART Global Initialization.....	1607
Table 12-142. UART Configure Register Access Mode.....	1607
Table 12-143. UART Configure Register Access Submode TCR_TLR.....	1607
Table 12-144. UART Configure Register Access Submode MSR_SPR.....	1607
Table 12-145. UART Configure Register Access Submode XOFF.....	1608
Table 12-146. DMA Mode Settings.....	1608
Table 12-147. Load FIFO Triggers Defined by the FCR.....	1608
Table 12-148. Load FIFO Triggers Defined by the TLR.....	1608
Table 12-149. Load FIFO Triggers Defined by the Concatenated Value.....	1608
Table 12-150. UART Baud Rate Settings.....	1609
Table 12-151. UART Interrupt Settings.....	1609
Table 12-152. UART Protocol Settings.....	1609
Table 12-153. UART Mode Selection.....	1609
Table 12-154. UART Multi-drop Parity Address Match Mode Configuration.....	1610
Table 12-155. UART Hardware Flow Control Configuration.....	1610
Table 12-156. UART Software Flow Control Configuration.....	1610
Table 12-157. SIR Mode Receive Settings.....	1611
Table 12-158. SIR Mode Transmit Settings.....	1611
Table 12-159. MIR Mode Receive Settings.....	1611
Table 12-160. MIR Mode Transmit Settings.....	1612
Table 12-161. FIR Mode Receive Settings.....	1612
Table 12-162. FIR Mode Transmit Settings.....	1612
Table 12-163. MCU_CPSW0 Module Allocation within Device Domains.....	1615
Table 12-164. RMII I/O Description.....	1619
Table 12-165. RGMII I/O Description.....	1620
Table 12-166. MCU_CPSW0 Integration Attributes.....	1623
Table 12-167. MCU_CPSW0 Clocks and Resets.....	1623
Table 12-168. MCU_CPSW0 Hardware Requests.....	1624
Table 12-169. Learned Address Control Bits.....	1632
Table 12-170. Free (Unused) Address Table Entry Bit Values.....	1633
Table 12-171. Multicast Address Table Entry Bit Values.....	1633
Table 12-172. VLAN/Multicast Address Table Entry Bit Values.....	1633
Table 12-173. Unicast Address Table Entry Bit Values.....	1634
Table 12-174. OUI Unicast Address Table Entry Bit Values.....	1635

Table 12-175. Unicast Address Table Entry Bit Values.....	1636
Table 12-176. VLAN Table Entry.....	1637
Table 12-177. Example of TX Configuration.....	1659
Table 12-178. Example of RX Configuration.....	1659
Table 12-179. In-Band Data.....	1662
Table 12-180. Embedded Memories.....	1663
Table 12-181. ECC RAM to CPSW RAM Mapping.....	1664
Table 12-182. ECC Submodule Header Data Bit to Encoder/Decoder Mapping.....	1665
Table 12-183. Switch Latency.....	1669
Table 12-184. Emulation Control Input.....	1669
Table 12-185. Rx Statistics Summary.....	1681
Table 12-186. Tx Statistics Summary.....	1682
Table 12-187. Rx Statistics Summary.....	1685
Table 12-188. Tx Statistics Summary.....	1686
Table 12-189. ADD_VAL feature.....	1690
Table 12-190. Values of Message Type Field.....	1702
Table 12-191. Values of Message Type Field.....	1702
Table 12-192. Values of Message Type Field.....	1703
Table 12-193. Error Encoding on the TXST_PKT_ERR[3:0] Output.....	1705
Table 12-194. MDIO Clause 45 Address Frame Format.....	1712
Table 12-195. MDIO Clause 45 Read Frame Format.....	1712
Table 12-196. MDIO Clause 45 Write Frame Format.....	1712
Table 12-197. CPSW0 Module Allocation within Device Domains.....	1716
Table 12-198. RMII I/O Description.....	1721
Table 12-199. RGMII I/O Description.....	1722
Table 12-200. CPSW0 Integration Attributes.....	1726
Table 12-201. CPSW0 Clocks and Resets.....	1726
Table 12-202. CPSW0 Hardware Requests.....	1728
Table 12-203. Learned Address Control Bits.....	1734
Table 12-204. Learned Address Control Bits.....	1736
Table 12-205. Free (Unused) Address Table Entry Bit Values.....	1736
Table 12-206. Multicast Address Table Entry Bit Values.....	1736
Table 12-207. VLAN/Multicast Address Table Entry Bit Values.....	1737
Table 12-208. Unicast Address Table Entry Bit Values.....	1738
Table 12-209. OUI Unicast Address Table Entry Bit Values.....	1739
Table 12-210. Unicast Address Table Entry Bit Values.....	1740
Table 12-211. Multicast Address Table Entry Bit Values.....	1741
Table 12-212. Inner VLAN Table Entry.....	1742
Table 12-213. Outer VLAN Table Entry.....	1743
Table 12-214. EtherType Table Entry.....	1743
Table 12-215. IPv4 Table Entry.....	1744
Table 12-216. IPv6 Table Entry High.....	1744
Table 12-217. IPv6 Table Entry Low.....	1744
Table 12-218. Example of TX Configuration.....	1769
Table 12-219. Example of RX Configuration.....	1769
Table 12-220. In-Band Data.....	1772
Table 12-221. Embedded Memories.....	1773
Table 12-222. ECC RAM to CPSW RAM Mapping.....	1774
Table 12-223. ECC Submodule Header Data Bit to Encoder/Decoder Mapping.....	1775
Table 12-224. Switch Latency.....	1782
Table 12-225. Emulation Control Input.....	1782
Table 12-226. Rx Statistics Summary.....	1794
Table 12-227. Tx Statistics Summary.....	1795
Table 12-228. Rx Statistics Summary.....	1798
Table 12-229. Tx Statistics Summary.....	1799
Table 12-230. ADD_VAL feature.....	1803
Table 12-231. Values of Message Type Field.....	1815
Table 12-232. Values of Message Type Field.....	1815
Table 12-233. Error Encoding on the TXST_PKT_ERR[3:0] Output.....	1817
Table 12-234. MDIO Clause 45 Address Frame Format.....	1825
Table 12-235. MDIO Clause 45 Read Frame Format.....	1825

Table 12-236. MDIO Clause 45 Write Frame Format.....	1825
Table 12-237. PCIe Subsystem Allocation within Device Domains.....	1829
Table 12-238. PCIe Subsystem I/O Signals.....	1832
Table 12-239. PCIe Subsystem Integration Attributes.....	1835
Table 12-240. PCIe Subsystem Clocks and Resets.....	1835
Table 12-241. PCIe Subsystem Hardware Requests.....	1838
Table 12-242. PCIe Core Interrupt Events.....	1855
Table 12-243. PCIe Root Port Inbound PCIe to AXI Address Translation Registers for one BAR.....	1863
Table 12-244. PCIe End Point Inbound PCIe to AXI Address Translation Registers for one BAR.....	1864
Table 12-245. AXI_A*USER Sideband Signal Mapping.....	1865
Table 12-246. QoS Ingress CCHANID Mapping.....	1870
Table 12-247. AXI to VBUSM Bus Error Mapping.....	1871
Table 12-248. VBUSM to AXI Bus Error Mapping.....	1872
Table 12-249. LTSSM State Encoding.....	1873
Table 12-250. USB Allocation Within Device Domains.....	1874
Table 12-251. USB3SS0 Input/Output Description.....	1880
Table 12-252. USB3SS1 Input/Output Description.....	1880
Table 12-253. USB Integration Attributes.....	1885
Table 12-254. USB Clocks and Resets.....	1885
Table 12-255. USB Hardware Requests.....	1885
Table 12-256. USB3.0 Register Configuration.....	1889
Table 12-257. SerDes Allocation Within Device Domains.....	1890
Table 12-258. SerDes Supported Standards.....	1895
Table 12-259. SERDES0 Input/Output Description.....	1897
Table 12-260. SERDES1 Input/Output Description.....	1898
Table 12-261. SERDES2 Input/Output Description.....	1900
Table 12-262. SERDES3 Input/Output Description.....	1901
Table 12-263. 2-L SerDes Integration Attributes.....	1904
Table 12-264. 2-L SerDes Clocks and Resets.....	1904
Table 12-265. 2-L SerDes Hardware Requests.....	1905
Table 12-266. SERDES0 Supported Configurations.....	1906
Table 12-267. SERDES1 Supported Configurations.....	1906
Table 12-268. SERDES2 Supported Configurations.....	1906
Table 12-269. SERDES3 Supported Configurations.....	1907
Table 12-270. USB0 Muxing to Serdeses.....	1907
Table 12-271. USB1 Muxing to Serdeses.....	1907
Table 12-272. ICSSG1 SGMII Lane 0 Muxing to Serdeses.....	1907
Table 12-273. ICSSG1 SGMII Lane 1 Muxing to Serdeses.....	1907
Table 12-274. 2-L SerDes Internal Reference Clock Selection (REFCLK).....	1908
Table 12-275. 2-L SerDes Internal Reference Clock Selection (REFCLK1).....	1908
Table 12-276. SerDes Allocation Within Device Domains.....	1910
Table 12-277. 4-L SerDes Supported Standards.....	1912
Table 12-278. 4-L SerDes Input/Output Description.....	1914
Table 12-279. 4-L SerDes Integration Attributes.....	1917
Table 12-280. 4-L SerDes Clocks and Resets.....	1917
Table 12-281. 4-L SerDes Hardware Requests.....	1917
Table 12-282. SERDES4 Supported Configurations.....	1918
Table 12-283. 4-L SerDes Internal Reference Clock Selection.....	1918
Table 12-284. FSS Allocation Across Device Domains.....	1921
Table 12-285. MCU_FSS0 I/O Signals.....	1924
Table 12-286. MCU_FSS0 Integration Attributes.....	1925
Table 12-287. MCU_FSS0 Clocks and Resets.....	1925
Table 12-288. MCU_FSS0 Hardware Requests.....	1926
Table 12-289. FSS Allowed Interface Combinations.....	1927
Table 12-290. Memory Address Equation.....	1929
Table 12-291. FSS Memory Regions.....	1930
Table 12-292. OSPI Allocation Across Device Domains.....	1933
Table 12-293. OSPI I/O Signals.....	1936
Table 12-294. OSPI I/O Connectivity to External SPI Devices.....	1936
Table 12-295. FSS0_OSPI Integration Attributes.....	1939
Table 12-296. MCU_FSS0_OSPI Clocks and Resets.....	1939

Table 12-297. FSS0_OSPI Hardware Requests.....	1939
Table 12-298. OSPI Modes.....	1941
Table 12-299. OSPI Memory Map.....	1944
Table 12-300. OSPI Events.....	1945
Table 12-301. SRAM Access Priority.....	1952
Table 12-302. Flash SPI Modes.....	1956
Table 12-303. READ and WRITE Instruction Configuration.....	1957
Table 12-304. HyperBus Allocation Across Device Domains.....	1966
Table 12-305. HyperBus I/O Signals.....	1968
Table 12-306. MCU_FSS0_HPB0 Integration Attributes.....	1970
Table 12-307. MCU_FSS0_HPB0 Clocks and Resets.....	1970
Table 12-308. MCU_FSS0_HPB0 Hardware Requests.....	1971
Table 12-309. HyperBus FIFOs.....	1974
Table 12-310. HyperFlash Access Sequence.....	1975
Table 12-311. HyperRAM Access Sequence.....	1976
Table 12-312. GPMC I/O Signals (Master Mode).....	1983
Table 12-313. GPMC Pin Multiplexing Options.....	1984
Table 12-314. GPMC0 Integration Attributes.....	1987
Table 12-315. GPMC0 Clocks and Resets.....	1987
Table 12-316. GPMC0 Hardware Requests.....	1988
Table 12-317. GPMC Clocks.....	1991
Table 12-318. GPMC Output Clock Configuration.....	1991
Table 12-319. GPMC Local Power-Management Features.....	1991
Table 12-320. GPMC Interrupt Events.....	1991
Table 12-321. Idle Cycle Insertion Configuration.....	2003
Table 12-322. Chip-Select Configuration for NAND Interfacing.....	2032
Table 12-323. ECC Enable Settings.....	2040
Table 12-324. Flattened BCH Codeword Mapping (512 Bytes + 104 Bits).....	2045
Table 12-325. Aligned Message Byte Mapping in 8-Bit NAND.....	2046
Table 12-326. Aligned Message Byte Mapping in 16-Bit NAND.....	2046
Table 12-327. Aligned Nibble Mapping of Message in 8-Bit NAND.....	2046
Table 12-328. Misaligned Nibble Mapping of Message in 8-Bit NAND.....	2046
Table 12-329. Aligned Nibble Mapping of Message in 16-Bit NAND.....	2047
Table 12-330. Misaligned Nibble Mapping of Message in 16-Bit NAND (1 Unused Nibble).....	2047
Table 12-331. Misaligned Nibble Mapping of Message in 16-Bit NAND (2 Unused Nibbles).....	2047
Table 12-332. Misaligned Nibble Mapping of Message in 16-Bit NAND (3 Unused Nibbles).....	2047
Table 12-333. Prefetch Mode Configuration.....	2058
Table 12-334. Write-Posting Mode Configuration.....	2060
Table 12-335. Typical GPMC Setup Signals.....	2063
Table 12-336. Useful Timing Parameters on the Memory Side.....	2064
Table 12-337. Calculating GPMC Timing Parameters.....	2065
Table 12-338. AC Characteristics for Asynchronous Read Access.....	2066
Table 12-339. GPMC Timing Parameters for Asynchronous Read Access.....	2067
Table 12-340. AC Characteristics for Asynchronous Single Write (Memory Side).....	2067
Table 12-341. GPMC Timing Parameters for Asynchronous Single Write.....	2068
Table 12-342. Supported Memory Interfaces.....	2069
Table 12-343. NAND Interface Bus Operations Summary.....	2070
Table 12-344. NOR Interface Bus Operations Summary.....	2071
Table 12-345. GPMC Configuration in NOR Mode.....	2073
Table 12-346. GPMC Configuration in NAND Mode.....	2074
Table 12-347. NOR Memory Type.....	2074
Table 12-348. NOR Chip-Select Configuration.....	2074
Table 12-349. NOR Timings Configuration.....	2075
Table 12-350. WAIT Pin Configuration.....	2075
Table 12-351. Enable Chip-Select.....	2075
Table 12-352. NAND Memory Type.....	2075
Table 12-353. NAND Chip-Select Configuration.....	2075
Table 12-354. Asynchronous Read and Write Operations.....	2075
Table 12-355. ECC Engine.....	2075
Table 12-356. Prefetch and Write-Posting Engine.....	2076
Table 12-357. WAIT Pin Configuration.....	2076



Table 12-358. Enable Chip-Select.....	2076
Table 12-359. Mode Parameters Check List.....	2077
Table 12-360. Access Type Parameters Check List.....	2077
Table 12-361. Timing Parameters.....	2078
Table 12-362. NAND Formulas Description.....	2080
Table 12-363. Synchronous NOR Formulas Description.....	2081
Table 12-364. Asynchronous NOR Formulas Description.....	2085
Table 12-365. ELM Allocation Across Device Domains.....	2088
Table 12-366. ELM0 Integration Attributes.....	2091
Table 12-367. ELM0 Clocks and Resets.....	2091
Table 12-368. ELM0 Hardware Requests.....	2091
Table 12-369. Local Power-Management Features.....	2093
Table 12-370. ELM Events.....	2093
Table 12-371. ELM_LOCATION_STATUS_i Value Decoding.....	2094
Table 12-372. ELM Processing Initialization.....	2096
Table 12-373. ELM Processing Completion for Continuous Mode.....	2096
Table 12-374. ELM Processing Completion for Page Mode.....	2097
Table 12-375. Use Case: Continuous Mode.....	2097
Table 12-376. 16-Bit NAND Sector Buffer Address Map.....	2098
Table 12-377. Use Case: Page Mode.....	2099
Table 12-378. MMCSD Allocation Across Device Domains.....	2102
Table 12-379. MMCSDi I/O Signals.....	2106
Table 12-380. Relationship Between Configuration and Name of Response Type.....	2110
Table 12-381. MMCSD Integration Attributes.....	2114
Table 12-382. MMCSD Clocks and Resets.....	2114
Table 12-383. MMCSD Hardware Requests.....	2115
Table 12-384. MMCSDi Module Memory Map.....	2118
Table 12-385. BIST Logic Signals.....	2122
Table 12-386. Determination of Transfer Type.....	2148
Table 12-387. Host Controller Setup Sequence.....	2173
Table 12-388. Card Interface Detection Sequence.....	2174
Table 12-389. Command Queuing Initialization Sequence.....	2182
Table 12-390. Task Issuance Sequence.....	2183
Table 12-391. Task Execution and Completion Sequence.....	2185
Table 12-392. Task Discard and Clear Sequence.....	2186
Table 12-393. Error Detect and Recovery Sequence.....	2187
Table 12-394. UFS Standards.....	2189
Table 12-395. UFS Allocation Across Device Domains.....	2189
Table 12-396. UFS I/O Signals.....	2191
Table 12-397. UFS0 Integration Attributes.....	2193
Table 12-398. UFS0 Clocks and Resets.....	2193
Table 12-399. UFS0 Hardware Requests.....	2193
Table 12-400. Reset Values for Some UniPro Attributes.....	2202
Table 12-401. UFS Host Controller Commands.....	2204
Table 12-402. The Dual Connection M-PHY Configuration Bits.....	2210
Table 12-403. ECAP Allocation Across Device Domains.....	2212
Table 12-404. ECAP I/O Signals.....	2214
Table 12-405. ECAP Integration Attributes.....	2217
Table 12-406. ECAP Clocks and Resets.....	2217
Table 12-407. ECAP Hardware Requests.....	2217
Table 12-408. ECAP Control and Status Functional Registers.....	2229
Table 12-409. ECAP Initialization for CAP Mode Absolute Time, Rising Edge Trigger.....	2232
Table 12-410. ECAP Initialization for CAP Mode Absolute Time, Rising and Falling Edge Trigger.....	2234
Table 12-411. ECAP Initialization for CAP Mode Delta Time, Rising Edge Trigger.....	2236
Table 12-412. ECAP Initialization for CAP Mode Delta Time, Rising and Falling Edge Triggers.....	2238
Table 12-413. ECAP Initialization for APWM Mode.....	2240
Table 12-414. ECAP1 Initialization for Multichannel PWM Generation with Synchronization.....	2242
Table 12-415. ECAP2 Initialization for Multichannel PWM Generation with Synchronization.....	2242
Table 12-416. ECAP3 Initialization for Multichannel PWM Generation with Synchronization.....	2242
Table 12-417. ECAP4 Initialization for Multichannel PWM Generation with Synchronization.....	2242
Table 12-418. ECAP1 Initialization for Multichannel PWM Generation with Phase Control.....	2245

Table 12-419. ECAP2 Initialization for Multichannel PWM Generation with Phase Control.....	2245
Table 12-420. ECAP3 Initialization for Multichannel PWM Generation with Phase Control.....	2245
Table 12-421. EPWM Allocation Across Device Domains.....	2246
Table 12-422. EPWM Subsystems I/O Signals.....	2256
Table 12-423. EPWM Integration Attributes.....	2260
Table 12-424. EPWM Clocks and Resets.....	2260
Table 12-425. EPWM Hardware Requests.....	2260
Table 12-426. Device Limitations for the EPWM Functional Interfaces.....	2268
Table 12-427. Submodule Configuration Parameters.....	2275
Table 12-428. EPWM Time-Base Submodule Registers.....	2279
Table 12-429. EPWM Time-Base Submodule Key Signals.....	2279
Table 12-430. EPWM Counter-Compare Submodule Registers.....	2286
Table 12-431. EPWM Counter-Compare Submodule Key Signals.....	2287
Table 12-432. EPWM Action-Qualifier Submodule Registers.....	2292
Table 12-433. EPWM Action-Qualifier Submodule Possible Input Events.....	2293
Table 12-434. EPWM Action-Qualifier Event Priority for Up-Down-Count Mode.....	2295
Table 12-435. EPWM Action-Qualifier Event Priority for Up-Count Mode.....	2295
Table 12-436. EPWM Action-Qualifier Event Priority for Down-Count Mode.....	2295
Table 12-437. Behavior if CMPA/CMPB is Greater than the Period.....	2296
Table 12-438. EPWMx Initialization for <a href="#">Figure 12-409</a> .....	2299
Table 12-439. EPWMx Run Time Changes for <a href="#">Figure 12-409</a> .....	2299
Table 12-440. EPWMx Initialization for <a href="#">Figure 12-410</a> .....	2301
Table 12-441. EPWMx Run Time Changes for <a href="#">Figure 12-410</a> .....	2301
Table 12-442. EPWMx Initialization for <a href="#">Figure 12-411</a> .....	2303
Table 12-443. EPWMx Run Time Changes for <a href="#">Figure 12-411</a> .....	2303
Table 12-444. EPWMx Initialization for <a href="#">Figure 12-412</a> .....	2305
Table 12-445. EPWMx Run Time Changes for <a href="#">Figure 12-412</a> .....	2305
Table 12-446. EPWMx Initialization for <a href="#">Figure 12-413</a> .....	2307
Table 12-447. EPWMx Run Time Changes for <a href="#">Figure 12-413</a> .....	2307
Table 12-448. EPWMx Initialization for <a href="#">Figure 12-414</a> .....	2309
Table 12-449. EPWMx Run Time Changes for <a href="#">Figure 12-414</a> .....	2309
Table 12-450. Dead-Band Generator Submodule Registers.....	2310
Table 12-451. Classical Dead-Band Operating Modes.....	2312
Table 12-452. EPWM-Chopper Submodule Registers.....	2314
Table 12-453. EPWM Trip-Zone Submodule Registers.....	2319
Table 12-454. Possible Actions On an EPWM Trip Event.....	2320
Table 12-455. EPWM Event-Trigger Submodule Registers.....	2323
Table 12-456. Resolution for PWM and HRPWM.....	2329
Table 12-457. HRPWM Submodule Registers.....	2330
Table 12-458. Relationship Between MEP Steps, PWM Frequency and Resolution.....	2331
Table 12-459. CMPA vs Duty (left), and [CMPA:CMPAHR] vs Duty (right).....	2332
Table 12-460. EPWM / HRPWM Module Control and Status Registers Grouped by Submodule.....	2334
Table 12-461. EQEP Allocation Across Device Domains.....	2340
Table 12-462. EQEP I/O Signals.....	2342
Table 12-463. EQEP Integration Attributes.....	2346
Table 12-464. EQEP Clocks and Resets.....	2346
Table 12-465. EQEP Hardware Requests.....	2346
Table 12-466. Device Limitations for the EQEP Functional Interfaces.....	2348
Table 12-467. Quadrature Decoder Truth Table.....	2352
Table 12-468. EQEP Control and Status Functional Registers.....	2368
Table 12-469. MCAN Allocation Across Device Domains.....	2369
Table 12-470. MCAN I/O Signals.....	2372
Table 12-471. MCU_MCAN Integration Attributes.....	2376
Table 12-472. MCU_MCAN Clocks and Resets.....	2376
Table 12-473. MCU_MCAN Hardware Requests.....	2377
Table 12-474. MCAN Integration Attributes.....	2380
Table 12-475. MCAN Clocks and Resets.....	2380
Table 12-476. MCAN Hardware Requests.....	2383
Table 12-477. Steps to Configure MCAN Module.....	2403
Table 12-478. DLC Coding.....	2405
Table 12-479. Rx Buffer/Rx FIFO Element Size.....	2417

Table 12-480. Example Filter Configuration for Rx Buffers.....	2418
Table 12-481. Possible Configurations for Message Transmission.....	2419
Table 12-482. Tx Buffer/Tx FIFO/Tx Queue Element Size.....	2420
Table 12-483. Message RAM Address Range.....	2424
Table 12-484. Rx Buffer/Rx FIFO Element Field Descriptions.....	2426
Table 12-485. Tx Buffer Element Field Descriptions.....	2428
Table 12-486. Tx Event FIFO Element Field Descriptions.....	2430
Table 12-487. Standard Message ID Filter Element Field Descriptions.....	2432
Table 12-488. Extended Message ID Filter Element Field Descriptions.....	2433
Table 12-489. ATL Allocation Across Device Domains.....	2436
Table 12-490. MCASP I/O Signals.....	2440
Table 12-491. Biphas-Mark Encoder.....	2448
Table 12-492. Preamble Codes.....	2449
Table 12-493. MCASP Integration Attributes.....	2450
Table 12-494. MCASP Clocks and Resets.....	2450
Table 12-495. MCASP Hardware Requests.....	2450
Table 12-496. MCASP TFU TDM Mode Settings.....	2462
Table 12-497. MCASP TFU DIT-Mode Example Settings.....	2463
Table 12-498. MCASP RFU Settings.....	2465
Table 12-499. Local Power-Management Features.....	2466
Table 12-500. Channel Status and User Data for Each DIT Block.....	2471
Table 12-501. TX Events.....	2480
Table 12-502. RX Events.....	2481
Table 12-503. Global Initialization of Surrounding Modules.....	2490
Table 12-504. MCASP Transmitters Global Initialization for DIT-Mode Operation.....	2491
Table 12-505. Transmit Format Unit Configuration for DIT-Transmission.....	2491
Table 12-506. Transmit Frame-Synchronization Generator Configuration for DIT-Transmission.....	2492
Table 12-507. Transmit Clock Generator Configuration in DIT-Mode.....	2492
Table 12-508. MCASP Pins Functional Configuration.....	2492
Table 12-509. DIT-Specific Subframe Fields Configuration.....	2492
Table 12-510. MCASP Receivers Global Initialization for TDM-Mode Operation.....	2493
Table 12-511. Receive Format Unit Configuration for TDM-Reception.....	2494
Table 12-512. Receive Frame-Synchronization Generator Configuration for TDM-Reception.....	2494
Table 12-513. Receive Clock Generator Configuration.....	2495
Table 12-514. MCASP Receiver Pins Functional Configuration.....	2496
Table 12-515. MCASP Transmitters Global Initialization for TDM-Mode Operation.....	2496
Table 12-516. Transmit Format Unit Configuration for TDM-Transmission.....	2497
Table 12-517. Transmit Frame-Synchronization Generator Configuration for TDM-Transmission.....	2498
Table 12-518. Transmit Clock Generator Configuration for TDM Cases.....	2498
Table 12-519. MCASP Transmit Pins Functional Configuration.....	2499
Table 12-520. Subprocess Call Summary for Main Sequence – MCASP DIT-/TDM- Transmission Polling Method.....	2500
Table 12-521. MCASP DIT-/TDM- Interrupt Transmission Model.....	2503
Table 12-522. MCASP DMA Transmission Model with Interrupt Events Servicing.....	2503
Table 12-523. Subprocess Call Summary for Main Sequence – MCASP Reception Polling Method.....	2505
Table 12-524. MCASP TDM- Interrupt Reception Model.....	2507
Table 12-525. MCASP DMA Reception Model with Interrupt Events Servicing.....	2507
Table 12-526. Subprocess Call Summary for Receive Interrupt Events Servicing.....	2510
Table 12-527. DSS and Display Peripherals Allocation Across Device Domains.....	2513
Table 12-528. DSS Pixel Format Interoperability between DISPC VP Output and DPI, eDP, and DSI.....	2520
Table 12-529. DSS DPI Parallel Interface I/O Signals.....	2521
Table 12-530. DISPC Video Port Register Fields for Active Matrix Display.....	2526
Table 12-531. DSI Interface I/O Signals.....	2531
Table 12-532. DP/eDP I/O Signals.....	2532
Table 12-533. DISPC Integration Attributes.....	2534
Table 12-534. DISPC Clocks and Resets.....	2534
Table 12-535. DISPC Hardware Requests.....	2535
Table 12-536. DSI Integration Attributes.....	2539
Table 12-537. DSI Clocks and Resets.....	2540
Table 12-538. EDP Integration Attributes.....	2542
Table 12-539. EDP Clocks and Resets.....	2542
Table 12-540. EDP Hardware Requests.....	2543

Table 12-541. DISPC Interrupts - First Level.....	2549
Table 12-542. DISPC Interrupts - Second Level - VID1 and VID2 Pipeline.....	2549
Table 12-543. DISPC Interrupts - Second Level - VIDL1 and VIDL2 Pipelines.....	2550
Table 12-544. DISPC Interrupts - Second Level - WB Pipeline.....	2551
Table 12-545. DISPC Interrupts - Second Level - VP1 and VP2 Outputs.....	2552
Table 12-546. DISPC Interrupts - Second Level - VP3 and VP4 Outputs.....	2555
Table 12-547. DISPC Register Settings for Accessing Image in Internal Memory.....	2559
Table 12-548. DISPC Flip/Mirror Configuration.....	2562
Table 12-549. DISPC Supported Pixel Data Formats.....	2571
Table 12-550. DISPC VID Replication: ARGB Pixel Formats Remapping into ARGB48-12121212.....	2587
Table 12-551. DISPC VID Line Buffer Width for Scaler Unit.....	2592
Table 12-552. DISPC Register Fields Associated to Coefficients for ARGB and Y Configuration in VID Horizontal Scaler.....	2594
Table 12-553. DISPC VID Scaler Vertical and Horizontal Accumulator Phases.....	2596
Table 12-554. DISPC VID CSC - YUV to RGB Register Settings.....	2597
Table 12-555. DISPC VID BCSH Register Settings for YUV to RGB.....	2599
Table 12-556. DISPC WB CSC - RGB to YUV Register Settings.....	2601
Table 12-557. DISPC Register Fields Associated to Coefficients for ARGB and Y Configuration in WB Horizontal Scaler.....	2603
Table 12-558. DISPC WB Scaler Vertical and Horizontal Accumulator Phases.....	2605
Table 12-559. DISPC Overlay Alpha Blending – ARGB.....	2610
Table 12-560. DISPC Overlay Color Bar Table.....	2614
Table 12-561. DISPC VP CPR and CSC Coefficients with Associated Register Fields.....	2615
Table 12-562. DISPC BT Mode Bit Function.....	2618
Table 12-563. DISPC BT Mode Status of Protection Bits in Function of F, V, and H.....	2618
Table 12-564. DISPC Internal Diagnostic Check Regions Parameters.....	2629
Table 12-565. DSS_COMMON Instances.....	2637
Table 12-566. DSS_COMMON Registers.....	2637
Table 12-567. DSS0_COMMON_DSS_REVISION Instances.....	2639
Table 12-568. DSS0_COMMON_DSS_REVISION Register Field Descriptions.....	2639
Table 12-569. DSS0_COMMON_DSS_SYSCONFIG Instances.....	2640
Table 12-570. DSS0_COMMON_DSS_SYSCONFIG Register Field Descriptions.....	2640
Table 12-571. DSS0_COMMON_DSS_SYSSTATUS Instances.....	2642
Table 12-572. DSS0_COMMON_DSS_SYSSTATUS Register Field Descriptions.....	2642
Table 12-573. DSS0_COMMON_DISPC_IRQSTATUS_RAW Instances.....	2644
Table 12-574. DSS0_COMMON_DISPC_IRQSTATUS_RAW Register Field Descriptions.....	2644
Table 12-575. DSS0_COMMON_DISPC_IRQSTATUS Instances.....	2646
Table 12-576. DSS0_COMMON_DISPC_IRQSTATUS Register Field Descriptions.....	2646
Table 12-577. DSS0_COMMON_DISPC_IRQENABLE_SET Instances.....	2648
Table 12-578. DSS0_COMMON_DISPC_IRQENABLE_SET Register Field Descriptions.....	2648
Table 12-579. DSS0_COMMON_DISPC_IRQENABLE_CLR Instances.....	2650
Table 12-580. DSS0_COMMON_DISPC_IRQENABLE_CLR Register Field Descriptions.....	2650
Table 12-581. DSS0_COMMON_VID_IRQENABLE_0 Instances.....	2652
Table 12-582. DSS0_COMMON_VID_IRQENABLE_0 Register Field Descriptions.....	2652
Table 12-583. DSS0_COMMON_VID_IRQENABLE_1 Instances.....	2654
Table 12-584. DSS0_COMMON_VID_IRQENABLE_1 Register Field Descriptions.....	2654
Table 12-585. DSS0_COMMON_VID_IRQENABLE_2 Instances.....	2656
Table 12-586. DSS0_COMMON_VID_IRQENABLE_2 Register Field Descriptions.....	2656
Table 12-587. DSS0_COMMON_VID_IRQENABLE_3 Instances.....	2658
Table 12-588. DSS0_COMMON_VID_IRQENABLE_3 Register Field Descriptions.....	2658
Table 12-589. DSS0_COMMON_VID_IRQSTATUS_0 Instances.....	2660
Table 12-590. DSS0_COMMON_VID_IRQSTATUS_0 Register Field Descriptions.....	2660
Table 12-591. DSS0_COMMON_VID_IRQSTATUS_1 Instances.....	2662
Table 12-592. DSS0_COMMON_VID_IRQSTATUS_1 Register Field Descriptions.....	2662
Table 12-593. DSS0_COMMON_VID_IRQSTATUS_2 Instances.....	2664
Table 12-594. DSS0_COMMON_VID_IRQSTATUS_2 Register Field Descriptions.....	2664
Table 12-595. DSS0_COMMON_VID_IRQSTATUS_3 Instances.....	2666
Table 12-596. DSS0_COMMON_VID_IRQSTATUS_3 Register Field Descriptions.....	2666
Table 12-597. DSS0_COMMON_VP_IRQENABLE_0 Instances.....	2668
Table 12-598. DSS0_COMMON_VP_IRQENABLE_0 Register Field Descriptions.....	2668
Table 12-599. DSS0_COMMON_VP_IRQENABLE_1 Instances.....	2670
Table 12-600. DSS0_COMMON_VP_IRQENABLE_1 Register Field Descriptions.....	2670
Table 12-601. DSS0_COMMON_VP_IRQENABLE_2 Instances.....	2672



Table 12-602. DSS0_COMMON_VP_IRQENABLE_2 Register Field Descriptions.....	2672
Table 12-603. DSS0_COMMON_VP_IRQENABLE_3 Instances.....	2674
Table 12-604. DSS0_COMMON_VP_IRQENABLE_3 Register Field Descriptions.....	2674
Table 12-605. DSS0_COMMON_VP_IRQSTATUS_0 Instances.....	2676
Table 12-606. DSS0_COMMON_VP_IRQSTATUS_0 Register Field Descriptions.....	2676
Table 12-607. DSS0_COMMON_VP_IRQSTATUS_1 Instances.....	2678
Table 12-608. DSS0_COMMON_VP_IRQSTATUS_1 Register Field Descriptions.....	2678
Table 12-609. DSS0_COMMON_VP_IRQSTATUS_2 Instances.....	2680
Table 12-610. DSS0_COMMON_VP_IRQSTATUS_2 Register Field Descriptions.....	2680
Table 12-611. DSS0_COMMON_VP_IRQSTATUS_3 Instances.....	2682
Table 12-612. DSS0_COMMON_VP_IRQSTATUS_3 Register Field Descriptions.....	2682
Table 12-613. DSS0_COMMON_WB_IRQENABLE Instances.....	2684
Table 12-614. DSS0_COMMON_WB_IRQENABLE Register Field Descriptions.....	2684
Table 12-615. DSS0_COMMON_WB_IRQSTATUS Instances.....	2686
Table 12-616. DSS0_COMMON_WB_IRQSTATUS Register Field Descriptions.....	2686
Table 12-617. DSS0_COMMON_DISPC_IRQ_EOI_FUNC Instances.....	2688
Table 12-618. DSS0_COMMON_DISPC_IRQ_EOI_FUNC Register Field Descriptions.....	2688
Table 12-619. DSS0_COMMON_DISPC_IRQ_EOI_SAFETY Instances.....	2689
Table 12-620. DSS0_COMMON_DISPC_IRQ_EOI_SAFETY Register Field Descriptions.....	2689
Table 12-621. DSS0_COMMON_DISPC_IRQ_EOI_SECURITY Instances.....	2690
Table 12-622. DSS0_COMMON_DISPC_IRQ_EOI_SECURITY Register Field Descriptions.....	2690
Table 12-623. DSS0_COMMON_DISPC_SECURE_DISABLE Instances.....	2691
Table 12-624. DSS0_COMMON_DISPC_SECURE_DISABLE Register Field Descriptions.....	2691
Table 12-625. DSS0_COMMON_DISPC_GLOBAL_MFLAG_ATTRIBUTE Instances.....	2692
Table 12-626. DSS0_COMMON_DISPC_GLOBAL_MFLAG_ATTRIBUTE Register Field Descriptions.....	2692
Table 12-627. DSS0_COMMON_DISPC_GLOBAL_OUTPUT_ENABLE Instances.....	2694
Table 12-628. DSS0_COMMON_DISPC_GLOBAL_OUTPUT_ENABLE Register Field Descriptions.....	2694
Table 12-629. DSS0_COMMON_DISPC_GLOBAL_BUFFER Instances.....	2695
Table 12-630. DSS0_COMMON_DISPC_GLOBAL_BUFFER Register Field Descriptions.....	2695
Table 12-631. DSS0_COMMON_DSS_CBA_CFG Instances.....	2697
Table 12-632. DSS0_COMMON_DSS_CBA_CFG Register Field Descriptions.....	2697
Table 12-633. DSS0_COMMON_DISPC_DBG_CONTROL Instances.....	2698
Table 12-634. DSS0_COMMON_DISPC_DBG_CONTROL Register Field Descriptions.....	2698
Table 12-635. DSS0_COMMON_DISPC_DBG_STATUS Instances.....	2700
Table 12-636. DSS0_COMMON_DISPC_DBG_STATUS Register Field Descriptions.....	2700
Table 12-637. DSS0_COMMON_DISPC_CLKGATING_DISABLE Instances.....	2701
Table 12-638. DSS0_COMMON_DISPC_CLKGATING_DISABLE Register Field Descriptions.....	2701
Table 12-639. DSS0_COMMON_FBDC_REVISION_1 Instances.....	2703
Table 12-640. DSS0_COMMON_FBDC_REVISION_1 Register Field Descriptions.....	2703
Table 12-641. DSS0_COMMON_FBDC_REVISION_2 Instances.....	2704
Table 12-642. DSS0_COMMON_FBDC_REVISION_2 Register Field Descriptions.....	2704
Table 12-643. DSS0_COMMON_FBDC_REVISION_3 Instances.....	2705
Table 12-644. DSS0_COMMON_FBDC_REVISION_3 Register Field Descriptions.....	2705
Table 12-645. DSS0_COMMON_FBDC_REVISION_4 Instances.....	2706
Table 12-646. DSS0_COMMON_FBDC_REVISION_4 Register Field Descriptions.....	2706
Table 12-647. DSS0_COMMON_FBDC_REVISION_5 Instances.....	2707
Table 12-648. DSS0_COMMON_FBDC_REVISION_5 Register Field Descriptions.....	2707
Table 12-649. DSS0_COMMON_FBDC_REVISION_6 Instances.....	2708
Table 12-650. DSS0_COMMON_FBDC_REVISION_6 Register Field Descriptions.....	2708
Table 12-651. DSS0_COMMON_FBDC_COMMON_CONTROL Instances.....	2709
Table 12-652. DSS0_COMMON_FBDC_COMMON_CONTROL Register Field Descriptions.....	2709
Table 12-653. DSS0_COMMON_FBDC_CONSTANT_COLOR_0 Instances.....	2710
Table 12-654. DSS0_COMMON_FBDC_CONSTANT_COLOR_0 Register Field Descriptions.....	2710
Table 12-655. DSS0_COMMON_FBDC_CONSTANT_COLOR_1 Instances.....	2711
Table 12-656. DSS0_COMMON_FBDC_CONSTANT_COLOR_1 Register Field Descriptions.....	2711
Table 12-657. DSS0_COMMON_DISPC_CONNECTIONS Instances.....	2712
Table 12-658. DSS0_COMMON_DISPC_CONNECTIONS Register Field Descriptions.....	2712
Table 12-659. DSS0_COMMON_DISPC_MSS_VP1 Instances.....	2714
Table 12-660. DSS0_COMMON_DISPC_MSS_VP1 Register Field Descriptions.....	2714
Table 12-661. DSS0_COMMON_DISPC_MSS_VP3 Instances.....	2715
Table 12-662. DSS0_COMMON_DISPC_MSS_VP3 Register Field Descriptions.....	2715



Table 12-663. DSS0_COMMON_GLOBAL_DMA_THREADSIZE Instances.....	2716
Table 12-664. DSS0_COMMON_GLOBAL_DMA_THREADSIZE Register Field Descriptions.....	2716
Table 12-665. DSS0_COMMON_GLOBAL_DMA_THREADSIZESTATUS Instances.....	2717
Table 12-666. DSS0_COMMON_GLOBAL_DMA_THREADSIZESTATUS Register Field Descriptions.....	2717
Table 12-667. DSS0_COMMON_GLOBAL_GOBITMODE Instances.....	2718
Table 12-668. DSS0_COMMON_GLOBAL_GOBITMODE Register Field Descriptions.....	2718
Table 12-669. DSS_VID Instances.....	2719
Table 12-670. DSS_VID Registers.....	2719
Table 12-671. DSS0_VID_ACCUH_0 Instances.....	2724
Table 12-672. DSS0_VID_ACCUH_0 Register Field Descriptions.....	2724
Table 12-673. DSS0_VID_ACCUH_1 Instances.....	2725
Table 12-674. DSS0_VID_ACCUH_1 Register Field Descriptions.....	2725
Table 12-675. DSS0_VID_ACCUH2_0 Instances.....	2726
Table 12-676. DSS0_VID_ACCUH2_0 Register Field Descriptions.....	2726
Table 12-677. DSS0_VID_ACCUH2_1 Instances.....	2727
Table 12-678. DSS0_VID_ACCUH2_1 Register Field Descriptions.....	2727
Table 12-679. DSS0_VID_ACCUV_0 Instances.....	2728
Table 12-680. DSS0_VID_ACCUV_0 Register Field Descriptions.....	2728
Table 12-681. DSS0_VID_ACCUV_1 Instances.....	2729
Table 12-682. DSS0_VID_ACCUV_1 Register Field Descriptions.....	2729
Table 12-683. DSS0_VID_ACCUV2_0 Instances.....	2730
Table 12-684. DSS0_VID_ACCUV2_0 Register Field Descriptions.....	2730
Table 12-685. DSS0_VID_ACCUV2_1 Instances.....	2731
Table 12-686. DSS0_VID_ACCUV2_1 Register Field Descriptions.....	2731
Table 12-687. DSS0_VID_ATTRIBUTES Instances.....	2732
Table 12-688. DSS0_VID_ATTRIBUTES Register Field Descriptions.....	2732
Table 12-689. DSS0_VID_ATTRIBUTES2 Instances.....	2737
Table 12-690. DSS0_VID_ATTRIBUTES2 Register Field Descriptions.....	2737
Table 12-691. DSS0_VID_BA_0 Instances.....	2739
Table 12-692. DSS0_VID_BA_0 Register Field Descriptions.....	2739
Table 12-693. DSS0_VID_BA_1 Instances.....	2740
Table 12-694. DSS0_VID_BA_1 Register Field Descriptions.....	2740
Table 12-695. DSS0_VID_BA_UV_0 Instances.....	2741
Table 12-696. DSS0_VID_BA_UV_0 Register Field Descriptions.....	2741
Table 12-697. DSS0_VID_BA_UV_1 Instances.....	2742
Table 12-698. DSS0_VID_BA_UV_1 Register Field Descriptions.....	2742
Table 12-699. DSS0_VID_BUF_SIZE_STATUS Instances.....	2743
Table 12-700. DSS0_VID_BUF_SIZE_STATUS Register Field Descriptions.....	2743
Table 12-701. DSS0_VID_BUF_THRESHOLD Instances.....	2744
Table 12-702. DSS0_VID_BUF_THRESHOLD Register Field Descriptions.....	2744
Table 12-703. DSS0_VID_CSC_COEF0 Instances.....	2745
Table 12-704. DSS0_VID_CSC_COEF0 Register Field Descriptions.....	2745
Table 12-705. DSS0_VID_CSC_COEF1 Instances.....	2746
Table 12-706. DSS0_VID_CSC_COEF1 Register Field Descriptions.....	2746
Table 12-707. DSS0_VID_CSC_COEF2 Instances.....	2747
Table 12-708. DSS0_VID_CSC_COEF2 Register Field Descriptions.....	2747
Table 12-709. DSS0_VID_CSC_COEF3 Instances.....	2748
Table 12-710. DSS0_VID_CSC_COEF3 Register Field Descriptions.....	2748
Table 12-711. DSS0_VID_CSC_COEF4 Instances.....	2749
Table 12-712. DSS0_VID_CSC_COEF4 Register Field Descriptions.....	2749
Table 12-713. DSS0_VID_CSC_COEF5 Instances.....	2750
Table 12-714. DSS0_VID_CSC_COEF5 Register Field Descriptions.....	2750
Table 12-715. DSS0_VID_CSC_COEF6 Instances.....	2751
Table 12-716. DSS0_VID_CSC_COEF6 Register Field Descriptions.....	2751
Table 12-717. DSS0_VID_FIRH Instances.....	2752
Table 12-718. DSS0_VID_FIRH Register Field Descriptions.....	2752
Table 12-719. DSS0_VID_FIRH2 Instances.....	2753
Table 12-720. DSS0_VID_FIRH2 Register Field Descriptions.....	2753
Table 12-721. DSS0_VID_FIRV Instances.....	2754
Table 12-722. DSS0_VID_FIRV Register Field Descriptions.....	2754
Table 12-723. DSS0_VID_FIRV2 Instances.....	2755

Table 12-724. DSS0_VID_FIRV2 Register Field Descriptions.....	2755
Table 12-725. DSS0_VID_FIR_COEF_H0_0 Instances.....	2756
Table 12-726. DSS0_VID_FIR_COEF_H0_0 Register Field Descriptions.....	2756
Table 12-727. DSS0_VID_FIR_COEF_H0_1 Instances.....	2757
Table 12-728. DSS0_VID_FIR_COEF_H0_1 Register Field Descriptions.....	2757
Table 12-729. DSS0_VID_FIR_COEF_H0_2 Instances.....	2758
Table 12-730. DSS0_VID_FIR_COEF_H0_2 Register Field Descriptions.....	2758
Table 12-731. DSS0_VID_FIR_COEF_H0_3 Instances.....	2759
Table 12-732. DSS0_VID_FIR_COEF_H0_3 Register Field Descriptions.....	2759
Table 12-733. DSS0_VID_FIR_COEF_H0_4 Instances.....	2760
Table 12-734. DSS0_VID_FIR_COEF_H0_4 Register Field Descriptions.....	2760
Table 12-735. DSS0_VID_FIR_COEF_H0_5 Instances.....	2761
Table 12-736. DSS0_VID_FIR_COEF_H0_5 Register Field Descriptions.....	2761
Table 12-737. DSS0_VID_FIR_COEF_H0_6 Instances.....	2762
Table 12-738. DSS0_VID_FIR_COEF_H0_6 Register Field Descriptions.....	2762
Table 12-739. DSS0_VID_FIR_COEF_H0_7 Instances.....	2763
Table 12-740. DSS0_VID_FIR_COEF_H0_7 Register Field Descriptions.....	2763
Table 12-741. DSS0_VID_FIR_COEF_H0_8 Instances.....	2764
Table 12-742. DSS0_VID_FIR_COEF_H0_8 Register Field Descriptions.....	2764
Table 12-743. DSS0_VID_FIR_COEF_H0_C_0 Instances.....	2765
Table 12-744. DSS0_VID_FIR_COEF_H0_C_0 Register Field Descriptions.....	2765
Table 12-745. DSS0_VID_FIR_COEF_H0_C_1 Instances.....	2766
Table 12-746. DSS0_VID_FIR_COEF_H0_C_1 Register Field Descriptions.....	2766
Table 12-747. DSS0_VID_FIR_COEF_H0_C_2 Instances.....	2767
Table 12-748. DSS0_VID_FIR_COEF_H0_C_2 Register Field Descriptions.....	2767
Table 12-749. DSS0_VID_FIR_COEF_H0_C_3 Instances.....	2768
Table 12-750. DSS0_VID_FIR_COEF_H0_C_3 Register Field Descriptions.....	2768
Table 12-751. DSS0_VID_FIR_COEF_H0_C_4 Instances.....	2769
Table 12-752. DSS0_VID_FIR_COEF_H0_C_4 Register Field Descriptions.....	2769
Table 12-753. DSS0_VID_FIR_COEF_H0_C_5 Instances.....	2770
Table 12-754. DSS0_VID_FIR_COEF_H0_C_5 Register Field Descriptions.....	2770
Table 12-755. DSS0_VID_FIR_COEF_H0_C_6 Instances.....	2771
Table 12-756. DSS0_VID_FIR_COEF_H0_C_6 Register Field Descriptions.....	2771
Table 12-757. DSS0_VID_FIR_COEF_H0_C_7 Instances.....	2772
Table 12-758. DSS0_VID_FIR_COEF_H0_C_7 Register Field Descriptions.....	2772
Table 12-759. DSS0_VID_FIR_COEF_H0_C_8 Instances.....	2773
Table 12-760. DSS0_VID_FIR_COEF_H0_C_8 Register Field Descriptions.....	2773
Table 12-761. DSS0_VID_FIR_COEF_H12_0 Instances.....	2774
Table 12-762. DSS0_VID_FIR_COEF_H12_0 Register Field Descriptions.....	2774
Table 12-763. DSS0_VID_FIR_COEF_H12_1 Instances.....	2775
Table 12-764. DSS0_VID_FIR_COEF_H12_1 Register Field Descriptions.....	2775
Table 12-765. DSS0_VID_FIR_COEF_H12_2 Instances.....	2776
Table 12-766. DSS0_VID_FIR_COEF_H12_2 Register Field Descriptions.....	2776
Table 12-767. DSS0_VID_FIR_COEF_H12_3 Instances.....	2777
Table 12-768. DSS0_VID_FIR_COEF_H12_3 Register Field Descriptions.....	2777
Table 12-769. DSS0_VID_FIR_COEF_H12_4 Instances.....	2778
Table 12-770. DSS0_VID_FIR_COEF_H12_4 Register Field Descriptions.....	2778
Table 12-771. DSS0_VID_FIR_COEF_H12_5 Instances.....	2779
Table 12-772. DSS0_VID_FIR_COEF_H12_5 Register Field Descriptions.....	2779
Table 12-773. DSS0_VID_FIR_COEF_H12_6 Instances.....	2780
Table 12-774. DSS0_VID_FIR_COEF_H12_6 Register Field Descriptions.....	2780
Table 12-775. DSS0_VID_FIR_COEF_H12_7 Instances.....	2781
Table 12-776. DSS0_VID_FIR_COEF_H12_7 Register Field Descriptions.....	2781
Table 12-777. DSS0_VID_FIR_COEF_H12_8 Instances.....	2782
Table 12-778. DSS0_VID_FIR_COEF_H12_8 Register Field Descriptions.....	2782
Table 12-779. DSS0_VID_FIR_COEF_H12_9 Instances.....	2783
Table 12-780. DSS0_VID_FIR_COEF_H12_9 Register Field Descriptions.....	2783
Table 12-781. DSS0_VID_FIR_COEF_H12_10 Instances.....	2784
Table 12-782. DSS0_VID_FIR_COEF_H12_10 Register Field Descriptions.....	2784
Table 12-783. DSS0_VID_FIR_COEF_H12_11 Instances.....	2785
Table 12-784. DSS0_VID_FIR_COEF_H12_11 Register Field Descriptions.....	2785

Table 12-785. DSS0_VID_FIR_COEF_H12_12 Instances.....	2786
Table 12-786. DSS0_VID_FIR_COEF_H12_12 Register Field Descriptions.....	2786
Table 12-787. DSS0_VID_FIR_COEF_H12_13 Instances.....	2787
Table 12-788. DSS0_VID_FIR_COEF_H12_13 Register Field Descriptions.....	2787
Table 12-789. DSS0_VID_FIR_COEF_H12_14 Instances.....	2788
Table 12-790. DSS0_VID_FIR_COEF_H12_14 Register Field Descriptions.....	2788
Table 12-791. DSS0_VID_FIR_COEF_H12_15 Instances.....	2789
Table 12-792. DSS0_VID_FIR_COEF_H12_15 Register Field Descriptions.....	2789
Table 12-793. DSS0_VID_FIR_COEF_H12_C_0 Instances.....	2790
Table 12-794. DSS0_VID_FIR_COEF_H12_C_0 Register Field Descriptions.....	2790
Table 12-795. DSS0_VID_FIR_COEF_H12_C_1 Instances.....	2791
Table 12-796. DSS0_VID_FIR_COEF_H12_C_1 Register Field Descriptions.....	2791
Table 12-797. DSS0_VID_FIR_COEF_H12_C_2 Instances.....	2792
Table 12-798. DSS0_VID_FIR_COEF_H12_C_2 Register Field Descriptions.....	2792
Table 12-799. DSS0_VID_FIR_COEF_H12_C_3 Instances.....	2793
Table 12-800. DSS0_VID_FIR_COEF_H12_C_3 Register Field Descriptions.....	2793
Table 12-801. DSS0_VID_FIR_COEF_H12_C_4 Instances.....	2794
Table 12-802. DSS0_VID_FIR_COEF_H12_C_4 Register Field Descriptions.....	2794
Table 12-803. DSS0_VID_FIR_COEF_H12_C_5 Instances.....	2795
Table 12-804. DSS0_VID_FIR_COEF_H12_C_5 Register Field Descriptions.....	2795
Table 12-805. DSS0_VID_FIR_COEF_H12_C_6 Instances.....	2796
Table 12-806. DSS0_VID_FIR_COEF_H12_C_6 Register Field Descriptions.....	2796
Table 12-807. DSS0_VID_FIR_COEF_H12_C_7 Instances.....	2797
Table 12-808. DSS0_VID_FIR_COEF_H12_C_7 Register Field Descriptions.....	2797
Table 12-809. DSS0_VID_FIR_COEF_H12_C_8 Instances.....	2798
Table 12-810. DSS0_VID_FIR_COEF_H12_C_8 Register Field Descriptions.....	2798
Table 12-811. DSS0_VID_FIR_COEF_H12_C_9 Instances.....	2799
Table 12-812. DSS0_VID_FIR_COEF_H12_C_9 Register Field Descriptions.....	2799
Table 12-813. DSS0_VID_FIR_COEF_H12_C_10 Instances.....	2800
Table 12-814. DSS0_VID_FIR_COEF_H12_C_10 Register Field Descriptions.....	2800
Table 12-815. DSS0_VID_FIR_COEF_H12_C_11 Instances.....	2801
Table 12-816. DSS0_VID_FIR_COEF_H12_C_11 Register Field Descriptions.....	2801
Table 12-817. DSS0_VID_FIR_COEF_H12_C_12 Instances.....	2802
Table 12-818. DSS0_VID_FIR_COEF_H12_C_12 Register Field Descriptions.....	2802
Table 12-819. DSS0_VID_FIR_COEF_H12_C_13 Instances.....	2803
Table 12-820. DSS0_VID_FIR_COEF_H12_C_13 Register Field Descriptions.....	2803
Table 12-821. DSS0_VID_FIR_COEF_H12_C_14 Instances.....	2804
Table 12-822. DSS0_VID_FIR_COEF_H12_C_14 Register Field Descriptions.....	2804
Table 12-823. DSS0_VID_FIR_COEF_H12_C_15 Instances.....	2805
Table 12-824. DSS0_VID_FIR_COEF_H12_C_15 Register Field Descriptions.....	2805
Table 12-825. DSS0_VID_FIR_COEF_V0_0 Instances.....	2806
Table 12-826. DSS0_VID_FIR_COEF_V0_0 Register Field Descriptions.....	2806
Table 12-827. DSS0_VID_FIR_COEF_V0_1 Instances.....	2807
Table 12-828. DSS0_VID_FIR_COEF_V0_1 Register Field Descriptions.....	2807
Table 12-829. DSS0_VID_FIR_COEF_V0_2 Instances.....	2808
Table 12-830. DSS0_VID_FIR_COEF_V0_2 Register Field Descriptions.....	2808
Table 12-831. DSS0_VID_FIR_COEF_V0_3 Instances.....	2809
Table 12-832. DSS0_VID_FIR_COEF_V0_3 Register Field Descriptions.....	2809
Table 12-833. DSS0_VID_FIR_COEF_V0_4 Instances.....	2810
Table 12-834. DSS0_VID_FIR_COEF_V0_4 Register Field Descriptions.....	2810
Table 12-835. DSS0_VID_FIR_COEF_V0_5 Instances.....	2811
Table 12-836. DSS0_VID_FIR_COEF_V0_5 Register Field Descriptions.....	2811
Table 12-837. DSS0_VID_FIR_COEF_V0_6 Instances.....	2812
Table 12-838. DSS0_VID_FIR_COEF_V0_6 Register Field Descriptions.....	2812
Table 12-839. DSS0_VID_FIR_COEF_V0_7 Instances.....	2813
Table 12-840. DSS0_VID_FIR_COEF_V0_7 Register Field Descriptions.....	2813
Table 12-841. DSS0_VID_FIR_COEF_V0_8 Instances.....	2814
Table 12-842. DSS0_VID_FIR_COEF_V0_8 Register Field Descriptions.....	2814
Table 12-843. DSS0_VID_FIR_COEF_V0_C_0 Instances.....	2815
Table 12-844. DSS0_VID_FIR_COEF_V0_C_0 Register Field Descriptions.....	2815
Table 12-845. DSS0_VID_FIR_COEF_V0_C_1 Instances.....	2816

Table 12-846. DSS0_VID_FIR_COEF_V0_C_1 Register Field Descriptions.....	2816
Table 12-847. DSS0_VID_FIR_COEF_V0_C_2 Instances.....	2817
Table 12-848. DSS0_VID_FIR_COEF_V0_C_2 Register Field Descriptions.....	2817
Table 12-849. DSS0_VID_FIR_COEF_V0_C_3 Instances.....	2818
Table 12-850. DSS0_VID_FIR_COEF_V0_C_3 Register Field Descriptions.....	2818
Table 12-851. DSS0_VID_FIR_COEF_V0_C_4 Instances.....	2819
Table 12-852. DSS0_VID_FIR_COEF_V0_C_4 Register Field Descriptions.....	2819
Table 12-853. DSS0_VID_FIR_COEF_V0_C_5 Instances.....	2820
Table 12-854. DSS0_VID_FIR_COEF_V0_C_5 Register Field Descriptions.....	2820
Table 12-855. DSS0_VID_FIR_COEF_V0_C_6 Instances.....	2821
Table 12-856. DSS0_VID_FIR_COEF_V0_C_6 Register Field Descriptions.....	2821
Table 12-857. DSS0_VID_FIR_COEF_V0_C_7 Instances.....	2822
Table 12-858. DSS0_VID_FIR_COEF_V0_C_7 Register Field Descriptions.....	2822
Table 12-859. DSS0_VID_FIR_COEF_V0_C_8 Instances.....	2823
Table 12-860. DSS0_VID_FIR_COEF_V0_C_8 Register Field Descriptions.....	2823
Table 12-861. DSS0_VID_FIR_COEF_V12_0 Instances.....	2824
Table 12-862. DSS0_VID_FIR_COEF_V12_0 Register Field Descriptions.....	2824
Table 12-863. DSS0_VID_FIR_COEF_V12_1 Instances.....	2825
Table 12-864. DSS0_VID_FIR_COEF_V12_1 Register Field Descriptions.....	2825
Table 12-865. DSS0_VID_FIR_COEF_V12_2 Instances.....	2826
Table 12-866. DSS0_VID_FIR_COEF_V12_2 Register Field Descriptions.....	2826
Table 12-867. DSS0_VID_FIR_COEF_V12_3 Instances.....	2827
Table 12-868. DSS0_VID_FIR_COEF_V12_3 Register Field Descriptions.....	2827
Table 12-869. DSS0_VID_FIR_COEF_V12_4 Instances.....	2828
Table 12-870. DSS0_VID_FIR_COEF_V12_4 Register Field Descriptions.....	2828
Table 12-871. DSS0_VID_FIR_COEF_V12_5 Instances.....	2829
Table 12-872. DSS0_VID_FIR_COEF_V12_5 Register Field Descriptions.....	2829
Table 12-873. DSS0_VID_FIR_COEF_V12_6 Instances.....	2830
Table 12-874. DSS0_VID_FIR_COEF_V12_6 Register Field Descriptions.....	2830
Table 12-875. DSS0_VID_FIR_COEF_V12_7 Instances.....	2831
Table 12-876. DSS0_VID_FIR_COEF_V12_7 Register Field Descriptions.....	2831
Table 12-877. DSS0_VID_FIR_COEF_V12_8 Instances.....	2832
Table 12-878. DSS0_VID_FIR_COEF_V12_8 Register Field Descriptions.....	2832
Table 12-879. DSS0_VID_FIR_COEF_V12_9 Instances.....	2833
Table 12-880. DSS0_VID_FIR_COEF_V12_9 Register Field Descriptions.....	2833
Table 12-881. DSS0_VID_FIR_COEF_V12_10 Instances.....	2834
Table 12-882. DSS0_VID_FIR_COEF_V12_10 Register Field Descriptions.....	2834
Table 12-883. DSS0_VID_FIR_COEF_V12_11 Instances.....	2835
Table 12-884. DSS0_VID_FIR_COEF_V12_11 Register Field Descriptions.....	2835
Table 12-885. DSS0_VID_FIR_COEF_V12_12 Instances.....	2836
Table 12-886. DSS0_VID_FIR_COEF_V12_12 Register Field Descriptions.....	2836
Table 12-887. DSS0_VID_FIR_COEF_V12_13 Instances.....	2837
Table 12-888. DSS0_VID_FIR_COEF_V12_13 Register Field Descriptions.....	2837
Table 12-889. DSS0_VID_FIR_COEF_V12_14 Instances.....	2838
Table 12-890. DSS0_VID_FIR_COEF_V12_14 Register Field Descriptions.....	2838
Table 12-891. DSS0_VID_FIR_COEF_V12_15 Instances.....	2839
Table 12-892. DSS0_VID_FIR_COEF_V12_15 Register Field Descriptions.....	2839
Table 12-893. DSS0_VID_FIR_COEF_V12_C_0 Instances.....	2840
Table 12-894. DSS0_VID_FIR_COEF_V12_C_0 Register Field Descriptions.....	2840
Table 12-895. DSS0_VID_FIR_COEF_V12_C_1 Instances.....	2841
Table 12-896. DSS0_VID_FIR_COEF_V12_C_1 Register Field Descriptions.....	2841
Table 12-897. DSS0_VID_FIR_COEF_V12_C_2 Instances.....	2842
Table 12-898. DSS0_VID_FIR_COEF_V12_C_2 Register Field Descriptions.....	2842
Table 12-899. DSS0_VID_FIR_COEF_V12_C_3 Instances.....	2843
Table 12-900. DSS0_VID_FIR_COEF_V12_C_3 Register Field Descriptions.....	2843
Table 12-901. DSS0_VID_FIR_COEF_V12_C_4 Instances.....	2844
Table 12-902. DSS0_VID_FIR_COEF_V12_C_4 Register Field Descriptions.....	2844
Table 12-903. DSS0_VID_FIR_COEF_V12_C_5 Instances.....	2845
Table 12-904. DSS0_VID_FIR_COEF_V12_C_5 Register Field Descriptions.....	2845
Table 12-905. DSS0_VID_FIR_COEF_V12_C_6 Instances.....	2846
Table 12-906. DSS0_VID_FIR_COEF_V12_C_6 Register Field Descriptions.....	2846



Table 12-907. DSS0_VID_FIR_COEF_V12_C_7 Instances.....	2847
Table 12-908. DSS0_VID_FIR_COEF_V12_C_7 Register Field Descriptions.....	2847
Table 12-909. DSS0_VID_FIR_COEF_V12_C_8 Instances.....	2848
Table 12-910. DSS0_VID_FIR_COEF_V12_C_8 Register Field Descriptions.....	2848
Table 12-911. DSS0_VID_FIR_COEF_V12_C_9 Instances.....	2849
Table 12-912. DSS0_VID_FIR_COEF_V12_C_9 Register Field Descriptions.....	2849
Table 12-913. DSS0_VID_FIR_COEF_V12_C_10 Instances.....	2850
Table 12-914. DSS0_VID_FIR_COEF_V12_C_10 Register Field Descriptions.....	2850
Table 12-915. DSS0_VID_FIR_COEF_V12_C_11 Instances.....	2851
Table 12-916. DSS0_VID_FIR_COEF_V12_C_11 Register Field Descriptions.....	2851
Table 12-917. DSS0_VID_FIR_COEF_V12_C_12 Instances.....	2852
Table 12-918. DSS0_VID_FIR_COEF_V12_C_12 Register Field Descriptions.....	2852
Table 12-919. DSS0_VID_FIR_COEF_V12_C_13 Instances.....	2853
Table 12-920. DSS0_VID_FIR_COEF_V12_C_13 Register Field Descriptions.....	2853
Table 12-921. DSS0_VID_FIR_COEF_V12_C_14 Instances.....	2854
Table 12-922. DSS0_VID_FIR_COEF_V12_C_14 Register Field Descriptions.....	2854
Table 12-923. DSS0_VID_FIR_COEF_V12_C_15 Instances.....	2855
Table 12-924. DSS0_VID_FIR_COEF_V12_C_15 Register Field Descriptions.....	2855
Table 12-925. DSS0_VID_GLOBAL_ALPHA Instances.....	2856
Table 12-926. DSS0_VID_GLOBAL_ALPHA Register Field Descriptions.....	2856
Table 12-927. DSS0_VID_MFLAG_THRESHOLD Instances.....	2857
Table 12-928. DSS0_VID_MFLAG_THRESHOLD Register Field Descriptions.....	2857
Table 12-929. DSS0_VID_PICTURE_SIZE Instances.....	2858
Table 12-930. DSS0_VID_PICTURE_SIZE Register Field Descriptions.....	2858
Table 12-931. DSS0_VID_PIXEL_INC Instances.....	2860
Table 12-932. DSS0_VID_PIXEL_INC Register Field Descriptions.....	2860
Table 12-933. DSS0_VID_PRELOAD Instances.....	2861
Table 12-934. DSS0_VID_PRELOAD Register Field Descriptions.....	2861
Table 12-935. DSS0_VID_ROW_INC Instances.....	2862
Table 12-936. DSS0_VID_ROW_INC Register Field Descriptions.....	2862
Table 12-937. DSS0_VID_SIZE Instances.....	2863
Table 12-938. DSS0_VID_SIZE Register Field Descriptions.....	2863
Table 12-939. DSS0_VID_BA_EXT_0 Instances.....	2864
Table 12-940. DSS0_VID_BA_EXT_0 Register Field Descriptions.....	2864
Table 12-941. DSS0_VID_BA_EXT_1 Instances.....	2865
Table 12-942. DSS0_VID_BA_EXT_1 Register Field Descriptions.....	2865
Table 12-943. DSS0_VID_BA_UV_EXT_0 Instances.....	2866
Table 12-944. DSS0_VID_BA_UV_EXT_0 Register Field Descriptions.....	2866
Table 12-945. DSS0_VID_BA_UV_EXT_1 Instances.....	2867
Table 12-946. DSS0_VID_BA_UV_EXT_1 Register Field Descriptions.....	2867
Table 12-947. DSS0_VID_CSC_COEF7 Instances.....	2868
Table 12-948. DSS0_VID_CSC_COEF7 Register Field Descriptions.....	2868
Table 12-949. DSS0_VID_ROW_INC_UV Instances.....	2869
Table 12-950. DSS0_VID_ROW_INC_UV Register Field Descriptions.....	2869
Table 12-951. DSS0_VID_TILE Instances.....	2870
Table 12-952. DSS0_VID_TILE Register Field Descriptions.....	2870
Table 12-953. DSS0_VID_TILE2 Instances.....	2871
Table 12-954. DSS0_VID_TILE2 Register Field Descriptions.....	2871
Table 12-955. DSS0_VID_FBDC_ATTRIBUTES Instances.....	2872
Table 12-956. DSS0_VID_FBDC_ATTRIBUTES Register Field Descriptions.....	2872
Table 12-957. DSS0_VID_FBDC_CLEAR_COLOR Instances.....	2873
Table 12-958. DSS0_VID_FBDC_CLEAR_COLOR Register Field Descriptions.....	2873
Table 12-959. DSS0_VID_CLUT_0 Instances.....	2874
Table 12-960. DSS0_VID_CLUT_0 Register Field Descriptions.....	2874
Table 12-961. DSS0_VID_CLUT_1 Instances.....	2875
Table 12-962. DSS0_VID_CLUT_1 Register Field Descriptions.....	2875
Table 12-963. DSS0_VID_CLUT_2 Instances.....	2876
Table 12-964. DSS0_VID_CLUT_2 Register Field Descriptions.....	2876
Table 12-965. DSS0_VID_CLUT_3 Instances.....	2877
Table 12-966. DSS0_VID_CLUT_3 Register Field Descriptions.....	2877
Table 12-967. DSS0_VID_CLUT_4 Instances.....	2878



Table 12-968. DSS0_VID_CLUT_4 Register Field Descriptions.....	2878
Table 12-969. DSS0_VID_CLUT_5 Instances.....	2879
Table 12-970. DSS0_VID_CLUT_5 Register Field Descriptions.....	2879
Table 12-971. DSS0_VID_CLUT_6 Instances.....	2880
Table 12-972. DSS0_VID_CLUT_6 Register Field Descriptions.....	2880
Table 12-973. DSS0_VID_CLUT_7 Instances.....	2881
Table 12-974. DSS0_VID_CLUT_7 Register Field Descriptions.....	2881
Table 12-975. DSS0_VID_CLUT_8 Instances.....	2882
Table 12-976. DSS0_VID_CLUT_8 Register Field Descriptions.....	2882
Table 12-977. DSS0_VID_CLUT_9 Instances.....	2883
Table 12-978. DSS0_VID_CLUT_9 Register Field Descriptions.....	2883
Table 12-979. DSS0_VID_CLUT_10 Instances.....	2884
Table 12-980. DSS0_VID_CLUT_10 Register Field Descriptions.....	2884
Table 12-981. DSS0_VID_CLUT_11 Instances.....	2885
Table 12-982. DSS0_VID_CLUT_11 Register Field Descriptions.....	2885
Table 12-983. DSS0_VID_CLUT_12 Instances.....	2886
Table 12-984. DSS0_VID_CLUT_12 Register Field Descriptions.....	2886
Table 12-985. DSS0_VID_CLUT_13 Instances.....	2887
Table 12-986. DSS0_VID_CLUT_13 Register Field Descriptions.....	2887
Table 12-987. DSS0_VID_CLUT_14 Instances.....	2888
Table 12-988. DSS0_VID_CLUT_14 Register Field Descriptions.....	2888
Table 12-989. DSS0_VID_CLUT_15 Instances.....	2889
Table 12-990. DSS0_VID_CLUT_15 Register Field Descriptions.....	2889
Table 12-991. DSS0_VID_SAFETY_ATTRIBUTES Instances.....	2890
Table 12-992. DSS0_VID_SAFETY_ATTRIBUTES Register Field Descriptions.....	2890
Table 12-993. DSS0_VID_SAFETY_CAPT_SIGNATURE Instances.....	2892
Table 12-994. DSS0_VID_SAFETY_CAPT_SIGNATURE Register Field Descriptions.....	2892
Table 12-995. DSS0_VID_SAFETY_POSITION Instances.....	2893
Table 12-996. DSS0_VID_SAFETY_POSITION Register Field Descriptions.....	2893
Table 12-997. DSS0_VID_SAFETY_REF_SIGNATURE Instances.....	2894
Table 12-998. DSS0_VID_SAFETY_REF_SIGNATURE Register Field Descriptions.....	2894
Table 12-999. DSS0_VID_SAFETY_SIZE Instances.....	2895
Table 12-1000. DSS0_VID_SAFETY_SIZE Register Field Descriptions.....	2895
Table 12-1001. DSS0_VID_SAFETY_LFSR_SEED Instances.....	2896
Table 12-1002. DSS0_VID_SAFETY_LFSR_SEED Register Field Descriptions.....	2896
Table 12-1003. DSS0_VID_LUMAKEY Instances.....	2897
Table 12-1004. DSS0_VID_LUMAKEY Register Field Descriptions.....	2897
Table 12-1005. DSS0_VID_DMA_BUFSIZE Instances.....	2898
Table 12-1006. DSS0_VID_DMA_BUFSIZE Register Field Descriptions.....	2898
Table 12-1007. DSS0_VID_CROP Instances.....	2899
Table 12-1008. DSS0_VID_CROP Register Field Descriptions.....	2899
Table 12-1009. DSS0_VID_SECURE Instances.....	2900
Table 12-1010. DSS0_VID_SECURE Register Field Descriptions.....	2900
Table 12-1011. DSS0_VID_PIPE_GO Instances.....	2901
Table 12-1012. DSS0_VID_PIPE_GO Register Field Descriptions.....	2901
Table 12-1013. DSS_OVR Instances.....	2902
Table 12-1014. DSS_OVR Registers.....	2902
Table 12-1015. DSS0_OVR_CONFIG Instances.....	2903
Table 12-1016. DSS0_OVR_CONFIG Register Field Descriptions.....	2903
Table 12-1017. DSS0_OVR_VIRTUALVP Instances.....	2905
Table 12-1018. DSS0_OVR_VIRTUALVP Register Field Descriptions.....	2905
Table 12-1019. DSS0_OVR_DEFAULT_COLOR Instances.....	2906
Table 12-1020. DSS0_OVR_DEFAULT_COLOR Register Field Descriptions.....	2906
Table 12-1021. DSS0_OVR_DEFAULT_COLOR2 Instances.....	2907
Table 12-1022. DSS0_OVR_DEFAULT_COLOR2 Register Field Descriptions.....	2907
Table 12-1023. DSS0_OVR_TRANS_COLOR_MAX Instances.....	2908
Table 12-1024. DSS0_OVR_TRANS_COLOR_MAX Register Field Descriptions.....	2908
Table 12-1025. DSS0_OVR_TRANS_COLOR_MAX2 Instances.....	2909
Table 12-1026. DSS0_OVR_TRANS_COLOR_MAX2 Register Field Descriptions.....	2909
Table 12-1027. DSS0_OVR_TRANS_COLOR_MIN Instances.....	2910
Table 12-1028. DSS0_OVR_TRANS_COLOR_MIN Register Field Descriptions.....	2910

Table 12-1029. DSS0_OVR_TRANS_COLOR_MIN2 Instances.....	2911
Table 12-1030. DSS0_OVR_TRANS_COLOR_MIN2 Register Field Descriptions.....	2911
Table 12-1031. DSS0_OVR_ATTRIBUTES_0 Instances.....	2912
Table 12-1032. DSS0_OVR_ATTRIBUTES_0 Register Field Descriptions.....	2912
Table 12-1033. DSS0_OVR_ATTRIBUTES_1 Instances.....	2913
Table 12-1034. DSS0_OVR_ATTRIBUTES_1 Register Field Descriptions.....	2913
Table 12-1035. DSS0_OVR_ATTRIBUTES_2 Instances.....	2914
Table 12-1036. DSS0_OVR_ATTRIBUTES_2 Register Field Descriptions.....	2914
Table 12-1037. DSS0_OVR_ATTRIBUTES_3 Instances.....	2915
Table 12-1038. DSS0_OVR_ATTRIBUTES_3 Register Field Descriptions.....	2915
Table 12-1039. DSS0_OVR_ATTRIBUTES_4 Instances.....	2916
Table 12-1040. DSS0_OVR_ATTRIBUTES_4 Register Field Descriptions.....	2916
Table 12-1041. DSS0_OVR_ATTRIBUTES2_0 Instances.....	2917
Table 12-1042. DSS0_OVR_ATTRIBUTES2_0 Register Field Descriptions.....	2917
Table 12-1043. DSS0_OVR_ATTRIBUTES2_1 Instances.....	2918
Table 12-1044. DSS0_OVR_ATTRIBUTES2_1 Register Field Descriptions.....	2918
Table 12-1045. DSS0_OVR_ATTRIBUTES2_2 Instances.....	2919
Table 12-1046. DSS0_OVR_ATTRIBUTES2_2 Register Field Descriptions.....	2919
Table 12-1047. DSS0_OVR_ATTRIBUTES2_3 Instances.....	2920
Table 12-1048. DSS0_OVR_ATTRIBUTES2_3 Register Field Descriptions.....	2920
Table 12-1049. DSS0_OVR_ATTRIBUTES2_4 Instances.....	2921
Table 12-1050. DSS0_OVR_ATTRIBUTES2_4 Register Field Descriptions.....	2921
Table 12-1051. DSS0_OVR_SECURE Instances.....	2922
Table 12-1052. DSS0_OVR_SECURE Register Field Descriptions.....	2922
Table 12-1053. DSS_VP Instances.....	2923
Table 12-1054. DSS_VP Registers.....	2923
Table 12-1055. DSS0_VP_CONFIG Instances.....	2926
Table 12-1056. DSS0_VP_CONFIG Register Field Descriptions.....	2926
Table 12-1057. DSS0_VP_CONTROL Instances.....	2929
Table 12-1058. DSS0_VP_CONTROL Register Field Descriptions.....	2929
Table 12-1059. DSS0_VP_CSC_COEF0 Instances.....	2932
Table 12-1060. DSS0_VP_CSC_COEF0 Register Field Descriptions.....	2932
Table 12-1061. DSS0_VP_CSC_COEF1 Instances.....	2933
Table 12-1062. DSS0_VP_CSC_COEF1 Register Field Descriptions.....	2933
Table 12-1063. DSS0_VP_CSC_COEF2 Instances.....	2934
Table 12-1064. DSS0_VP_CSC_COEF2 Register Field Descriptions.....	2934
Table 12-1065. DSS0_VP_DATA_CYCLE_0 Instances.....	2935
Table 12-1066. DSS0_VP_DATA_CYCLE_0 Register Field Descriptions.....	2935
Table 12-1067. DSS0_VP_DATA_CYCLE_1 Instances.....	2936
Table 12-1068. DSS0_VP_DATA_CYCLE_1 Register Field Descriptions.....	2936
Table 12-1069. DSS0_VP_DATA_CYCLE_2 Instances.....	2937
Table 12-1070. DSS0_VP_DATA_CYCLE_2 Register Field Descriptions.....	2937
Table 12-1071. DSS0_VP_LINE_NUMBER Instances.....	2938
Table 12-1072. DSS0_VP_LINE_NUMBER Register Field Descriptions.....	2938
Table 12-1073. DSS0_VP_POL_FREQ Instances.....	2939
Table 12-1074. DSS0_VP_POL_FREQ Register Field Descriptions.....	2939
Table 12-1075. DSS0_VP_SIZE_SCREEN Instances.....	2941
Table 12-1076. DSS0_VP_SIZE_SCREEN Register Field Descriptions.....	2941
Table 12-1077. DSS0_VP_TIMING_H Instances.....	2942
Table 12-1078. DSS0_VP_TIMING_H Register Field Descriptions.....	2942
Table 12-1079. DSS0_VP_TIMING_V Instances.....	2943
Table 12-1080. DSS0_VP_TIMING_V Register Field Descriptions.....	2943
Table 12-1081. DSS0_VP_CSC_COEF3 Instances.....	2944
Table 12-1082. DSS0_VP_CSC_COEF3 Register Field Descriptions.....	2944
Table 12-1083. DSS0_VP_CSC_COEF4 Instances.....	2945
Table 12-1084. DSS0_VP_CSC_COEF4 Register Field Descriptions.....	2945
Table 12-1085. DSS0_VP_CSC_COEF5 Instances.....	2946
Table 12-1086. DSS0_VP_CSC_COEF5 Register Field Descriptions.....	2946
Table 12-1087. DSS0_VP_CSC_COEF6 Instances.....	2947
Table 12-1088. DSS0_VP_CSC_COEF6 Register Field Descriptions.....	2947
Table 12-1089. DSS0_VP_CSC_COEF7 Instances.....	2948

Table 12-1090. DSS0_VP_CSC_COEF7 Register Field Descriptions.....	2948
Table 12-1091. DSS0_VP_SAFETY_ATTRIBUTES_0 Instances.....	2949
Table 12-1092. DSS0_VP_SAFETY_ATTRIBUTES_0 Register Field Descriptions.....	2949
Table 12-1093. DSS0_VP_SAFETY_ATTRIBUTES_1 Instances.....	2951
Table 12-1094. DSS0_VP_SAFETY_ATTRIBUTES_1 Register Field Descriptions.....	2951
Table 12-1095. DSS0_VP_SAFETY_ATTRIBUTES_2 Instances.....	2953
Table 12-1096. DSS0_VP_SAFETY_ATTRIBUTES_2 Register Field Descriptions.....	2953
Table 12-1097. DSS0_VP_SAFETY_ATTRIBUTES_3 Instances.....	2955
Table 12-1098. DSS0_VP_SAFETY_ATTRIBUTES_3 Register Field Descriptions.....	2955
Table 12-1099. DSS0_VP_SAFETY_ATTRIBUTES_4 Instances.....	2957
Table 12-1100. DSS0_VP_SAFETY_ATTRIBUTES_4 Register Field Descriptions.....	2957
Table 12-1101. DSS0_VP_SAFETY_ATTRIBUTES_5 Instances.....	2959
Table 12-1102. DSS0_VP_SAFETY_ATTRIBUTES_5 Register Field Descriptions.....	2959
Table 12-1103. DSS0_VP_SAFETY_ATTRIBUTES_6 Instances.....	2961
Table 12-1104. DSS0_VP_SAFETY_ATTRIBUTES_6 Register Field Descriptions.....	2961
Table 12-1105. DSS0_VP_SAFETY_ATTRIBUTES_7 Instances.....	2963
Table 12-1106. DSS0_VP_SAFETY_ATTRIBUTES_7 Register Field Descriptions.....	2963
Table 12-1107. DSS0_VP_SAFETY_CAPT_SIGNATURE_0 Instances.....	2965
Table 12-1108. DSS0_VP_SAFETY_CAPT_SIGNATURE_0 Register Field Descriptions.....	2965
Table 12-1109. DSS0_VP_SAFETY_CAPT_SIGNATURE_1 Instances.....	2966
Table 12-1110. DSS0_VP_SAFETY_CAPT_SIGNATURE_1 Register Field Descriptions.....	2966
Table 12-1111. DSS0_VP_SAFETY_CAPT_SIGNATURE_2 Instances.....	2967
Table 12-1112. DSS0_VP_SAFETY_CAPT_SIGNATURE_2 Register Field Descriptions.....	2967
Table 12-1113. DSS0_VP_SAFETY_CAPT_SIGNATURE_3 Instances.....	2968
Table 12-1114. DSS0_VP_SAFETY_CAPT_SIGNATURE_3 Register Field Descriptions.....	2968
Table 12-1115. DSS0_VP_SAFETY_CAPT_SIGNATURE_4 Instances.....	2969
Table 12-1116. DSS0_VP_SAFETY_CAPT_SIGNATURE_4 Register Field Descriptions.....	2969
Table 12-1117. DSS0_VP_SAFETY_CAPT_SIGNATURE_5 Instances.....	2970
Table 12-1118. DSS0_VP_SAFETY_CAPT_SIGNATURE_5 Register Field Descriptions.....	2970
Table 12-1119. DSS0_VP_SAFETY_CAPT_SIGNATURE_6 Instances.....	2971
Table 12-1120. DSS0_VP_SAFETY_CAPT_SIGNATURE_6 Register Field Descriptions.....	2971
Table 12-1121. DSS0_VP_SAFETY_CAPT_SIGNATURE_7 Instances.....	2972
Table 12-1122. DSS0_VP_SAFETY_CAPT_SIGNATURE_7 Register Field Descriptions.....	2972
Table 12-1123. DSS0_VP_SAFETY_POSITION_0 Instances.....	2973
Table 12-1124. DSS0_VP_SAFETY_POSITION_0 Register Field Descriptions.....	2973
Table 12-1125. DSS0_VP_SAFETY_POSITION_1 Instances.....	2974
Table 12-1126. DSS0_VP_SAFETY_POSITION_1 Register Field Descriptions.....	2974
Table 12-1127. DSS0_VP_SAFETY_POSITION_2 Instances.....	2975
Table 12-1128. DSS0_VP_SAFETY_POSITION_2 Register Field Descriptions.....	2975
Table 12-1129. DSS0_VP_SAFETY_POSITION_3 Instances.....	2976
Table 12-1130. DSS0_VP_SAFETY_POSITION_3 Register Field Descriptions.....	2976
Table 12-1131. DSS0_VP_SAFETY_POSITION_4 Instances.....	2977
Table 12-1132. DSS0_VP_SAFETY_POSITION_4 Register Field Descriptions.....	2977
Table 12-1133. DSS0_VP_SAFETY_POSITION_5 Instances.....	2978
Table 12-1134. DSS0_VP_SAFETY_POSITION_5 Register Field Descriptions.....	2978
Table 12-1135. DSS0_VP_SAFETY_POSITION_6 Instances.....	2979
Table 12-1136. DSS0_VP_SAFETY_POSITION_6 Register Field Descriptions.....	2979
Table 12-1137. DSS0_VP_SAFETY_POSITION_7 Instances.....	2980
Table 12-1138. DSS0_VP_SAFETY_POSITION_7 Register Field Descriptions.....	2980
Table 12-1139. DSS0_VP_SAFETY_REF_SIGNATURE_0 Instances.....	2981
Table 12-1140. DSS0_VP_SAFETY_REF_SIGNATURE_0 Register Field Descriptions.....	2981
Table 12-1141. DSS0_VP_SAFETY_REF_SIGNATURE_1 Instances.....	2982
Table 12-1142. DSS0_VP_SAFETY_REF_SIGNATURE_1 Register Field Descriptions.....	2982
Table 12-1143. DSS0_VP_SAFETY_REF_SIGNATURE_2 Instances.....	2983
Table 12-1144. DSS0_VP_SAFETY_REF_SIGNATURE_2 Register Field Descriptions.....	2983
Table 12-1145. DSS0_VP_SAFETY_REF_SIGNATURE_3 Instances.....	2984
Table 12-1146. DSS0_VP_SAFETY_REF_SIGNATURE_3 Register Field Descriptions.....	2984
Table 12-1147. DSS0_VP_SAFETY_REF_SIGNATURE_4 Instances.....	2985
Table 12-1148. DSS0_VP_SAFETY_REF_SIGNATURE_4 Register Field Descriptions.....	2985
Table 12-1149. DSS0_VP_SAFETY_REF_SIGNATURE_5 Instances.....	2986
Table 12-1150. DSS0_VP_SAFETY_REF_SIGNATURE_5 Register Field Descriptions.....	2986

Table 12-1151. DSS0_VP_SAFETY_REF_SIGNATURE_6 Instances.....	2987
Table 12-1152. DSS0_VP_SAFETY_REF_SIGNATURE_6 Register Field Descriptions.....	2987
Table 12-1153. DSS0_VP_SAFETY_REF_SIGNATURE_7 Instances.....	2988
Table 12-1154. DSS0_VP_SAFETY_REF_SIGNATURE_7 Register Field Descriptions.....	2988
Table 12-1155. DSS0_VP_SAFETY_SIZE_0 Instances.....	2989
Table 12-1156. DSS0_VP_SAFETY_SIZE_0 Register Field Descriptions.....	2989
Table 12-1157. DSS0_VP_SAFETY_SIZE_1 Instances.....	2990
Table 12-1158. DSS0_VP_SAFETY_SIZE_1 Register Field Descriptions.....	2990
Table 12-1159. DSS0_VP_SAFETY_SIZE_2 Instances.....	2991
Table 12-1160. DSS0_VP_SAFETY_SIZE_2 Register Field Descriptions.....	2991
Table 12-1161. DSS0_VP_SAFETY_SIZE_3 Instances.....	2992
Table 12-1162. DSS0_VP_SAFETY_SIZE_3 Register Field Descriptions.....	2992
Table 12-1163. DSS0_VP_SAFETY_SIZE_4 Instances.....	2993
Table 12-1164. DSS0_VP_SAFETY_SIZE_4 Register Field Descriptions.....	2993
Table 12-1165. DSS0_VP_SAFETY_SIZE_5 Instances.....	2994
Table 12-1166. DSS0_VP_SAFETY_SIZE_5 Register Field Descriptions.....	2994
Table 12-1167. DSS0_VP_SAFETY_SIZE_6 Instances.....	2995
Table 12-1168. DSS0_VP_SAFETY_SIZE_6 Register Field Descriptions.....	2995
Table 12-1169. DSS0_VP_SAFETY_SIZE_7 Instances.....	2996
Table 12-1170. DSS0_VP_SAFETY_SIZE_7 Register Field Descriptions.....	2996
Table 12-1171. DSS0_VP_SAFETY_LFSR_SEED Instances.....	2997
Table 12-1172. DSS0_VP_SAFETY_LFSR_SEED Register Field Descriptions.....	2997
Table 12-1173. DSS0_VP_GAMMA_TABLE_0 Instances.....	2998
Table 12-1174. DSS0_VP_GAMMA_TABLE_0 Register Field Descriptions.....	2998
Table 12-1175. DSS0_VP_GAMMA_TABLE_1 Instances.....	2999
Table 12-1176. DSS0_VP_GAMMA_TABLE_1 Register Field Descriptions.....	2999
Table 12-1177. DSS0_VP_GAMMA_TABLE_2 Instances.....	3000
Table 12-1178. DSS0_VP_GAMMA_TABLE_2 Register Field Descriptions.....	3000
Table 12-1179. DSS0_VP_GAMMA_TABLE_3 Instances.....	3001
Table 12-1180. DSS0_VP_GAMMA_TABLE_3 Register Field Descriptions.....	3001
Table 12-1181. DSS0_VP_GAMMA_TABLE_4 Instances.....	3002
Table 12-1182. DSS0_VP_GAMMA_TABLE_4 Register Field Descriptions.....	3002
Table 12-1183. DSS0_VP_GAMMA_TABLE_5 Instances.....	3003
Table 12-1184. DSS0_VP_GAMMA_TABLE_5 Register Field Descriptions.....	3003
Table 12-1185. DSS0_VP_GAMMA_TABLE_6 Instances.....	3004
Table 12-1186. DSS0_VP_GAMMA_TABLE_6 Register Field Descriptions.....	3004
Table 12-1187. DSS0_VP_GAMMA_TABLE_7 Instances.....	3005
Table 12-1188. DSS0_VP_GAMMA_TABLE_7 Register Field Descriptions.....	3005
Table 12-1189. DSS0_VP_GAMMA_TABLE_8 Instances.....	3006
Table 12-1190. DSS0_VP_GAMMA_TABLE_8 Register Field Descriptions.....	3006
Table 12-1191. DSS0_VP_GAMMA_TABLE_9 Instances.....	3007
Table 12-1192. DSS0_VP_GAMMA_TABLE_9 Register Field Descriptions.....	3007
Table 12-1193. DSS0_VP_GAMMA_TABLE_10 Instances.....	3008
Table 12-1194. DSS0_VP_GAMMA_TABLE_10 Register Field Descriptions.....	3008
Table 12-1195. DSS0_VP_GAMMA_TABLE_11 Instances.....	3009
Table 12-1196. DSS0_VP_GAMMA_TABLE_11 Register Field Descriptions.....	3009
Table 12-1197. DSS0_VP_GAMMA_TABLE_12 Instances.....	3010
Table 12-1198. DSS0_VP_GAMMA_TABLE_12 Register Field Descriptions.....	3010
Table 12-1199. DSS0_VP_GAMMA_TABLE_13 Instances.....	3011
Table 12-1200. DSS0_VP_GAMMA_TABLE_13 Register Field Descriptions.....	3011
Table 12-1201. DSS0_VP_GAMMA_TABLE_14 Instances.....	3012
Table 12-1202. DSS0_VP_GAMMA_TABLE_14 Register Field Descriptions.....	3012
Table 12-1203. DSS0_VP_GAMMA_TABLE_15 Instances.....	3013
Table 12-1204. DSS0_VP_GAMMA_TABLE_15 Register Field Descriptions.....	3013
Table 12-1205. DSS0_VP_SECURE Instances.....	3014
Table 12-1206. DSS0_VP_SECURE Register Field Descriptions.....	3014
Table 12-1207. DSS_WB Instances.....	3015
Table 12-1208. DSS_WB Registers.....	3015
Table 12-1209. DSS0_WB_ACCUH_0 Instances.....	3019
Table 12-1210. DSS0_WB_ACCUH_0 Register Field Descriptions.....	3019
Table 12-1211. DSS0_WB_ACCUH_1 Instances.....	3020



Table 12-1212. DSS0_WB_ACCUH_1 Register Field Descriptions.....	3020
Table 12-1213. DSS0_WB_ACCUH2_0 Instances.....	3021
Table 12-1214. DSS0_WB_ACCUH2_0 Register Field Descriptions.....	3021
Table 12-1215. DSS0_WB_ACCUH2_1 Instances.....	3022
Table 12-1216. DSS0_WB_ACCUH2_1 Register Field Descriptions.....	3022
Table 12-1217. DSS0_WB_ACCUV_0 Instances.....	3023
Table 12-1218. DSS0_WB_ACCUV_0 Register Field Descriptions.....	3023
Table 12-1219. DSS0_WB_ACCUV_1 Instances.....	3024
Table 12-1220. DSS0_WB_ACCUV_1 Register Field Descriptions.....	3024
Table 12-1221. DSS0_WB_ACCUV2_0 Instances.....	3025
Table 12-1222. DSS0_WB_ACCUV2_0 Register Field Descriptions.....	3025
Table 12-1223. DSS0_WB_ACCUV2_1 Instances.....	3026
Table 12-1224. DSS0_WB_ACCUV2_1 Register Field Descriptions.....	3026
Table 12-1225. DSS0_WB_ATTRIBUTES Instances.....	3027
Table 12-1226. DSS0_WB_ATTRIBUTES Register Field Descriptions.....	3027
Table 12-1227. DSS0_WB_ATTRIBUTES2 Instances.....	3031
Table 12-1228. DSS0_WB_ATTRIBUTES2 Register Field Descriptions.....	3031
Table 12-1229. DSS0_WB_BA_0 Instances.....	3033
Table 12-1230. DSS0_WB_BA_0 Register Field Descriptions.....	3033
Table 12-1231. DSS0_WB_BA_1 Instances.....	3034
Table 12-1232. DSS0_WB_BA_1 Register Field Descriptions.....	3034
Table 12-1233. DSS0_WB_BA_UV_0 Instances.....	3035
Table 12-1234. DSS0_WB_BA_UV_0 Register Field Descriptions.....	3035
Table 12-1235. DSS0_WB_BA_UV_1 Instances.....	3036
Table 12-1236. DSS0_WB_BA_UV_1 Register Field Descriptions.....	3036
Table 12-1237. DSS0_WB_BUF_SIZE_STATUS Instances.....	3037
Table 12-1238. DSS0_WB_BUF_SIZE_STATUS Register Field Descriptions.....	3037
Table 12-1239. DSS0_WB_BUF_THRESHOLD Instances.....	3038
Table 12-1240. DSS0_WB_BUF_THRESHOLD Register Field Descriptions.....	3038
Table 12-1241. DSS0_WB_CSC_COEF0 Instances.....	3039
Table 12-1242. DSS0_WB_CSC_COEF0 Register Field Descriptions.....	3039
Table 12-1243. DSS0_WB_CSC_COEF1 Instances.....	3040
Table 12-1244. DSS0_WB_CSC_COEF1 Register Field Descriptions.....	3040
Table 12-1245. DSS0_WB_CSC_COEF2 Instances.....	3041
Table 12-1246. DSS0_WB_CSC_COEF2 Register Field Descriptions.....	3041
Table 12-1247. DSS0_WB_CSC_COEF3 Instances.....	3042
Table 12-1248. DSS0_WB_CSC_COEF3 Register Field Descriptions.....	3042
Table 12-1249. DSS0_WB_CSC_COEF4 Instances.....	3043
Table 12-1250. DSS0_WB_CSC_COEF4 Register Field Descriptions.....	3043
Table 12-1251. DSS0_WB_CSC_COEF5 Instances.....	3044
Table 12-1252. DSS0_WB_CSC_COEF5 Register Field Descriptions.....	3044
Table 12-1253. DSS0_WB_CSC_COEF6 Instances.....	3045
Table 12-1254. DSS0_WB_CSC_COEF6 Register Field Descriptions.....	3045
Table 12-1255. DSS0_WB_FIRH Instances.....	3046
Table 12-1256. DSS0_WB_FIRH Register Field Descriptions.....	3046
Table 12-1257. DSS0_WB_FIRH2 Instances.....	3047
Table 12-1258. DSS0_WB_FIRH2 Register Field Descriptions.....	3047
Table 12-1259. DSS0_WB_FIRV Instances.....	3048
Table 12-1260. DSS0_WB_FIRV Register Field Descriptions.....	3048
Table 12-1261. DSS0_WB_FIRV2 Instances.....	3049
Table 12-1262. DSS0_WB_FIRV2 Register Field Descriptions.....	3049
Table 12-1263. DSS0_WB_FIR_COEF_H0_0 Instances.....	3050
Table 12-1264. DSS0_WB_FIR_COEF_H0_0 Register Field Descriptions.....	3050
Table 12-1265. DSS0_WB_FIR_COEF_H0_1 Instances.....	3051
Table 12-1266. DSS0_WB_FIR_COEF_H0_1 Register Field Descriptions.....	3051
Table 12-1267. DSS0_WB_FIR_COEF_H0_2 Instances.....	3052
Table 12-1268. DSS0_WB_FIR_COEF_H0_2 Register Field Descriptions.....	3052
Table 12-1269. DSS0_WB_FIR_COEF_H0_3 Instances.....	3053
Table 12-1270. DSS0_WB_FIR_COEF_H0_3 Register Field Descriptions.....	3053
Table 12-1271. DSS0_WB_FIR_COEF_H0_4 Instances.....	3054
Table 12-1272. DSS0_WB_FIR_COEF_H0_4 Register Field Descriptions.....	3054



Table 12-1273. DSS0_WB_FIR_COEF_H0_5 Instances.....	3055
Table 12-1274. DSS0_WB_FIR_COEF_H0_5 Register Field Descriptions.....	3055
Table 12-1275. DSS0_WB_FIR_COEF_H0_6 Instances.....	3056
Table 12-1276. DSS0_WB_FIR_COEF_H0_6 Register Field Descriptions.....	3056
Table 12-1277. DSS0_WB_FIR_COEF_H0_7 Instances.....	3057
Table 12-1278. DSS0_WB_FIR_COEF_H0_7 Register Field Descriptions.....	3057
Table 12-1279. DSS0_WB_FIR_COEF_H0_8 Instances.....	3058
Table 12-1280. DSS0_WB_FIR_COEF_H0_8 Register Field Descriptions.....	3058
Table 12-1281. DSS0_WB_FIR_COEF_H0_C_0 Instances.....	3059
Table 12-1282. DSS0_WB_FIR_COEF_H0_C_0 Register Field Descriptions.....	3059
Table 12-1283. DSS0_WB_FIR_COEF_H0_C_1 Instances.....	3060
Table 12-1284. DSS0_WB_FIR_COEF_H0_C_1 Register Field Descriptions.....	3060
Table 12-1285. DSS0_WB_FIR_COEF_H0_C_2 Instances.....	3061
Table 12-1286. DSS0_WB_FIR_COEF_H0_C_2 Register Field Descriptions.....	3061
Table 12-1287. DSS0_WB_FIR_COEF_H0_C_3 Instances.....	3062
Table 12-1288. DSS0_WB_FIR_COEF_H0_C_3 Register Field Descriptions.....	3062
Table 12-1289. DSS0_WB_FIR_COEF_H0_C_4 Instances.....	3063
Table 12-1290. DSS0_WB_FIR_COEF_H0_C_4 Register Field Descriptions.....	3063
Table 12-1291. DSS0_WB_FIR_COEF_H0_C_5 Instances.....	3064
Table 12-1292. DSS0_WB_FIR_COEF_H0_C_5 Register Field Descriptions.....	3064
Table 12-1293. DSS0_WB_FIR_COEF_H0_C_6 Instances.....	3065
Table 12-1294. DSS0_WB_FIR_COEF_H0_C_6 Register Field Descriptions.....	3065
Table 12-1295. DSS0_WB_FIR_COEF_H0_C_7 Instances.....	3066
Table 12-1296. DSS0_WB_FIR_COEF_H0_C_7 Register Field Descriptions.....	3066
Table 12-1297. DSS0_WB_FIR_COEF_H0_C_8 Instances.....	3067
Table 12-1298. DSS0_WB_FIR_COEF_H0_C_8 Register Field Descriptions.....	3067
Table 12-1299. DSS0_WB_FIR_COEF_H12_0 Instances.....	3068
Table 12-1300. DSS0_WB_FIR_COEF_H12_0 Register Field Descriptions.....	3068
Table 12-1301. DSS0_WB_FIR_COEF_H12_1 Instances.....	3069
Table 12-1302. DSS0_WB_FIR_COEF_H12_1 Register Field Descriptions.....	3069
Table 12-1303. DSS0_WB_FIR_COEF_H12_2 Instances.....	3070
Table 12-1304. DSS0_WB_FIR_COEF_H12_2 Register Field Descriptions.....	3070
Table 12-1305. DSS0_WB_FIR_COEF_H12_3 Instances.....	3071
Table 12-1306. DSS0_WB_FIR_COEF_H12_3 Register Field Descriptions.....	3071
Table 12-1307. DSS0_WB_FIR_COEF_H12_4 Instances.....	3072
Table 12-1308. DSS0_WB_FIR_COEF_H12_4 Register Field Descriptions.....	3072
Table 12-1309. DSS0_WB_FIR_COEF_H12_5 Instances.....	3073
Table 12-1310. DSS0_WB_FIR_COEF_H12_5 Register Field Descriptions.....	3073
Table 12-1311. DSS0_WB_FIR_COEF_H12_6 Instances.....	3074
Table 12-1312. DSS0_WB_FIR_COEF_H12_6 Register Field Descriptions.....	3074
Table 12-1313. DSS0_WB_FIR_COEF_H12_7 Instances.....	3075
Table 12-1314. DSS0_WB_FIR_COEF_H12_7 Register Field Descriptions.....	3075
Table 12-1315. DSS0_WB_FIR_COEF_H12_8 Instances.....	3076
Table 12-1316. DSS0_WB_FIR_COEF_H12_8 Register Field Descriptions.....	3076
Table 12-1317. DSS0_WB_FIR_COEF_H12_9 Instances.....	3077
Table 12-1318. DSS0_WB_FIR_COEF_H12_9 Register Field Descriptions.....	3077
Table 12-1319. DSS0_WB_FIR_COEF_H12_10 Instances.....	3078
Table 12-1320. DSS0_WB_FIR_COEF_H12_10 Register Field Descriptions.....	3078
Table 12-1321. DSS0_WB_FIR_COEF_H12_11 Instances.....	3079
Table 12-1322. DSS0_WB_FIR_COEF_H12_11 Register Field Descriptions.....	3079
Table 12-1323. DSS0_WB_FIR_COEF_H12_12 Instances.....	3080
Table 12-1324. DSS0_WB_FIR_COEF_H12_12 Register Field Descriptions.....	3080
Table 12-1325. DSS0_WB_FIR_COEF_H12_13 Instances.....	3081
Table 12-1326. DSS0_WB_FIR_COEF_H12_13 Register Field Descriptions.....	3081
Table 12-1327. DSS0_WB_FIR_COEF_H12_14 Instances.....	3082
Table 12-1328. DSS0_WB_FIR_COEF_H12_14 Register Field Descriptions.....	3082
Table 12-1329. DSS0_WB_FIR_COEF_H12_15 Instances.....	3083
Table 12-1330. DSS0_WB_FIR_COEF_H12_15 Register Field Descriptions.....	3083
Table 12-1331. DSS0_WB_FIR_COEF_H12_C_0 Instances.....	3084
Table 12-1332. DSS0_WB_FIR_COEF_H12_C_0 Register Field Descriptions.....	3084
Table 12-1333. DSS0_WB_FIR_COEF_H12_C_1 Instances.....	3085

Table 12-1334. DSS0_WB_FIR_COEF_H12_C_1 Register Field Descriptions.....	3085
Table 12-1335. DSS0_WB_FIR_COEF_H12_C_2 Instances.....	3086
Table 12-1336. DSS0_WB_FIR_COEF_H12_C_2 Register Field Descriptions.....	3086
Table 12-1337. DSS0_WB_FIR_COEF_H12_C_3 Instances.....	3087
Table 12-1338. DSS0_WB_FIR_COEF_H12_C_3 Register Field Descriptions.....	3087
Table 12-1339. DSS0_WB_FIR_COEF_H12_C_4 Instances.....	3088
Table 12-1340. DSS0_WB_FIR_COEF_H12_C_4 Register Field Descriptions.....	3088
Table 12-1341. DSS0_WB_FIR_COEF_H12_C_5 Instances.....	3089
Table 12-1342. DSS0_WB_FIR_COEF_H12_C_5 Register Field Descriptions.....	3089
Table 12-1343. DSS0_WB_FIR_COEF_H12_C_6 Instances.....	3090
Table 12-1344. DSS0_WB_FIR_COEF_H12_C_6 Register Field Descriptions.....	3090
Table 12-1345. DSS0_WB_FIR_COEF_H12_C_7 Instances.....	3091
Table 12-1346. DSS0_WB_FIR_COEF_H12_C_7 Register Field Descriptions.....	3091
Table 12-1347. DSS0_WB_FIR_COEF_H12_C_8 Instances.....	3092
Table 12-1348. DSS0_WB_FIR_COEF_H12_C_8 Register Field Descriptions.....	3092
Table 12-1349. DSS0_WB_FIR_COEF_H12_C_9 Instances.....	3093
Table 12-1350. DSS0_WB_FIR_COEF_H12_C_9 Register Field Descriptions.....	3093
Table 12-1351. DSS0_WB_FIR_COEF_H12_C_10 Instances.....	3094
Table 12-1352. DSS0_WB_FIR_COEF_H12_C_10 Register Field Descriptions.....	3094
Table 12-1353. DSS0_WB_FIR_COEF_H12_C_11 Instances.....	3095
Table 12-1354. DSS0_WB_FIR_COEF_H12_C_11 Register Field Descriptions.....	3095
Table 12-1355. DSS0_WB_FIR_COEF_H12_C_12 Instances.....	3096
Table 12-1356. DSS0_WB_FIR_COEF_H12_C_12 Register Field Descriptions.....	3096
Table 12-1357. DSS0_WB_FIR_COEF_H12_C_13 Instances.....	3097
Table 12-1358. DSS0_WB_FIR_COEF_H12_C_13 Register Field Descriptions.....	3097
Table 12-1359. DSS0_WB_FIR_COEF_H12_C_14 Instances.....	3098
Table 12-1360. DSS0_WB_FIR_COEF_H12_C_14 Register Field Descriptions.....	3098
Table 12-1361. DSS0_WB_FIR_COEF_H12_C_15 Instances.....	3099
Table 12-1362. DSS0_WB_FIR_COEF_H12_C_15 Register Field Descriptions.....	3099
Table 12-1363. DSS0_WB_FIR_COEF_V0_0 Instances.....	3100
Table 12-1364. DSS0_WB_FIR_COEF_V0_0 Register Field Descriptions.....	3100
Table 12-1365. DSS0_WB_FIR_COEF_V0_1 Instances.....	3101
Table 12-1366. DSS0_WB_FIR_COEF_V0_1 Register Field Descriptions.....	3101
Table 12-1367. DSS0_WB_FIR_COEF_V0_2 Instances.....	3102
Table 12-1368. DSS0_WB_FIR_COEF_V0_2 Register Field Descriptions.....	3102
Table 12-1369. DSS0_WB_FIR_COEF_V0_3 Instances.....	3103
Table 12-1370. DSS0_WB_FIR_COEF_V0_3 Register Field Descriptions.....	3103
Table 12-1371. DSS0_WB_FIR_COEF_V0_4 Instances.....	3104
Table 12-1372. DSS0_WB_FIR_COEF_V0_4 Register Field Descriptions.....	3104
Table 12-1373. DSS0_WB_FIR_COEF_V0_5 Instances.....	3105
Table 12-1374. DSS0_WB_FIR_COEF_V0_5 Register Field Descriptions.....	3105
Table 12-1375. DSS0_WB_FIR_COEF_V0_6 Instances.....	3106
Table 12-1376. DSS0_WB_FIR_COEF_V0_6 Register Field Descriptions.....	3106
Table 12-1377. DSS0_WB_FIR_COEF_V0_7 Instances.....	3107
Table 12-1378. DSS0_WB_FIR_COEF_V0_7 Register Field Descriptions.....	3107
Table 12-1379. DSS0_WB_FIR_COEF_V0_8 Instances.....	3108
Table 12-1380. DSS0_WB_FIR_COEF_V0_8 Register Field Descriptions.....	3108
Table 12-1381. DSS0_WB_FIR_COEF_V0_C_0 Instances.....	3109
Table 12-1382. DSS0_WB_FIR_COEF_V0_C_0 Register Field Descriptions.....	3109
Table 12-1383. DSS0_WB_FIR_COEF_V0_C_1 Instances.....	3110
Table 12-1384. DSS0_WB_FIR_COEF_V0_C_1 Register Field Descriptions.....	3110
Table 12-1385. DSS0_WB_FIR_COEF_V0_C_2 Instances.....	3111
Table 12-1386. DSS0_WB_FIR_COEF_V0_C_2 Register Field Descriptions.....	3111
Table 12-1387. DSS0_WB_FIR_COEF_V0_C_3 Instances.....	3112
Table 12-1388. DSS0_WB_FIR_COEF_V0_C_3 Register Field Descriptions.....	3112
Table 12-1389. DSS0_WB_FIR_COEF_V0_C_4 Instances.....	3113
Table 12-1390. DSS0_WB_FIR_COEF_V0_C_4 Register Field Descriptions.....	3113
Table 12-1391. DSS0_WB_FIR_COEF_V0_C_5 Instances.....	3114
Table 12-1392. DSS0_WB_FIR_COEF_V0_C_5 Register Field Descriptions.....	3114
Table 12-1393. DSS0_WB_FIR_COEF_V0_C_6 Instances.....	3115
Table 12-1394. DSS0_WB_FIR_COEF_V0_C_6 Register Field Descriptions.....	3115

Table 12-1395. DSS0_WB_FIR_COEF_V0_C_7 Instances.....	3116
Table 12-1396. DSS0_WB_FIR_COEF_V0_C_7 Register Field Descriptions.....	3116
Table 12-1397. DSS0_WB_FIR_COEF_V0_C_8 Instances.....	3117
Table 12-1398. DSS0_WB_FIR_COEF_V0_C_8 Register Field Descriptions.....	3117
Table 12-1399. DSS0_WB_FIR_COEF_V12_0 Instances.....	3118
Table 12-1400. DSS0_WB_FIR_COEF_V12_0 Register Field Descriptions.....	3118
Table 12-1401. DSS0_WB_FIR_COEF_V12_1 Instances.....	3119
Table 12-1402. DSS0_WB_FIR_COEF_V12_1 Register Field Descriptions.....	3119
Table 12-1403. DSS0_WB_FIR_COEF_V12_2 Instances.....	3120
Table 12-1404. DSS0_WB_FIR_COEF_V12_2 Register Field Descriptions.....	3120
Table 12-1405. DSS0_WB_FIR_COEF_V12_3 Instances.....	3121
Table 12-1406. DSS0_WB_FIR_COEF_V12_3 Register Field Descriptions.....	3121
Table 12-1407. DSS0_WB_FIR_COEF_V12_4 Instances.....	3122
Table 12-1408. DSS0_WB_FIR_COEF_V12_4 Register Field Descriptions.....	3122
Table 12-1409. DSS0_WB_FIR_COEF_V12_5 Instances.....	3123
Table 12-1410. DSS0_WB_FIR_COEF_V12_5 Register Field Descriptions.....	3123
Table 12-1411. DSS0_WB_FIR_COEF_V12_6 Instances.....	3124
Table 12-1412. DSS0_WB_FIR_COEF_V12_6 Register Field Descriptions.....	3124
Table 12-1413. DSS0_WB_FIR_COEF_V12_7 Instances.....	3125
Table 12-1414. DSS0_WB_FIR_COEF_V12_7 Register Field Descriptions.....	3125
Table 12-1415. DSS0_WB_FIR_COEF_V12_8 Instances.....	3126
Table 12-1416. DSS0_WB_FIR_COEF_V12_8 Register Field Descriptions.....	3126
Table 12-1417. DSS0_WB_FIR_COEF_V12_9 Instances.....	3127
Table 12-1418. DSS0_WB_FIR_COEF_V12_9 Register Field Descriptions.....	3127
Table 12-1419. DSS0_WB_FIR_COEF_V12_10 Instances.....	3128
Table 12-1420. DSS0_WB_FIR_COEF_V12_10 Register Field Descriptions.....	3128
Table 12-1421. DSS0_WB_FIR_COEF_V12_11 Instances.....	3129
Table 12-1422. DSS0_WB_FIR_COEF_V12_11 Register Field Descriptions.....	3129
Table 12-1423. DSS0_WB_FIR_COEF_V12_12 Instances.....	3130
Table 12-1424. DSS0_WB_FIR_COEF_V12_12 Register Field Descriptions.....	3130
Table 12-1425. DSS0_WB_FIR_COEF_V12_13 Instances.....	3131
Table 12-1426. DSS0_WB_FIR_COEF_V12_13 Register Field Descriptions.....	3131
Table 12-1427. DSS0_WB_FIR_COEF_V12_14 Instances.....	3132
Table 12-1428. DSS0_WB_FIR_COEF_V12_14 Register Field Descriptions.....	3132
Table 12-1429. DSS0_WB_FIR_COEF_V12_15 Instances.....	3133
Table 12-1430. DSS0_WB_FIR_COEF_V12_15 Register Field Descriptions.....	3133
Table 12-1431. DSS0_WB_FIR_COEF_V12_C_0 Instances.....	3134
Table 12-1432. DSS0_WB_FIR_COEF_V12_C_0 Register Field Descriptions.....	3134
Table 12-1433. DSS0_WB_FIR_COEF_V12_C_1 Instances.....	3135
Table 12-1434. DSS0_WB_FIR_COEF_V12_C_1 Register Field Descriptions.....	3135
Table 12-1435. DSS0_WB_FIR_COEF_V12_C_2 Instances.....	3136
Table 12-1436. DSS0_WB_FIR_COEF_V12_C_2 Register Field Descriptions.....	3136
Table 12-1437. DSS0_WB_FIR_COEF_V12_C_3 Instances.....	3137
Table 12-1438. DSS0_WB_FIR_COEF_V12_C_3 Register Field Descriptions.....	3137
Table 12-1439. DSS0_WB_FIR_COEF_V12_C_4 Instances.....	3138
Table 12-1440. DSS0_WB_FIR_COEF_V12_C_4 Register Field Descriptions.....	3138
Table 12-1441. DSS0_WB_FIR_COEF_V12_C_5 Instances.....	3139
Table 12-1442. DSS0_WB_FIR_COEF_V12_C_5 Register Field Descriptions.....	3139
Table 12-1443. DSS0_WB_FIR_COEF_V12_C_6 Instances.....	3140
Table 12-1444. DSS0_WB_FIR_COEF_V12_C_6 Register Field Descriptions.....	3140
Table 12-1445. DSS0_WB_FIR_COEF_V12_C_7 Instances.....	3141
Table 12-1446. DSS0_WB_FIR_COEF_V12_C_7 Register Field Descriptions.....	3141
Table 12-1447. DSS0_WB_FIR_COEF_V12_C_8 Instances.....	3142
Table 12-1448. DSS0_WB_FIR_COEF_V12_C_8 Register Field Descriptions.....	3142
Table 12-1449. DSS0_WB_FIR_COEF_V12_C_9 Instances.....	3143
Table 12-1450. DSS0_WB_FIR_COEF_V12_C_9 Register Field Descriptions.....	3143
Table 12-1451. DSS0_WB_FIR_COEF_V12_C_10 Instances.....	3144
Table 12-1452. DSS0_WB_FIR_COEF_V12_C_10 Register Field Descriptions.....	3144
Table 12-1453. DSS0_WB_FIR_COEF_V12_C_11 Instances.....	3145
Table 12-1454. DSS0_WB_FIR_COEF_V12_C_11 Register Field Descriptions.....	3145
Table 12-1455. DSS0_WB_FIR_COEF_V12_C_12 Instances.....	3146

Table 12-1456. DSS0_WB_FIR_COEF_V12_C_12 Register Field Descriptions.....	3146
Table 12-1457. DSS0_WB_FIR_COEF_V12_C_13 Instances.....	3147
Table 12-1458. DSS0_WB_FIR_COEF_V12_C_13 Register Field Descriptions.....	3147
Table 12-1459. DSS0_WB_FIR_COEF_V12_C_14 Instances.....	3148
Table 12-1460. DSS0_WB_FIR_COEF_V12_C_14 Register Field Descriptions.....	3148
Table 12-1461. DSS0_WB_FIR_COEF_V12_C_15 Instances.....	3149
Table 12-1462. DSS0_WB_FIR_COEF_V12_C_15 Register Field Descriptions.....	3149
Table 12-1463. DSS0_WB_MFLAG_THRESHOLD Instances.....	3150
Table 12-1464. DSS0_WB_MFLAG_THRESHOLD Register Field Descriptions.....	3150
Table 12-1465. DSS0_WB_PICTURE_SIZE Instances.....	3151
Table 12-1466. DSS0_WB_PICTURE_SIZE Register Field Descriptions.....	3151
Table 12-1467. DSS0_WB_SIZE Instances.....	3152
Table 12-1468. DSS0_WB_SIZE Register Field Descriptions.....	3152
Table 12-1469. DSS0_WB_POSITION Instances.....	3153
Table 12-1470. DSS0_WB_POSITION Register Field Descriptions.....	3153
Table 12-1471. DSS0_WB_CSC_COEF7 Instances.....	3154
Table 12-1472. DSS0_WB_CSC_COEF7 Register Field Descriptions.....	3154
Table 12-1473. DSS0_WB_ROW_INC Instances.....	3155
Table 12-1474. DSS0_WB_ROW_INC Register Field Descriptions.....	3155
Table 12-1475. DSS0_WB_ROW_INC_UV Instances.....	3156
Table 12-1476. DSS0_WB_ROW_INC_UV Register Field Descriptions.....	3156
Table 12-1477. DSS0_WB_BA_EXT_0 Instances.....	3157
Table 12-1478. DSS0_WB_BA_EXT_0 Register Field Descriptions.....	3157
Table 12-1479. DSS0_WB_BA_EXT_1 Instances.....	3158
Table 12-1480. DSS0_WB_BA_EXT_1 Register Field Descriptions.....	3158
Table 12-1481. DSS0_WB_BA_UV_EXT_0 Instances.....	3159
Table 12-1482. DSS0_WB_BA_UV_EXT_0 Register Field Descriptions.....	3159
Table 12-1483. DSS0_WB_BA_UV_EXT_1 Instances.....	3160
Table 12-1484. DSS0_WB_BA_UV_EXT_1 Register Field Descriptions.....	3160
Table 12-1485. DSS0_WB_SECURE Instances.....	3161
Table 12-1486. DSS0_WB_SECURE Register Field Descriptions.....	3161
Table 12-1487. DSI Clocking.....	3163
Table 12-1488. DSI Interrupts.....	3166
Table 12-1489. DSI Pixel Format Interoperability between DISPC and DSITX.....	3167
Table 12-1490. Secure Display Support.....	3168
Table 12-1491. DSI Direct Command Mode Registers.....	3170
Table 12-1492. DSI Main Settings Register Description.....	3170
Table 12-1493. Example of Required Time between Two Write Accesses.....	3174
Table 12-1494. Trigger Mapping.....	3180
Table 12-1495. TE Timeout Programming.....	3182
Table 12-1496. Interface Between Return Path and DPHY.....	3187
Table 12-1497. Timing Parameters.....	3198
Table 12-1498. Behavior of the DSITX Transmission when Both Video and Command are Running.....	3219
Table 12-1499. DPI and DSI Parameters - 1.....	3220
Table 12-1500. DPI and DSI Parameters - 2.....	3221
Table 12-1501. Example Video Operation.....	3222
Table 12-1502. VSYNC Parameter 1.....	3222
Table 12-1503. DSITX Video Stream Variable Refresh.....	3224
Table 12-1504. EDP Wrapper DPI Interface Signals.....	3226
Table 12-1505. EDP Secure Video Contention Protection.....	3227
Table 12-1506. EDP Wrapper Video Interface Pixel Mapping (MSB Mapping).....	3229
Table 12-1507. EDP Audio I2S Signals/Timing.....	3229
Table 12-1508. EDP DSC Usage Models.....	3231
Table 12-1509. EDP Clocks.....	3237
Table 12-1510. EDP Subsystem Interrupts.....	3237
Table 12-1511. EDP_INTR Interrupts.....	3238
Table 12-1512. EDP_INTR_ASF Interrupts.....	3238
Table 12-1513. EDP MHDPTX Controller Memories.....	3238
Table 12-1514. EDP DSC Memories.....	3239
Table 12-1515. EDP ECC_Aggregator (Based on Clock Group).....	3239
Table 12-1516. EDP Memory Map Region - Firewall Secure Mode Settings.....	3241



Table 12-1517. EDP MHDPTX Controller APB/SAPB Memory Map and Access Modes.....	3242
Table 12-1518. EDP MHDPTX Controller Software Events.....	3245
Table 12-1519. EDP HDCP Commands With Secret Information.....	3248
Table 12-1520. EDP PHY (SERDES) Programming Details.....	3252
Table 12-1521. CSI_RX_IF Modules Allocation Across Device Domains.....	3255
Table 12-1522. CSI_RX_IF Integration Attributes.....	3258
Table 12-1523. CSI_RX_IF Clocks and Resets.....	3258
Table 12-1524. CSI_RX_IF Hardware Requests.....	3258
Table 12-1525. CSI_RX_IF Inter-clock Dependencies.....	3263
Table 12-1526. CSI_RX_IF Interrupt Events Cross Table.....	3264
Table 12-1527. CSI_RX_IF_VBUS2APB_DEVICE_CONFIG bitfield details.....	3272
Table 12-1528. CSI_RX_IF Programming requirements for pixel modes and data sizes.....	3281
Table 12-1529. DPHY_RX Modules Allocation Across Device Domains.....	3283
Table 12-1530. DPHY_RX I/O Signals.....	3286
Table 12-1531. DPHY_RX Integration Attributes.....	3288
Table 12-1532. DPHY_RX Clocks and Resets.....	3288
Table 12-1533. Common Configuration-Related Setup.....	3292
Table 12-1534. Lane Configuration-Related Setup.....	3292
Table 12-1535. CSI_TX_IF0 Modules Allocation within Device Domains.....	3297
Table 12-1536. CSI_TX_IF Integration Attributes.....	3299
Table 12-1537. CSI_TX_IF Clocks and Resets.....	3299
Table 12-1538. CSI_TX_IF Hardware Requests.....	3300
Table 12-1539. CSI_TX_IF Color Bar format details.....	3303
Table 12-1540. CSI_TX_IF Inter-clock Dependencies.....	3304
Table 12-1541. CSI_TX_IF Interrupt Events Cross Table.....	3305
Table 12-1542. Lane Manager FSM State Description.....	3313
Table 12-1543. Data Lanes Control FSM State Description.....	3313
Table 12-1544. HS Clock Lane Control FSM State Description.....	3314
Table 12-1545. ULPS Clock Lane Control FSM State Description.....	3314
Table 12-1546. CSI_TX_IF configuration table for PSI_L.....	3315
Table 12-1547. CSI_TX_IF configuration table for PSI_L.....	3315
Table 12-1548. DPHY_TX Allocation Across Device Domains.....	3317
Table 12-1549. DPHY_TX I/O Signals in DSI Mode.....	3320
Table 12-1550. DPHY_TX I/O Signals in CSI Mode.....	3321
Table 12-1551. DPHY_TX Integration Attributes.....	3322
Table 12-1552. DPHY_TX Clocks and Resets.....	3322
Table 12-1553. VPFE Modules Allocation within Device Domains.....	3324
Table 12-1554. VPFE I/O Signals.....	3326
Table 12-1555. VPFE0 Integration Attributes.....	3329
Table 12-1556. VPFE0 Clocks and Resets.....	3329
Table 12-1557. VPFE0 Hardware Requests.....	3330
Table 12-1558. Basic Configuration of VPFE Registers.....	3334
Table 12-1559. Summary of VPFE Signal Pins and Common Input Devices.....	3335
Table 12-1560. Video Timing Reference Codes for SAV and EAV.....	3338
Table 12-1561. F, V, H Signal Descriptions.....	3338
Table 12-1562. F, H, V Protection (Error Correction) Bits.....	3339
Table 12-1563. Example for Decimation Pattern.....	3344
Table 12-1564. A-Law Table – Part 1.....	3345
Table 12-1565. A-Law Table – Part 2.....	3346
Table 12-1566. Storage Format in External Memory for Raw Data Mode.....	3349
Table 12-1567. Storage Format in External Memory for YCbCr/BT.656 Modes.....	3353
Table 12-1568. GTC Module Allocation within Device Domains.....	3354
Table 12-1569. GTC Integration Attributes.....	3355
Table 12-1570. GTC Clocks and Resets.....	3355
Table 12-1571. GTC Hardware Requests.....	3356
Table 12-1572. RTI Modules Allocation within Device Domains.....	3359
Table 12-1573. MCU_RTI Integration Attributes.....	3364
Table 12-1574. MCU_RTI Clocks and Resets.....	3364
Table 12-1575. MCU_RTI Hardware Requests.....	3364
Table 12-1576. RTI Integration Attributes.....	3367
Table 12-1577. RTI Clocks and Resets.....	3367



Table 12-1578. RTI Hardware Requests.....	3370
Table 12-1579. Timers Allocation Across Device Domains.....	3378
Table 12-1580. Timer I/O Signals.....	3382
Table 12-1581. MCU_TIMER Integration Attributes.....	3385
Table 12-1582. MCU_TIMER Clocks and Resets.....	3386
Table 12-1583. MCU_TIMER Hardware Requests.....	3388
Table 12-1584. TIMER Integration Attributes.....	3394
Table 12-1585. TIMER Clocks and Resets.....	3394
Table 12-1586. TIMER Hardware Requests.....	3406
Table 12-1587. IDLEMODE Settings.....	3413
Table 12-1588. Value Loaded in TIMER_TCR to Generate 1-ms Tick.....	3417
Table 12-1589. Prescaler/Timer Reload Values Versus Contexts.....	3420
Table 12-1590. Prescaler Clock Ratio Values.....	3422
Table 12-1591. Value and Corresponding Interrupt Period.....	3423
Table 12-1592. Global Initialization of Surrounding Modules.....	3427
Table 12-1593. Timer Module Global Initialization.....	3427
Table 12-1594. Timer Mode Configuration.....	3427
Table 12-1595. Timer Compare Mode Configuration.....	3428
Table 12-1596. Timer Capture Mode Configuration.....	3428
Table 12-1597. Initialize Capture Mode.....	3428
Table 12-1598. Detect Event.....	3429
Table 12-1599. Timer PWM Mode Configuration.....	3429
Table 12-1600. DCC Modules Allocation Across Device Domains.....	3430
Table 12-1601. MCU_DCC Integration Attributes.....	3434
Table 12-1602. MCU_DCC Clocks and Resets.....	3434
Table 12-1603. MCU_DCC Hardware Requests.....	3434
Table 12-1604. MCU_DCC Input Source Clock Mapping.....	3435
Table 12-1605. DCC Integration Attributes.....	3438
Table 12-1606. DCC Clocks and Resets.....	3439
Table 12-1607. DCC Hardware Requests.....	3440
Table 12-1608. DCC0 - DCC2 Input Source Clock Mapping.....	3443
Table 12-1609. DCC3 - DCC5 Input Source Clock Mapping.....	3445
Table 12-1610. DCC6 - DCC8 Input Source Clock Mapping.....	3447
Table 12-1611. DCC9 - DCC11 Input Source Clock Mapping.....	3449
Table 12-1612. DCC12 Input Source Clock Mapping.....	3450
Table 12-1613. DCC Instances.....	3457
Table 12-1614. DCC Registers.....	3457
Table 12-1615. DCC Registers.....	3457
Table 12-1616. DCC Registers.....	3458
Table 12-1617. DCC Registers.....	3458
Table 12-1618. DCC Registers.....	3459
Table 12-1619. DCC Registers.....	3459
Table 12-1620. DCC_GCTRL Instances.....	3460
Table 12-1621. DCC_GCTRL Register Field Descriptions.....	3460
Table 12-1622. DCC_REV Instances.....	3462
Table 12-1623. DCC_REV Register Field Descriptions.....	3462
Table 12-1624. DCC_CNTSEED0 Instances.....	3464
Table 12-1625. DCC_CNTSEED0 Register Field Descriptions.....	3464
Table 12-1626. DCC_VALIDSEED0 Instances.....	3465
Table 12-1627. DCC_VALIDSEED0 Register Field Descriptions.....	3465
Table 12-1628. DCC_CNTSEED1 Instances.....	3466
Table 12-1629. DCC_CNTSEED1 Register Field Descriptions.....	3466
Table 12-1630. DCC_STATUS Instances.....	3467
Table 12-1631. DCC_STATUS Register Field Descriptions.....	3467
Table 12-1632. DCC_CNT0 Instances.....	3469
Table 12-1633. DCC_CNT0 Register Field Descriptions.....	3469
Table 12-1634. DCC_VALID0 Instances.....	3470
Table 12-1635. DCC_VALID0 Register Field Descriptions.....	3470
Table 12-1636. DCC_CNT1 Instances.....	3471
Table 12-1637. DCC_CNT1 Register Field Descriptions.....	3471
Table 12-1638. DCC_CLKSRC1 Instances.....	3472

Table 12-1639. DCC_CLKSRC1 Register Field Descriptions.....	3472
Table 12-1640. DCC_CLKSRC0 Instances.....	3474
Table 12-1641. DCC_CLKSRC0 Register Field Descriptions.....	3474
Table 12-1642. DCC_GCTRL2 Instances.....	3476
Table 12-1643. DCC_GCTRL2 Register Field Descriptions.....	3476
Table 12-1644. DCC_STATUS2 Instances.....	3478
Table 12-1645. DCC_STATUS2 Register Field Descriptions.....	3478
Table 12-1646. DCC_ERRCNT Instances.....	3480
Table 12-1647. DCC_ERRCNT Register Field Descriptions.....	3480
Table 12-1648. ESM Modules Allocation within Device Domains.....	3481
Table 12-1649. ESM I/O Signals.....	3483
Table 12-1650. WKUP_ESM Integration Attributes.....	3486
Table 12-1651. WKUP_ESM Clocks and Resets.....	3486
Table 12-1652. WKUP_ESM Hardware Requests.....	3486
Table 12-1653. MCU_ESM Integration Attributes.....	3488
Table 12-1654. MCU_ESM Clocks and Resets.....	3489
Table 12-1655. MCU_ESM Hardware Requests.....	3489
Table 12-1656. ESM Integration Attributes.....	3491
Table 12-1657. ESM Clocks and Resets.....	3492
Table 12-1658. ESM Hardware Requests.....	3492
Table 12-1659. ESM Error Pin Scenarios.....	3503
Table 12-1660. ESM Error Pin Control Values.....	3503
Table 12-1661. MCRC Allocation Across Device Domains.....	3505
Table 12-1662. MCRC Integration Attributes.....	3506
Table 12-1663. MCRC Clocks and Resets.....	3506
Table 12-1664. MCRC Hardware Requests.....	3506
Table 12-1665. DMA Request and Counter Logic Operation According to CRC Mode.....	3513
Table 12-1666. Interrupt Conditions Per CRC Mode.....	3514
Table 12-1667. Interrupt Offset Mapping.....	3516
Table 12-1668. Device Modules and Subsystems with ECC Aggregator.....	3521
Table 12-1669. ECC Aggregator Integration Attributes.....	3525
Table 12-1670. ECC Aggregator Clocks and Resets.....	3525
Table 12-1671. ECC Aggregator Hardware Requests.....	3525



## About This Manual

This Technical Reference Manual (TRM) details the integration, the environment, the functional description, and the programming models for each peripheral and subsystem in the device.

## Related Documentation From Texas Instruments

For a complete listing of related documentation and development-support tools for the device, visit the Texas Instruments website at [www.ti.com](http://www.ti.com).

## Support Resources

[TI E2E™ support forums](#) are an engineer's go-to source for fast, verified answers and design help — straight from the experts. Search existing answers or ask your own question to get the quick design help you need.

Linked content is provided "AS IS" by the respective contributors. They do not constitute TI specifications and do not necessarily reflect TI's views; see TI's [Terms of Use](#).

## Glossary

[TI Glossary](#) This glossary lists and explains terms, acronyms, and definitions.

## Trademarks

TI E2E™ is a trademark of Texas Instruments.

HyperBus™ is a trademark of Mobiveil Inc.

Neon™, CoreSight™, and CoreLink™ are trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

DisplayPort™ is a trademark of VESA.

HD Radio™ is a trademark of iBiquity Digital Corporation.

HyperFlash™ and HyperRAM™ are trademarks of Cypress Semiconductor Corporation.

OpenCL™ is a trademark of Apple Inc.

Arm®, Cortex®, AMBA®, and Thumb® are registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

PowerVR® is a registered trademark of Imagination Technologies Limited.

VESA® is a registered trademark of VESA.

PCI-Express® and PCIe® are registered trademarks of PCI-SIG.

Cypress® is a registered trademark of Cypress Semiconductor Corporation.

OpenGL® is a registered trademark of Silicon Graphics, Inc.

MIPI® and UniPro® are registered trademarks of MIPI Alliance, Inc.

All trademarks are the property of their respective owners.

## Export Control Notice

Recipient agrees to not knowingly export or re-export, directly or indirectly, any product or technical data (as defined by the U.S., EU, and other Export Administration Regulations) including software, or any controlled product restricted by other applicable national regulations, received from disclosing party under nondisclosure obligations (if any), or any direct product of such technology, to any destination to which such export or re-export

is restricted or prohibited by U.S. or other applicable laws, without obtaining prior authorization from U.S. Department of Commerce and other competent Government authorities to the extent required by those laws.



This chapter introduces the features, subsystems, and architecture of the J721E Family of high-performance Systems-on-Chip (SoC).

---

**Note**

This document describes the superset architecture, processors and peripherals of the J721E DRA829/TDA4VM SoCs, which are part of the K3 Multicore SoC architecture platform. Not all features are available on each family of devices. The superset J721E device is available for preproduction software development. Software should constrain the features used to match the intended production device. For more information on the specific features, processors and peripherals available on a particular device, refer to the Device Comparison table in the corresponding device-specific Datasheet.

The DRA829/TDA4VM processors are hereinafter commonly referred to as *device* or *SoC*.

TI limits support for this family of SoCs to features that are supported via Software Development Kits (SDK). The SDK “build sheet” is available for download as part of each SDK and should be referenced to understand the subset of SoC hardware functionality that is available in software:

---

<b>1.1 Device Overview</b> .....	<b>80</b>
<b>1.2 Device Block Diagram</b> .....	<b>82</b>
<b>1.3 Device Main Domain</b> .....	<b>85</b>
<b>1.4 Device MCU Domain</b> .....	<b>107</b>
<b>1.5 Device WKUP Domain</b> .....	<b>114</b>
<b>1.6 Device Identification</b> .....	<b>117</b>



## 1.1 Device Overview

The DRA829 and TDA4VM SoCs are part of the K3 Multicore SoC architecture platform. The SoCs are targeted for automotive applications and aim to meet the complex processing needs of modern embedded products.

They are designed as a low power, high performance and highly integrated device architecture, adding significant enhancement on processing power, graphics capability, video and imaging processing, virtualization and coherent memory support. In addition, these SoCs support state of the art security and functional safety features.

Some of the main distinguished characteristics of the device are:

- 64-bit architecture with virtualization and coherent memory support, which leverages full processing capability of 64-bit Arm® Cortex®-A72
- Fully programmable industrial communication subsystems to enable future-proof designs for customers that need to adopt the new Gigabit Time-sensitive Networks (TSN) standards, but still need full support on legacy protocols and continuous system optimization over the product deployment
- Integration of vision hardware processing accelerators to facilitate extensive processing requirements in low power budget for automotive ADAS and machine vision applications
- Integration of a general-purpose microcontroller unit (MCU) with a dual Arm® Cortex®-R5F MCU subsystem, available for general purpose use as two cores or in lockstep, intended to help customers achieve functional safety goals for their end products
- Integration of a next-generation fixed and floating-point C71x Digital Signal Processor (DSP) that significantly boosts power over a broad range of general signal processing tasks for both general applications and automotive functions which also incorporates advanced techniques to improve control code efficiency and ease of programming such as branch prediction, protected pipeline, precise exception and virtual memory management
- Tightly coupled Matrix Multiplication Accelerator (MMA) that extends the C71x DSP architecture's scalar and vector facilities enabling deep learning and enhance vision, analytics and wide range of general applications. The achieved total TOPS (Tera Operations Per Second) performance significantly differentiates the device for single board computer in machine vision and deep learning applications
- Key display features including flexibility to interface with different panel types (eDP, DSI, DPI) with multi-layer hardware composition
- Integration of hardware features that help applications to achieve functional safety mechanisms
- Robust security architecture with sandboxed DMSC controller managing all secure configurations with high performance client-server messaging scheme between secure DMSC and all cores
- Simplified solution for power supply management, enabling lower cost system solution (on-die bias LDOs and power good comparators for minimal power sequencing requirements consistent with low cost supply design)

The device is composed of the following main subsystems, across different domains of the SoC, among others:

- One dual-core 64-bit Arm Cortex-A72 microprocessor subsystem at up to 2.0 GHz and up to 24K DMIPS (Dhrystone Million Instructions per Second)
- Up to three Microcontroller Units (MCU), based on dual-core Arm Cortex-R5F processor running at up to 1.0 GHz, up to 12K DMIPS
- Up to two TMS320C66x DSP CorePac modules running at up to 1.35 GHz, up to 40 GFLOPS
- One C71x floating point, vector DSP running at up to up to 1.0 GHz, up to 80 GFLOPS
- One deep-learning MMA, up to 8 TOPS (8b) at 1.0 GHz
- Up to two gigabit dual-core Programmable Real-Time Unit and Industrial Communication Subsystems (PRU\_ICSSG)
- Two Navigator Subsystems (NAVSS) for data movement and control
- One multi-pipeline Display Subsystem (DSS) with one MIPI® Display Serial Interface Controller (DSI) and shared MIPI D-PHY Transmitter (DPHY\_TX), one Embedded DisplayPort Transmitter (EDP) with shared Serializer/Deserializer (SERDES), and two MIPI Display Pixel Interface (DPI) ports
- Two Camera Streaming Interface Receivers (CSI\_RX\_IF) with dedicated MIPI D-PHYs (DPHY\_RX)
- One Camera Streaming Interface Transmitter (CSI\_TX\_IF) with MIPI D-PHY Transmitter (DPHY\_TX) shared with DSI

- One Vision Processing Accelerator (VPAC) with image signal processor
- One Depth and Motion Processing Accelerator (DMPAC)
- One dual-core multi-standard HD Video Decoder (DECODER)
- One dual-core multi-standard HD Video Encoder (ENCODER)
- One Graphics Processing Unit (GPU)
- One Device Management and Security Controller (DMSC)

The device provides a rich set of peripherals such as:

- General connectivity peripherals, including:
  - Two 12-bit general purpose Analog-to-Digital Converters (ADC)
  - Ten Inter-Integrated Circuit (I2C) interfaces
  - Three Improved Inter-Integrated Circuit (I3C) controllers
  - Eleven master/slave Multichannel Serial Peripheral Interfaces (MCSPI)
  - Twelve configurable Universal Asynchronous Receiver/Transmitter (UART) interfaces
  - Ten General-Purpose Input/Output (GPIO) modules
- High-speed interfaces, including:
  - Two Gigabit Ethernet Switch (CPSW) modules
  - Two Dual-Role-Device (DRD) Universal Serial Bus Subsystems (USBSS) with integrated PHY
  - Four Peripheral Component Interconnect express (PCIe) Gen3 subsystems
- Flash memory interfaces, including:
  - One Octal SPI (OSPI) interface and one Quad SPI (QSPI) or one QSPI and one HyperBus™
  - One General Purpose Memory Controller (GPMC) with Error Location Module (ELM) and 8- or 16-bit-wide data bus width (supports parallel NOR or NAND FLASH devices)
  - Three Multimedia Card/Secure Digital (MMCSD) controllers
  - One Universal Flash Storage (UFS) interface
- Industrial and control interfaces, including:
  - Sixteen Controller Area Network (MCAN) interfaces with flexible data rate support
  - Three Enhanced Capture (ECAP) modules
  - Six Enhanced Pulse-Width Modulation (EPWM) subsystems
  - Three Enhanced Quadrature Encoder Pulse (EQEP) modules
- Audio peripherals, including:
  - One Audio Tracking Logic (ATL)
  - Twelve Multichannel Audio Serial Port (MCASP) modules supporting up to 16 channels with independent TX/RX clock/sync domain
- One Video Processing Front End (VPFE) interface module

The device also integrates:

- Power distribution, reset controls and clock management components
- Power-management techniques for device power consumption minimization:
  - Adaptive Voltage Scaling (AVS)
  - Dynamic Frequency Scaling (DFS)
  - Gated clocks
  - Multiple voltage domains
  - Independently controlled power domains for major modules
  - Voltage and Temperature Management (VTM) module
  - Power-on Reset Generators (PRG)
  - Power Sleep Controllers (PSC)
- Optimized interconnect (CBASS) architecture to enable latency-critical real time network and IO applications
- Control modules (CTRL\_MMRs) mainly associated with device top-level configurations such as:
  - IO Pad and pin multiplexing configuration
  - PLL control and associated High-Speed Dividers (HSDIV)
  - Clock selection
  - Analog function controls

- Multicore Shared Memory Controller (MSMC)
- DDR Subsystem (DDRSS) with Error Correcting Code (ECC), supporting LPDDR4
- 1KB RAM with ECC support for C71x boot vectors
- 2KB RAM with ECC support for A72 and R5F boot vectors
- 512KB On-Chip SRAM protected by ECC
- One Global Time Counter (GTC) module
- Thirty 32-bit counter timers with compare and capture modes
- Debug and trace capabilities

The device includes different modules for functional safety requirements support:

- MCU island with dual lock step Arm Cortex-R5F
- Safety enabled interconnect with implemented features to help with Freedom From Interference (FFI)
- Twelve Real Time Interrupt (RTI) modules with Windowed Watchdog Timer (WWDT) functionality to monitor processor cores
- Sixteen Dual-Clock Comparators (DCC) to monitor clocking sources during run-time
- Three Error Signaling Modules (ESM) to enable error monitoring
- Temperature monitoring sensors
- ECC on all critical memories
- Dedicated hardware Memory Cyclic Redundancy Check (MCRC) blocks

The device supports the following main security functionalities among others:

- Secure Boot Management
- Public Key Accelerator (PKA) for large vector math operation
- Cryptographic acceleration (AES, 3DES, MD5, SHA1, SHA2-224, 256, 512 operation)
- Trusted Execution Environment (TEE)
- Secure storage support
- On-the-fly encryption and authentication support for OSPI interface

The device is partitioned into three functional domains, each containing specific processing cores and peripherals:

- Wake-up (WKUP) domain
- Microcontroller (MCU) domain
- MAIN domain

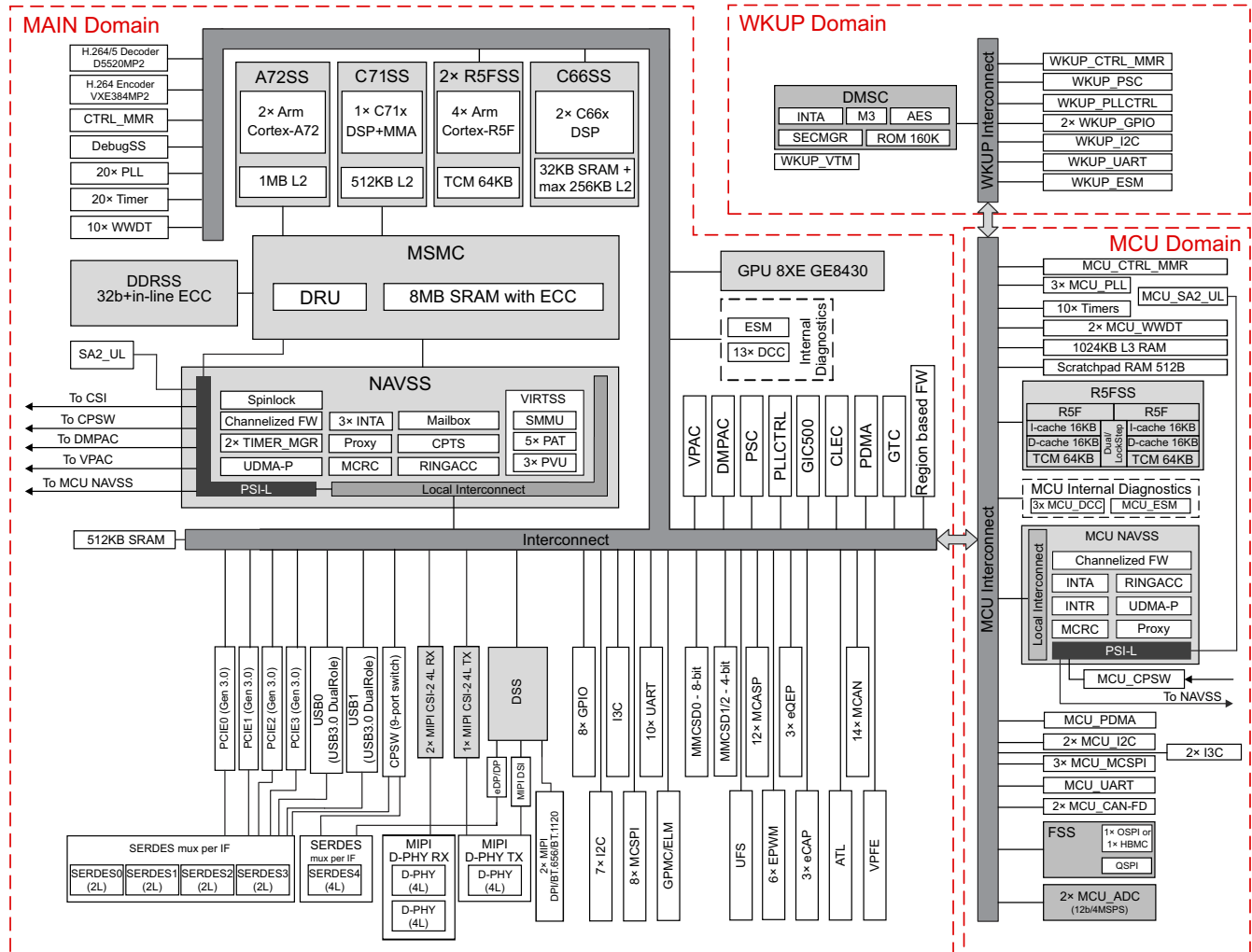
This domain fragmentation enables the device to achieve lower power dissipation profiles by allowing the power supplies to unused domains to be completely turned off and allows efficient addressing of safety requirements.

### Note

The MCU and WKUP domains are combined into a common MCU/WKUP domain in this family of devices, but for compatibility with other K3 platform SoCs, the domain naming and separation are maintained throughout this document (where applicable).

## 1.2 Device Block Diagram

Figure 1-1 shows the DRA829/TDA4VM SoCs top-level block diagram with domains partitions.



**Figure 1-1. Device Top-level Block Diagram**

Table 1-1 shows device IPs allocation within device domains.

**Table 1-1. Modules Allocation and Instances within Device Domains**

Module Full Name	Module Abbreviation	Domain		
		WKUP	MCU	MAIN
Dual-core Arm Cortex-A72 MPU	A72SS	-	-	1
Dual-core Arm Cortex-R5F Subsystem	R5FSS	-	1	2
C66x Digital Signal Processor Subsystem	C66SS	-	-	2
C71x Digital Signal Processor Subsystem with Matrix Multiplication Accelerator	C71SS	-	-	1
Arm Cortex-M3 based Device Management Security Controller	DMSC	1	-	-
Graphics Processing Unit (GE8430)	GPU	-	-	1
Multi-Standard HD Video Decoder (D5520MP2)	DECODER	-	-	1
Multi-Standard HD Video Encoder (VXE384MP2)	ENCODER	-	-	1
Vision Pre-processing Accelerator	VPAC	-	-	1
Depth and Motion Perception Accelerator	DMPAC	-	-	1
Mailbox	MAILBOX	-	-	1
Spinlock	SPINLOCK	-	-	1
Multicore Shared Memory Controller	MSMC	-	-	1

**Table 1-1. Modules Allocation and Instances within Device Domains (continued)**

Module Full Name	Module Abbreviation	Domain		
		WKUP	MCU	MAIN
DDR Subsystem	DDRSS	-	-	1
Virtualization Subsystem	VIRTSS	-	-	1
Peripheral Virtualization Unit	PVU	-	-	3
Page Based Address Translation Unit	PAT	-	-	5
Region-based Address Translation	RAT	1	2	13
Navigator Subsystem	NAVSS	-	1	1
Unified DMA Controller	UDMA	-	1	1
Ring Accelerator	RINGACC	-	1	1
Proxy	PROXY	-	1	1
Secure Proxy	SEC_PROXY	-	1	1
Interrupt Aggregator	INTR_AGGR	-	1	3
Peripheral Direct Memory Access	PDMA	-	4	13
Data Routing Unit	DRU	-	-	1
Common Platform Time Sync Module	CPTS	-	1	5
Timer Manager	TIMER_MGR	-	-	2
Analog-to-Digital Converter	ADC	-	2	-
General-Purpose Input/Output	GPIO	2	-	8
Inter-Integrated Circuit	I2C	1	2	7
Improved Inter-Integrated Circuit	I3C	-	2	1
Multichannel Serial Peripheral Interface	MCSPi	-	3	8
Universal Asynchronous Receiver/Transmitter	UART	1	1	10
Gigabit Ethernet Switch	CPSW	-	1	1
Peripheral Component Interconnect Express	PCIe	-	-	4
Universal Serial Bus Subsystem	USBSS	-	-	2
Serializer/Deserializer	SERDES	-	-	5
Flash Memory Subsystem	FSS	-	1	-
Octal Serial Peripheral Interface	OSPI	-	2	-
HyperBus Interface	HPB	-	1	-
General-Purpose Memory Controller	GPMC	-	-	1
Error Location Module	ELM	-	-	1
Multimedia Card/Secure Digital Interface	MMCSD	-	-	3
Universal Flash Storage	UFS	-	-	1
Enhanced Capture Module	ECAP	-	-	3
Enhanced Pulse Width Modulation Module	EPWM	-	-	6
Enhanced Quadrature Encoder Pulse Module	EQEP	-	-	3
Controller Area Network Interface	MCAN	-	2	14
Audio Tracking Logic	ATL	-	-	1
Multichannel Audio Serial Port	MCASP	-	-	12
Display Subsystem	DSS	-	-	1
MIPI Display Serial Interface	DSI	-	-	1
Embedded DisplayPort Transmitter	eDP	-	-	1
Camera Streaming Receiver Interface	CSI_RX_IF	-	-	2
Camera Streaming Transmitter Interface	CSI_TX_IF	-	-	1
Shared D-PHY Transmitter	DPHY_TX	-	-	1



**Table 1-1. Modules Allocation and Instances within Device Domains (continued)**

Module Full Name	Module Abbreviation	Domain		
		WKUP	MCU	MAIN
Video Processing Front End	VPFE	-	-	1
Global Timer Counter	GTC	-	-	1
Real Time Interrupt Windowed Watchdog Module	RTI	-	2	10
Timers	TIMER	-	10	20
Dual Clock Comparator	DCC	-	3	13
Error Signaling Module	ESM	1	1	1
Memory Cyclic Redundancy Check	MCRC	-	1	1

### Note

The supported set of features and peripherals is device part number dependent. For more information, see the device-specific Datasheet.

## 1.3 Device Main Domain

This section describes the modules integrated in the device MAIN domain.

[Figure 1-2](#) is a block diagram of the device MAIN domain.

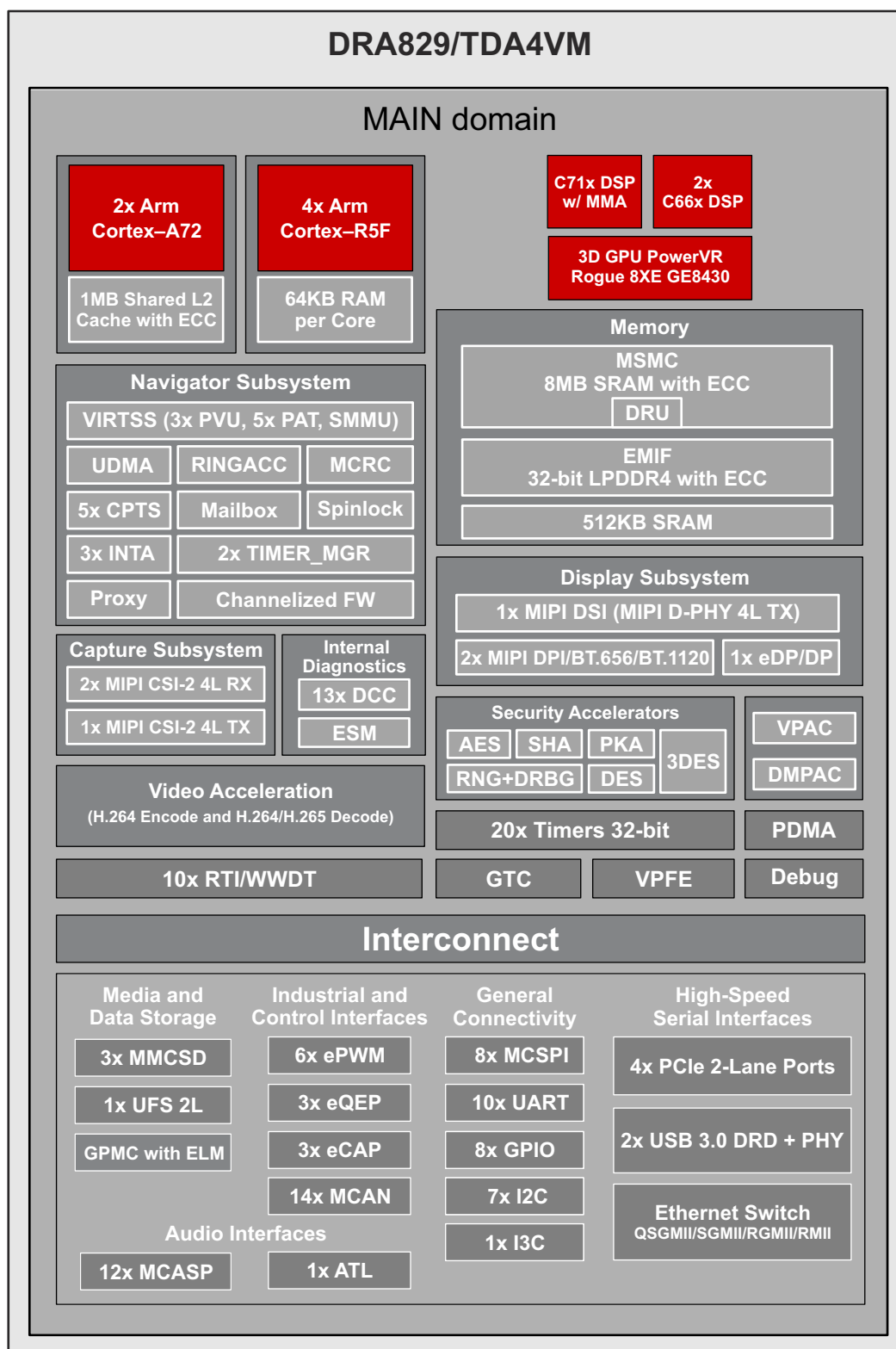


Figure 1-2. Device Main Domain Block Diagram

### Note

The supported set of features and peripherals is device part number dependent. For more information, see the device-specific Datasheet.

#### 1.3.1 Arm Cortex-A72 Subsystem

The SoC implements one dual-core Arm Cortex-A72 Microprocessor Unit (MPU). Each core has the following main features:

- Full Armv8-A architecture compliancy
- Advanced Single Instruction Multiple Data (SIMD) and floating point extension (Arm Neon™)
- Armv8 Cryptography Extensions
- Superscalar, variable length, out-of-order pipeline
- 48KB program and 32KB data Level 1 (L1) Cache
- 1MB shared Level 2 (L2) Cache
- ECC protection for L1 data cache and L2 Cache
- Parity protection for L1 Instruction Cache
- Dynamic branch prediction with Branch Target Buffer (BTB) and Global History Buffer (GHB) RAMs, return stack, and indirect predictor
- Arm General Interrupt Controller (GICv3) architecture
- Support for up to four timers within each Cortex-A72 core
- 512-bit wide, synchronous or asynchronous VBUSM.C master interface
- Arm CoreSight™ Debug and Trace Architecture
- Advanced power management for low power optimization
- SoC level dedicated RTI windowed watchdog timer per core

#### 1.3.2 Arm Cortex-R5F Processor

Integrated in MAIN domain are two instances of the dual-core Arm Cortex-R5F processor. Each dual-core instance supports the following main features:

- Armv7-R architecture
- Two modes of operation, boot-time configurable:
  - Split mode: two independently operating cores (Asymmetric Multi Processing, no coherence)
  - Lock (lockstep) mode: one main operating core with the other operating in lockstep
- 16KB instruction and 16KB data SECDED ECC protected L1 cache per core in split mode
- 64KB of Tightly Coupled Memory (TCM) per core
- Full-Precision Floating Point (VFPv3)
- 8 breakpoints, 8 watch points
- 16-region Memory Protection Unit (MPU)
- CoreSight Debug Access Port (DAP)
- CoreSight ETM-R5 interface
- Performance Monitoring Unit (PMU)
- 32-bit to 48-bit Region-based Address Translation (RAT) on memory access masters
- Integrated Vectored Interrupt Manager (VIM)
- Interfaces:
  - 64-bit VBUSM master pair (1 read, 1 write) for L3 memory accesses (per core)
  - 64-bit VBUSM slave for TCM access (per core)
  - 32-bit VBUSM master pair (1 read, 1 write) for peripheral access
  - 32-bit VBUSP master for peripheral access (per core)
  - 32-bit VBUSP slave configuration port (per core)
  - 32-bit VBUSP slave debug port

#### 1.3.3 C66x DSP Subsystem

Integrated in MAIN domain are two instances of C66x DSP based on the TI's standard TMS320C66x DSP CorePac module. Each C66x DSP supports the following main features:

- Fixed/Floating-point C66x CPU based on a superset of the C64x+ and C67x+ ISA
- Program memory controller (PMC) with 32KB L1 program memory (L1P), configurable as cache and/or SRAM
- Data memory controller (DMC) with 32KB L1 data memory (L1D), configurable as cache and/or SRAM
- Unified memory controller (UMC) with 288KB L2 memory, only part of which is cacheable
- Internal DMA (IDMA) engine
- External memory controller (EMC)
- Extended memory controller (XMC)
- Address extension/translation (32-bit to 48-bit)
- Memory protection for multiple segments and all internal L1/L2 RAMs
- Error detection and correction
- Integrated C66x CorePac interrupt controller (INTC)
- Debug/emulation capabilities

---

### Note

The C66x L1P memory is disabled (not supported) in this device.

---

## 1.3.4 C71x DSP Subsystem

Integrated single instance of C71x DSP supports the following main features:

- True 64-bit C71x CPU core with:
  - Instruction fetch unit
  - Instruction dispatch unit
  - Instruction decode unit
  - CPU dual data path with one 64-bit scalar side (side A) and one 512-bit vector side (side B)
  - CPU control logic
  - Test, debug, and interrupt logic
  - Enhanced Instruction Set Architecture (ISA)
  - OpenCL features
- Matrix Multiply Accelerator (MMA) as a special functional unit in C71x CorePac CPU
- L1 Program Memory Controller (PMC) with 32KB L1P memory, all cache
- L1 Data Memory Controller (DMC) with 48KB L1D memory, configurable as cache and/or SRAM
- L2 Unified Memory Controller (UMC) with 512KB L2 memory, configurable as cache and/or SRAM
- Multi-dimensional Streaming Engine (SE) - flexible, high bandwidth mechanism for reading large quantities of data into C71x DSP
- CorePac Memory Management Unit (CMMU)
- Power-down controller
- Debug capabilities

## 1.3.5 Graphics Processing Unit

Integrated in MAIN domain is one instance of Graphics Processing Unit (GPU) which accelerates 2-Dimensional (2D) and 3-Dimensional (3D) graphics and compute applications. It supports the following main features:

- Single core architectures based on PowerVR® Series8XE GE8430 with shared system level cache of 64KB
- Tile-based deferred rendering architecture
- Multi-threaded Unified Shading Cluster (USC) engines incorporating pixel, vertex and compute shader functionality
- Fully virtualized memory addressing for OS operation in a unified memory architecture (up to 64 GB physical address space and 1 TB virtual address space)
- PowerVR Texture Compression
- Advanced DMA driven operation for minimum host CPU interaction

- Support of Frame Buffer Compression (FBC) and Frame Buffer Compression Decompression (FBDC) features
- Programmable high-quality image anti-aliasing
- Hardware virtualization and hardware security
- API Support: RenderScript

### 1.3.6 Multi-Standard HD Video Decoder

Integrated in MAIN domain dual-core multi-standard HD Video Decoder (DECODER) is based on D5520MP2 PowerVR VPU (video processor unit). It has the following main features, among others:

- Decoding support of:
  - 1x 4Kp60 or
  - 2x 4Kp30 or
  - 4x 1080p60 or
  - 8x 1080p30
- Decoding formats supported:
  - H.265
  - H.264
  - WMV-9
  - VC-1
  - MPEG-2
  - H.263
  - DivX
  - AVC
  - RealVideo
  - VP8
  - VP6
  - Sorenson
  - JPEG
- Hardware scaling
- Rotation support: 90°, 180° and 270°
- Support of secure playback

### 1.3.7 Multi-Standard HD Video Encoder

Integrated in MAIN domain dual-core multi-standard HD Video Encoder (ENCODER) is based on VXE384MP2 PowerVR VPU (video processor unit). It has the following main features, among others:

- Encoding support of:
  - 1x 1080p60 video stream encoding or
  - 2x or 3x 1080p30 video stream encodings
- Encoding formats supported:
  - H.264 MVC
  - H.264 HP
  - H.264 MP
  - H.264 BP
  - MPEG-4 SP
  - H.263 BP
  - Still images in MPEG2 MP
  - Still images in JPEG
- Support of multiple slices per frame (H.264) under software control
- Search range of +/-128 horizontally and +/-104 vertically
- Integrated scaler
- Dedicated hardware support for
  - Integer motion estimation



- Sup-pel motion estimation
- Intra search
- Transform and inverse transform
- Quantization and inverse quantization
- Motion vector prediction and estimation
- Zigzag Scan, Run Length and Variable Length Encoding

### 1.3.8 Vision Pre-processing Accelerator

The Vision Pre-processing Accelerator (VPAC) provides a set of common vision primitive functions, performing various pixel data processing tasks, such as: color processing and enhancement, noise filtering, wide dynamic range (WDR) processing, lens distortion correction, pixel remap for de-warping, on-the-fly scale generation, on-the-fly pyramid generation, and offloads these common tasks from the main SoC processors (ARM, DSP, etc.). The VPAC includes the following processing and infrastructure sub-modules:

- Vision Imaging Sub-System (VISS) which provides raw data image processing such as:
  - Wide Dynamic Range (WDR) merge
  - Defect Pixel Correction (DPC)
  - Lens Shading Correction (LSC)
  - Global/Local Brightness and Contrast Enhancement (GLBCE)
  - Advanced Spatial Noise Filter (NSF4V)
  - Edge Enhancement (EE)
  - Demosaicing
  - Color conversion
- Lens Distortion Correction (LDC) block, which provides data reading from memory (DDR or on-chip) and applies perspective transformation as well as correction of lens distortion (including fisheye lenses).
- Multi-Scalar (MSC) block reads data from memory (DDR or on-chip) to internal shared level 2 (SL2) memory and generates up to 10 scaled outputs from one or two inputs, with various scaling ratios.
- Noise Filter (NF) block reads data from memory (DDR or on-chip) to internal SL2 memory and does Bilateral filtering to remove noise.
- Hardware Thread Scheduler (HTS) provides inter-processor communication among various VPAC sub-modules (VISS, LDC, MSC and NF) as well as with the local DMA Engine (UTC).
- Internal Shared Level 2 (SL2) memory for data exchange across VPAC sub-modules (VISS, LDC, MSC and NF) and from DDR/MSMC, using the K3 Data Movement Architecture (DMA) mechanism
- Load/Store Engine (LSE) which performs data load and store tasks on a the SL2 memory for the hardware accelerator algorithm cores

### 1.3.9 Depth and Motion Perception Accelerator

The Depth and Motion Perception Accelerator (DMPAC) computes dense stereo depth maps (*depth*) and dense optical flow vectors (*motion*) from camera inputs. The stereo and optical flow processing is partitioned into two top level sub-blocks: the Dense Optical Flow (DOF) engine and the Stereo Disparity Engine (SDE). The DOF and SDE blocks share a common local memory, DMA, external messaging and control infrastructure. The DMPAC provides the following main features, among others:

- Image resolution up to 2MPix (maximum horizontal resolution up to 2048 pixels; maximum vertical resolution up to 1024 pixels)
- Maximum throughput up to 220MPix/sec for DOF, and up to 84-100MPix/sec for SDE
- Simultaneous operation of Stereo (1MPix@30fps) and Optical flow (1MP@30fps)
- 12-bit fully packed luminance data input pixel data format (other formats supported through conversion)
- Packed 16-bit (disparity) or 32-bit (flow vector) output data formats
- Shared Level 2 (SL2) memory sub-system, which serves the data transfer of the DOF and SDE blocks
- Unified Transfer Controller (UTC), which serves as a DMA engine
- Hardware Thread Scheduler (HTS) block for messaging and control mechanism
- Counter, Timer and System Event Trace (CTSET) module, which provides event tracing capability for the Stereo and Optical Flow hardware threads

### 1.3.10 Navigator Subsystem

Instantiated in the MAIN domain one Navigator subsystem named NAVSS can be used for efficient transfer of data support between software, firmware and hardware in all combinations. It consists of the following main modules:

- Unified DMA Controller with the following main modes and features:
  - K3 DMA Architecture compliant Tx/Rx port implementation
  - K3 DMA Architecture compliant Packet-Oriented DMA Functionality (UDMA-P)
  - K3 DMA Architecture compliant Third Party Channel Controller
  - K3 DMA Architecture compliant Unified Transfer Controller
  - K3 DMA Architecture compliant Unified DMA channels which all share the execution hardware using time division multiplexing
- Ring Accelerator provides hardware acceleration to enable straightforward passing of work between a producer and a consumer and has the following main features:
  - Supports 1024 independent memory-mapped ring structures
  - Supports various modes for each ring based on usage and compatibility
  - Provides single-word deep shared incoming Transfer Response FIFO
  - Provides bit-wide source VBUSM read/write slave interface for accesses from DMA controller entities
- Proxy module with the following main features:
  - Provides proxy buffers to store large data bursts that a host can only access in smaller amounts
  - Keeps the large data coherent until the complete data has been accessed
  - Allows interleaved access between multiple hosts using multiple proxies
  - Supports a pre-configured number of target resources to proxy with pre-configured number of channels, size of each channel, and address offset per each target
- Secure proxy module is a modified version of the proxy module and in addition has the following main features:
  - Supports a number of threads, where each has their own independent proxy function
  - Supports a programmable fixed queue for each proxy thread
  - Supports multiple producers all writing to the same queue
  - Supports programmable thresholds for when to generate events
  - Supports a max message count for outbound proxy threads limiting the number of messages a thread can produce
- Interrupt Aggregator modules provide a centralized machine which handles the termination of system events to that they can be coherently processed by the host(s) in the system. Main features are as follows:
  - 64-bit VBUSP slave using 64-bit registers
  - Provide a set of TI Interrupt Architecture compliant interrupt status and mask registers which are used to pass specific event status to one or more host blocks.
  - Provide a set of Global Event Input (GEVI) counters which can count events delivered via an ingress Event Transport Lane (ETL)
  - Provides a set of Local Event Input (LEVI) to Global event registers which can be used to convert pulsed discrete interrupt inputs or clock synchronous rising edge events into Global events on an egress ETL
  - Provides a set of GEVI 'Multicast' registers which can take a Global event from an ingress ETL and generate two egress Global events on two egress ETL interfaces
- Virtualization Subsystem hardware modules with the following main features and blocks:
  - Implements all virtualization translation components
  - Implements NAVSS connectivity between Accelerator Cluster (AC), peripherals Module Subsystem (MODSS), Unified DMA (UDMA-P), and North Bridge (NB) ports
  - Supports 5 Page-based Address Translation (PAT) modules and their configuration
  - Supports 3 PVUs and their configuration
  - Supports 1 Arm SMMUv3 Translation Buffer Unit (TBU) and its configuration
  - Supports Arm SMMUv3 Translation Control Unit (TCU)
  - Supports external Arm AMBA® Distributed Translation Interface (DTI) connections

- Supports SMMU DTI interconnect
- Bridges all ARM AXI or APB interfaces to CBA VBUSM and VBUSP busses
- Peripheral Virtualization Unit module which provides TLBs (Translation Look-aside Buffers) for static virtual address translation on a CBA VBUSM bus with the following main features:
  - Implements a channelized TLB for virtual address translation
  - Supports a pre-configured number of entries per TLB channel
  - Supports TLB chaining to extend the search but at a latency penalty
  - Supports a 48-bit address size
  - Supports page sizes of 4KB, 16KB, 64KB, 2MB, 32MB, 512MB, 1GB, and 16GB
  - Support TLBs in software mode, where they are maintained by software only
  - Produces a fault interrupt when the TLB misses or upon a permission error
- Page-based Address Translation unit which performs page-based address translation on a CBA VBUSM bus with the following main features:
  - Implements a VBUSM retiming bridge with page based address translation
  - Supports up to 16384 pages, each occupying a programmable 4-KB, 16-KB, 64-KB, or 1-MB of address space
  - Supports a 48-bit output address size
  - Allows fast programming of the table, or fast generic SRAM access when the table is disabled
  - Supports *transaction* ID re-assignment
- Mailbox module to facilitate the communication between the various on-chip processors of the device by providing a queued mailbox-interrupt mechanism with the following main features:
  - 12 clusters
  - 32-bit message width
  - Message reception and queue-not-full notification using interrupts
  - Non-intrusive emulation
- Spinlock module (256 hardware semaphores) for synchronizing the processes running on multiple processors in the device.
- Two Timer Manager modules to support timing operations for the processes running on multiple processors, each with the following main features:
  - 1024 × 32-bit RAM-based independent timers (2048 in total)
  - Event interface to an interrupt aggregation module in the NAVSS subsystem with events triggering when a timer expires or when an expired timer is reset or deactivated
  - Host access to determine which timer(s) expired
  - 32 registers with individual timeout status (one bit per timer)
  - Groups of 16 timers separated into pages of 4-K address space
  - Timer bits within each page to read expiration status for each timer when software only has access to that page
  - 10 μs time to cycle through all of the timers
  - Host access to reset individual timers
  - Periodic hardware timers – a timer may be set to automatically reprogram itself upon expiration without software intervention.
- Time Sync modules to facilitate host control of time sync operations, each with the following main features:
  - Supports a selection of multiple external clock sources
  - Software control of time sync events via interrupt or polling
  - Supports 8 hardware timestamp push inputs
  - Supports timestamp counter compare output
  - Supports timestamp counter bit output
  - Supports 6 timestamp generator function outputs
  - 32-bit and 64-bit timestamp modes
- Memory Cyclic Redundancy Check module used to perform CRC to verify the integrity of a memory system with the following main features:
  - Four channels to perform background signature verification on any memory subsystem

- Data compression on 8-, 16-, 32-, and 64-bit data size
- Dedicated CRC value register per channel which contains the pre-determined CRC value
- Timed base event trigger from timer to initiate DMA data transfer
- Programmable 20-bit pattern counter per channel to count the number of data patterns for compression
- Three modes of operation: Auto, Semi-CPU, and Full-CPU
- Timeout interrupt generation if CRC is not performed within the time limit
- Per channel DMA request generation to initiate CRC value transfer

### 1.3.11 Region-based Address Translation Module

Integrated in MAIN domain thirteen instances of RAT module perform a region based address translation of a 32-bit input address into a 48-bit output address. Each RAT module provides the following main features:

- 16 regions with dedicated registers for attributes configuration:
  - Region base address
  - Region size
  - Translated base address
- Address translation for only enabled regions
- Region boundary crossing transactions error generation

### 1.3.12 Data Routing Unit

The Data Routing Unit (DRU) is a high bandwidth, flexible routing engine with programmable DMA transfer requests which enables performing of high speed data transfers between memory mapped slave endpoints, processor caches and shared caches. It behaves like a DMA transfer controller, moving data at MPU frequency and has the following main features:

- Programmable configuration registers for direct transfer request submission
- Read and write command queues
- Programmable priority for each queue
- Two dedicated ports (one read and one write) to generate independent read and write commands
- Support for region based and channelized firewall
- Independent 48-bit address fields for source and destinations
- Up to four dimensional data transfers
- Error detection and Correction

### 1.3.13 Display Subsystem

Instantiated in the MAIN domain Display Subsystem (DSS) is a flexible composition-enabled display subsystem that supports multiple high resolution display outputs. It consists of the following main modules:

- Display Controller (DISPC), with the following main features:
  - Support of multi-layer blending and transparency for each of display outputs
  - Supports write-back pipeline with scaling to enable memory-to-memory composition and/or to capture a display output for Ethernet video encoding
  - Supports gamma correction and programmable color control in both source and destination pipelines
  - Embedded DMA Controller with the following main features:
    - Support for 1D-only DMA transfers
    - Support for 48b addressable memory space
    - Support for memory fragmentation through external PAT at SoC level
    - Integrated shared buffer management for pipelines within the same DMA controller group
    - Programmable DMA requests management
    - Support for source image flip along X and Y-axis
    - Support for secure access to firewall protected frame buffer in DDR memory
  - Two input display processing Video Pipelines, each supporting:
    - Wide range of input RGB source pixel formats
    - Wide range of input YUV source pixel formats

- Programmable poly-phase filter (scaler)
- Programmable color space conversion
- Programmable Brightness/Contrast/Hue/Saturation
- Programmable Gamma Correction LUT
- Luma Key generation
- 10-bit processing pipeline
- Two input display processing Video Lite Pipelines, each supporting:
  - Wide range of input RGB source pixel formats
  - Wide range of input YUV source pixel formats
  - YUV420 to YUV422 chroma up-sampling using an average filter
  - YUV422 to YUV444 chroma up-sampling using a 4-tap filter based on Catmull-Rom algorithm
  - Programmable color space conversion
  - Programmable Brightness/Contrast/Hue/Saturation
  - Programmable Gamma Correction LUT
  - Luma Key generation
  - 10-bit processing pipeline
- One Write-back (WB) pipeline, supporting:
  - Wide range of destination RGB pixel formats
  - Wide range of destination YUV pixel formats
  - Programmable poly-phase filter (scaler)
  - Output capture and Memory-to-memory (M2M) operation modes
- Four Overlay Managers (OVR), each supporting:
  - Input pixel format: ARGB48-12121212
  - Output pixel format: ARGB48-12121212
  - Overlay of the input pipelines
  - Up to 5 input layers blending
  - Transparency color key
  - Alpha blending support: Embedded pixel alpha (ARGB and RGBA), global pixel, and combination of global pixel and pixel alpha
  - Z-order programmable (full flexibility)
  - Color bar test pattern insertion
  - Any overlay output can be selected to drive the Write-back pipeline
- Four Video Port (VP) display outputs, each supporting:
  - 36-bit per pixel on the RGB output interface
  - Independent programmable timing generator, supporting up to 600 MHz pixel clock video formats
  - Independent programmable 10-bit gamma correction
  - Independent programmable multiple cycles output format on 8/9/12/16-bit interface
  - Selection between RGB and YUV422 output pixel
  - Configurable VP output mode
- Internal diagnostic features:
  - Supports up to 4 programmable (position/size) check regions on the DISPC video port display outputs
  - Support for 1 check region on each input video pipeline output
  - MISR (Multiple Input Signature Register) used on each check region to perform data correctness check and/or freeze frame detection
- Local power features:
  - Low power saving modes
  - On-the-fly Dynamic Frequency Scaling (DFS) support
  - Capability to associate all buffers a single pipeline for a display self-refresh
- System interconnect ports:
  - Two 128-bit VBUSM master interfaces for data read/write
  - One 32-bit VBUSP slave interface for configuration



- Frame Buffer Decompression Core (FBDC), that performs a decompression on lossless compressed images on a tile-by-tile basis.
- MIPI Display Serial Interface (DSI) transmitter host controller, with the following main features:
  - Compliance with MIPI DSI 1.3.1 and previous protocol specifications
  - Compliance with Stereoscopic Display Format (SDF) specification
  - Video and command operational modes
  - Both burst and non-burst modes for video mode data transmission
  - Up to 4 virtual channels via command mode
  - Bi-directional communication and escape mode
  - Pixel clock rate range: 25-330 MHz
  - Programmable display resolutions
  - 16/18/24/30/36-bit RGB input data formats for video mode
  - RGB16, RGB18 packed, and RGB24 input data formats for command mode
  - All generic data types defined by MIPI
  - Display Command Set (DCS) transparent to the protocol engine
  - ECC on the APB interface
  - Data splitter for 2-, 3-, or 4-data lane configuration
  - Connection to a single MIPI D-PHY complex I/O through an 8-bit Protocol Peripheral Interface (PPI)
  - Tearing effect (TE) input signals for command mode display
  - Bus contention recovery
  - Video mode pattern generator: color bar pattern image and D-PHY BET testing pattern
  - APB slave interface with 32-bit data and address for configuration
- The MIPI DSI Physical Layer (D-PHY) module with the following main features:
  - Compliance with MIPI D-PHY 1.2 physical layer interface specification and features
  - 1, 2 or 4 data lanes, in addition to clock signaling
  - Maximum data rate up to 2.5 Gbps per data lane
  - Protocol Peripheral Interface (PPI)
  - HS continuous and burst mode
  - Low-Power (LP), Ultra-Lower Power Mode (ULPM), and Shutdown modes
  - Forward direction and reverse direction escape modes
  - Automatic termination control in both high-speed and low-power modes
  - Single 32-bit VBUSP slave interface
- Embedded DisplayPort (eDP) transmitter host controller with the following main features:
  - Compliance with VESA® DisplayPort™ (DP) 1.3 (with 1.4 DSC/FEC support) specification
  - Compliance with VESA Embedded DisplayPort (eDP) 1.4 specification
  - Static configuration of either DP or eDP mode
  - Link rates up to High Bit Rate 3 (HBR3)
  - Pixel clock rate range: 25-600 MHz
  - 8, 10, and 12 bpc (bits per component), in RGB/YCbCr444 colorimetry formats (CEA-861 compliant) and YCbCr422 (via simple decimation)
  - Data splitter for 1-, 2-, or 4-data lane configuration
  - Single Stream Transport (SST)
  - Multiple Stream Transport (MST)
  - High-bandwidth Digital Content Protection (HDCP) data encryption via an embedded HDCP core
  - Display Stream Compression (DSC) encoded stream data transport via an embedded DSC core
  - Forward Error Correction (FEC) encoder with/without DSC enabled in DP mode
  - Single Stream Transport (SST)
  - Audio transport features
  - Metadata transport via Main Stream Attribute (MSA) packet or via SDP
  - APB slave ports for TX/PHY controller configuration
  - SAPB (secure) slave port for secure connection
  - Video source muxing options

- One 32-bit VBUSP slave interface used for configuration
- ECC on the critical memories
- Parity check on the configuration interface
- Encoder self-check diagnostics support in the DSC core
- Injection of ECC and parity errors
- eDP (Physical Layer) SERDES and Aux PHY modules with the following main features:
  - DP1.3, HBR3 and eDP1.4a HBR3 throughput
  - 1, 2, or 4 lanes at 1.62Gbps, 2.7Gbps, 5.4Gbps, and 8.1Gbps per lane
  - Additional link rates (2.16, 2.43, 3.24, 4.32Gbps) per lane in eDP mode
  - Reduced differential voltage swing (0.2/0.25/0.30/0.35/0.40/0.45) in eDP mode
  - Hot Plug Detect (HPD) for connection detection and interrupt from sink
  - Integrated Low Jitter, Fixed Bandwidth PLL
  - DisplayPort physical layer functionalities:
    - Scrambler
    - 8/10-bit encoder (within the eDP transmitter)
    - Inter Lane Skew Insertion
    - Training Pattern Generation – TPS1,2,3,4 PRBS7 and 80-bit custom training pattern generation (bypassing the scrambler and encoder)
- 1 Mbps AUX PHY for link training, DPCD register access, HDCP authentication and EDID access

### 1.3.14 Camera Subsystem

Camera Subsystem unites two camera streaming interfaces – receiver and transmitter, allowing the device to stream video inputs from multiple cameras to the video processing accelerator (VPAC) or to internal memory and to output CSI-2 protocol image data to any device that supports MIPI CSI-2 protocol. Main modules are as follows:

- Camera Streaming Interface Receiver (CSI\_RX\_IF) with the following main features:
  - Compliant to MIPI CSI-2 v1.3+ and MIPI CSI-2 v2.0
  - Supports up to 16 virtual channels per input
  - Supports one 4MP camera or eight 2MP camera streams
  - Supports data rate up to 2.5 Gbps per lane (wire rate)
  - Supports 1, 2, 3, or 4 Data Lane connections to MIPI D-PHY Receiver (DPHY\_RX)
  - Over 25 different programmable formats including YUV420, YUV422, RGB, Raw, and User Defined
  - Supports four independent (simultaneous) output streams:
    - Two VP 32-bit streams to VISS inputs of VPAC image processing accelerator
    - One (up to 4 channels) PPI 16-bit pixel retransmission interface to Camera Streaming Interface Transmitter (CSI\_TX\_IF)
    - One (up to 32 Channels) DMA interface through a 128-bit Packet Streaming Interface Link (PSI\_L) connection to NAVSS for transfers to memory:
- Functional and data path error interrupts
- ECC support
- MIPI D-PHY Receiver (DPHY\_RX) with the following main features:
  - Allows the device to input video streams from external sensor cameras and other CSI2 compliant sources
  - Compliant to MIPI D-PHY standard v1.2
  - Supports up to 4 data and 1 clock lanes
  - Supports up to 2.5 Gbps (with deskew) and 1.5 Gbps (without deskew) per data lane
  - Clock lane Control / Interface logic type: CIL-SCNN for HS and low power receiving
  - Data lane Control / Interface logic type: CIL-SFAN for HS and low power receiving
  - Data lanes can be independently operated in HS or ULP mode
  - Swapping of DP/DN signals within each clock/data pair
- Camera Streaming Interface Transmitter (CSI\_TX\_IF) with the following main features:
  - Compliant to MIPI CSI-2 v1.3+, MIPI CSI-2 v2.0, and MIPI D-PHY v1.2
  - Data rate up to 2.5 Gbps per lane (wire rate)

- Supports 1, 2, 3, or 4 Data Lane connections to MIPI D-PHY Transmitter (DPHY\_TX)
- Over 25 different programmable formats including YUV420, YUV422, RGB, Raw, and User Defined
- Support of 16 virtual channels
- Support of four configurable input streams

### 1.3.15 Shared D-PHY Transmitter

The DPHY\_TX module provides an option for video output interfacing by implementing a four lane, MIPI D-PHY Transmitter. DPHY\_TX module supports the following main features:

- Compliancy to MIPI D-PHY Standard version 1.2
- Supports up to 4 data lanes
- Supports up to 2.5 Gbps (with deskew) and 1.5 Gbps (without deskew) per data lane
- Supports Escape mode
- Data Lanes can be independently operated in HS or ULP mode
- Includes a CMN block with reference generators / resistor calibration and an integrated PLL
- Fault detection

### 1.3.16 Video Processing Front End

Video Processing Front End (VPFE) is an input interface module that receives raw (unprocessed) image/video data or YUV digital video data from external imaging peripherals (such as image sensors, video decoders, etc.) and performs DMA transfers to store the captured data in the system DDR memory. VPFE module supports the following features:

- Includes a buffer memory for interfacing to the DMA at the chip level and preventing the Charge-Coupled Device Controller (CCDC) from overflowing
- Support for conventional Bayer pattern and Foveon sensor formats
- Generates HD/VD timing signals and field ID to an external timing generator or can synchronize to the external timing generator
- Support for progressive (non-interlaced) and interlaced sensors
- Support for up to 110-MHz sensor clock
- Support for REC656/CCIR-656 standard (YCbCr 422 format, either 8- or 16-bit)
- Support for YCbCr 422 format, either 8- or 16-bit with discrete HSYNC and VSYNC signals
- Support for up to 16-bit input
- Generates optical black clamping signals
- Support for digital clamping and black level compensation
- Support for 10-bit to 8-bit A-law compression
- Support for a low-pass filter prior to writing to DDR
- Support for generating output to range from 16-bits to 8-bits wide
- Support for down-sampling via programmable culling patterns
- Ability to control output to the DDR via an external write enable signal
- Support for up to 16K pixels (image size) in both the horizontal and vertical directions
- Region-based Address Translation (RAT) module for converting legacy 32-bit to 48-bit addressing scheme in the write DMA path

### 1.3.17 Multicore Shared Memory Controller

Multicore Shared Memory Controller (MSMC) provides high-bandwidth resource access both to and from all of the connected processing elements and the rest of the system and supports the following main features:

- 8MB (4 banks x 2MB) SRAM with ECC
- 512-bit processor port bus and 40-bit physical address bus
- Coherent unified bi-directional interfaces to connect to processors or device masters
- One infrastructure master interface
- Single external memory master interface
- Support of distributed virtual system
- Internal DMA engine – DRU (Data Routing Unit)
- Bandwidth management with starvation bound

- Two-level Quality-of-Service (QoS) support for real-time/non-real-time split
- Security firewall for SRAM/cache and external memory
- ECC error protection
- One interconnect messaging interface that supports DMA/pre-fetch requests to DRU
- Trace and debug features
- Support of dynamic clock gating on all logic units

### 1.3.18 DDR Subsystem

Integrated in MAIN domain one instance of DDR Subsystem (DDRSS) is used as an interface to external SDRAM devices which can be utilized for storing program or data. DDRSS provides the following main features:

- Support of LPDDR4 memory type
- 32-bit memory bus interface with in-line ECC
- Up to 8 GB memory address range
- System bus interface: little endian only with 512-bit data width
- Configuration bus Interface: little endian only with 32-bit data width
- Support of dual rank configuration
- Support of automatic idle power saving mode when no or low activity is detected
- Class of Service (CoS) - three latency classes supported
- Prioritized refresh scheduling
- Statistical counters for performance management

### 1.3.19 Region-based Address Translation Module

Integrated in MAIN domain thirteen instances of Region-based Address Translation (RAT) module perform a region based address translation of a 32-bit input address into a 48-bit output address. Each RAT module provides the following main features:

- 16 regions with dedicated registers for attributes configuration:
  - Region base address
  - Region size
  - Translated base address
- Address translation for only enabled regions
- Region boundary crossing transactions error generation

### 1.3.20 General Purpose Input/Output Interface

Eight General Purpose Input/Output (GPIO) modules integrated in MAIN domain provide dedicated general-purpose pins that can be configured as either inputs or outputs. Modules main features are:

- Support of 9 banks x 16 GPIO pins
- Support of up to 9 banks of interrupt capable GPIOs
- Interrupts can be triggered by rising and/or falling edge, specified for each interrupt capable GPIO pin
- Set/clear functionality per individual GPIO pin

### 1.3.21 Inter-Integrated Circuit Interface

Device MAIN domain contains seven multi-master Inter-Integrated Circuit (I2C) interfaces, each with the following main features:

- Compliancy to the Philips I2C-bus specification version 2.1
- Support of standard mode (up to 100 Kbps) and fast mode (up to 400 Kbps)
- Support of 7-bit and 10-bit device addressing modes
- Support of multi-master transmitter/slave receiver and receiver/slave transmitter modes
- Built-in FIFOs with programmable size of 8 to 64 bytes for buffered read or write
- 8-bit-wide data access
- Support of Auto Idle, Idle Request/Idle Acknowledge handshake, and Asynchronous Wakeup mechanisms
- Low power consumption

### 1.3.22 Improved Inter-Integrated Circuit Interface

Device MAIN domain contains one multi-master Improved Inter-Integrated Circuit (I3C) module with the following main features:

- Supports Single Data Rate (SDR) and High Data Rate – Dual Data Rate (HD-DDR) communication modes
- Support of Common Command Codes (CCC)
- Hot-Join capability
- In Band Interrupts (IBI)
- Support of Dynamic Address Assignment (DAA)
- Support of Static Addressing (SA)
- FIFO Buffers
- Registers to store the parameters for the response to an IBI interrupt from a number of slaves

### 1.3.23 Multi-channel Serial Peripheral Interface

Integrated in MAIN domain eight Multi-channel Serial Peripheral Interface (MCSPI) modules have the following main features:

- Serial clock with programmable frequency, polarity, and phase for each channel
- Wide selection of MCSPI word lengths, ranging from 4 to 32 bits
- Up to four master channels, or single channel in slave mode
- Support of different master multichannel modes
- Single interrupt line for multiple interrupt source events
- Support of start-bit write command
- Support of start-bit pause and break sequence
- Built-in FIFO available for a single channel

### 1.3.24 Universal Asynchronous Receiver/Transmitter

Integrated into MAIN domain are ten configurable Universal Asynchronous Receiver/Transmitter (UART) interfaces with the following main features:

- 16C750-compatible
- Support of RS-485 external transceiver auto flow control
- Dual 64-byte FIFOs – one per each received and transmitted data paths
- Programmable and selectable transmit and receive FIFO trigger levels for DMA and interrupt generation
- Programmable sleep mode
- Baud rates up to 3.6 Mbps with 48 MHz functional clock
- Auto-baud between 1200 bits/s and 115.2 Kbits/s (only when 48 MHz function clock is used)
- Support of IrDA 1.4 Slow Infrared (SIR), Medium Infrared (MIR), and Fast Infrared (FIR) communications
- Support of Consumer Infrared Remote control mode (CIR) with programmable data encoding

### 1.3.25 Gigabit Ethernet Switch

Integrated in the MAIN domain one instance of 9-port Gigabit Ethernet Switch (CPSW) subsystem provides Ethernet packet communication for the device. CPSW provides the following features:

- Eight Ethernet ports with selectable SGMII, RGMII and RMII interfaces and an internal Communications Port Programming Interface (CPPI) port
- Synchronous 10/100/1000 Mbit operation
- Flexible logical FIFO-based packet buffer structure
- Support of eight priority level Quality Of Service (QOS) - 802.1p
- Support for Audio/Video Bridging (P802.1Qav/D6.0 and 802.1Qaz)
- Ethernet port reset isolation
- Support for IEEE 1588 Clock Synchronization (2008 Annex D, Annex E and Annex F)
- Differentiated Services Code Point (DSCP) Priority Mapping (IPv4 and IPv6)
- IPV4/IPV6 UDP/TCP checksum offload
- Priority Based Flow Control (802.1QBB) and Flow Control (802.3x) Support
- Wire rate switching (802.1d)



- Store and Forward Switching
- Non-Blocking switch fabric
- Support of Time Sensitive Network - IEEE P802.3br/D2.0 Interspersing Express Traffic and IEEE 802.1Qbv/D2.2 Enhancements for Scheduled Traffic
- Address Lookup Engine (ALE) with 1024 ALE table entries
- EtherStats and 802.3 Stats Remote Network Monitoring (RMON) statistics gathering (per port statistics)
- Maximum frame size of 2020 bytes
- Management Data Input/Output (MDIO) module for PHY Management
- Host port CPPI Streaming Packet Interface
- Digital loopback and FIFO loopback modes supported
- Emulation support
- Full duplex mode supported in 10/100/1000 Mbps. Half-duplex mode supported only in 10/100 Mbps modes only.
- RAM Error Detection and Correction (SECCDED)

### 1.3.26 Peripheral Component Interconnect Express Subsystem

Integrated in the MAIN domain are four Peripheral Component Interconnect express (PCIe) subsystems with shared SerDes lines, each providing the following main features:

- Compliant to PCI-Express® Base Specification, Revision 4.0 (Version 0.7)
- One or two-lane configuration with up to 8.0 Gbps/lane (Gen3). Can be used as 2-lane controller, configurable in 1x1 or 1x2 mode.
- Gen3 (8 Gbps 128/130-bit encoding), Gen2 (5 Gbps 8/10-bit encoding), and Gen1 (2.5 Gbps 8/10-bit encoding) with auto-negotiation
- Dual mode: Root Port (RP) or End Point (EP) operation modes, selectable via bootstrap pins
- Dynamic PIPE width change when switching between Gen1/2/3 modes
- Constant 32-bit PIPE width for Gen1/2/3 modes
- Maximum payload size of 256 bytes
- Maximum remote read request size of 4KB
- Address Translation Services (ATS)
- Single-root I/O Virtualization (SR-IOV) with Physical Functions (PF) and Virtual Functions (VF) in End Point mode – 6 PF and 16 VF
- Four virtual channels (VC)
- Maximum number of non-posted outstanding transactions: 32
- Resizable Base Address Registers (BAR) capability
- Separate Reference Clock with Independent Spread (SRIS)
- Legacy, MSI and MSI-X Interrupt Support
- 32 outbound address translation regions
- Precision time measurement (PTM)

### 1.3.27 Universal Serial Bus (USB) Subsystem

Instantiated in MAIN domain two Universal Serial Bus (USB) subsystems with integrated PHY have the following main features:

- Dual-Role Device (DRD) capability
- Support of USB2.0 (up to 480 Mbps) standard
- Support of USB3.0 (5 Gbps) standard
- Support of Peripheral (aka Device) mode at Super Speed (SS at 5 Gbps), High Speed (HS at 480 Mbps), and Full Speed (FS at 12 Mbps)
- Support of Host mode at SS (5 Gbps), HS (480 Mbps), FS (12 Mbps), and Low Speed (LS at 1.5 Mbps)
- Support of static peripheral and static host operations
- Support of Host Negotiation Protocol (HNP)
- Support of USB3 low power protocol states (U0, U1, U2, and U3)
- USB Type-C connector with internal data lane swapping
- Each USB instance contains a single xHCI compliant with xHCI 1.0 specification with internal DMA controller

- Shared SerDes for USB 3.0 operation
- ECC on internal RAMs
- Embedded USB 2.0 PHY:
  - Fully compliant with UTMI+ Level 3 specification revision 1.0
  - Supports HS, FS and LS data rates
  - Charger Downstream Port (CDP) as per Battery Charging Specification, Revision 1.2
  - Supports USB low-power states: suspend and link power management (LPM)
  - Supports multiple reference clocks for PLL
  - Supports PLL standalone and bypass features

### 1.3.28 SerDes

Integrated in the MAIN domain are five instances of high-speed differential interface implemented with Serializer/Deserializer (SERDES) Multi-protocol Multi-link modules:

- The 2L SerDes instantiations which are an implementation of a multi-protocol PCIe/USB/Ethernet PHY, consisting of a PIPE compliant multi-protocol (PCIe Gen1/2/3 and USB 3.0/3.1) PCS, IEEE 802.3 clause 72 link training modules, supporting different components (TAP controller, APB interface, PHY power control and Clock/Reset controller), and Physical Medium Attachment (PMA)
- The 4L SerDes is an implementation of a multi-protocol PHY, consisting of a PIPE compliant multi-protocol Physical Coding Sublayer (PCS), supporting components (TAP controller, APB interface, PHY power control and Clock/Reset controller), and PMA.

Device supports four 2L SerDes modules, each with the following main features:

- Two bidirectional lanes
- Supported lane configurations: x1 and x2
- Supported standards:
  - PCIe Gen 1, 2 and 3
  - USB 3.1 Gen 1
  - Ethernet - SGMII/QSGMII
- Supports 'L1 PM Sub-states with CLKREQ' ECN
- Raw SerDes / Ethernet PHY 32/20-bit data interface with separate Rx and Tx source synchronous clocks
- For automotive safety applications, supports logic to monitor for successful PHY power state/data rate changes and generate an interrupt upon failure
- Provides option to automatically re-configure the PHY for a non-default configuration upon start-up (PHY auto-configuration)
- Supports multitude of high-speed data rates including (Gb/s): 1.25, 1.5, 2.5, 3.0, 3.125, 5.0, 6.0, 6.25, 8.0
- PHY (PMA integrated with PCS):
  - ECNs included: L1 PM Sub-states CLKREQ; SRIS
  - USB3.1 per Universal Serial Bus 3.1 Specification, Revision 1.0
  - QSGMII Specification revision 1.2
- Selectable serial pin polarity reversal for both transmit and receive paths
- Up to two reference clock sources for mixed standard multi-lane link applications
- Capable of equalizing up to 30dB total loss channels
- Receiver data recovery path:
  - Includes 5-tap DFE with adaptive training.
  - Supports far-end transmit pre-cursor and post-cursor adaptive training
  - Includes adaptive offset correction and gain control
  - Periodic on-the-fly training updates maintain optimization with drifting environment
- Receiver clock recovery path:
  - Tracks frequency offset of +/-5700ppm or greater where +/- 700 PPM is considered static and the remainder is from spread-spectrum techniques (SSC)
  - Includes CTLE with adaptive equalization
  - Includes adaptive phase offset correction

- Supports on-the-fly eye and bathtub curve diagramming with 8-bit voltage amplitude resolution and up to 1/64 UI time resolution
- Data path built-in self-test (BIST) with programmable pattern generation and error detection
- Serial bit stream and parallel word loopback for both line and parallel side
- Analog Test Bus (ATB) for internal node monitoring during characterization
- 8-bit ADC provides digitized ATB measurement results over configuration bus
- Supports DC and AC JTAG (boundary scan) per IEEE 1149.6
- Automatic calibration of pin termination resistors

Device supports one 4L SerDes with the following main features:

- Four bidirectional lanes
- Supported lane configurations: all lane configurations from x1 to x4
- Supported standards:
  - SGMII/QSGMII
  - Embedded DisplayPort Tx (UI\_Rate\_1 - UI\_Rate\_8)
- Single protocol, independent multiple links (where N is the number of lanes) support for:
  - eDP Tx link can be 1, 2, or 4 lanes
  - N SGMII links.
- Simultaneous multi-protocol support for various combinations (including Multiple Ethernet links + 1x eDP Tx link)
- Raw SerDes interface with separate Tx and Rx source synchronous clocks
- Supports PCIe 'L1 PM Sub-states with CLKREQ'
- For automotive safety applications, supports logic to monitor for successful PHY power state/data rate changes and generate an interrupt upon failure
- Dual on-chip PLLs which accepts a wide range of reference clocks
- Supports spread spectrum generation (up to 5000 PPM)
- PLL lock status
- Single off-chip resistor required
- Automatic calibration for on-chip termination resistors
- 20-bit/16-bit parallel data transceiver interface
- Support for external differential reference clock
- Support for internal single-ended reference clock
- Decision feedback equalization with 3-tap adaptive decision feedback equalization with adaptive CTLE and offset correction
- Supports bifurcation link combinations from 1 to 4 lanes per link
- Robust clock/data recovery tracks static frequency offset of +/- 350 PPM, and spread spectrum up to 5000 PPM
- Support for IEEE 1149.6 boundary scan
- Serial and parallel loop-back functions
- BIST functions for manufacturing test including multiple pattern generator/error detector
- Analog Test Bus for test and characterization
- 8-bit digitized ATB measurement system
- On demand eye/bathtub curve measurement capability

### 1.3.29 General Purpose Memory Controller with Error Location Module

Integrated in MAIN domain is one instance of General-Purpose Memory Controller (GPMC) with Error Location Module (ELM). Dedicated for interfacing with external memory devices it has the following main features:

- Support of 8- or 16-bit-wide data path to external memory devices
- Supports up to 4 independent chip-select regions of programmable size and programmable base addresses on 16MB, 32MB, 64MB, or 128MB boundary in a total address space of 1GB
- Support of the following wide range of external memories/devices:
  - Asynchronous or synchronous 8-bit wide memory or device (non-burst device)

- Asynchronous or synchronous 16-bit wide memory or device
- 16-bit non-multiplexed NOR flash device
- 16-bit address and data multiplexed NOR flash device
- 8-bit and 16-bit NAND flash device
- 16-bit pseudo-SRAM (pSRAM) device
- Supports various interface protocols when communicating with external memory or external devices:
  - Asynchronous read/write access
  - Asynchronous read page access (4, 8, and 16 Word16)
  - Synchronous read/write access
  - Synchronous read burst access without wrap capability (4, 8, and 16 Word16)
  - Synchronous read burst access with wrap capability (4, 8, and 16 Word16)
- Supports up to 16-bit on-the-fly error code detection using the Bose-Chaudhuri-Hocquenghem (BCH) or Hamming code to improve the reliability of NAND with a minimum effect on software (NAND flash with 512-byte page size or greater)
- ELM module which used in a conjunction with the GPMC and provides ECC calculation (up to 16-bit) for NAND support and ability to work in both page-based and continuous modes, has the following main features:
  - 4, 8, and 16 bits per 512-byte block error-location, based on BCH algorithms
  - Eight simultaneous processing contexts
  - Page-based and continuous modes
  - Interrupt generation on error-location process completion

### 1.3.30 Multimedia Card/Secure Digital Interface

Device MAIN domain supports three Multimedia Card/Secure Digital (MMCSD) controllers with the following main features:

- One controller with 8-bit wide data bus
- Two controllers with 4-bit wide data bus
- Support of eMMC5.1 Host Specification (JESD84-B51)
- Support of SD Host Controller Standard Specification - SDIO 4.0
- Integrated DMA controller supporting SD Advanced DMA - ADMA2 and ADMA3
- Multimedia card features:
  - Backward compatible with earlier eMMC standards
  - SDR, 3.3 V / 1.8 V, 4-bit bus width, 0-26 MHz, 13 MB/s
  - SDR, 3.3 V / 1.8 V, 1-bit bus width, 0-26 MHz, 3.25 MB/s
  - SDR, 3.3 V / 1.8 V, 4-bit bus width, 0-52 MHz, 26 MB/s
  - SDR, 3.3 V / 1.8 V, 1-bit bus width, 0-52 MHz, 6.5 MB/s
  - DDR, 3.3 V / 1.8 V, 4-bit bus width, 0-52 MHz, 52 MB/s
  - SDR, 1.8 V, 0-200 MHz, 4-bit bus width, 100 MB/s
- SD card support: SDIO, SDR12, SDR25, SDR50, DDR50
- System bus interface: CBA 4.0 VBUSM master port with 64-bit data width and 64-bit address, little endian only
- Configuration bus interface: CBA 4.0 VBUSM with 32-bit data width, 32-bit aligned accesses only, linear incrementing addressing mode, little endian only

### 1.3.31 Universal Flash Storage Interface

Device MAIN domain supports one standard-based serial interface engine - Universal Flash Storage (UFS) interface with an integrated M-PHY. UFS provides the following main features:

- Support of Universal Flash Storage Host (UFS2.1, JESD220C)
- Support of Universal Flash Storage Host Controller Interface (UFSHCI, JESD223C)
- Support of UFS interconnect - MIPI UniPro (v.1.60, 2013) and MIPI M-PHY (v3.1, 2014)
- Supports speeds 1.46 Gbps (Gear 1), 2.91 Gbps (Gear 2) and 5.83 Gbps (Gear 3), 2-lanes

- Support of all UFS mandatory SCSI commands required by JEDEC specification
- Master bus Interface: one 64-bit master interface for data transfers, 48-bit address width, embedded DMA within the host controller with support of little-endian mode only
- Configuration Bus Interface: one 32-bit slave interface for configuration access, support of linear incrementing addressing mode and only aligned accesses
- ECC support on internal RAMs
- M-PHY Features: supports reference clocks of 19.2 MHz and 26 MHz

### 1.3.32 Enhanced Capture Module

Integrated in the MAIN domain three Enhanced Capture (ECAP) modules provide accurate timing for different events. When not being used for event capture, its resources can be used to generate a single channel of asymmetrical PWM waveforms (configurable as either one capture input, or as one auxiliary PWM output). Each ECAP module supports the following main features:

- 32-bit time base counter
- 4 x 32 bits event time-stamp capture registers
- 4 stage sequencer (Mod4 counter), synchronized to external events
- Independent edge polarity selection for up to four sequenced time-stamp capture events
- Input capture signal pre-scaling (from 1 to 16)
- Interrupt capabilities on any of the four capture events
- Support of different capture modes (single shot capture, continuous mode capture, absolute timestamp capture or delta mode time-stamp capture)

### 1.3.33 Enhanced Pulse-Width Modulation Module

Integrated into MAIN domain are six Enhanced Pulse-Width Modulation (EPWM) modules, each with the following main features:

- Dedicated 16-bit time-base counter with period and frequency control
- Two independent PWM outputs that can be used in different configurations (with single-edge operation, with dual-edge symmetric operation or one independent PWM output with dual-edge asymmetric operation)
- Asynchronous override control of PWM signals during fault conditions
- Programmable phase-control support for lag or lead operation relative to other EPWM modules
- Dead-band generation with independent rising and falling edge delay control
- Programmable trip zone allocation of both latched and un-latched fault conditions
- Events enabling to trigger both CPU interrupts and start of ADC conversions

### 1.3.34 Enhanced Quadrature Encoder Pulse Module

Integrated in MAIN domain are three 32-bit Enhanced Quadrature Encoder Pulse (EQEP) modules for position, speed, and frequency measurements support. Each of the modules has the following main features:

- Input synchronization
- Three stage/six stage digital noise filter
- Quadrature decoder unit
- Position counter and control unit for position measurement
- Quadrature edge capture unit for low speed measurement
- Unit time base for speed/frequency measurement
- Watchdog timer for detecting stalls

### 1.3.35 Controller Area Network

Fourteen Controller Area Network (MCAN) interfaces integrated into MAIN domain and supporting both classic CAN and CAN FD (CAN with Flexible Data-Rate) specifications have the following main features:

- Conforms with CAN Protocol version 2.0 part A, B and ISO 11898-1:2015
- Full CAN FD (up to 64 data bytes) support
- SAE J1939 support
- AUTOSAR support for DRA829 and TDA4VM families of devices
- Up to 32 dedicated transmit buffers and 64 dedicated receive buffers



- Two configurable receive FIFOs, up to 64 elements each
- Configurable transmit FIFO, up to 32 elements
- Configurable transmit queue, up to 32 elements
- Configurable transmit event FIFO, up to 32 elements
- Up to 128 filter elements
- Maskable interrupts, two interrupt lines
- Timestamp Counter

### 1.3.36 Audio Tracking Logic

Instantiated in MAIN domain one Audio Tracking Logic (ATL) module, which is used by HD Radio™ applications to synchronize the digital audio output to the baseband clock, supports the following main features:

- One ATL module, containing four ATL instances, for HD Radio support and asynchronous sample rate conversion assistance
- Each instance tracks the time error between two syncs (local Audio Word Select [AWS] and Baseband Word Select [BWS])
- Each instance selects between 28 mux choices for BWS and 30 mux choices for AWS
- Each instance generates modulated ATCLK clock signals with software-initiated pulse stealing
- Selection between interface or functional clock to run error counting timers and to derive modulated clock outputs
- Clock and reset management: receives clock and reset signals from the device PSC module
- Hardware reset
- Local software reset

### 1.3.37 Multi-channel Audio Serial Port

Instantiated in MAIN domain are twelve instances of Multi-channel Audio Serial Ports (MCASP). The MCASP is a general purpose audio serial port, useful for Time-Division Multiplexed (TDM) stream, Inter-IC Sound (I2S) protocols reception and transmission as well as for an inter-component Digital audio Interface Transmission (DIT). MCASP modules have the following main features:

- Connection to audio Analog-to-Digital Converters (ADC), Digital-to-Analog Converters (DAC), codec, digital audio interface receiver (DIR), and Sony/Philips Digital Interface (S/PDIF) transmit physical layer components
- Support of Time Division Multiplexed (TDM) interface, Inter-IC Sound (I2S) standard, and similar bit stream formats
- Integrated Digital Audio Interface Transmitter (DIT) with enhanced channel status/user data RAM and support of S/PDIF, IEC60958-1, and AES-3 formats
- Independent serializer for each AXRx channel of each MCASP module
- A single 32-bit buffer per serializer for transmit and receive operations
- Support for two DMA requests (one per direction)
- One transmit and one receive interrupt requests common for all serializers
- Two independent clock generator modules for transmit and receive allowing the MCASP to receive and transmit at different rates
- Support of up to 16 serial data pins on MCASP0, up to 12 serial data pins on MCASP1, 8 serial data pins on MCASP10 and MCASP11, 6 serial data pins on MCASP2 and 4 serial data pins on MCASP3 through MCASP9
- MCASP10 internal connection to eDP for audio support

### 1.3.38 Timers

Three different types of timer modules are instantiated in the MAIN domain:

- One instance of Global Time Counter (GTC) module that can be used for time synchronization and debug trace time stamping with the following main features:
  - 64-bit up counter
  - No rollover during the lifetime of the device
  - Compatible with Armv8 system counter requirements

- Outputs reflected binary (Gray) encoded timer value for system timer bus distribution to other modules
- Selectable counter bit output as a push event that can be used by CPTS modules, timers or interface protocols
- Ten instances of Real Time Interrupt (RTI) windowed watchdog module providing timer functionality for operation systems and benchmarking code with the following main features:
  - Two independent 64 bit counter blocks
  - Four configurable compare registers for generating operating system ticks
  - Free running counter 0 can be incremented by either the internal pre-scale counter or by an external event
  - Selectable RTI clock input (derived from any of the available clock sources)
  - Windowed Watchdog Timer (WWDT) feature
  - Some RTI modules are pre-dedicated to specific processors cores
- Twenty instances of Timer module with support of the following main features:
  - Free running 32-bit upward counter
  - Generates a 1-ms tick with a 32.768-kHz functional clock
  - Interrupts generated on overflow, compare and capture
  - Supported modes of operation: compare and capture, auto-reload and start-stop
  - Programmable divider clock source ( $2^n$ , where  $n = [0-8]$ )
  - Dedicated input trigger for capture mode and dedicated output trigger/PWM signal
  - On-the-fly read/write register (while counting) for systems operation and benchmarking code
  - Each odd numbered timer instance may be optionally cascaded with the previous even numbered timer instance to form up to a 64-bit timer

### 1.3.39 Internal Diagnostics Modules

Instantiated in MAIN domain are different internal diagnostics modules which provide monitoring and diagnostic functions required to achieve certain safety compliance levels:

- Thirteen instances of Dual Clock Comparator (DCC) modules, used to determine the accuracy of a clock signal during the time execution of an application, each having the following main features:
  - Two independent counter blocks count clock pulses from each clock source
  - Each counter block is programmable, however, for proper operation the counters must be programmed with seed values that respect the ratio of the two clock frequencies
  - Configurable time base for error signal
  - Error signal generation when one of the clocks is out of spec
  - Clock frequency measurement
- One instance of Error Signaling Module (ESM) for safety-related events and/or errors aggregation from throughout the device into one location supports the following main features:
  - Up to 1024 level or pulse error event inputs
  - Selectable low and high priority interrupt error pin prioritization of each error event
  - Error pin to signal severe device failure
  - Configurable time base for error signal
  - Error forcing capability
  - Internal redundant flops on safety critical fields
- Multiple ECC aggregator modules supporting ECC mechanism for providing increased system reliability via reduction of memory software errors by allowing single bit errors to be detected and corrected (SEC) and double bit errors to be detected (DED). Applied to different memories in many of the subsystems, each of the ECC aggregators has the following main features:
  - Reduces memory software errors via single error correction (SEC) and double error detection (DED)
  - Provides a mechanism to control and monitor the ECC RAMs in a module or subsystem
  - Supports software readable status of ECC errors (single and double-bit) and associated info such as RAM address and data bit or bits that are in error
  - Aggregates level pending status from the ECC RAMs in two interrupts to the device CPU – interrupt for correctable error (SEC) and interrupt for uncorrectable error (DED)
  - Supports up to 256 ECC endpoints (either ECC RAM or interconnect ECC component)

- Single bit error detection via parity checking results in a non-correctable error interrupt
- Memory Cyclic Redundancy Check (MCRC) module used to perform CRC to verify the integrity of a memory system – part of the NAVSS

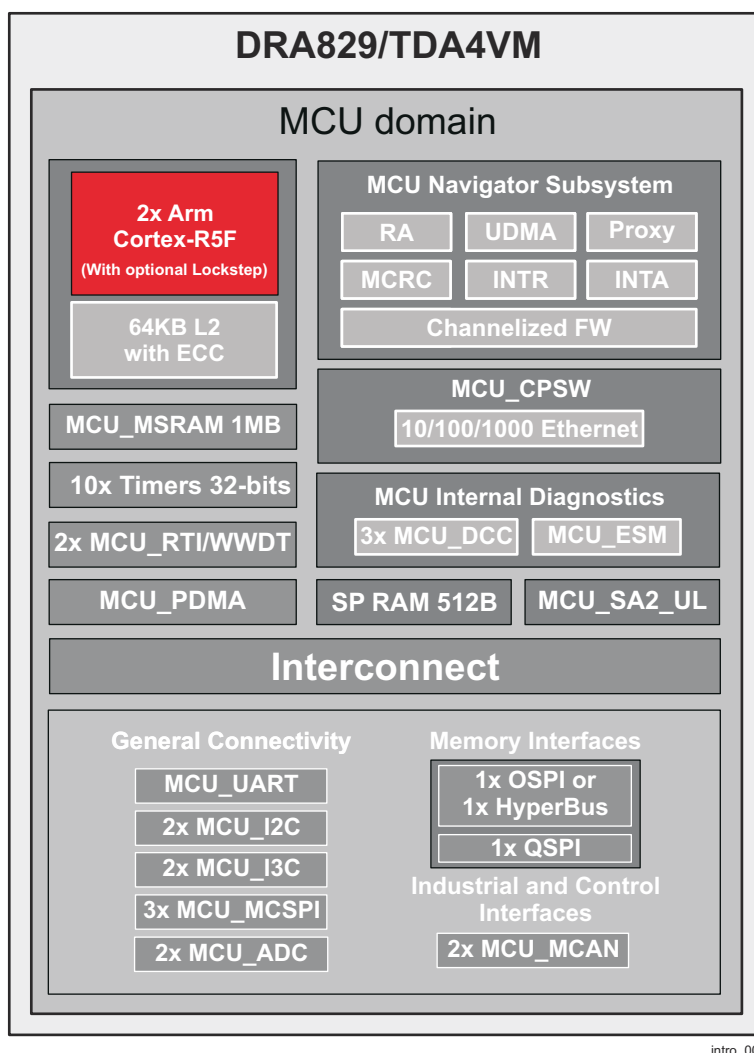
## 1.4 Device MCU Domain

The Microcontroller Unit Subsystem is a “chip-in-a-chip” concept and operates using a separate voltage supply, clock sources and resets and includes the components needed for device management. This allows the MCU to function continuously regardless of the state of the rest of the device, including periods in which the rest of the device is held in reset or powered down. Most communication between the MCU island and the rest of the SoC can (and, in a safety case, should) be conducted via internal SPI connections. These connections allow for safe communications to isolate the MCU island from faults in the rest of the SoC. The MCU island integrates communication peripherals such as MCSPI and MCAN which are expected to be used for safety critical communication, so if the main SoC goes down, the MCU island can communicate this information to the rest of the system. Isolation of the MCU Dual-Core Arm Cortex-R5F Microprocessor Subsystem helps to provide Freedom from Interference (FFI).

The MCU subsystem has integrated:

- 192 KB of read-only memory used for first stage booting (MCU ROM)
- 1024KB L3 RAM for program/data storage with the following features:
  - Supports various ECC features available via the ECC aggregator modules
  - Supports linear and cache-line wrap addressing modes
  - Supports configurable data-path widths of 32, 64, 128, 256 and 512 bits
  - Supports configurable FIFO depths for command, write data, read data and write status
- Configurable Scratchpad RAM (512 bytes) aimed for storing debug and context information across chip resets

[Figure 1-3](#) shows the block diagram of the device MCU domain.



intro\_002

**Figure 1-3. Device MCU Domain Block Diagram**

### Note

The supported set of features and peripherals is device part number dependent. For more information, see the device-specific Datasheet.

### 1.4.1 MCU Arm Cortex-R5F Processor

Integrated in MCU domain one instance of Arm dual-core Cortex-R5F processor supports the following main features:

- Armv7-R architecture
- Two modes of operation, boot-time configurable:
  - Split mode: two independently operating cores (Asymmetric Multi Processing, no coherence)
  - Lock (lockstep) mode: one main operating core with the other operating in lockstep
- 16KB instruction and 16KB data SECDED ECC protected L1 cache per core in split mode
- 64KB of Tightly Coupled Memory (TCM) per core
- Full-Precision Floating Point (VFPv3)
- 8 breakpoints, 8 watch points
- 16-region Memory Protection Unit (MPU)

- CoreSight Debug Access Port (DAP)
- CoreSight ETM-R5 interface
- Performance Monitoring Unit (PMU)
- 32-bit to 48-bit Region-based Address Translation (RAT) on memory access masters
- Integrated Vectored Interrupt Manager (VIM)
- Interfaces:
  - 64-bit VBUSM master pair (1 read, 1 write) for L3 memory accesses (per core)
  - 64-bit VBUSM slave for TCM access (per core)
  - 32-bit VBUSM master pair (1 read, 1 write) for peripheral access
  - 32-bit VBUSP master for peripheral access (per core)
  - 32-bit VBUSP slave configuration port (per core)
  - 32-bit VBUSP slave debug port

### 1.4.2 MCU Region-based Address Translation Module

Integrated in MCU domain two instances of Region-based Address Translation module perform a region based address translation of a 32-bit input address into a 48-bit output address. Each RAT module provides the following main features:

- 16 regions with dedicated registers for attributes configuration:
  - Region base address
  - Region size
  - Translated base address
- Address translation for only enabled regions
- Region boundary crossing transactions error generation

### 1.4.3 MCU Navigator Subsystem

Instantiated in the MCU domain one Navigator subsystem named MCU\_NAVSS can be used for efficient transfer of data support between software, firmware and hardware in all combinations. It consists of the following main modules:

- Unified DMA Controller with the following main modes and features:
  - K3 DMA Architecture compliant Tx/Rx port implementation
  - K3 DMA Architecture compliant Packet-Oriented DMA Functionality (UDMA-P)
  - K3 DMA Architecture compliant Third Party Channel Controller
  - K3 DMA Architecture compliant Unified Transfer Controller
  - K3 DMA Architecture compliant Unified DMA channels which all share the execution hardware using time division multiplexing
- Ring Accelerator provides hardware acceleration to enable straightforward passing of work between a producer and a consumer and has the following main features:
  - Supports 286 independent memory-mapped ring structures
  - Supports various modes for each ring based on usage and compatibility
  - Provides single-word deep shared incoming Transfer Response FIFO
  - Provides bit-wide source VBUSM read/write slave interface for accesses from DMA controller entities
- Proxy module with the following main features:
  - Provides proxy buffers to store large data bursts that a host can only access in smaller amounts
  - Keeps the large data coherent until the complete data has been accessed
  - Allows interleaved access between multiple hosts using multiple proxies
  - Supports a pre-configured number of target resources to proxy with pre-configured number of channels, size of each channel, and address offset per each target
- Secure proxy module is a modified version of the proxy module and in addition has the following main features:
  - Supports a number of threads, where each has their own independent proxy function
  - Supports a programmable fixed queue for each proxy thread
  - Supports multiple producers all writing to the same queue



- Supports programmable thresholds for when to generate events
- Supports a max message count for outbound proxy threads limiting the number of messages a thread can produce
- Interrupt Aggregator module provides a centralized machine which handles the termination of system events to that they can be coherently processed by the host(s) in the system. Main features are as follows:
  - 64-bit VBUSP slave using 64-bit registers
  - Provide a set of TI Interrupt Architecture compliant interrupt status and mask registers which are used to pass specific event status to one or more host blocks
  - Provide a set of Global Event Input (GEVI) counters which can count events delivered via an ingress Event Transport Lane (ETL)
  - Provides a set of Local Event Input (LEVI) to Global event registers which can be used to convert pulsed discrete interrupt inputs or clock synchronous rising edge events into Global events on an egress ETL
  - Provides a set of GEVI 'Multicast' registers which can take a Global event from an ingress ETL and generate two egress Global events on two egress ETL interfaces
- Time Sync module to facilitate host control of time sync operations with the following main features:
  - Supports a selection of multiple external clock sources
  - Software control of time sync events via interrupt or polling
  - Supports 8 hardware timestamp push inputs
  - Supports timestamp counter compare output
  - Supports timestamp counter bit output
  - Supports 6 timestamp generator function outputs
  - 32-bit and 64-bit timestamp modes
- Memory Cyclic Redundancy Check module performs CRC to verify the integrity of a memory system with the following main features:
  - Four channels to perform background signature verification on any memory subsystem
  - Data compression on 8-, 16-, 32-, and 64-bit data size
  - Dedicated CRC value register per channel which contains the pre-determined CRC value
  - Timed base event trigger from timer to initiate DMA data transfer
  - Programmable 20-bit pattern counter per channel to count the number of data patterns for compression
  - Three modes of operation: Auto, Semi-CPU, and Full-CPU
  - Timeout interrupt generation if CRC is not performed within the time limit
  - Per channel DMA request generation to initiate CRC value transfer

#### 1.4.4 MCU Analog-to-Digital Converter

Analog-to-Digital Converter (MCU\_ADC) module contains a single 12-bit ADC which can be multiplexed to any 1 of 8 analog inputs (channels). Integrated in the MCU domain are two instances, each with the following main features:

- 4 MSPS rate with a 60 MHz sample clock
- Single-ended or differential input options
- Each ADC module can be configured and transformed into digital test inputs
- Programmable 16 steps Finite State Machine (FSM) sequencer

#### 1.4.5 MCU Inter-Integrated Circuit Interface

Device MCU domain contains two multi-master Inter-Integrated Circuit (MCU\_I2C) interfaces, each with the following main features:

- Compliancy to the Philips I2C-bus specification version 2.1
- Support of standard mode (up to 100 Kbps) and fast mode (up to 400 Kbps)
- Support of 7-bit and 10-bit device addressing modes
- Support of multi-master transmitter/slave receiver and receiver/slave transmitter modes
- Built-in FIFOs with programmable size of 8 to 64 bytes for buffered read or write
- 8-bit-wide data access
- Support of Auto Idle, Idle Request/Idle Acknowledge handshake, and Asynchronous Wakeup mechanisms

- Low power consumption

#### **1.4.6 MCU Improved Inter-Integrated Circuit Interface**

Device MCU domain contains two multi-master Improved Inter-Integrated Circuit (MCU\_I3C) modules, each with the following main features:

- Supports Single Data Rate (SDR) and High Data Rate – Dual Data Rate (HD-DDR) communication modes
- Support of Common Command Codes (CCC)
- Hot-Join capability
- In Band Interrupts (IBI)
- Support of Dynamic Address Assignment (DAA)
- Support of Static Addressing (SA)
- FIFO Buffers
- Registers to store the parameters for the response to an IBI interrupt from a number of slaves

#### **1.4.7 MCU Multi-channel Serial Peripheral Interface**

Integrated in MCU domain three Multi-channel Serial Peripheral Interface (MCU\_MCSPI) modules where each has the following main features:

- Serial clock with programmable frequency, polarity, and phase for each channel
- Wide selection of MCSPI word lengths, ranging from 4 to 32 bits
- Up to four master channels, or single channel in slave mode
- Support of different master multichannel modes
- Single interrupt line for multiple interrupt source events
- Support of start-bit write command
- Support of start-bit pause and break sequence
- Built-in FIFO available for a single channel

#### **1.4.8 MCU Universal Asynchronous Receiver/Transmitter**

Integrated into MCU domain is one 16C750-compatible and configurable Universal Asynchronous Receiver/Transmitter (MCU\_UART) interface with the following main features:

- Support of RS-485 external transceiver auto flow control
- Dual 64-byte FIFOs – one per each received and transmitted data paths
- Programmable and selectable transmit and receive FIFO trigger levels for DMA and interrupt generation
- Programmable sleep mode
- Baud rates up to 3.6 Mbps with 48 MHz functional clock
- Auto-baud between 1200 bits/s and 115.2 Kbits/s (only when 48 MHz function clock is used)
- Support of IrDA 1.4 Slow Infrared (SIR), Medium Infrared (MIR), and Fast Infrared (FIR) communications
- Support of Consumer Infrared Remote control mode (CIR) with programmable data encoding

#### **1.4.9 MCU Gigabit Ethernet Switch**

Integrated in the MCU domain is one instance of Gigabit Ethernet Switch (MCU\_CPSW) subsystem which provides Ethernet packet communication for the device. MCU\_CPSW has the following main features:

- One Ethernet port with selectable RGMII or RMII interfaces and an internal Communications Port Programming Interface (CPPI) port
- Synchronous 10/100/1000 Mbit operation
- Flexible logical FIFO-based packet buffer structure
- Support of eight priority level Quality Of Service (QOS) - 802.1p
- Support for Audio/Video Bridging (P802.1Qav/D6.0 and 802.1Qaz)
- Ethernet port reset isolation
- Support for IEEE 1588 Clock Synchronization (2008 Annex D, Annex E and Annex F)
- Differentiated Services Code Point (DSCP) Priority Mapping (IPv4 and IPv6)
- Energy Efficient Ethernet (EEE) support (802.3az)
- Priority Based Flow Control (802.3x) Support
- Wire rate switching (802.1d)

- Store and Forward Switching
- Non-Blocking switch fabric
- Support of Time Sensitive Network - IEEE P802.3br/D2.0 Interspersing Express Traffic and IEEE 802.1Qbv/D2.2 Enhancements for Scheduled Traffic
- Address Lookup Engine (ALE) with 64 ALE table entries
- EtherStats and 802.3 Stats Remote Network Monitoring (RMON) statistics gathering (per port statistics)
- Ethernet Mac transmit to Ethernet Mac receive Loopback mode (digital loopback) supported
- Maximum frame size of 2024 bytes
- Management Data Input/Output (MDIO) module for PHY Management with Clause 45 support
- Host port CPPI Streaming Packet Interface
- Digital loopback and FIFO loopback modes supported
- Emulation support
- Full duplex mode supported in 10/100/1000 Mbps. Half-duplex mode supported only in 10/100 Mbps modes only.
- RAM Error Detection and Correction (SECCDED)

#### **1.4.10 MCU Octal Serial Peripheral Interface and HyperBus Memory Controller as a Flash Subsystem**

Integrated in MCU domain one Flash Subsystem (MCU\_FSS) provides access to external flash devices via Octal Serial Peripheral Interface (OSPI) and HyperBus interface along with encryption/decryption, authentication, and in-line ECC protection. MCU\_FSS supports the following main features:

- Provides one OSPI and one QSPI or one QSPI and one HyperBus flash interfaces
- Primary OSPI0/HyperBus interfaces support:
  - Execute in place (XIP) operation
  - 32-byte Block Copy (BC) operation
  - ECC and/or authentication with four configurable authentication regions and authentication on 32-byte or 64-byte blocks
- Secondary 4-bits OSPI1 interface supports:
  - 32-byte BC operation
  - ECC
- OSPI interfaces support:
  - Up to 4 devices using different chip selects
  - Single, dual, quad (QSPI mode), or octal (on OSPI0 only) I/O instructions
  - Dual Quad-SPI mode for fast boot applications
  - Memory mapped 'direct' and software triggered 'indirect' modes of operation
  - Software triggered 'indirect' mode of operation for performing low latency and non-processor intensive flash data transfers.
  - DDR Mode and Data Terminal Ready DTR protocol (including Octal DDR protocol with DQS for Octal-SPI devices)
  - Programmable write protected regions
  - Programmable delays between transactions
  - Programmable baud rate generator to generate OSPI clocks
  - Programmable interrupt generation
  - Programmable data decoder for enabling continuous addressing mode for each of connected devices and auto-detection of boundaries between devices
  - Bidirectional CRC on Multiple-SPI interface and handling of ECC errors for flash devices with embedded correction engine mechanisms
- HyperBus interface supports:
  - Up to 2 devices using two memory chip selects
  - Connection to Cypress® HyperFlash™ or HyperRAM™ devices
  - Up to 166 MHz maximum memory bus operation for reads
  - Up to 16 outstanding read transactions
  - Linear incrementing mode for reads and writes

- Asynchronous bus clock

#### 1.4.11 MCU Controller Area Network

Two Controller Area Network interfaces integrated into MCU domain (MCU\_MCAN) and supporting both classic CAN and CAN FD (CAN with Flexible Data-Rate) specifications have the following main features:

- Conforms with CAN Protocol version 2.0 part A, B and ISO 11898-1:2015
- Full CAN FD (up to 64 data bytes) support
- SAE J1939 support
- AUTOSAR support for DRA829 and TDA4VM families of devices
- Up to 32 dedicated transmit buffers and 64 dedicated receive buffers
- Two configurable receive FIFOs, up to 64 elements each
- Configurable transmit FIFO, up to 32 elements
- Configurable transmit queue, up to 32 elements
- Configurable transmit event FIFO, up to 32 elements
- Up to 128 filter elements
- Maskable interrupts, two interrupt lines
- Timestamp Counter

#### 1.4.12 MCU Timers

Two different types of timer modules are instantiated in the MCU domain:

- Two instances of Real Time Interrupt (MCU\_RTI) windowed watchdog module providing timer functionality for operation systems and benchmarking code with the following main features:
  - Two independent 64 bit counter blocks
  - Four configurable compare registers for generating operating system ticks
  - Free running counter 0 can be incremented by either the internal pre-scale counter or by an external event
  - Selectable RTI clock input (derived from any of the available clock sources)
  - Windowed Watchdog Timer (WWDT) feature
  - Some RTI modules are pre-dedicated to specific processor cores
- Ten instances of Timer module with support of the following main features:
  - Free running 32-bit upward counter
  - Generates a 1-ms tick with a 32.768-kHz functional clock
  - Interrupts generated on overflow, compare and capture
  - Supported modes of operation: compare and capture, auto-reload and start-stop
  - Programmable divider clock source ( $2^n$ , where  $n = [0-8]$ )
  - Dedicated input trigger for capture mode and dedicated output trigger/PWM signal
  - On-the-fly read/write register (while counting) for systems operation and benchmarking code
  - Each odd numbered timer instance may be optionally cascaded with the previous even numbered timer instance to form up to a 64-bit timer

#### 1.4.13 MCU Internal Diagnostics Modules

Instantiated in MCU domain are different internal diagnostics modules which provide monitoring and diagnostic functions required to achieve certain safety compliance levels:

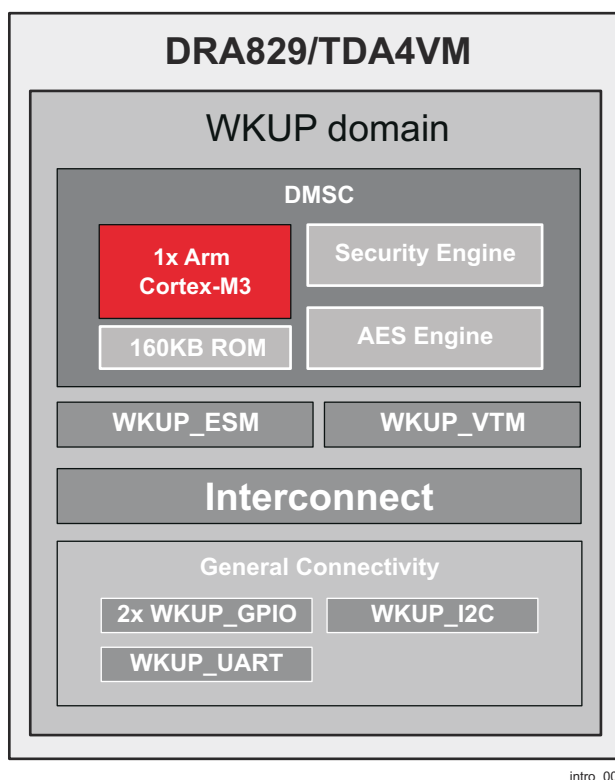
- Three instances of Dual Clock Comparator (MCU\_DCC) modules, used to determine the accuracy of a clock signal during the time execution of an application, each having the following main features:
  - Two independent counter blocks count clock pulses from each clock source
  - Each counter block is programmable, however, for proper operation the counters must be programmed with seed values that respect the ratio of the two clock frequencies
  - Configurable time base for error signal
  - Error signal generation when one of the clocks is out of spec
  - Clock frequency measurement
- One instance of Error Signaling Module (MCU\_ESM) for safety-related events and/or errors aggregation from throughout the device into one location supports the following main features:

- Up to 1024 level or pulse error event inputs
- Selectable low and high priority interrupt error pin prioritization of each error event
- Error pin to signal severe device failure
- Configurable time base for error signal
- Error forcing capability
- Internal redundant flops on safety critical fields
- Multiple ECC aggregator modules supporting ECC mechanism for providing increased system reliability via reduction of memory software errors by allowing single bit errors to be detected and corrected (SEC) and double bit errors to be detected (DED). Applied to different memories in many of the subsystems, each of the ECC aggregators has the following main features:
  - Reduces memory software errors via single error correction (SEC) and double error detection (DED)
  - Provides a mechanism to control and monitor the ECC RAMs in a module or subsystem
  - Supports software readable status of ECC errors (single and double-bit) and associated info such as RAM address and data bit or bits that are in error
  - Aggregates level pending status from the ECC RAMs in two interrupts to the device CPU – interrupt for correctable error (SEC) and interrupt for uncorrectable error (DED)
  - Supports up to 256 ECC endpoints (either ECC RAM or interconnect ECC component)
  - Single bit error detection via parity checking results in a non-correctable error interrupt
- Memory Cyclic Redundancy Check module used to perform CRC to verify the integrity of a memory system – part of MCU\_NAVSS

## 1.5 Device WKUP Domain

Wake-up Subsystem (WKUPSS) is defined as an isolated domain to control operating states of the device. It supports low-power modes of operations and safety precautions during power transitions. Located in the WKUPSS are dedicated subsystems for device management and security functions.

Figure 1-4 shows the block diagram of the device WKUP domain.



**Figure 1-4. Device WKUP Domain Block Diagram**



### Note

The supported set of features and peripherals is device part number dependent. For more information, see the device-specific Datasheet.

#### 1.5.1 WKUP Device Management and Security Controller

Integrated in WKUP domain is the Device Management and Security Controller (WKUP\_DMSC) which provides control over the device boot sequencing, device management, power management, and security. With the factory-sealed firmware, DMSC main functions include:

- Device management
- On-chip power management and wake-up control
- Device boot configuration and sequence
- Secure boot setup
- Decryption routines
- Firewall control for isolation and Security
- Runtime Security Management and resource allocation

Arm Cortex-M3 based DMSC acts as a system security master and protects critical security assets during run-time. As part of booting a High Security (HS) device, DMSC uses on-chip keys to establish root-of-trust and authenticate images to reinforce trust. DMSC controls the power management of device, hence is responsible to bring device cleanly out of reset and enforce clock and reset rules. DMSC power management functions are critical to bring device to low power modes and sense wakeup events to bring device back to active state. DMSC acts also as main boot processor and as such is the very first subsystem that is brought out of reset after device power-on-reset

Main components of the DMSC are:

- Arm Cortex-M3 processor core (ARMv7-M architecture profile)
- 160 KB ROM to allow boot sequence, authentication and provide security service (M3 accessible only)
- Two separate local memory banks for Instruction code (I-code) and Data space (D-code) with single error correction and double error detection
- Firewall enabled 32-bit VBUSP CBASS interconnect
- Interrupt Aggregator with support of up to 80 interrupt inputs to the DMSC
- Four dual-mode 32-bit timers
- One RTI/WWDT module capable of issuing warm reset to the SoC
- DMSC control module - contains various control, configuration and status MMRs for power management functions
- Debug and trace related modules
- Security Manager module for device security management, device type control (GP, EMU, HS), emulation and JTAG control, and key management
- AES engine with 128, 192 and 256-bits support and DPA/EMA countermeasures

#### 1.5.2 WKUP General Purpose Input/Output Interface

Two General Purpose Input/Output (WKUP\_GPIO) modules integrated in WKUP domain provide dedicated general-purpose pins that can be configured as either inputs or outputs. Modules main features are:

- Support of 9 banks x 16 GPIO pins
- Support of up to 9 banks of interrupt capable GPIOs
- Interrupts can be triggered by rising and/or falling edge, specified for each interrupt capable GPIO pin
- Set/clear functionality per individual GPIO pin

#### 1.5.3 WKUP Inter-Integrated Circuit Interface

Device WKUP domain contains one multi-master Inter-Integrated Circuit (WKUP\_I2C) interface with the following main features:

- Compliancy to the Philips I2C-bus specification version 2.1

- Support of standard mode (up to 100 Kbps) and fast mode (up to 400 Kbps)
- Support of 7-bit and 10-bit device addressing modes
- Support of multi-master transmitter/slave receiver and receiver/slave transmitter modes
- Built-in FIFOs with programmable size of 8 to 64 bytes for buffered read or write
- 8-bit-wide data access
- Support of Auto Idle, Idle Request/Idle Acknowledge handshake, and Asynchronous Wakeup mechanisms
- Low power consumption

#### 1.5.4 WKUP Universal Asynchronous Receiver/Transmitter

Integrated into WKUP domain is one 16C750-compatible and configurable Universal Asynchronous Receiver/Transmitter (WKUP\_UART) interface with the following main features:

- Support of RS-485 external transceiver auto flow control
- Dual 64-byte FIFOs – one per each received and transmitted data paths
- Programmable and selectable transmit and receive FIFO trigger levels for DMA and interrupt generation
- Programmable sleep mode
- Baud rates up to 3.6 Mbps with 48 MHz functional clock
- Auto-baud between 1200 bits/s and 115.2 Kbits/s (only when 48 MHz function clock is used)
- Support of IrDA 1.4 Slow Infrared (SIR), Medium Infrared (MIR), and Fast Infrared (FIR) communications
- Support of Consumer Infrared Remote control mode (CIR) with programmable data encoding

#### 1.5.5 WKUP Internal Diagnostics Modules

Instantiated in WKUP domain are different internal diagnostics modules which provide monitoring and diagnostic functions required to achieve certain safety compliance levels:

- One Error Signaling Module (WKUP\_ESM) for safety-related events and/or errors aggregation from throughout the device into one location supports the following main features:
  - Up to 1024 level or pulse error event inputs
  - Selectable low and high priority interrupt error pin prioritization of each error event
  - Error pin to signal severe device failure
  - Configurable time base for error signal
  - Error forcing capability
  - Internal redundant flops on safety critical fields
- Multiple ECC aggregator modules supporting ECC mechanism for providing increased system reliability via reduction of memory software errors by allowing single bit errors to be detected and corrected (SEC) and double bit errors to be detected (DED). Applied to different memories in many of the subsystems, each of the ECC aggregators has the following main features:
  - Reduces memory software errors via single error correction (SEC) and double error detection (DED)
  - Provides a mechanism to control and monitor the ECC RAMs in a module or subsystem
  - Supports software readable status of ECC errors (single and double-bit) and associated info such as RAM address and data bit or bits that are in error
  - Aggregates level pending status from the ECC RAMs in two interrupts to the device CPU – interrupt for correctable error (SEC) and interrupt for uncorrectable error (DED)
  - Supports up to 256 ECC endpoints (either ECC RAM or interconnect ECC component)
  - Single bit error detection via parity checking results in a non-correctable error interrupt

## 1.6 Device Identification

The manufacturer identity, the boundary scan part number, and the silicon revision of the device can be read in the CTRLMMR\_WKUP\_JTAGID register. See [Table 1-2](#) and [Table 1-3](#), for more information.

**Table 1-2. Device JTAG ID**

CTRLMMR_WKUP_JTAGID Register Field	Value	Comment
[31-28] VARIANT	See <a href="#">Table 1-3</a>	Silicon Revision (SR) identifier.
[27-12] PARTNO	See <a href="#">Table 1-3</a>	Part number for boundary scan.
[11-1] MFG	0x17	Manufacturer identity (TI).
[0] LSB	0x1	Always reads 1.

**Table 1-3. Device JTAG ID Values**

Silicon Type	VARIANT	PARTNO	CTRLMMR_WKUP_JTAGID Register Value
DRA829 SR1.0 TDA4VM SR1.0	0x0	0xBB64	0x0BB6402F
DRA829 SR1.1 TDA4VM SR1.1	0x1	0xBB64	0x1BB6402F
DRA829 SR2.0 TDA4VM SR2.0	0x2	0xBB64	0x2BB6402F



This chapter summarizes the memory map address regions for the device.

<b>2.1 MAIN Domain Memory Map.....</b>	<b>119</b>
<b>2.2 MCU Domain Memory Map.....</b>	<b>173</b>
<b>2.3 WKUP Domain Memory Map.....</b>	<b>177</b>
<b>2.4 Processors View Memory Map.....</b>	<b>178</b>
<b>2.5 Region-based Address Translation.....</b>	<b>179</b>

## 2.1 MAIN Domain Memory Map

Table 2-1 shows the MAIN domain memory map.

### Note

The memory locations not shown in Table 2-1 are either unallocated or reserved and not used. Accesses to these locations are not recommended and should be avoided.

**Table 2-1. MAIN Domain Memory Map**

Region Name	Start Address	End Address	Size
PSRAM2KECC0_RAM	0x0000000000	0x00000007FF	2 KB
CTRL_MMR0_CFG0	0x0000100000	0x000011FFFF	128 KB
PSRAM2KECC0_RAM	0x0000200000	0x00002003FF	1 KB
EFUSE0	0x0000300000	0x00003000FF	256 B
PSC0	0x0000400000	0x0000400FFF	4 KB
PLLCTRL0	0x0000410000	0x00004101FF	512 B
PBIST6	0x0000420000	0x00004203FF	1 KB
DFTSS0	0x0000500000	0x00005003FF	1 KB
GPIO0	0x0000600000	0x00006000FF	256 B
GPIO1	0x0000601000	0x00006010FF	256 B
GPIO2	0x0000610000	0x00006100FF	256 B
GPIO3	0x0000611000	0x00006110FF	256 B
GPIO4	0x0000620000	0x00006200FF	256 B
GPIO5	0x0000621000	0x00006210FF	256 B
GPIO6	0x0000630000	0x00006300FF	256 B
GPIO7	0x0000631000	0x00006310FF	256 B
PLL0_CFG	0x0000680000	0x000069FFFF	128 KB
ESM0_CFG	0x0000700000	0x0000700FFF	4 KB
PSRAM2KECC0_ECC_AGGR	0x0000780000	0x00007803FF	1 KB
PSRAM2KECC0_ECC_AGGR	0x0000784000	0x00007843FF	1 KB
DCC0	0x0000800000	0x000080003F	64 B
DCC1	0x0000804000	0x000080403F	64 B
DCC2	0x0000808000	0x000080803F	64 B
DCC3	0x000080C000	0x000080C03F	64 B
DCC4	0x0000810000	0x000081003F	64 B
DCC5	0x0000814000	0x000081403F	64 B
DCC6	0x0000818000	0x000081803F	64 B
DCC7	0x000081C000	0x000081C03F	64 B
DCC8	0x0000820000	0x000082003F	64 B
DCC9	0x0000824000	0x000082403F	64 B
DCC10	0x0000828000	0x000082803F	64 B
DCC11	0x000082C000	0x000082C03F	64 B
DCC12	0x0000830000	0x000083003F	64 B
GPIOMUX_INTRTR0_INTR_ROUTER_CFG	0x0000A00000	0x0000A007FF	2 KB
MAIN2MCU_LVL_INTRTR0_CFG	0x0000A10000	0x0000A107FF	2 KB



**Table 2-1. MAIN Domain Memory Map (continued)**

Region Name	Start Address	End Address	Size
MAIN2MCU_PLS_INTRTR0_CFG	0x0000A20000	0x0000A207FF	2 KB
CMPEVENT_INTRTR0_INTR_ROUTER_CFG	0x0000A30000	0x0000A303FF	1 KB
TIMESYNC_INTRTR0_INTR_ROUTER_CFG	0x0000A40000	0x0000A407FF	2 KB
R5FSS0_INTROUTER0_INTR_ROUTER_CFG	0x0000A60000	0x0000A61FFF	8 KB
R5FSS1_INTROUTER0_INTR_ROUTER_CFG	0x0000A70000	0x0000A71FFF	8 KB
GTC0_GTC_CFG0	0x0000A80000	0x0000A803FF	1 KB
GTC0_GTC_CFG1	0x0000A90000	0x0000A93FFF	16 KB
GTC0_GTC_CFG2	0x0000AA0000	0x0000AA3FFF	16 KB
GTC0_GTC_CFG3	0x0000AB0000	0x0000AB3FFF	16 KB
C66SS0_INTROUTER0_INTR_ROUTER_CFG	0x0000AC0000	0x0000AC0FFF	4 KB
C66SS1_INTROUTER0_INTR_ROUTER_CFG	0x0000AD0000	0x0000AD0FFF	4 KB
CBASS_INFRA0_ERR	0x0000B00000	0x0000B003FF	1 KB
CBASS_FW0_ERR	0x0000B08000	0x0000B083FF	1 KB
ECC_AGGR0_ECC_AGGR	0x0000C02000	0x0000C023FF	1 KB
COMPUTE_CLUSTER0_GIC_TRANSLATER	0x0001000000	0x00013FFFFFFF	4 MB
COMPUTE_CLUSTER0_GIC_DISTRIBUTOR	0x0001800000	0x000180FFFF	64 KB
COMPUTE_CLUSTER0_GIC_MESSAGE_BASED_SPIS	0x0001810000	0x000181FFFF	64 KB
COMPUTE_CLUSTER0_GIC_ITS	0x0001820000	0x000182FFFF	64 KB
COMPUTE_CLUSTER0_GIC_REDISTRIBUTOR_CONTROL_LPI_0	0x0001900000	0x000190FFFF	64 KB
COMPUTE_CLUSTER0_GIC_REDISTRIBUTOR_SGI_PPI_0	0x0001910000	0x000191FFFF	64 KB
COMPUTE_CLUSTER0_GIC_REDISTRIBUTOR_CONTROL_LPI_1	0x0001920000	0x000192FFFF	64 KB
COMPUTE_CLUSTER0_GIC_REDISTRIBUTOR_SGI_PPI_1	0x0001930000	0x000193FFFF	64 KB
COMPUTE_CLUSTER0_GIC_REDISTRIBUTOR_CONTROL_LPI_2	0x0001940000	0x000194FFFF	64 KB
COMPUTE_CLUSTER0_GIC_REDISTRIBUTOR_SGI_PPI_2	0x0001950000	0x000195FFFF	64 KB
COMPUTE_CLUSTER0_GIC_REDISTRIBUTOR_CONTROL_LPI_3	0x0001960000	0x000196FFFF	64 KB
COMPUTE_CLUSTER0_GIC_REDISTRIBUTOR_SGI_PPI_3	0x0001970000	0x000197FFFF	64 KB
COMPUTE_CLUSTER0_GIC_REDISTRIBUTOR_CONTROL_LPI_4	0x0001980000	0x000198FFFF	64 KB
COMPUTE_CLUSTER0_GIC_REDISTRIBUTOR_SGI_PPI_4	0x0001990000	0x000199FFFF	64 KB
COMPUTE_CLUSTER0_GIC_REDISTRIBUTOR_CONTROL_LPI_5	0x00019A0000	0x00019AFFFF	64 KB
COMPUTE_CLUSTER0_GIC_REDISTRIBUTOR_SGI_PPI_5	0x00019B0000	0x00019BFFFF	64 KB
COMPUTE_CLUSTER0_GIC_REDISTRIBUTOR_CONTROL_LPI_6	0x00019C0000	0x00019CFFFF	64 KB
COMPUTE_CLUSTER0_GIC_REDISTRIBUTOR_SGI_PPI_6	0x00019D0000	0x00019DFFFF	64 KB
COMPUTE_CLUSTER0_GIC_REDISTRIBUTOR_CONTROL_LPI_7	0x00019E0000	0x00019EFFFF	64 KB
COMPUTE_CLUSTER0_GIC_REDISTRIBUTOR_SGI_PPI_7	0x00019F0000	0x00019FFFFFFF	64 KB
I2C0_CFG	0x0002000000	0x00020000FF	256 B
I2C1_CFG	0x0002010000	0x00020100FF	256 B
I2C2_CFG	0x0002020000	0x00020200FF	256 B
I2C3_CFG	0x0002030000	0x00020300FF	256 B
I2C4_CFG	0x0002040000	0x00020400FF	256 B
I2C5_CFG	0x0002050000	0x00020500FF	256 B
I2C6_CFG	0x0002060000	0x00020600FF	256 B
I3C0_MMR_MMRVBP	0x00020A0000	0x00020A01FF	512 B
I3C0_VBP2APB_WRAP_CORE_VBP_MIPI_I3C_MST	0x00020A8000	0x00020A83FF	1 KB
MCSPi0_CFG	0x0002100000	0x00021003FF	1 KB
MCSPi1_CFG	0x0002110000	0x00021103FF	1 KB

**Table 2-1. MAIN Domain Memory Map (continued)**

Region Name	Start Address	End Address	Size
MCSPi2_CFG	0x0002120000	0x00021203FF	1 KB
MCSPi3_CFG	0x0002130000	0x00021303FF	1 KB
MCSPi4_CFG	0x0002140000	0x00021403FF	1 KB
MCSPi5_CFG	0x0002150000	0x00021503FF	1 KB
MCSPi6_CFG	0x0002160000	0x00021603FF	1 KB
MCSPi7_CFG	0x0002170000	0x00021703FF	1 KB
RTI0_CFG	0x0002200000	0x00022000FF	256 B
RTI1_CFG	0x0002210000	0x00022100FF	256 B
RTI15_CFG	0x00022F0000	0x00022F00FF	256 B
RTI16_CFG	0x0002300000	0x00023000FF	256 B
RTI24_CFG	0x0002380000	0x00023800FF	256 B
RTI25_CFG	0x0002390000	0x00023900FF	256 B
RTI28_CFG	0x00023C0000	0x00023C00FF	256 B
RTI29_CFG	0x00023D0000	0x00023D00FF	256 B
RTI30_CFG	0x00023E0000	0x00023E00FF	256 B
RTI31_CFG	0x00023F0000	0x00023F00FF	256 B
TIMER0_CFG	0x0002400000	0x00024003FF	1 KB
TIMER1_CFG	0x0002410000	0x00024103FF	1 KB
TIMER2_CFG	0x0002420000	0x00024203FF	1 KB
TIMER3_CFG	0x0002430000	0x00024303FF	1 KB
TIMER4_CFG	0x0002440000	0x00024403FF	1 KB
TIMER5_CFG	0x0002450000	0x00024503FF	1 KB
TIMER6_CFG	0x0002460000	0x00024603FF	1 KB
TIMER7_CFG	0x0002470000	0x00024703FF	1 KB
TIMER8_CFG	0x0002480000	0x00024803FF	1 KB
TIMER9_CFG	0x0002490000	0x00024903FF	1 KB
TIMER10_CFG	0x00024A0000	0x00024A03FF	1 KB
TIMER11_CFG	0x00024B0000	0x00024B03FF	1 KB
TIMER12_CFG	0x00024C0000	0x00024C03FF	1 KB
TIMER13_CFG	0x00024D0000	0x00024D03FF	1 KB
TIMER14_CFG	0x00024E0000	0x00024E03FF	1 KB
TIMER15_CFG	0x00024F0000	0x00024F03FF	1 KB
TIMER16_CFG	0x0002500000	0x00025003FF	1 KB
TIMER17_CFG	0x0002510000	0x00025103FF	1 KB
TIMER18_CFG	0x0002520000	0x00025203FF	1 KB
TIMER19_CFG	0x0002530000	0x00025303FF	1 KB
MCAN0_SS	0x0002700000	0x00027000FF	256 B
MCAN0_CFG	0x0002701000	0x00027011FF	512 B
MCAN0_MSGMEM_RAM	0x0002708000	0x000270FFFF	32 KB
MCAN1_SS	0x0002710000	0x00027100FF	256 B
MCAN1_CFG	0x0002711000	0x00027111FF	512 B
MCAN1_MSGMEM_RAM	0x0002718000	0x000271FFFF	32 KB
MCAN2_SS	0x0002720000	0x00027200FF	256 B
MCAN2_CFG	0x0002721000	0x00027211FF	512 B
MCAN2_MSGMEM_RAM	0x0002728000	0x000272FFFF	32 KB

**Table 2-1. MAIN Domain Memory Map (continued)**

Region Name	Start Address	End Address	Size
MCAN3_SS	0x0002730000	0x00027300FF	256 B
MCAN3_CFG	0x0002731000	0x00027311FF	512 B
MCAN3_MSGMEM_RAM	0x0002738000	0x000273FFFF	32 KB
MCAN4_SS	0x0002740000	0x00027400FF	256 B
MCAN4_CFG	0x0002741000	0x00027411FF	512 B
MCAN4_MSGMEM_RAM	0x0002748000	0x000274FFFF	32 KB
MCAN5_SS	0x0002750000	0x00027500FF	256 B
MCAN5_CFG	0x0002751000	0x00027511FF	512 B
MCAN5_MSGMEM_RAM	0x0002758000	0x000275FFFF	32 KB
MCAN6_SS	0x0002760000	0x00027600FF	256 B
MCAN6_CFG	0x0002761000	0x00027611FF	512 B
MCAN6_MSGMEM_RAM	0x0002768000	0x000276FFFF	32 KB
MCAN7_SS	0x0002770000	0x00027700FF	256 B
MCAN7_CFG	0x0002771000	0x00027711FF	512 B
MCAN7_MSGMEM_RAM	0x0002778000	0x000277FFFF	32 KB
MCAN8_SS	0x0002780000	0x00027800FF	256 B
MCAN8_CFG	0x0002781000	0x00027811FF	512 B
MCAN8_MSGMEM_RAM	0x0002788000	0x000278FFFF	32 KB
MCAN9_SS	0x0002790000	0x00027900FF	256 B
MCAN9_CFG	0x0002791000	0x00027911FF	512 B
MCAN9_MSGMEM_RAM	0x0002798000	0x000279FFFF	32 KB
MCAN10_SS	0x00027A0000	0x00027A00FF	256 B
MCAN10_CFG	0x00027A1000	0x00027A11FF	512 B
MCAN10_MSGMEM_RAM	0x00027A8000	0x00027AFFFF	32 KB
MCAN11_SS	0x00027B0000	0x00027B00FF	256 B
MCAN11_CFG	0x00027B1000	0x00027B11FF	512 B
MCAN11_MSGMEM_RAM	0x00027B8000	0x00027BFFFF	32 KB
MCAN12_SS	0x00027C0000	0x00027C00FF	256 B
MCAN12_CFG	0x00027C1000	0x00027C11FF	512 B
MCAN12_MSGMEM_RAM	0x00027C8000	0x00027CFFFF	32 KB
MCAN13_SS	0x00027D0000	0x00027D00FF	256 B
MCAN13_CFG	0x00027D1000	0x00027D11FF	512 B
MCAN13_MSGMEM_RAM	0x00027D8000	0x00027DFFFF	32 KB
PDMA5_REGS	0x00027E0000	0x00027E03FF	1 KB
UART0	0x0002800000	0x00028001FF	512 B
UART1	0x0002810000	0x00028101FF	512 B
UART2	0x0002820000	0x00028201FF	512 B
UART3	0x0002830000	0x00028301FF	512 B
UART4	0x0002840000	0x00028401FF	512 B
UART5	0x0002850000	0x00028501FF	512 B
UART6	0x0002860000	0x00028601FF	512 B
UART7	0x0002870000	0x00028701FF	512 B
UART8	0x0002880000	0x00028801FF	512 B
UART9	0x0002890000	0x00028901FF	512 B
PCIE0_CORE_PCIE_INTD_CFG_INTD_CFG	0x0002900000	0x0002900FFF	4 KB

**Table 2-1. MAIN Domain Memory Map (continued)**

Region Name	Start Address	End Address	Size
PCIE0_CORE_VMAP_HP_MMRS	0x0002904000	0x00029043FF	1 KB
PCIE0_CORE_VMAP_LP_MMRS	0x0002905000	0x00029053FF	1 KB
PCIE0_CORE_CPTS_CFG_CPTS_VBUSP	0x0002906000	0x00029063FF	1 KB
PCIE0_CORE_USER_CFG_USER_CFG	0x0002907000	0x00029073FF	1 KB
PCIE1_CORE_PCIE_INTD_CFG_INTD_CFG	0x0002910000	0x0002910FFF	4 KB
PCIE1_CORE_VMAP_HP_MMRS	0x0002914000	0x00029143FF	1 KB
PCIE1_CORE_VMAP_LP_MMRS	0x0002915000	0x00029153FF	1 KB
PCIE1_CORE_CPTS_CFG_CPTS_VBUSP	0x0002916000	0x00029163FF	1 KB
PCIE1_CORE_USER_CFG_USER_CFG	0x0002917000	0x00029173FF	1 KB
PCIE2_CORE_PCIE_INTD_CFG_INTD_CFG	0x0002920000	0x0002920FFF	4 KB
PCIE2_CORE_VMAP_HP_MMRS	0x0002924000	0x00029243FF	1 KB
PCIE2_CORE_VMAP_LP_MMRS	0x0002925000	0x00029253FF	1 KB
PCIE2_CORE_CPTS_CFG_CPTS_VBUSP	0x0002926000	0x00029263FF	1 KB
PCIE2_CORE_USER_CFG_USER_CFG	0x0002927000	0x00029273FF	1 KB
PCIE3_CORE_PCIE_INTD_CFG_INTD_CFG	0x0002930000	0x0002930FFF	4 KB
PCIE3_CORE_VMAP_HP_MMRS	0x0002934000	0x00029343FF	1 KB
PCIE3_CORE_VMAP_LP_MMRS	0x0002935000	0x00029353FF	1 KB
PCIE3_CORE_CPTS_CFG_CPTS_VBUSP	0x0002936000	0x00029363FF	1 KB
PCIE3_CORE_USER_CFG_USER_CFG	0x0002937000	0x00029373FF	1 KB
COMPUTE_CLUSTER0_SS_CFG	0x0002980000	0x00029801FF	512 B
COMPUTE_CLUSTER0_CTL_CFG	0x0002990000	0x0002997FFF	32 KB
PCIE0_CORE_ECC_AGGR0	0x0002A00000	0x0002A003FF	1 KB
PCIE0_CORE_ECC_AGGR1	0x0002A01000	0x0002A013FF	1 KB
PCIE1_CORE_ECC_AGGR0	0x0002A02000	0x0002A023FF	1 KB
PCIE1_CORE_ECC_AGGR1	0x0002A03000	0x0002A033FF	1 KB
PCIE2_CORE_ECC_AGGR0	0x0002A04000	0x0002A043FF	1 KB
PCIE2_CORE_ECC_AGGR1	0x0002A05000	0x0002A053FF	1 KB
PCIE3_CORE_ECC_AGGR0	0x0002A06000	0x0002A063FF	1 KB
PCIE3_CORE_ECC_AGGR1	0x0002A07000	0x0002A073FF	1 KB
USB0_RAM0_INJ_CFG	0x0002A10000	0x0002A103FF	1 KB
USB0_ECC_AGGR	0x0002A13000	0x0002A133FF	1 KB
USB1_ECC_AGGR	0x0002A16000	0x0002A163FF	1 KB
USB1_RAM0_INJ_CFG	0x0002A17000	0x0002A173FF	1 KB
CPSW0_ECC	0x0002A21000	0x0002A213FF	1 KB
SA2_UL0_ECC_AGGR	0x0002A23000	0x0002A233FF	1 KB
MMCSD0_ECC_AGGR_RXMEM	0x0002A24000	0x0002A243FF	1 KB
MMCSD0_ECC_AGGR_TXMEM	0x0002A25000	0x0002A253FF	1 KB
MMCSD1_ECC_AGGR_RXMEM	0x0002A26000	0x0002A263FF	1 KB
MMCSD1_ECC_AGGR_TXMEM	0x0002A27000	0x0002A273FF	1 KB
UFS0_HCLK_ECC_AGGR_CFG	0x0002A28000	0x0002A283FF	1 KB
UFS0_IPS_TCLK_ERR_INJ_CFG	0x0002A2A000	0x0002A2A3FF	1 KB
MSRAM16KX256E0_ECC_AGGR_REGS	0x0002A2F000	0x0002A2F3FF	1 KB
CSI_RX_IF0_ECC_AGGR_CFG	0x0002A30000	0x0002A303FF	1 KB
CSI_RX_IF1_ECC_AGGR_CFG	0x0002A31000	0x0002A313FF	1 KB
CSI_TX_IF0_ECC_AGGR_CFG	0x0002A38000	0x0002A383FF	1 KB

**Table 2-1. MAIN Domain Memory Map (continued)**

Region Name	Start Address	End Address	Size
CSI_TX_IF0_ECC_AGGR_BYTE_CFG	0x0002A38400	0x0002A387FF	1 KB
MCAN8_ECC_AGGR	0x0002A40000	0x0002A403FF	1 KB
MCAN9_ECC_AGGR	0x0002A41000	0x0002A413FF	1 KB
MCAN10_ECC_AGGR	0x0002A42000	0x0002A423FF	1 KB
MCAN11_ECC_AGGR	0x0002A43000	0x0002A433FF	1 KB
MCAN12_ECC_AGGR	0x0002A44000	0x0002A443FF	1 KB
MCAN13_ECC_AGGR	0x0002A45000	0x0002A453FF	1 KB
VPAC0_KSDW_ECC_AGGR_CFG	0x0002A60000	0x0002A603FF	1 KB
VPAC0_PAR_VPAC_VISS0_S_VBUSP_KSDW_ECC_AGGR_CFG	0x0002A61000	0x0002A613FF	1 KB
VPAC0_PAR_VPAC_LDC0_S_VBUSP_KSDW_ECC_AGGR_CFG	0x0002A63000	0x0002A633FF	1 KB
R5FSS0_CORE0_ECC_AGGR	0x0002A68000	0x0002A683FF	1 KB
R5FSS1_CORE0_ECC_AGGR	0x0002A69000	0x0002A693FF	1 KB
DMPAC0_KSDW_ECC_AGGR_CFG	0x0002A6A000	0x0002A6A3FF	1 KB
MMCSD2_ECC_AGGR_TXMEM	0x0002A70000	0x0002A703FF	1 KB
MMCSD2_ECC_AGGR_RXMEM	0x0002A71000	0x0002A713FF	1 KB
I3C0_P_ECC_AGGR_CFG	0x0002A74000	0x0002A743FF	1 KB
I3C0_S_ECC_AGGR_CFG	0x0002A75000	0x0002A753FF	1 KB
MCAN0_ECC_AGGR	0x0002A78000	0x0002A783FF	1 KB
MCAN1_ECC_AGGR	0x0002A79000	0x0002A793FF	1 KB
MCAN2_ECC_AGGR	0x0002A7A000	0x0002A7A3FF	1 KB
MCAN3_ECC_AGGR	0x0002A7B000	0x0002A7B3FF	1 KB
MCAN4_ECC_AGGR	0x0002A7C000	0x0002A7C3FF	1 KB
MCAN5_ECC_AGGR	0x0002A7D000	0x0002A7D3FF	1 KB
MCAN6_ECC_AGGR	0x0002A7E000	0x0002A7E3FF	1 KB
MCAN7_ECC_AGGR	0x0002A7F000	0x0002A7F3FF	1 KB
CBASS_DEBUG0_ERR	0x0002A80000	0x0002A803FF	1 KB
CBASS_HC2_0_ERR	0x0002A83000	0x0002A833FF	1 KB
CBASS_AC_CFG0_ERR	0x0002A84000	0x0002A843FF	1 KB
CBASS_AC0_ERR	0x0002A85000	0x0002A853FF	1 KB
CBASS_DATADEBUG0_ERR	0x0002A86000	0x0002A863FF	1 KB
CBASS_HC0_ERR	0x0002A87000	0x0002A873FF	1 KB
CBASS_CS10_ERR	0x0002A88000	0x0002A883FF	1 KB
CBASS_HC_CFG0_ERR	0x0002A89000	0x0002A893FF	1 KB
CBASS_MCASP_G0_0_ERR	0x0002A8A000	0x0002A8A3FF	1 KB
CBASS_MCASP_G1_0_ERR	0x0002A8B000	0x0002A8B3FF	1 KB
CBASS_RC0_ERR	0x0002A8C000	0x0002A8C3FF	1 KB
CBASS_RC_CFG0_ERR	0x0002A8D000	0x0002A8D3FF	1 KB
CBASS_AASRC0_ERR	0x0002A8E000	0x0002A8E3FF	1 KB
CBASS_IPPHY0_ERR	0x0002A8F000	0x0002A8F3FF	1 KB
DSS_EDP0_MHDPTX_WRAPPER_ECC_AGGR_CORE_CFG	0x0002AC0000	0x0002AC03FF	1 KB
DSS_EDP0_MHDPTX_WRAPPER_ECC_AGGR_PHY_CFG	0x0002AC1000	0x0002AC13FF	1 KB
DSS_EDP0_MHDPTX_WRAPPER_ECC_AGGR_DSC_CFG	0x0002AC2000	0x0002AC23FF	1 KB
ECC_AGGR16_REGS	0x0002AF0000	0x0002AF03FF	1 KB
ECC_AGGR17_REGS	0x0002AF1000	0x0002AF13FF	1 KB
ECC_AGGR18_REGS	0x0002AF2000	0x0002AF23FF	1 KB



**Table 2-1. MAIN Domain Memory Map (continued)**

Region Name	Start Address	End Address	Size
ECC_AGGR19_REGS	0x0002AF3000	0x0002AF33FF	1 KB
ECC_AGGR4_REGS	0x0002AF4000	0x0002AF43FF	1 KB
ECC_AGGR5_REGS	0x0002AF5000	0x0002AF53FF	1 KB
ECC_AGGR6_REGS	0x0002AF6000	0x0002AF63FF	1 KB
ECC_AGGR10_REGS	0x0002AFA000	0x0002AFA3FF	1 KB
ECC_AGGR11_REGS	0x0002AFB000	0x0002AFB3FF	1 KB
MCASP0_CFG	0x0002B00000	0x0002B01FFF	8 KB
MCASP0_DMA	0x0002B08000	0x0002B083FF	1 KB
MCASP1_CFG	0x0002B10000	0x0002B11FFF	8 KB
MCASP1_DMA	0x0002B18000	0x0002B183FF	1 KB
MCASP2_CFG	0x0002B20000	0x0002B21FFF	8 KB
MCASP2_DMA	0x0002B28000	0x0002B283FF	1 KB
MCASP3_CFG	0x0002B30000	0x0002B31FFF	8 KB
MCASP3_DMA	0x0002B38000	0x0002B383FF	1 KB
MCASP4_CFG	0x0002B40000	0x0002B41FFF	8 KB
MCASP4_DMA	0x0002B48000	0x0002B483FF	1 KB
MCASP5_CFG	0x0002B50000	0x0002B51FFF	8 KB
MCASP5_DMA	0x0002B58000	0x0002B583FF	1 KB
MCASP6_CFG	0x0002B60000	0x0002B61FFF	8 KB
MCASP6_DMA	0x0002B68000	0x0002B683FF	1 KB
MCASP7_CFG	0x0002B70000	0x0002B71FFF	8 KB
MCASP7_DMA	0x0002B78000	0x0002B783FF	1 KB
MCASP8_CFG	0x0002B80000	0x0002B81FFF	8 KB
MCASP8_DMA	0x0002B88000	0x0002B883FF	1 KB
MCASP9_CFG	0x0002B90000	0x0002B91FFF	8 KB
MCASP9_DMA	0x0002B98000	0x0002B983FF	1 KB
MCASP10_CFG	0x0002BA0000	0x0002BA1FFF	8 KB
MCASP10_DMA	0x0002BA8000	0x0002BA83FF	1 KB
MCASP11_CFG	0x0002BB0000	0x0002BB1FFF	8 KB
MCASP11_DMA	0x0002BB8000	0x0002BB83FF	1 KB
AASRC0_CFG	0x0002D00000	0x0002D03FFF	16 KB
AASRC0_DATA_R0	0x0002D10000	0x0002D13FFF	16 KB
AASRC0_DATA_R1	0x0002D20000	0x0002D23FFF	16 KB
VPFE0_MMRS	0x0002F00000	0x0002F00FFF	4 KB
VPFE0_VPFE	0x0002F08000	0x0002F08FFF	4 KB
MLB0_MMR_MMRVBP	0x0002F80000	0x0002F801FF	512 B
MLB0_MLBDIM_WRAP_ECC_AGGR_VBP	0x0002F81000	0x0002F813FF	1 KB
MLB0_VBP2APB_WRAP_MLB_CFG_VBP_MLBDIM	0x0002F82000	0x0002F823FF	1 KB
MLB0_RAT_WRAP_RAT_CFG_VBP_MMRS	0x0002F83000	0x0002F83FFF	4 KB
EHRPWM0_EPWM	0x0003000000	0x00030000FF	256 B
EHRPWM0_EHRPWM	0x0003008000	0x00030080FF	256 B
EHRPWM1_EPWM	0x0003010000	0x00030100FF	256 B
EHRPWM1_EHRPWM	0x0003018000	0x00030180FF	256 B
EHRPWM2_EPWM	0x0003020000	0x00030200FF	256 B
EHRPWM2_EHRPWM	0x0003028000	0x00030280FF	256 B

**Table 2-1. MAIN Domain Memory Map (continued)**

Region Name	Start Address	End Address	Size
EHRPWM3_EPWM	0x0003030000	0x00030300FF	256 B
EHRPWM3_EHRPWM	0x0003038000	0x00030380FF	256 B
EHRPWM4_EPWM	0x0003040000	0x00030400FF	256 B
EHRPWM4_EHRPWM	0x0003048000	0x00030480FF	256 B
EHRPWM5_EPWM	0x0003050000	0x00030500FF	256 B
EHRPWM5_EHRPWM	0x0003058000	0x00030580FF	256 B
ECAP0_CTL_STS	0x0003100000	0x00031000FF	256 B
ECAP1_CTL_STS	0x0003110000	0x00031100FF	256 B
ECAP2_CTL_STS	0x0003120000	0x00031200FF	256 B
ATL0_REG	0x00031F0000	0x00031F03FF	1 KB
EQEP0_REG	0x0003200000	0x00032000FF	256 B
EQEP1_REG	0x0003210000	0x00032100FF	256 B
EQEP2_REG	0x0003220000	0x00032200FF	256 B
PBIST1	0x0003300000	0x00033003FF	1 KB
PBIST3	0x0003310000	0x00033103FF	1 KB
PBIST0	0x0003320000	0x00033203FF	1 KB
PBIST2	0x0003330000	0x00033303FF	1 KB
PBIST4	0x0003340000	0x00033403FF	1 KB
PBIST7	0x0003360000	0x00033603FF	1 KB
PBIST5	0x0003370000	0x00033703FF	1 KB
PBIST9	0x0003380000	0x00033803FF	1 KB
PBIST10	0x0003390000	0x00033903FF	1 KB
GPU0_PBIST_CFG	0x00033A0000	0x00033A03FF	1 KB
USART_PSILSS16_MMRS	0x0003400000	0x0003400FFF	4 KB
MISC_PSILSS12_MMRS	0x0003404000	0x0003404FFF	4 KB
DEBUG_PSILSS4_MMRS	0x0003408000	0x0003408FFF	4 KB
AASRC_PSILSS1_MMRS	0x000340C000	0x000340CFFF	4 KB
CSI_PSILSS0_MMRS	0x0003410000	0x0003410FFF	4 KB
MSRAM16KX256E0_RAM	0x0003600000	0x000367FFFF	512 KB
NAVSS0_NBSS_CFG_REGS0_MMRS	0x0003800000	0x00038000FF	256 B
NAVSS0_NBSS_CFG_ECCAGGR0_REGS	0x0003801000	0x00038013FF	1 KB
NAVSS0_NBSS_NB0_CFG_MMRS	0x0003802000	0x00038020FF	256 B
NAVSS0_NBSS_NB1_CFG_MMRS	0x0003803000	0x00038030FF	256 B
NAVSS0_NBSS_CFG_MSMC0_SLV_VIRTID_CFG_MMRS	0x0003810000	0x00038100FF	256 B
USB0_MMR_MMRVBP_USBSS_CMN	0x0004104000	0x00041040FF	256 B
USB0_PHY2	0x0004108000	0x00041083FF	1 KB
USB1_MMR_MMRVBP_USBSS_CMN	0x0004114000	0x00041140FF	256 B
USB1_PHY2	0x0004118000	0x00041183FF	1 KB
ENCODER0_REG_AXI	0x0004205000	0x00042053FF	1 KB
DECODER0_IMG_VIDEO_BUS4_MMU	0x0004301000	0x00043011FF	512 B
DECODER0_MSVDX_AXI	0x0004301800	0x0004301807	8 B
DECODER0_IMG_VIDEO_BUS4_MMU2	0x0004321000	0x00043211FF	512 B
DECODER0_MSVDX_AXI2	0x0004321800	0x0004321807	8 B
CSI_TX_IF0_TX_SHIM_VBUSP_MMR_CSI2TXIF	0x0004400000	0x0004400FFF	4 KB
CSI_TX_IF0_VBUS2APB_WRAP_VBUSP_APB_CSI2TX	0x0004404000	0x0004404FFF	4 KB

**Table 2-1. MAIN Domain Memory Map (continued)**

Region Name	Start Address	End Address	Size
CSI_TX_IF0_CP_INTD_CFG_INTD_CFG	0x0004408000	0x0004408FFF	4 KB
DPHY_TX0	0x0004480000	0x0004480FFF	4 KB
CSI_RX_IF0_RX_SHIM_VBUSP_MMR_CSI2RXIF	0x0004500000	0x0004500FFF	4 KB
CSI_RX_IF0_VBUS2APB_WRAP_VBUSP_APB_CSI2RX	0x0004504000	0x0004504FFF	4 KB
CSI_RX_IF0_CP_INTD_CFG_INTD_CFG	0x0004508000	0x0004508FFF	4 KB
CSI_RX_IF1_RX_SHIM_VBUSP_MMR_CSI2RXIF	0x0004510000	0x0004510FFF	4 KB
CSI_RX_IF1_VBUS2APB_WRAP_VBUSP_APB_CSI2RX	0x0004514000	0x0004514FFF	4 KB
CSI_RX_IF1_CP_INTD_CFG_INTD_CFG	0x0004518000	0x0004518FFF	4 KB
DPHY_RX0_VBUS2APB_WRAP_VBUSP_K3_DPHY_RX	0x0004580000	0x0004580FFF	4 KB
DPHY_RX0_MMR_SLV_K3_DPHY_WRAP	0x0004581000	0x00045810FF	256 B
DPHY_RX1_VBUS2APB_WRAP_VBUSP_K3_DPHY_RX	0x0004590000	0x0004590FFF	4 KB
DPHY_RX1_MMR_SLV_K3_DPHY_WRAP	0x0004591000	0x00045910FF	256 B
DSS_DSI0_DSI_TOP_ECC_AGGR_SYS_CFG	0x0004700000	0x00047003FF	1 KB
DSS_DSI0_DSI_WRAP_MMR_VBUSP_CFG_DSI_WRAP	0x0004710000	0x00047100FF	256 B
DSS_DSI0_DSI_TOP_VBUSP_CFG_DSI_0_DSI	0x0004800000	0x00048FFFFFFF	1 MB
DSS0_DISPC_0_COMMON_M	0x0004A00000	0x0004A0FFFF	64 KB
DSS0_VIDL1	0x0004A20000	0x0004A2FFFF	64 KB
DSS0_VIDL2	0x0004A30000	0x0004A3FFFF	64 KB
DSS0_VID1	0x0004A50000	0x0004A5FFFF	64 KB
DSS0_VID2	0x0004A60000	0x0004A6FFFF	64 KB
DSS0_OVR1	0x0004A70000	0x0004A7FFFF	64 KB
DSS0_VP1	0x0004A80000	0x0004A8FFFF	64 KB
DSS0_OVR2	0x0004A90000	0x0004A9FFFF	64 KB
DSS0_VP2	0x0004AA0000	0x0004AAFFFF	64 KB
DSS0_OVR3	0x0004AB0000	0x0004ABFFFF	64 KB
DSS0_VP3	0x0004AC0000	0x0004ACFFFF	64 KB
DSS0_OVR4	0x0004AD0000	0x0004ADFFFF	64 KB
DSS0_VP4	0x0004AE0000	0x0004AEFFFF	64 KB
DSS0_WB	0x0004AF0000	0x0004AFFFFFFF	64 KB
SA2_UL0	0x0004E00000	0x0004E00FFF	4 KB
SA2_UL0_MMRA	0x0004E01000	0x0004E011FF	512 B
SA2_UL0_EIP_76	0x0004E10000	0x0004E1007F	128 B
SA2_UL0_EIP_29T2	0x0004E20000	0x0004E2FFFF	64 KB
UFS0_SYSCFG_SS_CFG	0x0004E80000	0x0004E800FF	256 B
UFS0_P2A_WRAP_CFG_VBP_UFSHCI	0x0004E84000	0x0004E85FFF	8 KB
DSS_EDP0_INTG_CFG_VP	0x0004F40000	0x0004F400FF	256 B
DSS_EDP0_V2A_S_CORE_VP_REGS_SAPB	0x0004F48000	0x0004F480FF	256 B
MMCSD0_CTL_CFG	0x0004F80000	0x0004F80FFF	4 KB
MMCSD0_SS_CFG	0x0004F88000	0x0004F883FF	1 KB
MMCSD2_SS_CFG	0x0004F90000	0x0004F903FF	1 KB
MMCSD2_CTL_CFG	0x0004F98000	0x0004F98FFF	4 KB
MMCSD1_CTL_CFG	0x0004FB0000	0x0004FB0FFF	4 KB
MMCSD1_SS_CFG	0x0004FB8000	0x0004FB83FF	1 KB
SERDES_16G0	0x0005000000	0x0005000FFF	64 KB
SERDES_16G1	0x0005010000	0x000501FFFF	64 KB

**Table 2-1. MAIN Domain Memory Map (continued)**

Region Name	Start Address	End Address	Size
SERDES_16G2	0x0005020000	0x000502FFFF	64 KB
SERDES_16G3	0x0005030000	0x000503FFFF	64 KB
SERDES_10G0	0x0005050000	0x000505FFFF	64 KB
ELM0	0x0005380000	0x0005380FFF	4 KB
GPMC0_CFG	0x0005390000	0x00053903FF	1 KB
R5FSS0_COMPARE_CFG	0x0005B00000	0x0005B000FF	256 B
R5FSS0_ECC_AGGR	0x0005B10000	0x0005B103FF	1 KB
R5FSS1_COMPARE_CFG	0x0005B20000	0x0005B200FF	256 B
R5FSS1_ECC_AGGR	0x0005B30000	0x0005B303FF	1 KB
R5FSS0_CORE0_ATCM	0x0005C00000	0x0005C07FFF	32 KB
R5FSS0_CORE0_BTCM	0x0005C10000	0x0005C17FFF	32 KB
R5FSS0_CORE1_ATCM <sup>(1)</sup>	0x0005D00000	0x0005D07FFF	32 KB
R5FSS0_CORE1_BTCM <sup>(1)</sup>	0x0005D10000	0x0005D17FFF	32 KB
R5FSS1_CORE0_ATCM	0x0005E00000	0x0005E07FFF	32 KB
R5FSS1_CORE0_BTCM	0x0005E10000	0x0005E17FFF	32 KB
R5FSS1_CORE1_ATCM <sup>(1)</sup>	0x0005F00000	0x0005F07FFF	32 KB
R5FSS1_CORE1_BTCM <sup>(1)</sup>	0x0005F10000	0x0005F17FFF	32 KB
USB0_VBP2APB_WRAP_CONTROLLER_VBP_CORE_ADDR_MAP	0x0006000000	0x00063FFFFFFF	4 MB
USB1_VBP2APB_WRAP_CONTROLLER_VBP_CORE_ADDR_MAP	0x0006400000	0x00067FFFFFFF	4 MB
DEBUGSS1_SYS	0x0008000000	0x0008000FFF	4 KB
DEBUGSS0_SYS	0x0008004000	0x0008004FFF	4 KB
CCDEBUGSS0_SYS	0x0008008000	0x0008008FFF	4 KB
C66DEBUGSS0_SYS	0x0008010000	0x0008010FFF	4 KB
C66DEBUGSS1_SYS	0x0008020000	0x0008020FFF	4 KB
STM0_STIMULUS	0x0009000000	0x0009FFFFFF	16 MB
DSS_EDP0_V2A_CORE_VP_REGS_APB	0x000A000000	0x000A03FFFF	256 KB
PRU_ICSSG0_DRAM0_SLV_RAM	0x000B000000	0x000B001FFF	8 KB
PRU_ICSSG0_DRAM1_SLV_RAM	0x000B002000	0x000B003FFF	8 KB
PRU_ICSSG0_PR1_RTU0_PR1_RTU0_IRAM_RAM	0x000B004000	0x000B005FFF	8 KB
PRU_ICSSG0_PR1_RTU1_PR1_RTU1_IRAM_RAM	0x000B006000	0x000B007FFF	8 KB
PRU_ICSSG0_RAT_SLICE0_CFG	0x000B008000	0x000B008FFF	4 KB
PRU_ICSSG0_RAT_SLICE1_CFG	0x000B009000	0x000B009FFF	4 KB
PRU_ICSSG0_RAM_SLV_RAM	0x000B010000	0x000B01FFFF	64 KB
PRU_ICSSG0_PR1_ICSS_INTC_INTC_SLV	0x000B020000	0x000B021FFF	8 KB
PRU_ICSSG0_PR1_PDSP0_IRAM	0x000B022000	0x000B0220FF	256 B
PRU_ICSSG0_PR1_PDSP0_IRAM_DEBUG	0x000B022400	0x000B0224FF	256 B
PRU_ICSSG0_PR1_RTU0_PR1_RTU0_IRAM	0x000B023000	0x000B0230FF	256 B
PRU_ICSSG0_PR1_RTU0_PR1_RTU0_IRAM_DEBUG	0x000B023400	0x000B0234FF	256 B
PRU_ICSSG0_PR1_RTU1_PR1_RTU1_IRAM	0x000B023800	0x000B0238FF	256 B
PRU_ICSSG0_PR1_RTU1_PR1_RTU1_IRAM_DEBUG	0x000B023C00	0x000B023CFF	256 B
PRU_ICSSG0_PR1_PDSP1_IRAM	0x000B024000	0x000B0240FF	256 B
PRU_ICSSG0_PR1_PDSP1_IRAM_DEBUG	0x000B024400	0x000B0244FF	256 B
PRU_ICSSG0_PR1_PROT_SLV	0x000B024C00	0x000B024CFF	256 B
PRU_ICSSG0_PR1_CFG_SLV	0x000B026000	0x000B0261FF	512 B
PRU_ICSSG0_PA_STAT_WRAP_PA_SLV_QSTAT	0x000B027000	0x000B027FFF	4 KB

**Table 2-1. MAIN Domain Memory Map (continued)**

Region Name	Start Address	End Address	Size
PRU_ICSSG0_PR1_ICSS_UART_UART_SLV	0x000B028000	0x000B02803F	64 B
PRU_ICSSG0_PR1_TASKS_MGR_PRU0_PR1_TASKS_MGR_PRU0_MMR	0x000B02A000	0x000B02A0FF	256 B
PRU_ICSSG0_PR1_TASKS_MGR_RTU0_PR1_TASKS_MGR_RTU0_MMR	0x000B02A100	0x000B02A1FF	256 B
PRU_ICSSG0_PR1_TASKS_MGR_PRU1_PR1_TASKS_MGR_PRU1_MMR	0x000B02A200	0x000B02A2FF	256 B
PRU_ICSSG0_PR1_TASKS_MGR_RTU1_PR1_TASKS_MGR_RTU1_MMR	0x000B02A300	0x000B02A3FF	256 B
PRU_ICSSG0_PA_STAT_WRAP_PA_SLV_CSTAT	0x000B02C000	0x000B02CFFF	4 KB
PRU_ICSSG0_IEP0	0x000B02E000	0x000B02EFFF	4 KB
PRU_ICSSG0_IEP1	0x000B02F000	0x000B02FFFF	4 KB
PRU_ICSSG0_PR1_ICSS_ECAP0_ECAP_SLV	0x000B030000	0x000B0300FF	256 B
PRU_ICSSG0_PR1_MII_RT_PR1_MII_RT_CFG	0x000B032000	0x000B0320FF	256 B
PRU_ICSSG0_PR1_MII_RT_PR1_SGMII0_CFG_SGMII0	0x000B032100	0x000B0321FF	256 B
PRU_ICSSG0_PR1_MII_RT_PR1_SGMII1_CFG_SGMII1	0x000B032200	0x000B0322FF	256 B
PRU_ICSSG0_PR1_MDIO_V1P7_MDIO	0x000B032400	0x000B0324FF	256 B
PRU_ICSSG0_PR1_MII_RT_PR1_MII_RT_G_CFG_REGS_G	0x000B033000	0x000B033FFF	4 KB
PRU_ICSSG0_PR1_PDSP0_IRAM_RAM	0x000B034000	0x000B037FFF	16 KB
PRU_ICSSG0_PR1_PDSP1_IRAM_RAM	0x000B038000	0x000B03BFFF	16 KB
PRU_ICSSG0_PA_STAT_WRAP_PA_SLV	0x000B03C000	0x000B03C0FF	256 B
PRU_ICSSG1_DRAM0_SLV_RAM	0x000B100000	0x000B101FFF	8 KB
PRU_ICSSG1_DRAM1_SLV_RAM	0x000B102000	0x000B103FFF	8 KB
PRU_ICSSG1_PR1_RTU0_PR1_RTU0_IRAM_RAM	0x000B104000	0x000B105FFF	8 KB
PRU_ICSSG1_PR1_RTU1_PR1_RTU1_IRAM_RAM	0x000B106000	0x000B107FFF	8 KB
PRU_ICSSG1_RAT_SLICE0_CFG	0x000B108000	0x000B108FFF	4 KB
PRU_ICSSG1_RAT_SLICE1_CFG	0x000B109000	0x000B109FFF	4 KB
PRU_ICSSG1_RAM_SLV_RAM	0x000B110000	0x000B11FFFF	64 KB
PRU_ICSSG1_PR1_ICSS_INTC_INTC_SLV	0x000B120000	0x000B121FFF	8 KB
PRU_ICSSG1_PR1_PDSP0_IRAM	0x000B122000	0x000B1220FF	256 B
PRU_ICSSG1_PR1_PDSP0_IRAM_DEBUG	0x000B122400	0x000B1224FF	256 B
PRU_ICSSG1_PR1_RTU0_PR1_RTU0_IRAM	0x000B123000	0x000B1230FF	256 B
PRU_ICSSG1_PR1_RTU0_PR1_RTU0_IRAM_DEBUG	0x000B123400	0x000B1234FF	256 B
PRU_ICSSG1_PR1_RTU1_PR1_RTU1_IRAM	0x000B123800	0x000B1238FF	256 B
PRU_ICSSG1_PR1_RTU1_PR1_RTU1_IRAM_DEBUG	0x000B123C00	0x000B123CFF	256 B
PRU_ICSSG1_PR1_PDSP1_IRAM	0x000B124000	0x000B1240FF	256 B
PRU_ICSSG1_PR1_PDSP1_IRAM_DEBUG	0x000B124400	0x000B1244FF	256 B
PRU_ICSSG1_PR1_PROT_SLV	0x000B124C00	0x000B124CFF	256 B
PRU_ICSSG1_PR1_CFG_SLV	0x000B126000	0x000B1261FF	512 B
PRU_ICSSG1_PA_STAT_WRAP_PA_SLV_QSTAT	0x000B127000	0x000B127FFF	4 KB
PRU_ICSSG1_PR1_ICSS_UART_UART_SLV	0x000B128000	0x000B12803F	64 B
PRU_ICSSG1_PR1_TASKS_MGR_PRU0_PR1_TASKS_MGR_PRU0_MMR	0x000B12A000	0x000B12A0FF	256 B
PRU_ICSSG1_PR1_TASKS_MGR_RTU0_PR1_TASKS_MGR_RTU0_MMR	0x000B12A100	0x000B12A1FF	256 B
PRU_ICSSG1_PR1_TASKS_MGR_PRU1_PR1_TASKS_MGR_PRU1_MMR	0x000B12A200	0x000B12A2FF	256 B
PRU_ICSSG1_PR1_TASKS_MGR_RTU1_PR1_TASKS_MGR_RTU1_MMR	0x000B12A300	0x000B12A3FF	256 B
PRU_ICSSG1_PA_STAT_WRAP_PA_SLV_CSTAT	0x000B12C000	0x000B12CFFF	4 KB
PRU_ICSSG1_IEP0	0x000B12E000	0x000B12EFFF	4 KB
PRU_ICSSG1_IEP1	0x000B12F000	0x000B12FFFF	4 KB
PRU_ICSSG1_PR1_ICSS_ECAP0_ECAP_SLV	0x000B130000	0x000B1300FF	256 B



**Table 2-1. MAIN Domain Memory Map (continued)**

Region Name	Start Address	End Address	Size
PRU_ICSSG1_PR1_MII_RT_PR1_MII_RT_CFG	0x000B132000	0x000B1320FF	256 B
PRU_ICSSG1_PR1_MII_RT_PR1_SGMII0_CFG_SGMII0	0x000B132100	0x000B1321FF	256 B
PRU_ICSSG1_PR1_MII_RT_PR1_SGMII1_CFG_SGMII1	0x000B132200	0x000B1322FF	256 B
PRU_ICSSG1_PR1_MDIO_V1P7_MDIO	0x000B132400	0x000B1324FF	256 B
PRU_ICSSG1_PR1_MII_RT_PR1_MII_RT_G_CFG_REGS_G	0x000B133000	0x000B133FFF	4 KB
PRU_ICSSG1_PR1_PDSP0_IRAM_RAM	0x000B134000	0x000B137FFF	16 KB
PRU_ICSSG1_PR1_PDSP1_IRAM_RAM	0x000B138000	0x000B13BFFF	16 KB
PRU_ICSSG1_PA_STAT_WRAP_PA_SLV	0x000B13C000	0x000B13C0FF	256 B
PRU_ICSSG0_ECC_AGGR	0x000BF00000	0x000BF003FF	1 KB
PRU_ICSSG1_ECC_AGGR	0x000BF01000	0x000BF013FF	1 KB
CPSW0_NUSS	0x000C000000	0x000C1FFFFFFF	2 MB
PCIE0_CORE_DBN_CFG_PCIE_CORE	0x000D000000	0x000D7FFFFFFF	8 MB
PCIE1_CORE_DBN_CFG_PCIE_CORE	0x000D800000	0x000DFFFFFFF	8 MB
PCIE2_CORE_DBN_CFG_PCIE_CORE	0x000E000000	0x000E7FFFFFFF	8 MB
PCIE3_CORE_DBN_CFG_PCIE_CORE	0x000E800000	0x000EFFFFFFF	8 MB
VPAC0_VPAC_REGS_VPAC_REGS_CFG_IP_MMRS	0x000F000000	0x000F0003FF	1 KB
VPAC0_CTSET2_WRAP_CFG_CTSET2_CFG	0x000F002000	0x000F003FFF	8 KB
VPAC0_CP_INTD_CFG_INTD_CFG	0x000F004000	0x000F004FFF	4 KB
VPAC0_HTS_S_VBUSP	0x000F008000	0x000F00FFFF	32 KB
VPAC0_PAR_VPAC_LDC0_S_VBUSP_MMR_VBUSP	0x000F020000	0x000F0203FF	1 KB
VPAC0_PAR_VPAC_LDC0_S_VBUSP_VPAC_LDC_LSE_CFG_VP	0x000F020400	0x000F0205FF	512 B
VPAC0_PAR_VPAC_LDC0_S_VBUSP_PIXWRINTF_DUALY_LUTCFG_DUALY_LUT	0x000F020800	0x000F020FFF	2 KB
VPAC0_PAR_VPAC_LDC0_S_VBUSP_PIXWRINTF_DUALC_LUTCFG_DUALC_LUT	0x000F021000	0x000F0217FF	2 KB
VPAC0_PAR_VPAC_LDC0_S_VBUSP_MEMCFG_LOOP_MESH_VBUSPI_MESH_MEM	0x000F022000	0x000F023FFF	8 KB
VPAC0_PAR_VPAC_LDC0_S_VBUSP_MEMCFG_LOOP_Y_VBUSPI_Y_MEM	0x000F028000	0x000F02FFFF	32 KB
VPAC0_PAR_VPAC_LDC0_S_VBUSP_MEMCFG_LOOP_CBCR_VBUSPI_CBCR_MEM	0x000F030000	0x000F037FFF	32 KB
VPAC0_PAR_VPAC_VISS0_S_VBUSP_MMR_CFG_VISS_TOP	0x000F080000	0x000F0801FF	512 B
VPAC0_PAR_VPAC_VISS0_S_VBUSP_VPAC_VISS_LSE_CFG_VP	0x000F080400	0x000F0805FF	512 B
VPAC0_PAR_VPAC_VISS0_S_VBUSP_K3_GLBCE_TOP_CFG_GLBCE	0x000F083800	0x000F083FFF	2 KB
VPAC0_PAR_VPAC_VISS0_S_VBUSP_K3_GLBCE_TOP_STATMEM_CFG_GLBCE_STATMEM	0x000F084000	0x000F087FFF	16 KB
VPAC0_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_CFA_VBUSP_FLEXCFA	0x000F088000	0x000F08FFFF	32 KB
VPAC0_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_FCC_VBUSP_FLEXCC	0x000F090000	0x000F0907FF	2 KB
VPAC0_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_FCC_VBUSP_FLEXCC_CONTRASTC1	0x000F090800	0x000F090FFF	2 KB
VPAC0_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_FCC_VBUSP_FLEXCC_CONTRASTC2	0x000F091000	0x000F0917FF	2 KB
VPAC0_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_FCC_VBUSP_FLEXCC_CONTRASTC3	0x000F091800	0x000F091FFF	2 KB
VPAC0_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_FCC_VBUSP_FLEXCC_Y8R8	0x000F092000	0x000F0927FF	2 KB
VPAC0_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_FCC_VBUSP_FLEXCC_C8G8	0x000F092800	0x000F092FFF	2 KB

**Table 2-1. MAIN Domain Memory Map (continued)**

Region Name	Start Address	End Address	Size
VPAC0_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_FCC_VBUSP_FLEXCC_S8B8	0x000F093000	0x000F0937FF	2 KB
VPAC0_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_FCC_VBUSP_FLEXCC_HIST	0x000F093800	0x000F093FFF	2 KB
VPAC0_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_FCC_VBUSP_FLEXCC_LINE	0x000F098000	0x000F0987FF	2 KB
VPAC0_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_MMR_S_VBUSP_RAWFE_CFG	0x000F0A0000	0x000F0A03FF	1 KB
VPAC0_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_H3A_WRAP_CFG_RAWFE_H3A_CFG	0x000F0A0400	0x000F0A04FF	256 B
VPAC0_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_LUT3_RAM_RAWFE_PWL_LUT3_RAM	0x000F0A0800	0x000F0A0FFF	2 KB
VPAC0_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_LUT2_RAM_RAWFE_PWL_LUT2_RAM	0x000F0A1000	0x000F0A17FF	2 KB
VPAC0_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_LUT1_RAM_RAWFE_PWL_LUT1_RAM	0x000F0A1800	0x000F0A1FFF	2 KB
VPAC0_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_WDR_LUT_RAM_RAWFE_WDR_LUT_RAM	0x000F0A2000	0x000F0A27FF	2 KB
VPAC0_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_H3A_LUT_RAM_RAWFE_H3A_LUT_RAM	0x000F0A2800	0x000F0A2FFF	2 KB
VPAC0_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_DPC_RAM_RAWFE_DPC_LUT_RAM	0x000F0A3000	0x000F0A33FF	1 KB
VPAC0_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_DPC_LRAM_RAWFE_DPC_LRAM	0x000F0A4000	0x000F0A5FFF	8 KB
VPAC0_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_LSC_RAM_RAWFE_LSC_LUT_RAM	0x000F0A8000	0x000F0AFFFF	32 KB
VPAC0_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_H3A_WRAP_ARAM_RAWFE_H3A_ARAM	0x000F0B0000	0x000F0B1FFF	8 KB
VPAC0_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_H3A_WRAP_LRAM_RAWFE_H3A_LRAM	0x000F0B2000	0x000F0B3FFF	8 KB
VPAC0_PAR_VPAC_VISS0_S_VBUSP_VISS_NSF4V_CFG_MMR_VBUSP_NSF4VCORE	0x000F0C0000	0x000F0C03FF	1 KB
VPAC0_PAR_VPAC_VISS0_S_VBUSP_VISS_NSF4V_CFG_MEM_MMRRAM_VBUSP_MMR_RAM	0x000F0C4000	0x000F0C7FFF	16 KB
VPAC0_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_EE_VBUSP_FLEXEE	0x000F0D0000	0x000F0D7FFF	32 KB
VPAC0_PAR_VPAC_MSC_CFG_VP_CFG_VP	0x000F1C0000	0x000F1C07FF	2 KB
VPAC0_PAR_VPAC_MSC_CFG_VP_LSE_CFG_VP	0x000F1C0800	0x000F1C09FF	512 B
VPAC0_PAR_VPAC_NF_S_VBUSP_MMR_VBUSP_NF_CFG	0x000F1C2000	0x000F1C2FFF	4 KB
VPAC0_PAR_VPAC_NF_S_VBUSP_VPAC_NF_LSE_CFG_VP	0x000F1C3000	0x000F1C31FF	512 B
VPAC0_DRU_UTC_VPAC0_DRU_MMR_CFG_DRU_DRU	0x000F200000	0x000F203FFF	16 KB
VPAC0_DRU_UTC_VPAC0_DRU_MMR_CFG_DRU_DRU_SET	0x000F204000	0x000F207FFF	16 KB
VPAC0_DRU_UTC_VPAC0_DRU_MMR_CFG_DRU_DRU_QUEUE	0x000F208000	0x000F20FFFF	32 KB
VPAC0_DRU_UTC_VPAC0_DRU_MMR_CFG_DRU_DRU_CHNRT	0x000F240000	0x000F25FFFF	128 KB
VPAC0_DRU_UTC_VPAC0_DRU_MMR_CFG_DRU_DRU_CHRT	0x000F260000	0x000F27FFFF	128 KB
VPAC0_DRU_UTC_VPAC0_DRU_MMR_CFG_DRU_DRU_CHATOMIC_DEBUG	0x000F280000	0x000F29FFFF	128 KB
VPAC0_DRU_UTC_VPAC0_DRU_MMR_CFG_DRU_DRU_CAUSE	0x000F2E0000	0x000F2FFFFFFF	128 KB
VPAC0_DRU_UTC_VPAC1_DRU_MMR_CFG_DRU_DRU	0x000F300000	0x000F303FFF	16 KB
VPAC0_DRU_UTC_VPAC1_DRU_MMR_CFG_DRU_DRU_SET	0x000F304000	0x000F307FFF	16 KB
VPAC0_DRU_UTC_VPAC1_DRU_MMR_CFG_DRU_DRU_QUEUE	0x000F308000	0x000F30FFFF	32 KB
VPAC0_DRU_UTC_VPAC1_DRU_MMR_CFG_DRU_DRU_CHNRT	0x000F340000	0x000F35FFFF	128 KB
VPAC0_DRU_UTC_VPAC1_DRU_MMR_CFG_DRU_DRU_CHRT	0x000F360000	0x000F37FFFF	128 KB
VPAC0_DRU_UTC_VPAC1_DRU_MMR_CFG_DRU_DRU_CHATOMIC_DEBUG	0x000F380000	0x000F39FFFF	128 KB

**Table 2-1. MAIN Domain Memory Map (continued)**

Region Name	Start Address	End Address	Size
VPAC0_DRU_UTC_VPAC1_DRU_MMR_CFG_DRU_DRU_CAUSE	0x000F3E0000	0x000F3FFFFFFF	128 KB
DMPAC0_DMPAC_REGS_DMPAC_REGS_CFG_IP_MMRS	0x000F400000	0x000F4003FF	1 KB
DMPAC0_CP_INTD_CFG_INTD_CFG	0x000F401000	0x000F401FFF	4 KB
DMPAC0 HTS_S_VBUSP	0x000F408000	0x000F40FFFF	32 KB
DMPAC0_CTSET2_WRAP_CFG_CTSET2_CFG	0x000F420000	0x000F421FFF	8 KB
DMPAC0_DMPAC_FOCO_0_CFG_SLV_DMPAC_FOCO_CORE_FOCO_REGS_CFG_IP_MMRS	0x000F424000	0x000F42403F	64 B
DMPAC0_DMPAC_FOCO_0_CFG_SLV_VPAC_FOCO_LSE_CFG_VP	0x000F424200	0x000F4243FF	512 B
DMPAC0_DMPAC_FOCO_1_CFG_SLV_DMPAC_FOCO_CORE_FOCO_REGS_CFG_IP_MMRS	0x000F428000	0x000F42803F	64 B
DMPAC0_DMPAC_FOCO_1_CFG_SLV_VPAC_FOCO_LSE_CFG_VP	0x000F428200	0x000F4283FF	512 B
DMPAC0_PAR_DOF_CFG_VP_MMR_VBUSP_DOFCORE	0x000F480000	0x000F480FFF	4 KB
DMPAC0_PAR_DOF_CFG_VP_MEM_MMRRAM_VBUSP_MMR_RAM	0x000F4C0000	0x000F4FFFFFFF	256 KB
DMPAC0_PAR_PAR_SDE_S_VBUSP_MMR_VBUSP_MMR	0x000F500000	0x000F500FFF	4 KB
DMPAC0_PAR_PAR_SDE_S_VBUSP_MEM_MMRRAM_VBUSP_MMR_RAM	0x000F540000	0x000F57FFFF	256 KB
DMPAC0_DRU_UTC_DMPAC0_DRU_MMR_CFG_DRU_DRU	0x000F600000	0x000F603FFF	16 KB
DMPAC0_DRU_UTC_DMPAC0_DRU_MMR_CFG_DRU_DRU_SET	0x000F604000	0x000F607FFF	16 KB
DMPAC0_DRU_UTC_DMPAC0_DRU_MMR_CFG_DRU_DRU_QUEUE	0x000F608000	0x000F60FFFF	32 KB
DMPAC0_DRU_UTC_DMPAC0_DRU_MMR_CFG_DRU_DRU_CHNRT	0x000F640000	0x000F65FFFF	128 KB
DMPAC0_DRU_UTC_DMPAC0_DRU_MMR_CFG_DRU_DRU_CHRT	0x000F660000	0x000F67FFFF	128 KB
DMPAC0_DRU_UTC_DMPAC0_DRU_MMR_CFG_DRU_DRU_CHATOMIC_DEB UG	0x000F680000	0x000F69FFFF	128 KB
DMPAC0_DRU_UTC_DMPAC0_DRU_MMR_CFG_DRU_DRU_CAUSE	0x000F6E0000	0x000F6FFFFFFF	128 KB
PCIE0_DAT0	0x0010000000	0x0017FFFFFFF	128 MB
PCIE1_DAT0	0x0018000000	0x001FFFFFFF	128 MB
GPMC0_DATA	0x0020000000	0x0027FFFFFFF	128 MB
NAVSS0_MSRAM0_SLV_RAM	0x0030000000	0x003000FFFF	64 KB
NAVSS0_MODSS_INTA0_CFG	0x0030800000	0x003080001F	32 B
NAVSS0_MODSS_INTA1_CFG	0x0030801000	0x003080101F	32 B
NAVSS0_UDMASS_INTA0_CFG	0x0030802000	0x003080201F	32 B
NAVSS0_MODSS_INTA0_CFG_IMAP	0x0030900000	0x0030907FFF	32 KB
NAVSS0_MODSS_INTA1_CFG_IMAP	0x0030908000	0x003090FFFF	32 KB
NAVSS0_UDMASS_INTA0_IMAP	0x0030940000	0x003097FFFF	256 KB
NAVSS0_NAV_DDR0_VIRTID_CFG_MMRS	0x0030A02000	0x0030A020FF	256 B
NAVSS0_NAV_DDR1_VIRTID_CFG_MMRS	0x0030A03000	0x0030A030FF	256 B
NAVSS0_UDMASS_UDMAP0_CFG_TCHAN	0x0030B00000	0x0030B1FFFF	128 KB
NAVSS0_UDMASS_UDMAP0_CFG_RCHAN	0x0030C00000	0x0030C0FFFF	64 KB
NAVSS0_UDMASS_UDMAP0_CFG_RFLOW	0x0030D00000	0x0030D07FFF	32 KB
NAVSS0_SPINLOCK	0x0030E00000	0x0030E07FFF	32 KB
NAVSS0_TIMERMGR0_CFG	0x0030E80000	0x0030E801FF	512 B
NAVSS0_TIMERMGR1_CFG	0x0030E81000	0x0030E811FF	512 B
NAVSS0_TIMERMGR0_CFG_OES	0x0030F00000	0x0030F00FFF	4 KB
NAVSS0_TIMERMGR1_CFG_OES	0x0030F01000	0x0030F01FFF	4 KB
NAVSS0_ECCAGGR0_REGS	0x0031000000	0x00310003FF	1 KB
NAVSS0_UDMASS_ECCAGGR0_CFG_REGS	0x0031001000	0x00310013FF	1 KB
NAVSS0_VIRTSS_ECCAGGR_CFG	0x0031002000	0x00310023FF	1 KB

**Table 2-1. MAIN Domain Memory Map (continued)**

Region Name	Start Address	End Address	Size
NAVSS0_PAT0_CFG_MMRS	0x0031010000	0x00310100FF	256 B
NAVSS0_PAT1_CFG_MMRS	0x0031011000	0x00310110FF	256 B
NAVSS0_PAT2_CFG_MMRS	0x0031012000	0x00310120FF	256 B
NAVSS0_PAT3_CFG_MMRS	0x0031013000	0x00310130FF	256 B
NAVSS0_PAT4_CFG_MMRS	0x0031014000	0x00310140FF	256 B
NAVSS0_UDMASS_INTA0_CFG_GCNTCFG	0x0031040000	0x0031043FFF	16 KB
NAVSS0_UDMASS_RINGACC0_CFG	0x0031080000	0x00310BFFFF	256 KB
NAVSS0_CFG	0x00310C0000	0x00310C00FF	256 B
NAVSS0_CPTS	0x00310D0000	0x00310D03FF	1 KB
NAVSS0_INTR0_INTR_ROUTER_CFG	0x00310E0000	0x00310E3FFF	16 KB
NAVSS0_UDMASS_INTA0_CFG_L2G	0x0031100000	0x0031100FFF	4 KB
NAVSS0_UDMASS_INTA0_CFG_MCAST	0x0031110000	0x0031113FFF	16 KB
NAVSS0_PROXY0_CFG_BUF_CFG	0x0031120000	0x00311200FF	256 B
NAVSS0_PROXY_BUF	0x0031130000	0x0031133FFF	16 KB
NAVSS0_SEC_PROXY0_CFG_MMRS	0x0031140000	0x00311400FF	256 B
NAVSS0_UDMASS_UDMAP0_CFG	0x0031150000	0x00311500FF	256 B
NAVSS0_UDMASS_RINGACC0_GCFG	0x0031160000	0x00311603FF	1 KB
NAVSS0_PSILSS_LOCAL_CFG_MMRS	0x0031170000	0x0031170FFF	4 KB
NAVSS0_PSILSS_GLOBAL_CFG_MMRS	0x0031180000	0x0031180FFF	4 KB
NAVSS0_PSILSS_TR_CFG_MMRS	0x0031190000	0x0031190FFF	4 KB
NAVSS0_MCRC	0x0031F70000	0x0031F70FFF	4 KB
NAVSS0_UDMASS_PSILCFG0_CFG_PROXY	0x0031F78000	0x0031F781FF	512 B
NAVSS0_MAILBOX_REGS0	0x0031F80000	0x0031F801FF	512 B
NAVSS0_MAILBOX_REGS1	0x0031F81000	0x0031F811FF	512 B
NAVSS0_MAILBOX_REGS2	0x0031F82000	0x0031F821FF	512 B
NAVSS0_MAILBOX_REGS3	0x0031F83000	0x0031F831FF	512 B
NAVSS0_MAILBOX_REGS4	0x0031F84000	0x0031F841FF	512 B
NAVSS0_MAILBOX_REGS5	0x0031F85000	0x0031F851FF	512 B
NAVSS0_MAILBOX_REGS6	0x0031F86000	0x0031F861FF	512 B
NAVSS0_MAILBOX_REGS7	0x0031F87000	0x0031F871FF	512 B
NAVSS0_MAILBOX_REGS8	0x0031F88000	0x0031F881FF	512 B
NAVSS0_MAILBOX_REGS9	0x0031F89000	0x0031F891FF	512 B
NAVSS0_MAILBOX_REGS10	0x0031F8A000	0x0031F8A1FF	512 B
NAVSS0_MAILBOX_REGS11	0x0031F8B000	0x0031F8B1FF	512 B
NAVSS0_UDMASS_RINGACC0_CFG_MON	0x0032000000	0x003201FFFF	128 KB
NAVSS0_TIMERMGR0_CFG_TIMERS	0x0032200000	0x003223FFFF	256 KB
NAVSS0_TIMERMGR1_CFG_TIMERS	0x0032240000	0x003227FFFF	256 KB
NAVSS0_SEC_PROXY0_CFG_RT	0x0032400000	0x00324FFFFFFF	1 MB
NAVSS0_SEC_PROXY0_CFG_SCFG	0x0032800000	0x00328FFFFFFF	1 MB
NAVSS0_SEC_PROXY0_SRC_TARGET_DATA	0x0032C00000	0x0032CFFFFFFF	1 MB
NAVSS0_PROXY_TARGET0_DATA	0x0033000000	0x003303FFFF	256 KB
NAVSS0_PROXY0_BUF_CFG	0x0033400000	0x003343FFFF	256 KB
NAVSS0_UDMASS_INTA0_CFG_GCNTRTI	0x0033800000	0x00339FFFFFFF	2 MB
NAVSS0_MODSS_INTA0_CFG_INTR	0x0033C00000	0x0033C3FFFF	256 KB
NAVSS0_MODSS_INTA1_CFG_INTR	0x0033C40000	0x0033C7FFFF	256 KB

**Table 2-1. MAIN Domain Memory Map (continued)**

Region Name	Start Address	End Address	Size
NAVSS0_UDMASS_INTA0_CFG_INTR	0x0033D00000	0x0033DFFFFFFF	1 MB
NAVSS0_UDMASS_UDMAP0_CFG_RCHANRT	0x0034000000	0x00340FFFFFFF	1 MB
NAVSS0_UDMASS_UDMAP0_CFG_TCHANRT	0x0035000000	0x00351FFFFFFF	2 MB
NAVSS0_PAT0_CFG_SCRATCH	0x0036200000	0x003620FFFF	64 KB
NAVSS0_PAT1_CFG_SCRATCH	0x0036210000	0x003621FFFF	64 KB
NAVSS0_PAT2_CFG_SCRATCH	0x0036220000	0x003622FFFF	64 KB
NAVSS0_PAT3_CFG_SCRATCH	0x0036230000	0x0036231FFF	8 KB
NAVSS0_PAT4_CFG_SCRATCH	0x0036240000	0x0036241FFF	8 KB
NAVSS0_PAT0_CFG_TABLE	0x0036400000	0x003643FFFF	256 KB
NAVSS0_PAT1_CFG_TABLE	0x0036440000	0x003647FFFF	256 KB
NAVSS0_PAT2_CFG_TABLE	0x0036480000	0x00364BFFFF	256 KB
NAVSS0_PAT3_CFG_TABLE	0x00364C0000	0x00364C7FFF	32 KB
NAVSS0_PAT4_CFG_TABLE	0x0036500000	0x0036507FFF	32 KB
NAVSS0_TCU_CFG	0x0036600000	0x00367FFFFFFF	2 MB
NAVSS0_UDMASS_RINGACC0_SRC_FIFOS	0x0038000000	0x00383FFFFFFF	4 MB
NAVSS0_UDMASS_RINGACC0_CFG_RT	0x003C000000	0x003C3FFFFFFF	4 MB
CBASS_INFRA0_FW	0x0045000000	0x004501FFFF	128 KB
COMPUTE_CLUSTER0_A72SS0_COMMON_FW	0x0045040400	0x00450407FF	1 KB
COMPUTE_CLUSTER0_C71SS0_L2_FW	0x0045042000	0x00450423FF	1 KB
COMPUTE_CLUSTER0_C71SS0_MDMA_FW	0x0045042400	0x00450427FF	1 KB
COMPUTE_CLUSTER0_DRU_FW	0x0045047000	0x00450473FF	1 KB
COMPUTE_CLUSTER0_DRU_MMR_FW	0x0045048000	0x004504FFFF	32 KB
CBASS_IPPHY0_FW	0x0045200000	0x004523FFFF	256 KB
CBASS_RC0_FW	0x0045240000	0x0045243FFF	16 KB
CBASS_RC_CFG0_FW	0x0045250000	0x004525FFFF	64 KB
CBASS_CSIO_FW	0x0045260000	0x0045267FFF	32 KB
CBASS_DATADEBUG0_FW	0x0045268000	0x004526FFFF	32 KB
CBASS_MCASP_G1_0_FW	0x0045270000	0x0045277FFF	32 KB
CBASS_HC0_FW	0x0045278000	0x0045279FFF	8 KB
CBASS_HC_CFG0_FW	0x0045280000	0x0045287FFF	32 KB
CBASS_MCASP_G0_0_FW	0x0045288000	0x004528BFFF	16 KB
CBASS_AASRC0_FW	0x004528C000	0x004528FFFF	16 KB
CBASS_AC_CFG0_FW	0x00452A0000	0x00452AFFFF	64 KB
NAVSS0_UDMASS_DMSC_FW	0x0045400000	0x004547FFFF	512 KB
NAVSS0_MODSS_DMSC_FW	0x0045480000	0x004549FFFF	128 KB
NAVSS0_VIRTSS_DMSC_FW	0x00454A0000	0x00454A7FFF	32 KB
DMPAC0_CFG_DMSC_FW	0x00455D8000	0x00455DBFFF	16 KB
DMPAC0_SL2_DMSC_FW	0x00455E0000	0x00455E0FFF	4 KB
VPAC0_DMSC_VPAC_SCRPFW	0x00455E8000	0x00455EBFFF	16 KB
VPAC0_DMSC_VPAC_SCRMFW	0x00455F0000	0x00455F0FFF	4 KB
COMPUTE_CLUSTER0_DMSC_PRIVID	0x0045830000	0x0045833FFF	16 KB
VPAC0_DMSC_VPAC_SCRMISC	0x0045860000	0x0045861FFF	8 KB
COMPUTE_CLUSTER0_DMSC_EMULATION	0x0045900000	0x0045903FFF	16 KB
MAIN_SEC_MMR0_DBG_CTRL	0x0045944000	0x0045947FFF	16 KB
COMPUTE_CLUSTER0_DMSC_BOOT	0x0045A00000	0x0045A0FFFF	64 KB



**Table 2-1. MAIN Domain Memory Map (continued)**

Region Name	Start Address	End Address	Size
MAIN_SEC_MMR0_BOOT_CTRL	0x0045A40000	0x0045A43FFF	16 KB
CBASS_AC0_GLB	0x0045B08000	0x0045B083FF	1 KB
NAVSS0_MODSS_DMSC_GLB	0x0045B0A000	0x0045B0A3FF	1 KB
NAVSS0_UDMASS_DMSC_GLB	0x0045B0B000	0x0045B0B3FF	1 KB
NAVSS0_VIRTSS_DMSC_GLB	0x0045B0B800	0x0045B0BBFF	1 KB
CBASS_INFRA0_GLB	0x0045B0C000	0x0045B0C3FF	1 KB
DMPAC0_CFG_DMSC_GLB	0x0045B0D000	0x0045B0D3FF	1 KB
DMPAC0_SL2_DMSC_GLB	0x0045B0D400	0x0045B0D7FF	1 KB
VPAC0_DMSC_VPAC_SCRMGLB	0x0045B0D800	0x0045B0DBFF	1 KB
VPAC0_DMSC_VPAC_SCRPGLB	0x0045B0DC00	0x0045B0DFFF	1 KB
COMPUTE_CLUSTER0_A72SS0_COMMON_FW_GLB	0x0045B10400	0x0045B107FF	1 KB
COMPUTE_CLUSTER0_C71SS0_L2_FW_GLB	0x0045B12000	0x0045B123FF	1 KB
COMPUTE_CLUSTER0_C71SS0_MDMA_FW_GLB	0x0045B12400	0x0045B127FF	1 KB
COMPUTE_CLUSTER0_DRU_FW_GLB	0x0045B17000	0x0045B173FF	1 KB
COMPUTE_CLUSTER0_DRU_MMR_FW_GLB	0x0045B18000	0x0045B183FF	1 KB
CBASS_AASRC0_GLB	0x0045B20000	0x0045B203FF	1 KB
CBASS_CSIO_GLB	0x0045B20400	0x0045B207FF	1 KB
CBASS_DATADEBUG0_GLB	0x0045B20800	0x0045B20BFF	1 KB
CBASS_HC0_GLB	0x0045B20C00	0x0045B20FFF	1 KB
CBASS_HC_CFG0_GLB	0x0045B21000	0x0045B213FF	1 KB
CBASS_IPPHY0_GLB	0x0045B21400	0x0045B217FF	1 KB
CBASS_MCASP_G0_0_GLB	0x0045B21800	0x0045B21BFF	1 KB
CBASS_MCASP_G1_0_GLB	0x0045B21C00	0x0045B21FFF	1 KB
CBASS_RC0_GLB	0x0045B22000	0x0045B223FF	1 KB
CBASS_RC_CFG0_GLB	0x0045B22400	0x0045B227FF	1 KB
CBASS_HC2_0_GLB	0x0045B22800	0x0045B22BFF	1 KB
CBASS_AC_CFG0_GLB	0x0045B22C00	0x0045B22FFF	1 KB
COMPUTE_CLUSTER0_A72SS0	0x0060000000	0x0060FFFFFF	16 MB
NAVSS0_SRAM0	0x0060000000	0x007FFFFFFFFF	512 MB
NAVSS0_SRAM1	0x0060000000	0x007FFFFFFFFF	512 MB
COMPUTE_CLUSTER0_RESERVED1	0x0061000000	0x0061FFFFFF	16 MB
COMPUTE_CLUSTER0_RESERVED2	0x0062000000	0x0062FFFFFF	16 MB
COMPUTE_CLUSTER0_RESERVED3	0x0063000000	0x0063FFFFFF	16 MB
COMPUTE_CLUSTER0_C71SS0	0x0064000000	0x0064FFFFFF	16 MB
COMPUTE_CLUSTER0_RESERVED5	0x0065000000	0x0065FFFFFF	16 MB
COMPUTE_CLUSTER0_RESERVED6	0x0066000000	0x0066FFFFFF	16 MB
COMPUTE_CLUSTER0_RESERVED7	0x0067000000	0x0067FFFFFF	16 MB
COMPUTE_CLUSTER0_RESERVED8	0x0068000000	0x0068FFFFFF	16 MB
COMPUTE_CLUSTER0_RESERVED9	0x0069000000	0x0069FFFFFF	16 MB
COMPUTE_CLUSTER0_RESERVED10	0x006A000000	0x006AFFFFFF	16 MB
COMPUTE_CLUSTER0_RESERVED11	0x006B000000	0x006BFFFFFF	16 MB
COMPUTE_CLUSTER0_CPU12	0x006C000000	0x006CFFFFFF	16 MB
COMPUTE_CLUSTER0_DRU_CFG	0x006D000000	0x006D003FFF	16 KB
COMPUTE_CLUSTER0_DRU_SET	0x006D004000	0x006D007FFF	16 KB
COMPUTE_CLUSTER0_DRU_QUEUE	0x006D008000	0x006D00FFFF	32 KB

**Table 2-1. MAIN Domain Memory Map (continued)**

Region Name	Start Address	End Address	Size
COMPUTE_CLUSTER0_DRU_MEM_ATT0	0x006D010000	0x006D01FFFF	64 KB
COMPUTE_CLUSTER0_DRU_MEM_ATT1	0x006D020000	0x006D02FFFF	64 KB
COMPUTE_CLUSTER0_DRU_MEM_ATT2	0x006D030000	0x006D03FFFF	64 KB
COMPUTE_CLUSTER0_DRU_CHNRT	0x006D040000	0x006D05FFFF	128 KB
COMPUTE_CLUSTER0_DRU_CHRT	0x006D060000	0x006D07FFFF	128 KB
COMPUTE_CLUSTER0_DRU_CHATOMIC_DEBUG	0x006D080000	0x006D09FFFF	128 KB
COMPUTE_CLUSTER0_DRU_CHCORE	0x006D0A0000	0x006D0BFFFF	128 KB
COMPUTE_CLUSTER0_DRU_CAUSE	0x006D0E0000	0x006D0FFFFFFF	128 KB
COMPUTE_CLUSTER0_MSMC_CFGS0	0x006E000000	0x006EFFFFFFF	16 MB
COMPUTE_CLUSTER0_UNALLOCATED0	0x006F000000	0x006FFFFFFF	16 MB
COMPUTE_CLUSTER0_MSMC_SRAM	0x0070000000	0x00707FFFFFFF	8 MB
COMPUTE_CLUSTER0_MSMC_ATOMIC_COUNTERS	0x0074000000	0x0077FFFFFFF	64 MB
COMPUTE_CLUSTER0_CLEC_REGS	0x0078000000	0x007FFFFFFF	128 MB
NAVSS0_DDR0_MEM	0x0080000000	0x00FFFFFFFFF	2 GB
NAVSS0_DDR1_MEM	0x0080000000	0x00FFFFFFFFF	2 GB
NAVSS0_DDR0_MEM1	0x0800000000	0x0FFFFFFFFF	32 GB
NAVSS0_DDR1_MEM1	0x0800000000	0x0FFFFFFFFF	32 GB
PCIE0_DAT1	0x4000000000	0x40FFFFFFFFF	4 GB
PCIE1_DAT1	0x4100000000	0x41FFFFFFFFF	4 GB
PCIE2_DAT1	0x4200000000	0x42FFFFFFFFF	4 GB
PCIE3_DAT1	0x4300000000	0x43FFFFFFFFF	4 GB
PCIE2_DAT0	0x4400000000	0x4407FFFFFFF	128 MB
PCIE3_DAT0	0x4410000000	0x4417FFFFFFF	128 MB
NAVSS0_PAT0_SRC_PAT	0x4800000000	0x483FFFFFFF	1 GB
NAVSS0_PAT1_SRC_PAT	0x4840000000	0x487FFFFFFF	1 GB
NAVSS0_PAT2_SRC_PAT	0x4880000000	0x48BFFFFFFF	1 GB
NAVSS0_PAT3_SRC_PAT	0x4900000000	0x497FFFFFFF	2 GB
NAVSS0_PAT4_SRC_PAT	0x4980000000	0x49FFFFFFF	2 GB
NAVSS0 Alias Memory Map (See <a href="#">Table 2-2</a> )	0x4A1F700000	0x4BF3C3FFFFFFF	7629 MB
DEBUGSS_WRAP0_ROM_TABLE_0_0	0x4C00000000	0x4C00000FFFF	4 KB
DEBUGSS_WRAP0_RESV0_0	0x4C00001000	0x4C00001FFFF	4 KB
DEBUGSS_WRAP0_CFGAP0	0x4C00002000	0x4C000020FFF	256 B
DEBUGSS_WRAP0_APBAP0	0x4C00002100	0x4C000021FFF	256 B
DEBUGSS_WRAP0_AXIAP0	0x4C00002200	0x4C000022FFF	256 B
DEBUGSS_WRAP0_PWRAP0	0x4C00002300	0x4C000023FFF	256 B
DEBUGSS_WRAP0_PVIEW0	0x4C00002400	0x4C000024FFF	256 B
DEBUGSS_WRAP0_JTAGAP0	0x4C00002500	0x4C000025FFF	256 B
DEBUGSS_WRAP0_SECAP0	0x4C00002600	0x4C000026FFF	256 B
DEBUGSS_WRAP0_CORTEX0_CFG0	0x4C00002700	0x4C000027FFF	256 B
DEBUGSS_WRAP0_CORTEX1_CFG0	0x4C00002800	0x4C000028FFF	256 B
DEBUGSS_WRAP0_CORTEX2_CFG0	0x4C00002900	0x4C000029FFF	256 B
DEBUGSS_WRAP0_CORTEX3_CFG0	0x4C00002A00	0x4C00002AFF	256 B
DEBUGSS_WRAP0_CORTEX4_CFG0	0x4C00002B00	0x4C00002BFF	256 B
DEBUGSS_WRAP0_CORTEX5_CFG0	0x4C00002C00	0x4C00002CFF	256 B
DEBUGSS_WRAP0_CORTEX6_CFG0	0x4C00002D00	0x4C00002DFF	256 B

**Table 2-1. MAIN Domain Memory Map (continued)**

Region Name	Start Address	End Address	Size
DEBUGSS_WRAP0_CORTEX7_CFG0	0x4C00002E00	0x4C00002EFF	256 B
DEBUGSS_WRAP0_CORTEX8_CFG0	0x4C00002F00	0x4C00002FFF	256 B
DEBUGSS_WRAP0_RESV1_0	0x4C00003000	0x4C00003FFF	4 KB
DEBUGSS_WRAP0_RESV2_0	0x4C00004000	0x4C02003FFF	32 MB
DEBUGSS_WRAP0_ROM_TABLE_1_0	0x4C20000000	0x4C20000FFF	4 KB
DEBUGSS_WRAP0_CSCTI0	0x4C20001000	0x4C20001FFF	4 KB
DEBUGSS_WRAP0_DRM0	0x4C20002000	0x4C20002FFF	4 KB
DEBUGSS_WRAP0_RESV3_0	0x4C20003000	0x4C20003FFF	4 KB
DEBUGSS_WRAP0_CSTPIU0	0x4C20004000	0x4C20004FFF	4 KB
DEBUGSS_WRAP0_CTF0	0x4C20005000	0x4C20005FFF	4 KB
DEBUGSS_WRAP0_RESV4_0	0x4C20006000	0x4C21005FFF	16 MB
COMPUTE_CLUSTER0_CCROM	0x4C30000000	0x4C30000FFF	4 KB
DEBUGSS_WRAP0_EXT_APB0	0x4C30000000	0x4C3FFFFFFF	256 MB
COMPUTE_CLUSTER0_CTSET	0x4C30100000	0x4C30101FFF	8 KB
COMPUTE_CLUSTER0_CTI0	0x4C30102000	0x4C30102FFF	4 KB
COMPUTE_CLUSTER0_CTI1	0x4C30103000	0x4C30103FFF	4 KB
COMPUTE_CLUSTER0_CTI2	0x4C30104000	0x4C30104FFF	4 KB
COMPUTE_CLUSTER0_CTI3	0x4C30105000	0x4C30105FFF	4 KB
COMPUTE_CLUSTER0_CTI4	0x4C30106000	0x4C30106FFF	4 KB
COMPUTE_CLUSTER0_CTI5	0x4C30107000	0x4C30107FFF	4 KB
COMPUTE_CLUSTER0_CTI7	0x4C30109000	0x4C30109FFF	4 KB
COMPUTE_CLUSTER0_AGGR0	0x4C30140000	0x4C3017FFFF	256 KB
COMPUTE_CLUSTER0_AGGR1	0x4C30180000	0x4C301BFFFF	256 KB
COMPUTE_CLUSTER0_CPAC0	0x4C30400000	0x4C307FFFFF	4 MB
COMPUTE_CLUSTER0_CPAC4	0x4C31400000	0x4C317FFFFF	4 MB
CCDEBUGSS0_ROM	0x4C3C000000	0x4C3C000FFF	4 KB
CCDEBUGSS0_ATB_REPLICATOR_CFG_CXATBREPLICATOR_CFG	0x4C3C004000	0x4C3C004FFF	4 KB
CCDEBUGSS0_TBR_VBUSP_WRAP_TBR_CFG_TBR_CFG	0x4C3C005000	0x4C3C005FFF	4 KB
CCDEBUGSS0_ARM_CTI_0_CFG_CSCTI_CFG	0x4C3C006000	0x4C3C006FFF	4 KB
CCDEBUGSS0_ARM_CTI_1_CFG_CSCTI_CFG	0x4C3C008000	0x4C3C008FFF	4 KB
CCDEBUGSS0_ARM_CTI_2_CFG_CSCTI_CFG	0x4C3C009000	0x4C3C009FFF	4 KB
CCDEBUGSS0_ARM_CTI_3_CFG_CSCTI_CFG	0x4C3C00A000	0x4C3C00AFFF	4 KB
CCDEBUGSS0_ARM_CTI_4_CFG_CSCTI_CFG	0x4C3C00B000	0x4C3C00BFFF	4 KB
CCDEBUGSS0_ARM_CTI_5_CFG_CSCTI_CFG	0x4C3C00C000	0x4C3C00CFFF	4 KB
CCDEBUGSS0_ARM_CTI_6_CFG_CSCTI_CFG	0x4C3C00D000	0x4C3C00DFFF	4 KB
CCDEBUGSS0_ARM_CTI_7_CFG_CSCTI_CFG	0x4C3C00E000	0x4C3C00EFFF	4 KB
CCDEBUGSS0_ARM_CTI_8_CFG_CSCTI_CFG	0x4C3C00F000	0x4C3C00FFFF	4 KB
DEBUGSS0_ROM	0x4C3C010000	0x4C3C010FFF	4 KB
DEBUGSS0_CTSET2_WRAP_CFG_CTSET2_CFG	0x4C3C012000	0x4C3C013FFF	8 KB
DEBUGSS0_ATB_REPLICATOR_CFG_CXATBREPLICATOR_CFG	0x4C3C014000	0x4C3C014FFF	4 KB
DEBUGSS0_TBR_VBUSP_WRAP_TBR_CFG_TBR_CFG	0x4C3C015000	0x4C3C015FFF	4 KB
DEBUGSS0_ARM_CTI_0_CFG_CSCTI_CFG	0x4C3C016000	0x4C3C016FFF	4 KB
DEBUGSS0_ARM_CTI_1_CFG_CSCTI_CFG	0x4C3C018000	0x4C3C018FFF	4 KB
DEBUGSS0_ARM_CTI_2_CFG_CSCTI_CFG	0x4C3C019000	0x4C3C019FFF	4 KB
DEBUGSS0_ARM_CTI_3_CFG_CSCTI_CFG	0x4C3C01A000	0x4C3C01AFFF	4 KB

**Table 2-1. MAIN Domain Memory Map (continued)**

Region Name	Start Address	End Address	Size
DEBUGSS0_ARM_CTI_4_CFG_CSCTI_CFG	0x4C3C01B000	0x4C3C01BFFF	4 KB
DEBUGSS0_ARM_CTI_5_CFG_CSCTI_CFG	0x4C3C01C000	0x4C3C01CFFF	4 KB
DEBUGSS0_ARM_CTI_6_CFG_CSCTI_CFG	0x4C3C01D000	0x4C3C01DFFF	4 KB
DEBUGSS0_ARM_CTI_7_CFG_CSCTI_CFG	0x4C3C01E000	0x4C3C01EFFF	4 KB
DEBUGSS0_ARM_CTI_8_CFG_CSCTI_CFG	0x4C3C01F000	0x4C3C01FFFF	4 KB
DEBUGSS1_ROM	0x4C3C020000	0x4C3C020FFF	4 KB
DEBUGSS1_CTSET2_WRAP_CFG_CTSET2_CFG	0x4C3C022000	0x4C3C023FFF	8 KB
DEBUGSS1_ATB_REPLICATOR_CFG_CXATBREPLICATOR_CFG	0x4C3C024000	0x4C3C024FFF	4 KB
DEBUGSS1_TBR_VBUSP_WRAP_TBR_CFG_TBR_CFG	0x4C3C025000	0x4C3C025FFF	4 KB
DEBUGSS1_ARM_CTI_0_CFG_CSCTI_CFG	0x4C3C026000	0x4C3C026FFF	4 KB
DEBUGSS1_ARM_CTI_1_CFG_CSCTI_CFG	0x4C3C028000	0x4C3C028FFF	4 KB
DEBUGSS1_ARM_CTI_2_CFG_CSCTI_CFG	0x4C3C029000	0x4C3C029FFF	4 KB
DEBUGSS1_ARM_CTI_3_CFG_CSCTI_CFG	0x4C3C02A000	0x4C3C02AFFF	4 KB
DEBUGSS1_ARM_CTI_4_CFG_CSCTI_CFG	0x4C3C02B000	0x4C3C02BFFF	4 KB
DEBUGSS1_ARM_CTI_5_CFG_CSCTI_CFG	0x4C3C02C000	0x4C3C02CFFF	4 KB
DEBUGSS1_ARM_CTI_6_CFG_CSCTI_CFG	0x4C3C02D000	0x4C3C02DFFF	4 KB
DEBUGSS1_ARM_CTI_7_CFG_CSCTI_CFG	0x4C3C02E000	0x4C3C02EFFF	4 KB
DEBUGSS1_ARM_CTI_8_CFG_CSCTI_CFG	0x4C3C02F000	0x4C3C02FFFF	4 KB
C66DEBUGSS0_ROM	0x4C3C030000	0x4C3C030FFF	4 KB
C66DEBUGSS0_ATB_REPLICATOR_CFG_CXATBREPLICATOR_CFG	0x4C3C034000	0x4C3C034FFF	4 KB
C66DEBUGSS0_TBR_VBUSP_WRAP_TBR_CFG_TBR_CFG	0x4C3C035000	0x4C3C035FFF	4 KB
C66DEBUGSS0_ARM_CTI_0_CFG_CSCTI_CFG	0x4C3C036000	0x4C3C036FFF	4 KB
C66DEBUGSS0_ARM_CTI_1_CFG_CSCTI_CFG	0x4C3C038000	0x4C3C038FFF	4 KB
C66DEBUGSS0_ARM_CTI_2_CFG_CSCTI_CFG	0x4C3C039000	0x4C3C039FFF	4 KB
C66DEBUGSS0_ARM_CTI_3_CFG_CSCTI_CFG	0x4C3C03A000	0x4C3C03AFFF	4 KB
C66DEBUGSS0_ARM_CTI_4_CFG_CSCTI_CFG	0x4C3C03B000	0x4C3C03BFFF	4 KB
C66DEBUGSS0_ARM_CTI_5_CFG_CSCTI_CFG	0x4C3C03C000	0x4C3C03CFFF	4 KB
C66DEBUGSS0_ARM_CTI_6_CFG_CSCTI_CFG	0x4C3C03D000	0x4C3C03DFFF	4 KB
C66DEBUGSS0_ARM_CTI_7_CFG_CSCTI_CFG	0x4C3C03E000	0x4C3C03EFFF	4 KB
C66DEBUGSS0_ARM_CTI_8_CFG_CSCTI_CFG	0x4C3C03F000	0x4C3C03FFFF	4 KB
C66DEBUGSS1_ROM	0x4C3C040000	0x4C3C040FFF	4 KB
C66DEBUGSS1_ATB_REPLICATOR_CFG_CXATBREPLICATOR_CFG	0x4C3C044000	0x4C3C044FFF	4 KB
C66DEBUGSS1_TBR_VBUSP_WRAP_TBR_CFG_TBR_CFG	0x4C3C045000	0x4C3C045FFF	4 KB
C66DEBUGSS1_ARM_CTI_0_CFG_CSCTI_CFG	0x4C3C046000	0x4C3C046FFF	4 KB
C66DEBUGSS1_ARM_CTI_1_CFG_CSCTI_CFG	0x4C3C048000	0x4C3C048FFF	4 KB
C66DEBUGSS1_ARM_CTI_2_CFG_CSCTI_CFG	0x4C3C049000	0x4C3C049FFF	4 KB
C66DEBUGSS1_ARM_CTI_3_CFG_CSCTI_CFG	0x4C3C04A000	0x4C3C04AFFF	4 KB
C66DEBUGSS1_ARM_CTI_4_CFG_CSCTI_CFG	0x4C3C04B000	0x4C3C04BFFF	4 KB
C66DEBUGSS1_ARM_CTI_5_CFG_CSCTI_CFG	0x4C3C04C000	0x4C3C04CFFF	4 KB
C66DEBUGSS1_ARM_CTI_6_CFG_CSCTI_CFG	0x4C3C04D000	0x4C3C04DFFF	4 KB
C66DEBUGSS1_ARM_CTI_7_CFG_CSCTI_CFG	0x4C3C04E000	0x4C3C04EFFF	4 KB
C66DEBUGSS1_ARM_CTI_8_CFG_CSCTI_CFG	0x4C3C04F000	0x4C3C04FFFF	4 KB
STM0_CXSTM	0x4C3D200000	0x4C3D200FFF	4 KB
STM0_CTI_CSCTI	0x4C3D201000	0x4C3D201FFF	4 KB
DEBGSUSPENDRTR0_INTR_ROUTER_CFG	0x4C3D300000	0x4C3D300FFF	4 KB

**Table 2-1. MAIN Domain Memory Map (continued)**

Region Name	Start Address	End Address	Size
CPT2_AGGR0_MMR	0x4C3E100000	0x4C3E1000FF	256 B
CPT2_AGGR0_MEM0	0x4C3E120000	0x4C3E120FFF	4 KB
CPT2_AGGR0_MEM1	0x4C3E121000	0x4C3E121FFF	4 KB
CPT2_AGGR0_MEM2	0x4C3E122000	0x4C3E122FFF	4 KB
CPT2_AGGR0_MEM3	0x4C3E123000	0x4C3E123FFF	4 KB
CPT2_AGGR0_MEM4	0x4C3E124000	0x4C3E124FFF	4 KB
CPT2_AGGR0_MEM5	0x4C3E125000	0x4C3E125FFF	4 KB
CPT2_AGGR0_MEM6	0x4C3E126000	0x4C3E126FFF	4 KB
CPT2_AGGR0_MEM7	0x4C3E127000	0x4C3E127FFF	4 KB
CPT2_AGGR0_MEM8	0x4C3E128000	0x4C3E128FFF	4 KB
CPT2_AGGR0_MEM9	0x4C3E129000	0x4C3E129FFF	4 KB
CPT2_AGGR0_MEM10	0x4C3E12A000	0x4C3E12AFFF	4 KB
CPT2_AGGR0_MEM11	0x4C3E12B000	0x4C3E12BFFF	4 KB
CPT2_AGGR0_MEM12	0x4C3E12C000	0x4C3E12CFFF	4 KB
CPT2_AGGR0_MEM13	0x4C3E12D000	0x4C3E12DFFF	4 KB
CPT2_AGGR0_MEM14	0x4C3E12E000	0x4C3E12EFFF	4 KB
CPT2_AGGR0_MEM15	0x4C3E12F000	0x4C3E12FFFF	4 KB
CPT2_AGGR0_MEM16	0x4C3E130000	0x4C3E130FFF	4 KB
CPT2_AGGR0_MEM17	0x4C3E131000	0x4C3E131FFF	4 KB
CPT2_AGGR0_MEM18	0x4C3E132000	0x4C3E132FFF	4 KB
CPT2_AGGR0_MEM19	0x4C3E133000	0x4C3E133FFF	4 KB
CPT2_AGGR0_MEM20	0x4C3E134000	0x4C3E134FFF	4 KB
CPT2_AGGR0_MEM21	0x4C3E135000	0x4C3E135FFF	4 KB
CPT2_AGGR0_MEM22	0x4C3E136000	0x4C3E136FFF	4 KB
CPT2_AGGR0_MEM23	0x4C3E137000	0x4C3E137FFF	4 KB
CPT2_AGGR0_MEM24	0x4C3E138000	0x4C3E138FFF	4 KB
CPT2_AGGR0_MEM25	0x4C3E139000	0x4C3E139FFF	4 KB
CPT2_AGGR0_MEM26	0x4C3E13A000	0x4C3E13AFFF	4 KB
CPT2_AGGR0_MEM27	0x4C3E13B000	0x4C3E13BFFF	4 KB
CPT2_AGGR0_MEM28	0x4C3E13C000	0x4C3E13CFFF	4 KB
CPT2_AGGR0_MEM29	0x4C3E13D000	0x4C3E13DFFF	4 KB
CPT2_AGGR0_MEM30	0x4C3E13E000	0x4C3E13EFFF	4 KB
CPT2_AGGR0_MEM31	0x4C3E13F000	0x4C3E13FFFF	4 KB
CPT2_AGGR1_MMR	0x4C3E140000	0x4C3E1400FF	256 B
CPT2_AGGR1_STP2ATB_CFG	0x4C3E140100	0x4C3E1401FF	256 B
CPT2_AGGR1_MEM0	0x4C3E160000	0x4C3E160FFF	4 KB
CPT2_AGGR1_MEM1	0x4C3E161000	0x4C3E161FFF	4 KB
CPT2_AGGR1_MEM2	0x4C3E162000	0x4C3E162FFF	4 KB
CPT2_AGGR1_MEM3	0x4C3E163000	0x4C3E163FFF	4 KB
CPT2_AGGR1_MEM4	0x4C3E164000	0x4C3E164FFF	4 KB
CPT2_AGGR1_MEM5	0x4C3E165000	0x4C3E165FFF	4 KB
CPT2_AGGR1_MEM6	0x4C3E166000	0x4C3E166FFF	4 KB
CPT2_AGGR1_MEM7	0x4C3E167000	0x4C3E167FFF	4 KB
CPT2_AGGR1_MEM8	0x4C3E168000	0x4C3E168FFF	4 KB
CPT2_AGGR1_MEM9	0x4C3E169000	0x4C3E169FFF	4 KB



**Table 2-1. MAIN Domain Memory Map (continued)**

Region Name	Start Address	End Address	Size
CPT2_AGGR1_MEM10	0x4C3E16A000	0x4C3E16AFFF	4 KB
CPT2_AGGR1_MEM11	0x4C3E16B000	0x4C3E16BFFF	4 KB
CPT2_AGGR1_MEM12	0x4C3E16C000	0x4C3E16CFFF	4 KB
CPT2_AGGR1_MEM13	0x4C3E16D000	0x4C3E16DFFF	4 KB
CPT2_AGGR1_MEM14	0x4C3E16E000	0x4C3E16EFFF	4 KB
CPT2_AGGR1_MEM15	0x4C3E16F000	0x4C3E16FFFF	4 KB
CPT2_AGGR1_MEM16	0x4C3E170000	0x4C3E170FFF	4 KB
CPT2_AGGR1_MEM17	0x4C3E171000	0x4C3E171FFF	4 KB
CPT2_AGGR1_MEM18	0x4C3E172000	0x4C3E172FFF	4 KB
CPT2_AGGR1_MEM19	0x4C3E173000	0x4C3E173FFF	4 KB
CPT2_AGGR1_MEM20	0x4C3E174000	0x4C3E174FFF	4 KB
CPT2_AGGR1_MEM21	0x4C3E175000	0x4C3E175FFF	4 KB
CPT2_AGGR1_MEM22	0x4C3E176000	0x4C3E176FFF	4 KB
CPT2_AGGR1_MEM23	0x4C3E177000	0x4C3E177FFF	4 KB
CPT2_AGGR1_MEM24	0x4C3E178000	0x4C3E178FFF	4 KB
CPT2_AGGR1_MEM25	0x4C3E179000	0x4C3E179FFF	4 KB
CPT2_AGGR1_MEM26	0x4C3E17A000	0x4C3E17AFFF	4 KB
CPT2_AGGR1_MEM27	0x4C3E17B000	0x4C3E17BFFF	4 KB
CPT2_AGGR1_MEM28	0x4C3E17C000	0x4C3E17CFFF	4 KB
CPT2_AGGR1_MEM29	0x4C3E17D000	0x4C3E17DFFF	4 KB
CPT2_AGGR1_MEM30	0x4C3E17E000	0x4C3E17EFFF	4 KB
CPT2_AGGR1_MEM31	0x4C3E17F000	0x4C3E17FFFF	4 KB
CPT2_AGGR2_MMR	0x4C3E180000	0x4C3E1800FF	256 B
CPT2_AGGR2_STP2ATB_CFG	0x4C3E180100	0x4C3E1801FF	256 B
CPT2_AGGR2_MEM0	0x4C3E1A0000	0x4C3E1A0FFF	4 KB
CPT2_AGGR2_MEM1	0x4C3E1A1000	0x4C3E1A1FFF	4 KB
CPT2_AGGR2_MEM2	0x4C3E1A2000	0x4C3E1A2FFF	4 KB
CPT2_AGGR2_MEM3	0x4C3E1A3000	0x4C3E1A3FFF	4 KB
CPT2_AGGR2_MEM4	0x4C3E1A4000	0x4C3E1A4FFF	4 KB
CPT2_AGGR2_MEM5	0x4C3E1A5000	0x4C3E1A5FFF	4 KB
CPT2_AGGR2_MEM6	0x4C3E1A6000	0x4C3E1A6FFF	4 KB
CPT2_AGGR2_MEM7	0x4C3E1A7000	0x4C3E1A7FFF	4 KB
CPT2_AGGR2_MEM8	0x4C3E1A8000	0x4C3E1A8FFF	4 KB
CPT2_AGGR2_MEM9	0x4C3E1A9000	0x4C3E1A9FFF	4 KB
CPT2_AGGR2_MEM10	0x4C3E1AA000	0x4C3E1AAFFF	4 KB
CPT2_AGGR2_MEM11	0x4C3E1AB000	0x4C3E1ABFFF	4 KB
CPT2_AGGR2_MEM12	0x4C3E1AC000	0x4C3E1ACFFF	4 KB
CPT2_AGGR2_MEM13	0x4C3E1AD000	0x4C3E1ADFFF	4 KB
CPT2_AGGR2_MEM14	0x4C3E1AE000	0x4C3E1AEFFF	4 KB
CPT2_AGGR2_MEM15	0x4C3E1AF000	0x4C3E1AFFFF	4 KB
CPT2_AGGR2_MEM16	0x4C3E1B0000	0x4C3E1B0FFF	4 KB
CPT2_AGGR2_MEM17	0x4C3E1B1000	0x4C3E1B1FFF	4 KB
CPT2_AGGR2_MEM18	0x4C3E1B2000	0x4C3E1B2FFF	4 KB
CPT2_AGGR2_MEM19	0x4C3E1B3000	0x4C3E1B3FFF	4 KB
CPT2_AGGR2_MEM20	0x4C3E1B4000	0x4C3E1B4FFF	4 KB

**Table 2-1. MAIN Domain Memory Map (continued)**

Region Name	Start Address	End Address	Size
CPT2_AGGR2_MEM21	0x4C3E1B5000	0x4C3E1B5FFF	4 KB
CPT2_AGGR2_MEM22	0x4C3E1B6000	0x4C3E1B6FFF	4 KB
CPT2_AGGR2_MEM23	0x4C3E1B7000	0x4C3E1B7FFF	4 KB
CPT2_AGGR2_MEM24	0x4C3E1B8000	0x4C3E1B8FFF	4 KB
CPT2_AGGR2_MEM25	0x4C3E1B9000	0x4C3E1B9FFF	4 KB
CPT2_AGGR2_MEM26	0x4C3E1BA000	0x4C3E1BAFFF	4 KB
CPT2_AGGR2_MEM27	0x4C3E1BB000	0x4C3E1BBFFF	4 KB
CPT2_AGGR2_MEM28	0x4C3E1BC000	0x4C3E1BCFFF	4 KB
CPT2_AGGR2_MEM29	0x4C3E1BD000	0x4C3E1BDFFF	4 KB
CPT2_AGGR2_MEM30	0x4C3E1BE000	0x4C3E1BEFFF	4 KB
CPT2_AGGR2_MEM31	0x4C3E1BF000	0x4C3E1BFFFF	4 KB
C66SS0_VBUSP_CFG_ADTF	0x4C3EE00000	0x4C3EE003FF	1 KB
C66SS1_VBUSP_CFG_ADTF	0x4C3EE01000	0x4C3EE013FF	1 KB
DEBUGSS_WRAP0_ROM_TABLE_0_1	0x4C40000000	0x4C40000FFF	4 KB
DEBUGSS_WRAP0_RESV0_1	0x4C40001000	0x4C40001FFF	4 KB
DEBUGSS_WRAP0_CFGAP1	0x4C40002000	0x4C400020FF	256 B
DEBUGSS_WRAP0_APBAP1	0x4C40002100	0x4C400021FF	256 B
DEBUGSS_WRAP0_AXIAP1	0x4C40002200	0x4C400022FF	256 B
DEBUGSS_WRAP0_PWRAP1	0x4C40002300	0x4C400023FF	256 B
DEBUGSS_WRAP0_PVIEW1	0x4C40002400	0x4C400024FF	256 B
DEBUGSS_WRAP0_JTAGAP1	0x4C40002500	0x4C400025FF	256 B
DEBUGSS_WRAP0_SECAP1	0x4C40002600	0x4C400026FF	256 B
DEBUGSS_WRAP0_CORTEX0_CFG1	0x4C40002700	0x4C400027FF	256 B
DEBUGSS_WRAP0_CORTEX1_CFG1	0x4C40002800	0x4C400028FF	256 B
DEBUGSS_WRAP0_CORTEX2_CFG1	0x4C40002900	0x4C400029FF	256 B
DEBUGSS_WRAP0_CORTEX3_CFG1	0x4C40002A00	0x4C40002AFF	256 B
DEBUGSS_WRAP0_CORTEX4_CFG1	0x4C40002B00	0x4C40002BFF	256 B
DEBUGSS_WRAP0_CORTEX5_CFG1	0x4C40002C00	0x4C40002CFF	256 B
DEBUGSS_WRAP0_CORTEX6_CFG1	0x4C40002D00	0x4C40002DFF	256 B
DEBUGSS_WRAP0_CORTEX7_CFG1	0x4C40002E00	0x4C40002EFF	256 B
DEBUGSS_WRAP0_CORTEX8_CFG1	0x4C40002F00	0x4C40002FFF	256 B
DEBUGSS_WRAP0_RESV1_1	0x4C40003000	0x4C40003FFF	4 KB
DEBUGSS_WRAP0_RESV2_1	0x4C40004000	0x4C42003FFF	32 MB
DEBUGSS_WRAP0_ROM_TABLE_1_1	0x4C60000000	0x4C60000FFF	4 KB
DEBUGSS_WRAP0_CSCTI1	0x4C60001000	0x4C60001FFF	4 KB
DEBUGSS_WRAP0_DRM1	0x4C60002000	0x4C60002FFF	4 KB
DEBUGSS_WRAP0_RESV3_1	0x4C60003000	0x4C60003FFF	4 KB
DEBUGSS_WRAP0_CSTPIU1	0x4C60004000	0x4C60004FFF	4 KB
DEBUGSS_WRAP0_CTF1	0x4C60005000	0x4C60005FFF	4 KB
DEBUGSS_WRAP0_RESV4_1	0x4C60006000	0x4C61005FFF	16 MB
DEBUGSS_WRAP0_EXT_APB1	0x4C70000000	0x4C7FFFFFFF	256 MB
COMPUTE_CLUSTER0_MSMC_PBIST	0x4D10000000	0x4D1000FFFF	64 KB
COMPUTE_CLUSTER0_A72SS0_PBIST0	0x4D10010000	0x4D1001FFFF	64 KB
COMPUTE_CLUSTER0_C71SS0_PBIST	0x4D10050000	0x4D100503FF	1 KB
COMPUTE_CLUSTER0_MSMC_ECC_AGGR0	0x4D20000000	0x4D200003FF	1 KB

**Table 2-1. MAIN Domain Memory Map (continued)**

Region Name	Start Address	End Address	Size
COMPUTE_CLUSTER0_MSMC_ECC_AGGR1	0x4D20000400	0x4D200007FF	1 KB
COMPUTE_CLUSTER0_MSMC_ECC_AGGR2	0x4D20000800	0x4D20000BFF	1 KB
COMPUTE_CLUSTER0_A72SS0_COMMON_ECC_AGGR	0x4D20010000	0x4D200103FF	1 KB
COMPUTE_CLUSTER0_A72SS0_CORE0_ECC_AGGR	0x4D20010400	0x4D200107FF	1 KB
COMPUTE_CLUSTER0_A72SS0_CORE1_ECC_AGGR	0x4D20010800	0x4D20010BFF	1 KB
COMPUTE_CLUSTER0_C71SS0_ECC_AGGR	0x4D20050000	0x4D200503FF	1 KB
COMPUTE_CLUSTER0_ECC_AGGR_CTL	0x4D200B0000	0x4D200B03FF	1 KB
COMPUTE_CLUSTER0_ECC_AGGR_VBUS	0x4D200B0400	0x4D200B07FF	1 KB
COMPUTE_CLUSTER0_ECC_AGGR_CFG	0x4D200B0800	0x4D200B0BFF	1 KB
COMPUTE_CLUSTER0_ECC_AGGR	0x4D200C0000	0x4D200C03FF	1 KB
COMPUTE_CLUSTER0_CC_REGS	0x4D21000000	0x4D2100FFFF	64 KB
C66SS0_VBUSP_CFG_PBISTCFG	0x4D80000000	0x4D8000FFFF	64 KB
C66SS0_C66_SDMA_L2SRAM_0	0x4D80800000	0x4D8083FFFF	256 KB
C66SS0_C66_SDMA_L1DSRAM	0x4D80F00000	0x4D80F07FFF	32 KB
C66SS1_VBUSP_CFG_PBISTCFG	0x4D81000000	0x4D8100FFFF	64 KB
C66SS1_C66_SDMA_L2SRAM_0	0x4D81800000	0x4D8183FFFF	256 KB
C66SS1_C66_SDMA_L1DSRAM	0x4D81F00000	0x4D81F07FFF	32 KB
R5FSS0_CORE0_ICACHE	0x4E00000000	0x4E007FFFFF	8 MB
R5FSS0_CORE0_DCACHE	0x4E00800000	0x4E00FFFFFF	8 MB
R5FSS0_CORE1_ICACHE	0x4E01000000	0x4E017FFFFF	8 MB
R5FSS0_CORE1_DCACHE	0x4E01800000	0x4E01FFFFFF	8 MB
R5FSS1_CORE0_ICACHE	0x4E10000000	0x4E107FFFFF	8 MB
R5FSS1_CORE0_DCACHE	0x4E10800000	0x4E10FFFFFF	8 MB
R5FSS1_CORE1_ICACHE	0x4E11000000	0x4E117FFFFF	8 MB
R5FSS1_CORE1_DCACHE	0x4E11800000	0x4E11FFFFFF	8 MB
VPAC0_VPAC_TOP_PAC_BASE_MEM_SLV_CBASS_STRIPE_MSRAM_SLV	0x4F00000000	0x4F0007FFFF	512 KB
DMPAC0_DMPAC_TOP_DOE_INFRA_DMPAC_BASE_MEM_SLV_CBASS_STRIPE_MSRAM_SLV	0x4F01000000	0x4F0107FFFF	512 KB

- (1) These regions are used when R5FSS0 and R5FSS1 work in split mode. For more information about split and lockstep modes, see [Section 6.3.3.2 MCU Cortex-R5F Core](#). For more information about ATCM and BTCM, see [Section 6.3.3.2 Tightly-Coupled Memories \(TCMs\)](#).

**Table 2-2. NAVSS0 Alias Memory Map**

Region Name	Start Address	End Address	Size
NAVSS0_ALIAS64K_MCRC0_S_CFG_MCRC64	0x4A1F700000	0x4A1F70FFFF	64 KB
NAVSS0_ALIAS64K_PSILCFG0_CFG_PROXY	0x4A1F780000	0x4A1F781FFF	8 KB
NAVSS0_ALIAS64K_MAILBOX0_CFG_REGS0	0x4A1F800000	0x4A1F801FFF	8 KB
NAVSS0_ALIAS64K_MAILBOX0_CFG_REGS1	0x4A1F810000	0x4A1F811FFF	8 KB
NAVSS0_ALIAS64K_MAILBOX0_CFG_REGS2	0x4A1F820000	0x4A1F821FFF	8 KB
NAVSS0_ALIAS64K_MAILBOX0_CFG_REGS3	0x4A1F830000	0x4A1F831FFF	8 KB
NAVSS0_ALIAS64K_MAILBOX0_CFG_REGS4	0x4A1F840000	0x4A1F841FFF	8 KB
NAVSS0_ALIAS64K_MAILBOX0_CFG_REGS5	0x4A1F850000	0x4A1F851FFF	8 KB
NAVSS0_ALIAS64K_MAILBOX0_CFG_REGS6	0x4A1F860000	0x4A1F861FFF	8 KB
NAVSS0_ALIAS64K_MAILBOX0_CFG_REGS7	0x4A1F870000	0x4A1F871FFF	8 KB
NAVSS0_ALIAS64K_MAILBOX0_CFG_REGS8	0x4A1F880000	0x4A1F881FFF	8 KB
NAVSS0_ALIAS64K_MAILBOX0_CFG_REGS9	0x4A1F890000	0x4A1F891FFF	8 KB
NAVSS0_ALIAS64K_MAILBOX0_CFG_REGS10	0x4A1F8A0000	0x4A1F8A1FFF	8 KB

**Table 2-2. NAVSS0 Alias Memory Map (continued)**

Region Name	Start Address	End Address	Size
NAVSS0_ALIAS64K_MAILBOX0_CFG_REGS11	0x4A1F8B0000	0x4A1F8B1FFF	8 KB
NAVSS0_ALIAS64K_RINGACC0_CFG_MON	0x4A20000000	0x4A201FFFFFFF	2 MB
NAVSS0_ALIAS64K_TIMERMGR0_CFG_TIMERS	0x4A22000000	0x4A223FFFFFFF	4 MB
NAVSS0_ALIAS64K_TIMERMGR1_CFG_TIMERS	0x4A22400000	0x4A227FFFFFFF	4 MB
NAVSS0_ALIAS64K_SEC_PROXY0_CFG_RT	0x4A24000000	0x4A24FFFFFFF	16 MB
NAVSS0_ALIAS64K_SEC_PROXY0_CFG_SCFG	0x4A28000000	0x4A28FFFFFFF	16 MB
NAVSS0_ALIAS64K_SEC_PROXY0_SRC_TARGET_DATA	0x4A2C000000	0x4A2CFFFFFFF	16 MB
NAVSS0_ALIAS64K_PROXY0_SRC_TARGET0_DATA	0x4A30000000	0x4A303FFFFFFF	4 MB
NAVSS0_ALIAS64K_PROXY0_CFG_BUF_CFG	0x4A34000000	0x4A343FFFFFFF	4 MB
NAVSS0_ALIAS64K_UDMASS_INTA0_CFG_GCNTRTI	0x4A38000000	0x4A39FFFFFFF	32 MB
NAVSS0_ALIAS64K_MODSS_INTA0_CFG_INTR	0x4A3C000000	0x4A3C3FFFFFFF	4 MB
NAVSS0_ALIAS64K_MODSS_INTA1_CFG_INTR	0x4A3C400000	0x4A3C7FFFFFFF	4 MB
NAVSS0_ALIAS64K_UDMASS_INTA0_CFG_INTR	0x4A3D000000	0x4A3DFFFFFFF	16 MB
NAVSS0_ALIAS64K_UDMAP0_CFG_RCHANRT	0x4A40000000	0x4A40FFFFFFF	16 MB
NAVSS0_ALIAS64K_UDMAP0_CFG_TCHANRT	0x4A50000000	0x4A51FFFFFFF	32 MB
NAVSS0_ALIAS64K_PAT0_CFG_SCRATCH	0x4A62000000	0x4A620FFFFFFF	1 MB
NAVSS0_ALIAS64K_PAT1_CFG_SCRATCH	0x4A62100000	0x4A621FFFFFFF	1 MB
NAVSS0_ALIAS64K_PAT2_CFG_SCRATCH	0x4A62200000	0x4A622FFFFFFF	1 MB
NAVSS0_ALIAS64K_PAT3_CFG_SCRATCH	0x4A62300000	0x4A6231FFFF	128 KB
NAVSS0_ALIAS64K_PAT4_CFG_SCRATCH	0x4A62400000	0x4A6241FFFF	128 KB
NAVSS0_ALIAS64K_PAT0_CFG_TABLE	0x4A64000000	0x4A643FFFFFFF	4 MB
NAVSS0_ALIAS64K_PAT1_CFG_TABLE	0x4A64400000	0x4A647FFFFFFF	4 MB
NAVSS0_ALIAS64K_PAT2_CFG_TABLE	0x4A64800000	0x4A64BFFFFFFF	4 MB
NAVSS0_ALIAS64K_PAT3_CFG_TABLE	0x4A64C00000	0x4A64C7FFFF	512 KB
NAVSS0_ALIAS64K_PAT4_CFG_TABLE	0x4A65000000	0x4A6507FFFF	512 KB
NAVSS0_ALIAS64K_TCU_CFG	0x4A66000000	0x4A67FFFFFFF	32 MB
NAVSS0_ALIAS64K_RINGACC0_SRC_FIFOS	0x4A80000000	0x4A83FFFFFFF	64 MB
NAVSS0_ALIAS64K_RINGACC0_CFG_RT	0x4AC0000000	0x4AC3FFFFFFF	64 MB
NAVSS0_VIRT_ALIAS_0_MODSS_INTA0_CFG	0x4B00800000	0x4B0080001F	32 B
NAVSS0_VIRT_ALIAS_0_MODSS_INTA1_CFG	0x4B00801000	0x4B0080101F	32 B
NAVSS0_VIRT_ALIAS_0_UDMASS_INTA0_CFG	0x4B00802000	0x4B0080201F	32 B
NAVSS0_VIRT_ALIAS_0_MODSS_INTA0_CFG_IMAP	0x4B00900000	0x4B00907FFF	32 KB
NAVSS0_VIRT_ALIAS_0_MODSS_INTA1_CFG_IMAP	0x4B00908000	0x4B0090FFFF	32 KB
NAVSS0_VIRT_ALIAS_0_UDMASS_INTA0_CFG_IMAP	0x4B00940000	0x4B0097FFFF	256 KB
NAVSS0_VIRT_ALIAS_0_NAV_DDR0_VIRTID_CFG_MMRS	0x4B00A02000	0x4B00A020FF	256 B
NAVSS0_VIRT_ALIAS_0_NAV_DDR1_VIRTID_CFG_MMRS	0x4B00A03000	0x4B00A030FF	256 B
NAVSS0_VIRT_ALIAS_0_UDMAP0_CFG_TCHAN	0x4B00B00000	0x4B00B1FFFF	128 KB
NAVSS0_VIRT_ALIAS_0_UDMAP0_CFG_RCHAN	0x4B00C00000	0x4B00C0FFFF	64 KB
NAVSS0_VIRT_ALIAS_0_UDMAP0_CFG_RFLOW	0x4B00D00000	0x4B00D07FFF	32 KB
NAVSS0_VIRT_ALIAS_0_SPINLOCK0_CFG	0x4B00E00000	0x4B00E07FFF	32 KB
NAVSS0_VIRT_ALIAS_0_TIMERMGR0_CFG	0x4B00E80000	0x4B00E801FF	512 B
NAVSS0_VIRT_ALIAS_0_TIMERMGR1_CFG	0x4B00E81000	0x4B00E811FF	512 B
NAVSS0_VIRT_ALIAS_0_TIMERMGR0_CFG_OES	0x4B00F00000	0x4B00F00FFF	4 KB
NAVSS0_VIRT_ALIAS_0_TIMERMGR1_CFG_OES	0x4B00F01000	0x4B00F01FFF	4 KB
NAVSS0_VIRT_ALIAS_0_ECCAGGR0	0x4B01000000	0x4B010003FF	1 KB

**Table 2-2. NAVSS0 Alias Memory Map (continued)**

Region Name	Start Address	End Address	Size
NAVSS0_VIRT_ALIAS_0_UDMASS_ECCAGGR_CFG	0x4B01001000	0x4B010013FF	1 KB
NAVSS0_VIRT_ALIAS_0_VIRTSS_ECCAGGR_CFG	0x4B01002000	0x4B010023FF	1 KB
NAVSS0_VIRT_ALIAS_0_PAT0_CFG_MMRS	0x4B01010000	0x4B010100FF	256 B
NAVSS0_VIRT_ALIAS_0_PAT1_CFG_MMRS	0x4B01011000	0x4B010110FF	256 B
NAVSS0_VIRT_ALIAS_0_PAT2_CFG_MMRS	0x4B01012000	0x4B010120FF	256 B
NAVSS0_VIRT_ALIAS_0_PAT3_CFG_MMRS	0x4B01013000	0x4B010130FF	256 B
NAVSS0_VIRT_ALIAS_0_PAT4_CFG_MMRS	0x4B01014000	0x4B010140FF	256 B
NAVSS0_VIRT_ALIAS_0_UDMASS_INTA0_CFG_GCNTCFG	0x4B01040000	0x4B01043FFF	16 KB
NAVSS0_VIRT_ALIAS_0_RINGACC0_CFG	0x4B01080000	0x4B010BFFFF	256 KB
NAVSS0_VIRT_ALIAS_0_REGS0_CFG_MMRS	0x4B010C0000	0x4B010C00FF	256 B
NAVSS0_VIRT_ALIAS_0_CPTS0_S_VBUSP_CPTS_VBUSP	0x4B010D0000	0x4B010D03FF	1 KB
NAVSS0_VIRT_ALIAS_0_INTR0_CFG_INTR_ROUTER_CFG	0x4B010E0000	0x4B010E3FFF	16 KB
NAVSS0_VIRT_ALIAS_0_UDMASS_INTA0_CFG_L2G	0x4B01100000	0x4B01100FFF	4 KB
NAVSS0_VIRT_ALIAS_0_UDMASS_INTA0_CFG_MCAST	0x4B01110000	0x4B01113FFF	16 KB
NAVSS0_VIRT_ALIAS_0_PROXY0_CFG_BUF_CFG_GCFG	0x4B01120000	0x4B011200FF	256 B
NAVSS0_VIRT_ALIAS_0_PROXY0_CFG_BUFRAM_SLV_RAM	0x4B01130000	0x4B01133FFF	16 KB
NAVSS0_VIRT_ALIAS_0_SEC_PROXY0_CFG_MMRS	0x4B01140000	0x4B011400FF	256 B
NAVSS0_VIRT_ALIAS_0_UDMAP0_CFG_GCFG	0x4B01150000	0x4B011500FF	256 B
NAVSS0_VIRT_ALIAS_0_RINGACC0_CFG_GCFG	0x4B01160000	0x4B011603FF	1 KB
NAVSS0_VIRT_ALIAS_0_PSILSS_LOCAL_CFG_MMRS	0x4B01170000	0x4B01170FFF	4 KB
NAVSS0_VIRT_ALIAS_0_PSILSS_GLOBAL_CFG_MMRS	0x4B01180000	0x4B01180FFF	4 KB
NAVSS0_VIRT_ALIAS_0_PSILSS_TR_CFG_MMRS	0x4B01190000	0x4B01190FFF	4 KB
NAVSS0_VIRT_ALIAS_0_MCRC0_S_CFG_MCRC64	0x4B01F70000	0x4B01F70FFF	4 KB
NAVSS0_VIRT_ALIAS_0_PSILCFG0_CFG_PROXY	0x4B01F78000	0x4B01F781FF	512 B
NAVSS0_VIRT_ALIAS_0_MAILBOX0_CFG_REGS0	0x4B01F80000	0x4B01F801FF	512 B
NAVSS0_VIRT_ALIAS_0_MAILBOX0_CFG_REGS1	0x4B01F81000	0x4B01F811FF	512 B
NAVSS0_VIRT_ALIAS_0_MAILBOX0_CFG_REGS2	0x4B01F82000	0x4B01F821FF	512 B
NAVSS0_VIRT_ALIAS_0_MAILBOX0_CFG_REGS3	0x4B01F83000	0x4B01F831FF	512 B
NAVSS0_VIRT_ALIAS_0_MAILBOX0_CFG_REGS4	0x4B01F84000	0x4B01F841FF	512 B
NAVSS0_VIRT_ALIAS_0_MAILBOX0_CFG_REGS5	0x4B01F85000	0x4B01F851FF	512 B
NAVSS0_VIRT_ALIAS_0_MAILBOX0_CFG_REGS6	0x4B01F86000	0x4B01F861FF	512 B
NAVSS0_VIRT_ALIAS_0_MAILBOX0_CFG_REGS7	0x4B01F87000	0x4B01F871FF	512 B
NAVSS0_VIRT_ALIAS_0_MAILBOX0_CFG_REGS8	0x4B01F88000	0x4B01F881FF	512 B
NAVSS0_VIRT_ALIAS_0_MAILBOX0_CFG_REGS9	0x4B01F89000	0x4B01F891FF	512 B
NAVSS0_VIRT_ALIAS_0_MAILBOX0_CFG_REGS10	0x4B01F8A000	0x4B01F8A1FF	512 B
NAVSS0_VIRT_ALIAS_0_MAILBOX0_CFG_REGS11	0x4B01F8B000	0x4B01F8B1FF	512 B
NAVSS0_VIRT_ALIAS_0_RINGACC0_CFG_MON	0x4B02000000	0x4B0201FFFF	128 KB
NAVSS0_VIRT_ALIAS_0_TIMERMGR0_CFG_TIMERS	0x4B02200000	0x4B0223FFFF	256 KB
NAVSS0_VIRT_ALIAS_0_TIMERMGR1_CFG_TIMERS	0x4B02240000	0x4B0227FFFF	256 KB
NAVSS0_VIRT_ALIAS_0_SEC_PROXY0_CFG_RT	0x4B02400000	0x4B024FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_0_SEC_PROXY0_CFG_SCFG	0x4B02800000	0x4B028FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_0_SEC_PROXY0_SRC_TARGET_DATA	0x4B02C00000	0x4B02CFFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_0_PROXY0_SRC_TARGET0_DATA	0x4B03000000	0x4B0303FFFF	256 KB
NAVSS0_VIRT_ALIAS_0_PROXY0_CFG_BUF_CFG	0x4B03400000	0x4B0343FFFF	256 KB
NAVSS0_VIRT_ALIAS_0_UDMASS_INTA0_CFG_GCNTRTI	0x4B03800000	0x4B039FFFFFFF	2 MB



**Table 2-2. NAVSS0 Alias Memory Map (continued)**

Region Name	Start Address	End Address	Size
NAVSS0_VIRT_ALIAS_0_MODSS_INTA0_CFG_INTR	0x4B03C00000	0x4B03C3FFFF	256 KB
NAVSS0_VIRT_ALIAS_0_MODSS_INTA1_CFG_INTR	0x4B03C40000	0x4B03C7FFFF	256 KB
NAVSS0_VIRT_ALIAS_0_UDMASS_INTA0_CFG_INTR	0x4B03D00000	0x4B03DFFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_0_UDMAP0_CFG_RCHANRT	0x4B04000000	0x4B040FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_0_UDMAP0_CFG_TCHANRT	0x4B05000000	0x4B051FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_0_PAT0_CFG_SCRATCH	0x4B06200000	0x4B0620FFFF	64 KB
NAVSS0_VIRT_ALIAS_0_PAT1_CFG_SCRATCH	0x4B06210000	0x4B0621FFFF	64 KB
NAVSS0_VIRT_ALIAS_0_PAT2_CFG_SCRATCH	0x4B06220000	0x4B0622FFFF	64 KB
NAVSS0_VIRT_ALIAS_0_PAT3_CFG_SCRATCH	0x4B06230000	0x4B06231FFF	8 KB
NAVSS0_VIRT_ALIAS_0_PAT4_CFG_SCRATCH	0x4B06240000	0x4B06241FFF	8 KB
NAVSS0_VIRT_ALIAS_0_PAT0_CFG_TABLE	0x4B06400000	0x4B0643FFFF	256 KB
NAVSS0_VIRT_ALIAS_0_PAT1_CFG_TABLE	0x4B06440000	0x4B0647FFFF	256 KB
NAVSS0_VIRT_ALIAS_0_PAT2_CFG_TABLE	0x4B06480000	0x4B064BFFFF	256 KB
NAVSS0_VIRT_ALIAS_0_PAT3_CFG_TABLE	0x4B064C0000	0x4B064C7FFF	32 KB
NAVSS0_VIRT_ALIAS_0_PAT4_CFG_TABLE	0x4B06500000	0x4B06507FFF	32 KB
NAVSS0_VIRT_ALIAS_0_TCU_CFG	0x4B06600000	0x4B067FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_0_RINGACC0_SRC_FIFOS	0x4B08000000	0x4B083FFFFFFF	4 MB
NAVSS0_VIRT_ALIAS_0_RINGACC0_CFG_RT	0x4B0C000000	0x4B0C3FFFFFFF	4 MB
NAVSS0_VIRT_ALIAS_1_MODSS_INTA0_CFG	0x4B10800000	0x4B1080001F	32 B
NAVSS0_VIRT_ALIAS_1_MODSS_INTA1_CFG	0x4B10801000	0x4B1080101F	32 B
NAVSS0_VIRT_ALIAS_1_UDMASS_INTA0_CFG	0x4B10802000	0x4B1080201F	32 B
NAVSS0_VIRT_ALIAS_1_MODSS_INTA0_CFG_IMAP	0x4B10900000	0x4B10907FFF	32 KB
NAVSS0_VIRT_ALIAS_1_MODSS_INTA1_CFG_IMAP	0x4B10908000	0x4B1090FFFF	32 KB
NAVSS0_VIRT_ALIAS_1_UDMASS_INTA0_CFG_IMAP	0x4B10940000	0x4B1097FFFF	256 KB
NAVSS0_VIRT_ALIAS_1_NAV_DDR0_VIRTID_CFG_MMRS	0x4B10A02000	0x4B10A020FF	256 B
NAVSS0_VIRT_ALIAS_1_NAV_DDR1_VIRTID_CFG_MMRS	0x4B10A03000	0x4B10A030FF	256 B
NAVSS0_VIRT_ALIAS_1_UDMAP0_CFG_TCHAN	0x4B10B00000	0x4B10B1FFFF	128 KB
NAVSS0_VIRT_ALIAS_1_UDMAP0_CFG_RCHAN	0x4B10C00000	0x4B10C0FFFF	64 KB
NAVSS0_VIRT_ALIAS_1_UDMAP0_CFG_RFLOW	0x4B10D00000	0x4B10D07FFF	32 KB
NAVSS0_VIRT_ALIAS_1_SPINLOCK0_CFG	0x4B10E00000	0x4B10E07FFF	32 KB
NAVSS0_VIRT_ALIAS_1_TIMERMGR0_CFG	0x4B10E80000	0x4B10E801FF	512 B
NAVSS0_VIRT_ALIAS_1_TIMERMGR1_CFG	0x4B10E81000	0x4B10E811FF	512 B
NAVSS0_VIRT_ALIAS_1_TIMERMGR0_CFG_OES	0x4B10F00000	0x4B10F00FFF	4 KB
NAVSS0_VIRT_ALIAS_1_TIMERMGR1_CFG_OES	0x4B10F01000	0x4B10F01FFF	4 KB
NAVSS0_VIRT_ALIAS_1_ECCAGGR0	0x4B11000000	0x4B110003FF	1 KB
NAVSS0_VIRT_ALIAS_1_UDMASS_ECCAGGR_CFG	0x4B11001000	0x4B110013FF	1 KB
NAVSS0_VIRT_ALIAS_1_VIRTSS_ECCAGGR_CFG	0x4B11002000	0x4B110023FF	1 KB
NAVSS0_VIRT_ALIAS_1_PAT0_CFG_MMRS	0x4B11010000	0x4B110100FF	256 B
NAVSS0_VIRT_ALIAS_1_PAT1_CFG_MMRS	0x4B11011000	0x4B110110FF	256 B
NAVSS0_VIRT_ALIAS_1_PAT2_CFG_MMRS	0x4B11012000	0x4B110120FF	256 B
NAVSS0_VIRT_ALIAS_1_PAT3_CFG_MMRS	0x4B11013000	0x4B110130FF	256 B
NAVSS0_VIRT_ALIAS_1_PAT4_CFG_MMRS	0x4B11014000	0x4B110140FF	256 B
NAVSS0_VIRT_ALIAS_1_UDMASS_INTA0_CFG_GCNTCFG	0x4B11040000	0x4B11043FFF	16 KB
NAVSS0_VIRT_ALIAS_1_RINGACC0_CFG	0x4B11080000	0x4B110BFFFF	256 KB
NAVSS0_VIRT_ALIAS_1_REGS0_CFG_MMRS	0x4B110C0000	0x4B110C00FF	256 B

**Table 2-2. NAVSS0 Alias Memory Map (continued)**

Region Name	Start Address	End Address	Size
NAVSS0_VIRT_ALIAS_1_CPTS0_S_VBUSP_CPTS_VBUSP	0x4B110D0000	0x4B110D03FF	1 KB
NAVSS0_VIRT_ALIAS_1_INTR0_CFG_INTR_ROUTER_CFG	0x4B110E0000	0x4B110E3FFF	16 KB
NAVSS0_VIRT_ALIAS_1_UDMASS_INTA0_CFG_L2G	0x4B11100000	0x4B11100FFF	4 KB
NAVSS0_VIRT_ALIAS_1_UDMASS_INTA0_CFG_MCAST	0x4B11110000	0x4B11113FFF	16 KB
NAVSS0_VIRT_ALIAS_1_PROXY0_CFG_BUF_CFG_GCFG	0x4B11120000	0x4B111200FF	256 B
NAVSS0_VIRT_ALIAS_1_PROXY0_CFG_BUFRAM_SLV_RAM	0x4B11130000	0x4B11133FFF	16 KB
NAVSS0_VIRT_ALIAS_1_SEC_PROXY0_CFG_MMRS	0x4B11140000	0x4B111400FF	256 B
NAVSS0_VIRT_ALIAS_1_UDMAP0_CFG_GCFG	0x4B11150000	0x4B111500FF	256 B
NAVSS0_VIRT_ALIAS_1_RINGACC0_CFG_GCFG	0x4B11160000	0x4B111603FF	1 KB
NAVSS0_VIRT_ALIAS_1_PSILSS_LOCAL_CFG_MMRS	0x4B11170000	0x4B11170FFF	4 KB
NAVSS0_VIRT_ALIAS_1_PSILSS_GLOBAL_CFG_MMRS	0x4B11180000	0x4B11180FFF	4 KB
NAVSS0_VIRT_ALIAS_1_PSILSS_TR_CFG_MMRS	0x4B11190000	0x4B11190FFF	4 KB
NAVSS0_VIRT_ALIAS_1_MCRC0_S_CFG_MCRC64	0x4B11F70000	0x4B11F70FFF	4 KB
NAVSS0_VIRT_ALIAS_1_PSILCFG0_CFG_PROXY	0x4B11F78000	0x4B11F781FF	512 B
NAVSS0_VIRT_ALIAS_1_MAILBOX0_CFG_REGS0	0x4B11F80000	0x4B11F801FF	512 B
NAVSS0_VIRT_ALIAS_1_MAILBOX0_CFG_REGS1	0x4B11F81000	0x4B11F811FF	512 B
NAVSS0_VIRT_ALIAS_1_MAILBOX0_CFG_REGS2	0x4B11F82000	0x4B11F821FF	512 B
NAVSS0_VIRT_ALIAS_1_MAILBOX0_CFG_REGS3	0x4B11F83000	0x4B11F831FF	512 B
NAVSS0_VIRT_ALIAS_1_MAILBOX0_CFG_REGS4	0x4B11F84000	0x4B11F841FF	512 B
NAVSS0_VIRT_ALIAS_1_MAILBOX0_CFG_REGS5	0x4B11F85000	0x4B11F851FF	512 B
NAVSS0_VIRT_ALIAS_1_MAILBOX0_CFG_REGS6	0x4B11F86000	0x4B11F861FF	512 B
NAVSS0_VIRT_ALIAS_1_MAILBOX0_CFG_REGS7	0x4B11F87000	0x4B11F871FF	512 B
NAVSS0_VIRT_ALIAS_1_MAILBOX0_CFG_REGS8	0x4B11F88000	0x4B11F881FF	512 B
NAVSS0_VIRT_ALIAS_1_MAILBOX0_CFG_REGS9	0x4B11F89000	0x4B11F891FF	512 B
NAVSS0_VIRT_ALIAS_1_MAILBOX0_CFG_REGS10	0x4B11F8A000	0x4B11F8A1FF	512 B
NAVSS0_VIRT_ALIAS_1_MAILBOX0_CFG_REGS11	0x4B11F8B000	0x4B11F8B1FF	512 B
NAVSS0_VIRT_ALIAS_1_RINGACC0_CFG_MON	0x4B12000000	0x4B1201FFFF	128 KB
NAVSS0_VIRT_ALIAS_1_TIMERMGR0_CFG_TIMERS	0x4B12200000	0x4B1223FFFF	256 KB
NAVSS0_VIRT_ALIAS_1_TIMERMGR1_CFG_TIMERS	0x4B12240000	0x4B1227FFFF	256 KB
NAVSS0_VIRT_ALIAS_1_SEC_PROXY0_CFG_RT	0x4B12400000	0x4B124FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_1_SEC_PROXY0_CFG_SCFG	0x4B12800000	0x4B128FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_1_SEC_PROXY0_SRC_TARGET_DATA	0x4B12C00000	0x4B12CFFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_1_PROXY0_SRC_TARGET0_DATA	0x4B13000000	0x4B1303FFFF	256 KB
NAVSS0_VIRT_ALIAS_1_PROXY0_CFG_BUF_CFG	0x4B13400000	0x4B1343FFFF	256 KB
NAVSS0_VIRT_ALIAS_1_UDMASS_INTA0_CFG_GCINTRI	0x4B13800000	0x4B139FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_1_MODSS_INTA0_CFG_INTR	0x4B13C00000	0x4B13C3FFFF	256 KB
NAVSS0_VIRT_ALIAS_1_MODSS_INTA1_CFG_INTR	0x4B13C40000	0x4B13C7FFFF	256 KB
NAVSS0_VIRT_ALIAS_1_UDMASS_INTA0_CFG_INTR	0x4B13D00000	0x4B13DFFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_1_UDMAP0_CFG_RCHANRT	0x4B14000000	0x4B140FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_1_UDMAP0_CFG_TCHANRT	0x4B15000000	0x4B151FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_1_PAT0_CFG_SCRATCH	0x4B16200000	0x4B1620FFFF	64 KB
NAVSS0_VIRT_ALIAS_1_PAT1_CFG_SCRATCH	0x4B16210000	0x4B1621FFFF	64 KB
NAVSS0_VIRT_ALIAS_1_PAT2_CFG_SCRATCH	0x4B16220000	0x4B1622FFFF	64 KB
NAVSS0_VIRT_ALIAS_1_PAT3_CFG_SCRATCH	0x4B16230000	0x4B16231FFF	8 KB
NAVSS0_VIRT_ALIAS_1_PAT4_CFG_SCRATCH	0x4B16240000	0x4B16241FFF	8 KB

**Table 2-2. NAVSS0 Alias Memory Map (continued)**

Region Name	Start Address	End Address	Size
NAVSS0_VIRT_ALIAS_1_PAT0_CFG_TABLE	0x4B16400000	0x4B1643FFFF	256 KB
NAVSS0_VIRT_ALIAS_1_PAT1_CFG_TABLE	0x4B16440000	0x4B1647FFFF	256 KB
NAVSS0_VIRT_ALIAS_1_PAT2_CFG_TABLE	0x4B16480000	0x4B164BFFFF	256 KB
NAVSS0_VIRT_ALIAS_1_PAT3_CFG_TABLE	0x4B164C0000	0x4B164C7FFF	32 KB
NAVSS0_VIRT_ALIAS_1_PAT4_CFG_TABLE	0x4B16500000	0x4B16507FFF	32 KB
NAVSS0_VIRT_ALIAS_1_TCU_CFG	0x4B16600000	0x4B167FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_1_RINGACC0_SRC_FIFOS	0x4B18000000	0x4B183FFFFFFF	4 MB
NAVSS0_VIRT_ALIAS_1_RINGACC0_CFG_RT	0x4B1C000000	0x4B1C3FFFFFFF	4 MB
NAVSS0_VIRT_ALIAS_2_MODSS_INTA0_CFG	0x4B20800000	0x4B2080001F	32 B
NAVSS0_VIRT_ALIAS_2_MODSS_INTA1_CFG	0x4B20801000	0x4B2080101F	32 B
NAVSS0_VIRT_ALIAS_2_UDMASS_INTA0_CFG	0x4B20802000	0x4B2080201F	32 B
NAVSS0_VIRT_ALIAS_2_MODSS_INTA0_CFG_IMAP	0x4B20900000	0x4B20907FFF	32 KB
NAVSS0_VIRT_ALIAS_2_MODSS_INTA1_CFG_IMAP	0x4B20908000	0x4B2090FFFFFF	32 KB
NAVSS0_VIRT_ALIAS_2_UDMASS_INTA0_CFG_IMAP	0x4B20940000	0x4B2097FFFF	256 KB
NAVSS0_VIRT_ALIAS_2_NAV_DDR0_VIRTID_CFG_MMRS	0x4B20A02000	0x4B20A020FF	256 B
NAVSS0_VIRT_ALIAS_2_NAV_DDR1_VIRTID_CFG_MMRS	0x4B20A03000	0x4B20A030FF	256 B
NAVSS0_VIRT_ALIAS_2_UDMAP0_CFG_TCHAN	0x4B20B00000	0x4B20B1FFFF	128 KB
NAVSS0_VIRT_ALIAS_2_UDMAP0_CFG_RCHAN	0x4B20C00000	0x4B20C0FFFF	64 KB
NAVSS0_VIRT_ALIAS_2_UDMAP0_CFG_RFLOW	0x4B20D00000	0x4B20D07FFF	32 KB
NAVSS0_VIRT_ALIAS_2_SPINLOCK0_CFG	0x4B20E00000	0x4B20E07FFF	32 KB
NAVSS0_VIRT_ALIAS_2_TIMERMGR0_CFG	0x4B20E80000	0x4B20E801FF	512 B
NAVSS0_VIRT_ALIAS_2_TIMERMGR1_CFG	0x4B20E81000	0x4B20E811FF	512 B
NAVSS0_VIRT_ALIAS_2_TIMERMGR0_CFG_OES	0x4B20F00000	0x4B20F00FFF	4 KB
NAVSS0_VIRT_ALIAS_2_TIMERMGR1_CFG_OES	0x4B20F01000	0x4B20F01FFF	4 KB
NAVSS0_VIRT_ALIAS_2_ECCAGGR0	0x4B21000000	0x4B210003FF	1 KB
NAVSS0_VIRT_ALIAS_2_UDMASS_ECCAGGR_CFG	0x4B21001000	0x4B210013FF	1 KB
NAVSS0_VIRT_ALIAS_2_VIRTSS_ECCAGGR_CFG	0x4B21002000	0x4B210023FF	1 KB
NAVSS0_VIRT_ALIAS_2_PAT0_CFG_MMRS	0x4B21010000	0x4B210100FF	256 B
NAVSS0_VIRT_ALIAS_2_PAT1_CFG_MMRS	0x4B21011000	0x4B210110FF	256 B
NAVSS0_VIRT_ALIAS_2_PAT2_CFG_MMRS	0x4B21012000	0x4B210120FF	256 B
NAVSS0_VIRT_ALIAS_2_PAT3_CFG_MMRS	0x4B21013000	0x4B210130FF	256 B
NAVSS0_VIRT_ALIAS_2_PAT4_CFG_MMRS	0x4B21014000	0x4B210140FF	256 B
NAVSS0_VIRT_ALIAS_2_UDMASS_INTA0_CFG_GCNTCFG	0x4B21040000	0x4B21043FFF	16 KB
NAVSS0_VIRT_ALIAS_2_RINGACC0_CFG	0x4B21080000	0x4B210BFFFF	256 KB
NAVSS0_VIRT_ALIAS_2_REGS0_CFG_MMRS	0x4B210C0000	0x4B210C00FF	256 B
NAVSS0_VIRT_ALIAS_2_CPTS0_S_VBUSP_CPTS_VBUSP	0x4B210D0000	0x4B210D03FF	1 KB
NAVSS0_VIRT_ALIAS_2_INTR0_CFG_INTR_ROUTER_CFG	0x4B210E0000	0x4B210E3FFF	16 KB
NAVSS0_VIRT_ALIAS_2_UDMASS_INTA0_CFG_L2G	0x4B21100000	0x4B21100FFF	4 KB
NAVSS0_VIRT_ALIAS_2_UDMASS_INTA0_CFG_MCAST	0x4B21110000	0x4B21113FFF	16 KB
NAVSS0_VIRT_ALIAS_2_PROXY0_CFG_BUF_CFG_GCFCG	0x4B21120000	0x4B211200FF	256 B
NAVSS0_VIRT_ALIAS_2_PROXY0_CFG_BUFRAM_SLV_RAM	0x4B21130000	0x4B21133FFF	16 KB
NAVSS0_VIRT_ALIAS_2_SEC_PROXY0_CFG_MMRS	0x4B21140000	0x4B211400FF	256 B
NAVSS0_VIRT_ALIAS_2_UDMAP0_CFG_GCFCG	0x4B21150000	0x4B211500FF	256 B
NAVSS0_VIRT_ALIAS_2_RINGACC0_CFG_GCFCG	0x4B21160000	0x4B211603FF	1 KB
NAVSS0_VIRT_ALIAS_2_PSILSS_LOCAL_CFG_MMRS	0x4B21170000	0x4B21170FFF	4 KB

**Table 2-2. NAVSS0 Alias Memory Map (continued)**

Region Name	Start Address	End Address	Size
NAVSS0_VIRT_ALIAS_2_PSILSS_GLOBAL_CFG_MMRS	0x4B21180000	0x4B21180FFF	4 KB
NAVSS0_VIRT_ALIAS_2_PSILSS_TR_CFG_MMRS	0x4B21190000	0x4B21190FFF	4 KB
NAVSS0_VIRT_ALIAS_2_MCRC0_S_CFG_MCRC64	0x4B21F70000	0x4B21F70FFF	4 KB
NAVSS0_VIRT_ALIAS_2_PSILCFG0_CFG_PROXY	0x4B21F78000	0x4B21F781FF	512 B
NAVSS0_VIRT_ALIAS_2_MAILBOX0_CFG_REGS0	0x4B21F80000	0x4B21F801FF	512 B
NAVSS0_VIRT_ALIAS_2_MAILBOX0_CFG_REGS1	0x4B21F81000	0x4B21F811FF	512 B
NAVSS0_VIRT_ALIAS_2_MAILBOX0_CFG_REGS2	0x4B21F82000	0x4B21F821FF	512 B
NAVSS0_VIRT_ALIAS_2_MAILBOX0_CFG_REGS3	0x4B21F83000	0x4B21F831FF	512 B
NAVSS0_VIRT_ALIAS_2_MAILBOX0_CFG_REGS4	0x4B21F84000	0x4B21F841FF	512 B
NAVSS0_VIRT_ALIAS_2_MAILBOX0_CFG_REGS5	0x4B21F85000	0x4B21F851FF	512 B
NAVSS0_VIRT_ALIAS_2_MAILBOX0_CFG_REGS6	0x4B21F86000	0x4B21F861FF	512 B
NAVSS0_VIRT_ALIAS_2_MAILBOX0_CFG_REGS7	0x4B21F87000	0x4B21F871FF	512 B
NAVSS0_VIRT_ALIAS_2_MAILBOX0_CFG_REGS8	0x4B21F88000	0x4B21F881FF	512 B
NAVSS0_VIRT_ALIAS_2_MAILBOX0_CFG_REGS9	0x4B21F89000	0x4B21F891FF	512 B
NAVSS0_VIRT_ALIAS_2_MAILBOX0_CFG_REGS10	0x4B21F8A000	0x4B21F8A1FF	512 B
NAVSS0_VIRT_ALIAS_2_MAILBOX0_CFG_REGS11	0x4B21F8B000	0x4B21F8B1FF	512 B
NAVSS0_VIRT_ALIAS_2_RINGACC0_CFG_MON	0x4B22000000	0x4B2201FFFF	128 KB
NAVSS0_VIRT_ALIAS_2_TIMERMGR0_CFG_TIMERS	0x4B22200000	0x4B2223FFFF	256 KB
NAVSS0_VIRT_ALIAS_2_TIMERMGR1_CFG_TIMERS	0x4B22240000	0x4B2227FFFF	256 KB
NAVSS0_VIRT_ALIAS_2_SEC_PROXY0_CFG_RT	0x4B22400000	0x4B224FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_2_SEC_PROXY0_CFG_SCFG	0x4B22800000	0x4B228FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_2_SEC_PROXY0_SRC_TARGET_DATA	0x4B22C00000	0x4B22CFFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_2_PROXY0_SRC_TARGET0_DATA	0x4B23000000	0x4B2303FFFF	256 KB
NAVSS0_VIRT_ALIAS_2_PROXY0_CFG_BUF_CFG	0x4B23400000	0x4B2343FFFF	256 KB
NAVSS0_VIRT_ALIAS_2_UDMASS_INTA0_CFG_GCINTRTI	0x4B23800000	0x4B239FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_2_MODSS_INTA0_CFG_INTR	0x4B23C00000	0x4B23C3FFFF	256 KB
NAVSS0_VIRT_ALIAS_2_MODSS_INTA1_CFG_INTR	0x4B23C40000	0x4B23C7FFFF	256 KB
NAVSS0_VIRT_ALIAS_2_UDMASS_INTA0_CFG_INTR	0x4B23D00000	0x4B23DFFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_2_UDMAP0_CFG_RCHANRT	0x4B24000000	0x4B240FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_2_UDMAP0_CFG_TCHANRT	0x4B25000000	0x4B251FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_2_PAT0_CFG_SCRATCH	0x4B26200000	0x4B2620FFFF	64 KB
NAVSS0_VIRT_ALIAS_2_PAT1_CFG_SCRATCH	0x4B26210000	0x4B2621FFFF	64 KB
NAVSS0_VIRT_ALIAS_2_PAT2_CFG_SCRATCH	0x4B26220000	0x4B2622FFFF	64 KB
NAVSS0_VIRT_ALIAS_2_PAT3_CFG_SCRATCH	0x4B26230000	0x4B26231FFF	8 KB
NAVSS0_VIRT_ALIAS_2_PAT4_CFG_SCRATCH	0x4B26240000	0x4B26241FFF	8 KB
NAVSS0_VIRT_ALIAS_2_PAT0_CFG_TABLE	0x4B26400000	0x4B2643FFFF	256 KB
NAVSS0_VIRT_ALIAS_2_PAT1_CFG_TABLE	0x4B26440000	0x4B2647FFFF	256 KB
NAVSS0_VIRT_ALIAS_2_PAT2_CFG_TABLE	0x4B26480000	0x4B264BFFFF	256 KB
NAVSS0_VIRT_ALIAS_2_PAT3_CFG_TABLE	0x4B264C0000	0x4B264C7FFF	32 KB
NAVSS0_VIRT_ALIAS_2_PAT4_CFG_TABLE	0x4B26500000	0x4B26507FFF	32 KB
NAVSS0_VIRT_ALIAS_2_TCU_CFG	0x4B26600000	0x4B267FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_2_RINGACC0_SRC_FIFOS	0x4B28000000	0x4B283FFFFFFF	4 MB
NAVSS0_VIRT_ALIAS_2_RINGACC0_CFG_RT	0x4B2C000000	0x4B2C3FFFFFFF	4 MB
NAVSS0_VIRT_ALIAS_3_MODSS_INTA0_CFG	0x4B30800000	0x4B3080001F	32 B
NAVSS0_VIRT_ALIAS_3_MODSS_INTA1_CFG	0x4B30801000	0x4B3080101F	32 B

**Table 2-2. NAVSS0 Alias Memory Map (continued)**

Region Name	Start Address	End Address	Size
NAVSS0_VIRT_ALIAS_3_UDMASS_INTA0_CFG	0x4B30802000	0x4B3080201F	32 B
NAVSS0_VIRT_ALIAS_3_MODSS_INTA0_CFG_IMAP	0x4B30900000	0x4B30907FFF	32 KB
NAVSS0_VIRT_ALIAS_3_MODSS_INTA1_CFG_IMAP	0x4B30908000	0x4B3090FFFF	32 KB
NAVSS0_VIRT_ALIAS_3_UDMASS_INTA0_CFG_IMAP	0x4B30940000	0x4B3097FFFF	256 KB
NAVSS0_VIRT_ALIAS_3_NAV_DDR0_VIRTID_CFG_MMRS	0x4B30A02000	0x4B30A020FF	256 B
NAVSS0_VIRT_ALIAS_3_NAV_DDR1_VIRTID_CFG_MMRS	0x4B30A03000	0x4B30A030FF	256 B
NAVSS0_VIRT_ALIAS_3_UDMAP0_CFG_TCHAN	0x4B30B00000	0x4B30B1FFFF	128 KB
NAVSS0_VIRT_ALIAS_3_UDMAP0_CFG_RCHAN	0x4B30C00000	0x4B30C0FFFF	64 KB
NAVSS0_VIRT_ALIAS_3_UDMAP0_CFG_RFLOW	0x4B30D00000	0x4B30D07FFF	32 KB
NAVSS0_VIRT_ALIAS_3_SPINLOCK0_CFG	0x4B30E00000	0x4B30E07FFF	32 KB
NAVSS0_VIRT_ALIAS_3_TIMERMGR0_CFG	0x4B30E80000	0x4B30E801FF	512 B
NAVSS0_VIRT_ALIAS_3_TIMERMGR1_CFG	0x4B30E81000	0x4B30E811FF	512 B
NAVSS0_VIRT_ALIAS_3_TIMERMGR0_CFG_OES	0x4B30F00000	0x4B30F00FFF	4 KB
NAVSS0_VIRT_ALIAS_3_TIMERMGR1_CFG_OES	0x4B30F01000	0x4B30F01FFF	4 KB
NAVSS0_VIRT_ALIAS_3_ECCAGGR0	0x4B31000000	0x4B310003FF	1 KB
NAVSS0_VIRT_ALIAS_3_UDMASS_ECCAGGR_CFG	0x4B31001000	0x4B310013FF	1 KB
NAVSS0_VIRT_ALIAS_3_VIRTSS_ECCAGGR_CFG	0x4B31002000	0x4B310023FF	1 KB
NAVSS0_VIRT_ALIAS_3_PAT0_CFG_MMRS	0x4B31010000	0x4B310100FF	256 B
NAVSS0_VIRT_ALIAS_3_PAT1_CFG_MMRS	0x4B31011000	0x4B310110FF	256 B
NAVSS0_VIRT_ALIAS_3_PAT2_CFG_MMRS	0x4B31012000	0x4B310120FF	256 B
NAVSS0_VIRT_ALIAS_3_PAT3_CFG_MMRS	0x4B31013000	0x4B310130FF	256 B
NAVSS0_VIRT_ALIAS_3_PAT4_CFG_MMRS	0x4B31014000	0x4B310140FF	256 B
NAVSS0_VIRT_ALIAS_3_UDMASS_INTA0_CFG_GCNTCFG	0x4B31040000	0x4B31043FFF	16 KB
NAVSS0_VIRT_ALIAS_3_RINGACC0_CFG	0x4B31080000	0x4B310BFFFF	256 KB
NAVSS0_VIRT_ALIAS_3_REGS0_CFG_MMRS	0x4B310C0000	0x4B310C00FF	256 B
NAVSS0_VIRT_ALIAS_3_CPTS0_S_VBUSB_CPTS_VBUSB	0x4B310D0000	0x4B310D03FF	1 KB
NAVSS0_VIRT_ALIAS_3_INTR0_CFG_INTR_ROUTER_CFG	0x4B310E0000	0x4B310E3FFF	16 KB
NAVSS0_VIRT_ALIAS_3_UDMASS_INTA0_CFG_L2G	0x4B31100000	0x4B31100FFF	4 KB
NAVSS0_VIRT_ALIAS_3_UDMASS_INTA0_CFG_MCAST	0x4B31110000	0x4B31113FFF	16 KB
NAVSS0_VIRT_ALIAS_3_PROXY0_CFG_BUF_CFG_GCFG	0x4B31120000	0x4B311200FF	256 B
NAVSS0_VIRT_ALIAS_3_PROXY0_CFG_BUFRAM_SLV_RAM	0x4B31130000	0x4B31133FFF	16 KB
NAVSS0_VIRT_ALIAS_3_SEC_PROXY0_CFG_MMRS	0x4B31140000	0x4B311400FF	256 B
NAVSS0_VIRT_ALIAS_3_UDMAP0_CFG_GCFG	0x4B31150000	0x4B311500FF	256 B
NAVSS0_VIRT_ALIAS_3_RINGACC0_CFG_GCFG	0x4B31160000	0x4B311603FF	1 KB
NAVSS0_VIRT_ALIAS_3_PSILSS_LOCAL_CFG_MMRS	0x4B31170000	0x4B31170FFF	4 KB
NAVSS0_VIRT_ALIAS_3_PSILSS_GLOBAL_CFG_MMRS	0x4B31180000	0x4B31180FFF	4 KB
NAVSS0_VIRT_ALIAS_3_PSILSS_TR_CFG_MMRS	0x4B31190000	0x4B31190FFF	4 KB
NAVSS0_VIRT_ALIAS_3_MCRC0_S_CFG_MCRC64	0x4B31F70000	0x4B31F70FFF	4 KB
NAVSS0_VIRT_ALIAS_3_PSILCFG0_CFG_PROXY	0x4B31F78000	0x4B31F781FF	512 B
NAVSS0_VIRT_ALIAS_3_MAILBOX0_CFG_REGS0	0x4B31F80000	0x4B31F801FF	512 B
NAVSS0_VIRT_ALIAS_3_MAILBOX0_CFG_REGS1	0x4B31F81000	0x4B31F811FF	512 B
NAVSS0_VIRT_ALIAS_3_MAILBOX0_CFG_REGS2	0x4B31F82000	0x4B31F821FF	512 B
NAVSS0_VIRT_ALIAS_3_MAILBOX0_CFG_REGS3	0x4B31F83000	0x4B31F831FF	512 B
NAVSS0_VIRT_ALIAS_3_MAILBOX0_CFG_REGS4	0x4B31F84000	0x4B31F841FF	512 B
NAVSS0_VIRT_ALIAS_3_MAILBOX0_CFG_REGS5	0x4B31F85000	0x4B31F851FF	512 B



**Table 2-2. NAVSS0 Alias Memory Map (continued)**

Region Name	Start Address	End Address	Size
NAVSS0_VIRT_ALIAS_3_MAILBOX0_CFG_REGS6	0x4B31F86000	0x4B31F861FF	512 B
NAVSS0_VIRT_ALIAS_3_MAILBOX0_CFG_REGS7	0x4B31F87000	0x4B31F871FF	512 B
NAVSS0_VIRT_ALIAS_3_MAILBOX0_CFG_REGS8	0x4B31F88000	0x4B31F881FF	512 B
NAVSS0_VIRT_ALIAS_3_MAILBOX0_CFG_REGS9	0x4B31F89000	0x4B31F891FF	512 B
NAVSS0_VIRT_ALIAS_3_MAILBOX0_CFG_REGS10	0x4B31F8A000	0x4B31F8A1FF	512 B
NAVSS0_VIRT_ALIAS_3_MAILBOX0_CFG_REGS11	0x4B31F8B000	0x4B31F8B1FF	512 B
NAVSS0_VIRT_ALIAS_3_RINGACC0_CFG_MON	0x4B32000000	0x4B3201FFFF	128 KB
NAVSS0_VIRT_ALIAS_3_TIMERMGR0_CFG_TIMERS	0x4B32200000	0x4B3223FFFF	256 KB
NAVSS0_VIRT_ALIAS_3_TIMERMGR1_CFG_TIMERS	0x4B32240000	0x4B3227FFFF	256 KB
NAVSS0_VIRT_ALIAS_3_SEC_PROXY0_CFG_RT	0x4B32400000	0x4B324FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_3_SEC_PROXY0_CFG_SCFG	0x4B32800000	0x4B328FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_3_SEC_PROXY0_SRC_TARGET_DATA	0x4B32C00000	0x4B32CFFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_3_PROXY0_SRC_TARGET0_DATA	0x4B33000000	0x4B3303FFFF	256 KB
NAVSS0_VIRT_ALIAS_3_PROXY0_CFG_BUF_CFG	0x4B33400000	0x4B3343FFFF	256 KB
NAVSS0_VIRT_ALIAS_3_UDMASS_INTA0_CFG_GCNTRTI	0x4B33800000	0x4B339FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_3_MODSS_INTA0_CFG_INTR	0x4B33C00000	0x4B33C3FFFF	256 KB
NAVSS0_VIRT_ALIAS_3_MODSS_INTA1_CFG_INTR	0x4B33C40000	0x4B33C7FFFF	256 KB
NAVSS0_VIRT_ALIAS_3_UDMASS_INTA0_CFG_INTR	0x4B33D00000	0x4B33DFFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_3_UDMAP0_CFG_RCHANRT	0x4B34000000	0x4B340FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_3_UDMAP0_CFG_TCHANRT	0x4B35000000	0x4B351FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_3_PAT0_CFG_SCRATCH	0x4B36200000	0x4B3620FFFF	64 KB
NAVSS0_VIRT_ALIAS_3_PAT1_CFG_SCRATCH	0x4B36210000	0x4B3621FFFF	64 KB
NAVSS0_VIRT_ALIAS_3_PAT2_CFG_SCRATCH	0x4B36220000	0x4B3622FFFF	64 KB
NAVSS0_VIRT_ALIAS_3_PAT3_CFG_SCRATCH	0x4B36230000	0x4B36231FFF	8 KB
NAVSS0_VIRT_ALIAS_3_PAT4_CFG_SCRATCH	0x4B36240000	0x4B36241FFF	8 KB
NAVSS0_VIRT_ALIAS_3_PAT0_CFG_TABLE	0x4B36400000	0x4B3643FFFF	256 KB
NAVSS0_VIRT_ALIAS_3_PAT1_CFG_TABLE	0x4B36440000	0x4B3647FFFF	256 KB
NAVSS0_VIRT_ALIAS_3_PAT2_CFG_TABLE	0x4B36480000	0x4B364BFFFF	256 KB
NAVSS0_VIRT_ALIAS_3_PAT3_CFG_TABLE	0x4B364C0000	0x4B364C7FFF	32 KB
NAVSS0_VIRT_ALIAS_3_PAT4_CFG_TABLE	0x4B36500000	0x4B36507FFF	32 KB
NAVSS0_VIRT_ALIAS_3_TCU_CFG	0x4B36600000	0x4B367FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_3_RINGACC0_SRC_FIFOS	0x4B38000000	0x4B383FFFFFFF	4 MB
NAVSS0_VIRT_ALIAS_3_RINGACC0_CFG_RT	0x4B3C000000	0x4B3C3FFFFFFF	4 MB
NAVSS0_VIRT_ALIAS_4_MODSS_INTA0_CFG	0x4B40800000	0x4B4080001F	32 B
NAVSS0_VIRT_ALIAS_4_MODSS_INTA1_CFG	0x4B40801000	0x4B4080101F	32 B
NAVSS0_VIRT_ALIAS_4_UDMASS_INTA0_CFG	0x4B40802000	0x4B4080201F	32 B
NAVSS0_VIRT_ALIAS_4_MODSS_INTA0_CFG_IMAP	0x4B40900000	0x4B40907FFF	32 KB
NAVSS0_VIRT_ALIAS_4_MODSS_INTA1_CFG_IMAP	0x4B40908000	0x4B4090FFFFFF	32 KB
NAVSS0_VIRT_ALIAS_4_UDMASS_INTA0_CFG_IMAP	0x4B40940000	0x4B4097FFFF	256 KB
NAVSS0_VIRT_ALIAS_4_NAV_DDR0_VIRTID_CFG_MMRS	0x4B40A02000	0x4B40A020FF	256 B
NAVSS0_VIRT_ALIAS_4_NAV_DDR1_VIRTID_CFG_MMRS	0x4B40A03000	0x4B40A030FF	256 B
NAVSS0_VIRT_ALIAS_4_UDMAP0_CFG_TCHAN	0x4B40B00000	0x4B40B1FFFF	128 KB
NAVSS0_VIRT_ALIAS_4_UDMAP0_CFG_RCHAN	0x4B40C00000	0x4B40C0FFFF	64 KB
NAVSS0_VIRT_ALIAS_4_UDMAP0_CFG_RFLOW	0x4B40D00000	0x4B40D07FFF	32 KB
NAVSS0_VIRT_ALIAS_4_SPINLOCK0_CFG	0x4B40E00000	0x4B40E07FFF	32 KB



**Table 2-2. NAVSS0 Alias Memory Map (continued)**

Region Name	Start Address	End Address	Size
NAVSS0_VIRT_ALIAS_4_TIMERMGR0_CFG	0x4B40E80000	0x4B40E801FF	512 B
NAVSS0_VIRT_ALIAS_4_TIMERMGR1_CFG	0x4B40E81000	0x4B40E811FF	512 B
NAVSS0_VIRT_ALIAS_4_TIMERMGR0_CFG_OES	0x4B40F00000	0x4B40F00FFF	4 KB
NAVSS0_VIRT_ALIAS_4_TIMERMGR1_CFG_OES	0x4B40F01000	0x4B40F01FFF	4 KB
NAVSS0_VIRT_ALIAS_4_ECCAGGR0	0x4B41000000	0x4B410003FF	1 KB
NAVSS0_VIRT_ALIAS_4_UDMASS_ECCAGGR_CFG	0x4B41001000	0x4B410013FF	1 KB
NAVSS0_VIRT_ALIAS_4_VIRTSS_ECCAGGR_CFG	0x4B41002000	0x4B410023FF	1 KB
NAVSS0_VIRT_ALIAS_4_PAT0_CFG_MMRS	0x4B41010000	0x4B410100FF	256 B
NAVSS0_VIRT_ALIAS_4_PAT1_CFG_MMRS	0x4B41011000	0x4B410110FF	256 B
NAVSS0_VIRT_ALIAS_4_PAT2_CFG_MMRS	0x4B41012000	0x4B410120FF	256 B
NAVSS0_VIRT_ALIAS_4_PAT3_CFG_MMRS	0x4B41013000	0x4B410130FF	256 B
NAVSS0_VIRT_ALIAS_4_PAT4_CFG_MMRS	0x4B41014000	0x4B410140FF	256 B
NAVSS0_VIRT_ALIAS_4_UDMASS_INTA0_CFG_GCNTCFG	0x4B41040000	0x4B41043FFF	16 KB
NAVSS0_VIRT_ALIAS_4_RINGACC0_CFG	0x4B41080000	0x4B410BFFFF	256 KB
NAVSS0_VIRT_ALIAS_4_REGS0_CFG_MMRS	0x4B410C0000	0x4B410C00FF	256 B
NAVSS0_VIRT_ALIAS_4_CPTS0_S_VBUSP_CPTS_VBUSP	0x4B410D0000	0x4B410D03FF	1 KB
NAVSS0_VIRT_ALIAS_4_INTR0_CFG_INTR_ROUTER_CFG	0x4B410E0000	0x4B410E3FFF	16 KB
NAVSS0_VIRT_ALIAS_4_UDMASS_INTA0_CFG_L2G	0x4B41100000	0x4B41100FFF	4 KB
NAVSS0_VIRT_ALIAS_4_UDMASS_INTA0_CFG_MCAST	0x4B41110000	0x4B41113FFF	16 KB
NAVSS0_VIRT_ALIAS_4_PROXY0_CFG_BUF_CFG_GCFG	0x4B41120000	0x4B411200FF	256 B
NAVSS0_VIRT_ALIAS_4_PROXY0_CFG_BUFRAM_SLV_RAM	0x4B41130000	0x4B41133FFF	16 KB
NAVSS0_VIRT_ALIAS_4_SEC_PROXY0_CFG_MMRS	0x4B41140000	0x4B411400FF	256 B
NAVSS0_VIRT_ALIAS_4_UDMAP0_CFG_GCFG	0x4B41150000	0x4B411500FF	256 B
NAVSS0_VIRT_ALIAS_4_RINGACC0_CFG_GCFG	0x4B41160000	0x4B411603FF	1 KB
NAVSS0_VIRT_ALIAS_4_PSILSS_LOCAL_CFG_MMRS	0x4B41170000	0x4B41170FFF	4 KB
NAVSS0_VIRT_ALIAS_4_PSILSS_GLOBAL_CFG_MMRS	0x4B41180000	0x4B41180FFF	4 KB
NAVSS0_VIRT_ALIAS_4_PSILSS_TR_CFG_MMRS	0x4B41190000	0x4B41190FFF	4 KB
NAVSS0_VIRT_ALIAS_4_MCRC0_S_CFG_MCRC64	0x4B41F70000	0x4B41F70FFF	4 KB
NAVSS0_VIRT_ALIAS_4_PSILCFG0_CFG_PROXY	0x4B41F78000	0x4B41F781FF	512 B
NAVSS0_VIRT_ALIAS_4_MAILBOX0_CFG_REGS0	0x4B41F80000	0x4B41F801FF	512 B
NAVSS0_VIRT_ALIAS_4_MAILBOX0_CFG_REGS1	0x4B41F81000	0x4B41F811FF	512 B
NAVSS0_VIRT_ALIAS_4_MAILBOX0_CFG_REGS2	0x4B41F82000	0x4B41F821FF	512 B
NAVSS0_VIRT_ALIAS_4_MAILBOX0_CFG_REGS3	0x4B41F83000	0x4B41F831FF	512 B
NAVSS0_VIRT_ALIAS_4_MAILBOX0_CFG_REGS4	0x4B41F84000	0x4B41F841FF	512 B
NAVSS0_VIRT_ALIAS_4_MAILBOX0_CFG_REGS5	0x4B41F85000	0x4B41F851FF	512 B
NAVSS0_VIRT_ALIAS_4_MAILBOX0_CFG_REGS6	0x4B41F86000	0x4B41F861FF	512 B
NAVSS0_VIRT_ALIAS_4_MAILBOX0_CFG_REGS7	0x4B41F87000	0x4B41F871FF	512 B
NAVSS0_VIRT_ALIAS_4_MAILBOX0_CFG_REGS8	0x4B41F88000	0x4B41F881FF	512 B
NAVSS0_VIRT_ALIAS_4_MAILBOX0_CFG_REGS9	0x4B41F89000	0x4B41F891FF	512 B
NAVSS0_VIRT_ALIAS_4_MAILBOX0_CFG_REGS10	0x4B41F8A000	0x4B41F8A1FF	512 B
NAVSS0_VIRT_ALIAS_4_MAILBOX0_CFG_REGS11	0x4B41F8B000	0x4B41F8B1FF	512 B
NAVSS0_VIRT_ALIAS_4_RINGACC0_CFG_MON	0x4B42000000	0x4B4201FFFF	128 KB
NAVSS0_VIRT_ALIAS_4_TIMERMGR0_CFG_TIMERS	0x4B42200000	0x4B4223FFFF	256 KB
NAVSS0_VIRT_ALIAS_4_TIMERMGR1_CFG_TIMERS	0x4B42240000	0x4B4227FFFF	256 KB
NAVSS0_VIRT_ALIAS_4_SEC_PROXY0_CFG_RT	0x4B42400000	0x4B424FFFFF	1 MB

**Table 2-2. NAVSS0 Alias Memory Map (continued)**

Region Name	Start Address	End Address	Size
NAVSS0_VIRT_ALIAS_4_SEC_PROXY0_CFG_SCFG	0x4B42800000	0x4B428FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_4_SEC_PROXY0_SRC_TARGET_DATA	0x4B42C00000	0x4B42CFFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_4_PROXY0_SRC_TARGET0_DATA	0x4B43000000	0x4B4303FFFFFFF	256 KB
NAVSS0_VIRT_ALIAS_4_PROXY0_CFG_BUF_CFG	0x4B43400000	0x4B4343FFFFFFF	256 KB
NAVSS0_VIRT_ALIAS_4_UDMASS_INTA0_CFG_GCINTRTI	0x4B43800000	0x4B439FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_4_MODSS_INTA0_CFG_INTR	0x4B43C00000	0x4B43C3FFFFFFF	256 KB
NAVSS0_VIRT_ALIAS_4_MODSS_INTA1_CFG_INTR	0x4B43C40000	0x4B43C7FFFFFFF	256 KB
NAVSS0_VIRT_ALIAS_4_UDMASS_INTA0_CFG_INTR	0x4B43D00000	0x4B43DFFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_4_UDMAP0_CFG_RCHANRT	0x4B44000000	0x4B440FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_4_UDMAP0_CFG_TCHANRT	0x4B45000000	0x4B451FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_4_PAT0_CFG_SCRATCH	0x4B46200000	0x4B4620FFFFFFF	64 KB
NAVSS0_VIRT_ALIAS_4_PAT1_CFG_SCRATCH	0x4B46210000	0x4B4621FFFFFFF	64 KB
NAVSS0_VIRT_ALIAS_4_PAT2_CFG_SCRATCH	0x4B46220000	0x4B4622FFFFFFF	64 KB
NAVSS0_VIRT_ALIAS_4_PAT3_CFG_SCRATCH	0x4B46230000	0x4B46231FFFF	8 KB
NAVSS0_VIRT_ALIAS_4_PAT4_CFG_SCRATCH	0x4B46240000	0x4B46241FFFF	8 KB
NAVSS0_VIRT_ALIAS_4_PAT0_CFG_TABLE	0x4B46400000	0x4B4643FFFFFFF	256 KB
NAVSS0_VIRT_ALIAS_4_PAT1_CFG_TABLE	0x4B46440000	0x4B4647FFFFFFF	256 KB
NAVSS0_VIRT_ALIAS_4_PAT2_CFG_TABLE	0x4B46480000	0x4B464BFFFFFFF	256 KB
NAVSS0_VIRT_ALIAS_4_PAT3_CFG_TABLE	0x4B464C0000	0x4B464C7FFFF	32 KB
NAVSS0_VIRT_ALIAS_4_PAT4_CFG_TABLE	0x4B46500000	0x4B46507FFFF	32 KB
NAVSS0_VIRT_ALIAS_4_TCU_CFG	0x4B46600000	0x4B467FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_4_RINGACC0_SRC_FIFOS	0x4B48000000	0x4B483FFFFFFF	4 MB
NAVSS0_VIRT_ALIAS_4_RINGACC0_CFG_RT	0x4B4C000000	0x4B4C3FFFFFFF	4 MB
NAVSS0_VIRT_ALIAS_5_MODSS_INTA0_CFG	0x4B50800000	0x4B5080001F	32 B
NAVSS0_VIRT_ALIAS_5_MODSS_INTA1_CFG	0x4B50801000	0x4B5080101F	32 B
NAVSS0_VIRT_ALIAS_5_UDMASS_INTA0_CFG	0x4B50802000	0x4B5080201F	32 B
NAVSS0_VIRT_ALIAS_5_MODSS_INTA0_CFG_IMAP	0x4B50900000	0x4B50907FFF	32 KB
NAVSS0_VIRT_ALIAS_5_MODSS_INTA1_CFG_IMAP	0x4B50908000	0x4B5090FFFFFFF	32 KB
NAVSS0_VIRT_ALIAS_5_UDMASS_INTA0_CFG_IMAP	0x4B50940000	0x4B5097FFFFFFF	256 KB
NAVSS0_VIRT_ALIAS_5_NAV_DDR0_VIRTID_CFG_MMRS	0x4B50A02000	0x4B50A020FF	256 B
NAVSS0_VIRT_ALIAS_5_NAV_DDR1_VIRTID_CFG_MMRS	0x4B50A03000	0x4B50A030FF	256 B
NAVSS0_VIRT_ALIAS_5_UDMAP0_CFG_TCHAN	0x4B50B00000	0x4B50B1FFFFFFF	128 KB
NAVSS0_VIRT_ALIAS_5_UDMAP0_CFG_RCHAN	0x4B50C00000	0x4B50C0FFFFFFF	64 KB
NAVSS0_VIRT_ALIAS_5_UDMAP0_CFG_RFLOW	0x4B50D00000	0x4B50D07FFFF	32 KB
NAVSS0_VIRT_ALIAS_5_SPINLOCK0_CFG	0x4B50E00000	0x4B50E07FFFF	32 KB
NAVSS0_VIRT_ALIAS_5_TIMERMGR0_CFG	0x4B50E80000	0x4B50E801FF	512 B
NAVSS0_VIRT_ALIAS_5_TIMERMGR1_CFG	0x4B50E81000	0x4B50E811FF	512 B
NAVSS0_VIRT_ALIAS_5_TIMERMGR0_CFG_OES	0x4B50F00000	0x4B50F00FFFF	4 KB
NAVSS0_VIRT_ALIAS_5_TIMERMGR1_CFG_OES	0x4B50F01000	0x4B50F01FFFF	4 KB
NAVSS0_VIRT_ALIAS_5_ECCAGGR0	0x4B51000000	0x4B510003FF	1 KB
NAVSS0_VIRT_ALIAS_5_UDMASS_ECCAGGR_CFG	0x4B51001000	0x4B510013FF	1 KB
NAVSS0_VIRT_ALIAS_5_VIRTSS_ECCAGGR_CFG	0x4B51002000	0x4B510023FF	1 KB
NAVSS0_VIRT_ALIAS_5_PAT0_CFG_MMRS	0x4B51010000	0x4B510100FF	256 B
NAVSS0_VIRT_ALIAS_5_PAT1_CFG_MMRS	0x4B51011000	0x4B510110FF	256 B
NAVSS0_VIRT_ALIAS_5_PAT2_CFG_MMRS	0x4B51012000	0x4B510120FF	256 B

**Table 2-2. NAVSS0 Alias Memory Map (continued)**

Region Name	Start Address	End Address	Size
NAVSS0_VIRT_ALIAS_5_PAT3_CFG_MMRS	0x4B51013000	0x4B510130FF	256 B
NAVSS0_VIRT_ALIAS_5_PAT4_CFG_MMRS	0x4B51014000	0x4B510140FF	256 B
NAVSS0_VIRT_ALIAS_5_UDMASS_INTA0_CFG_GCNTCFG	0x4B51040000	0x4B51043FFF	16 KB
NAVSS0_VIRT_ALIAS_5_RINGACC0_CFG	0x4B51080000	0x4B510BFFFF	256 KB
NAVSS0_VIRT_ALIAS_5_REGS0_CFG_MMRS	0x4B510C0000	0x4B510C00FF	256 B
NAVSS0_VIRT_ALIAS_5_CPTS0_S_VBUSP_CPTS_VBUSP	0x4B510D0000	0x4B510D03FF	1 KB
NAVSS0_VIRT_ALIAS_5_INTR0_CFG_INTR_ROUTER_CFG	0x4B510E0000	0x4B510E3FFF	16 KB
NAVSS0_VIRT_ALIAS_5_UDMASS_INTA0_CFG_L2G	0x4B51100000	0x4B51100FFF	4 KB
NAVSS0_VIRT_ALIAS_5_UDMASS_INTA0_CFG_MCAST	0x4B51110000	0x4B51113FFF	16 KB
NAVSS0_VIRT_ALIAS_5_PROXY0_CFG_BUF_CFG_GCFG	0x4B51120000	0x4B511200FF	256 B
NAVSS0_VIRT_ALIAS_5_PROXY0_CFG_BUFRAM_SLV_RAM	0x4B51130000	0x4B51133FFF	16 KB
NAVSS0_VIRT_ALIAS_5_SEC_PROXY0_CFG_MMRS	0x4B51140000	0x4B511400FF	256 B
NAVSS0_VIRT_ALIAS_5_UDMAP0_CFG_GCFG	0x4B51150000	0x4B511500FF	256 B
NAVSS0_VIRT_ALIAS_5_RINGACC0_CFG_GCFG	0x4B51160000	0x4B511603FF	1 KB
NAVSS0_VIRT_ALIAS_5_PSILSS_LOCAL_CFG_MMRS	0x4B51170000	0x4B51170FFF	4 KB
NAVSS0_VIRT_ALIAS_5_PSILSS_GLOBAL_CFG_MMRS	0x4B51180000	0x4B51180FFF	4 KB
NAVSS0_VIRT_ALIAS_5_PSILSS_TR_CFG_MMRS	0x4B51190000	0x4B51190FFF	4 KB
NAVSS0_VIRT_ALIAS_5_MCRC0_S_CFG_MCRC64	0x4B51F70000	0x4B51F70FFF	4 KB
NAVSS0_VIRT_ALIAS_5_PSILCFG0_CFG_PROXY	0x4B51F78000	0x4B51F781FF	512 B
NAVSS0_VIRT_ALIAS_5_MAILBOX0_CFG_REGS0	0x4B51F80000	0x4B51F801FF	512 B
NAVSS0_VIRT_ALIAS_5_MAILBOX0_CFG_REGS1	0x4B51F81000	0x4B51F811FF	512 B
NAVSS0_VIRT_ALIAS_5_MAILBOX0_CFG_REGS2	0x4B51F82000	0x4B51F821FF	512 B
NAVSS0_VIRT_ALIAS_5_MAILBOX0_CFG_REGS3	0x4B51F83000	0x4B51F831FF	512 B
NAVSS0_VIRT_ALIAS_5_MAILBOX0_CFG_REGS4	0x4B51F84000	0x4B51F841FF	512 B
NAVSS0_VIRT_ALIAS_5_MAILBOX0_CFG_REGS5	0x4B51F85000	0x4B51F851FF	512 B
NAVSS0_VIRT_ALIAS_5_MAILBOX0_CFG_REGS6	0x4B51F86000	0x4B51F861FF	512 B
NAVSS0_VIRT_ALIAS_5_MAILBOX0_CFG_REGS7	0x4B51F87000	0x4B51F871FF	512 B
NAVSS0_VIRT_ALIAS_5_MAILBOX0_CFG_REGS8	0x4B51F88000	0x4B51F881FF	512 B
NAVSS0_VIRT_ALIAS_5_MAILBOX0_CFG_REGS9	0x4B51F89000	0x4B51F891FF	512 B
NAVSS0_VIRT_ALIAS_5_MAILBOX0_CFG_REGS10	0x4B51F8A000	0x4B51F8A1FF	512 B
NAVSS0_VIRT_ALIAS_5_MAILBOX0_CFG_REGS11	0x4B51F8B000	0x4B51F8B1FF	512 B
NAVSS0_VIRT_ALIAS_5_RINGACC0_CFG_MON	0x4B52000000	0x4B5201FFFF	128 KB
NAVSS0_VIRT_ALIAS_5_TIMERMGR0_CFG_TIMERS	0x4B52200000	0x4B5223FFFF	256 KB
NAVSS0_VIRT_ALIAS_5_TIMERMGR1_CFG_TIMERS	0x4B52240000	0x4B5227FFFF	256 KB
NAVSS0_VIRT_ALIAS_5_SEC_PROXY0_CFG_RT	0x4B52400000	0x4B524FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_5_SEC_PROXY0_CFG_SCFG	0x4B52800000	0x4B528FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_5_SEC_PROXY0_SRC_TARGET_DATA	0x4B52C00000	0x4B52CFFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_5_PROXY0_SRC_TARGET0_DATA	0x4B53000000	0x4B5303FFFF	256 KB
NAVSS0_VIRT_ALIAS_5_PROXY0_CFG_BUF_CFG	0x4B53400000	0x4B5343FFFF	256 KB
NAVSS0_VIRT_ALIAS_5_UDMASS_INTA0_CFG_GCINTRTI	0x4B53800000	0x4B539FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_5_MODSS_INTA0_CFG_INTR	0x4B53C00000	0x4B53C3FFFF	256 KB
NAVSS0_VIRT_ALIAS_5_MODSS_INTA1_CFG_INTR	0x4B53C40000	0x4B53C7FFFF	256 KB
NAVSS0_VIRT_ALIAS_5_UDMASS_INTA0_CFG_INTR	0x4B53D00000	0x4B53DFFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_5_UDMAP0_CFG_RCHANRT	0x4B54000000	0x4B540FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_5_UDMAP0_CFG_TCHANRT	0x4B55000000	0x4B551FFFFFFF	2 MB

**Table 2-2. NAVSS0 Alias Memory Map (continued)**

Region Name	Start Address	End Address	Size
NAVSS0_VIRT_ALIAS_5_PAT0_CFG_SCRATCH	0x4B56200000	0x4B5620FFFF	64 KB
NAVSS0_VIRT_ALIAS_5_PAT1_CFG_SCRATCH	0x4B56210000	0x4B5621FFFF	64 KB
NAVSS0_VIRT_ALIAS_5_PAT2_CFG_SCRATCH	0x4B56220000	0x4B5622FFFF	64 KB
NAVSS0_VIRT_ALIAS_5_PAT3_CFG_SCRATCH	0x4B56230000	0x4B56231FFF	8 KB
NAVSS0_VIRT_ALIAS_5_PAT4_CFG_SCRATCH	0x4B56240000	0x4B56241FFF	8 KB
NAVSS0_VIRT_ALIAS_5_PAT0_CFG_TABLE	0x4B56400000	0x4B5643FFFF	256 KB
NAVSS0_VIRT_ALIAS_5_PAT1_CFG_TABLE	0x4B56440000	0x4B5647FFFF	256 KB
NAVSS0_VIRT_ALIAS_5_PAT2_CFG_TABLE	0x4B56480000	0x4B564BFFFF	256 KB
NAVSS0_VIRT_ALIAS_5_PAT3_CFG_TABLE	0x4B564C0000	0x4B564C7FFF	32 KB
NAVSS0_VIRT_ALIAS_5_PAT4_CFG_TABLE	0x4B56500000	0x4B56507FFF	32 KB
NAVSS0_VIRT_ALIAS_5_TCU_CFG	0x4B56600000	0x4B567FFFFF	2 MB
NAVSS0_VIRT_ALIAS_5_RINGACC0_SRC_FIFOS	0x4B58000000	0x4B583FFFFF	4 MB
NAVSS0_VIRT_ALIAS_5_RINGACC0_CFG_RT	0x4B5C000000	0x4B5C3FFFFF	4 MB
NAVSS0_VIRT_ALIAS_6_MODSS_INTA0_CFG	0x4B60800000	0x4B6080001F	32 B
NAVSS0_VIRT_ALIAS_6_MODSS_INTA1_CFG	0x4B60801000	0x4B6080101F	32 B
NAVSS0_VIRT_ALIAS_6_UDMASS_INTA0_CFG	0x4B60802000	0x4B6080201F	32 B
NAVSS0_VIRT_ALIAS_6_MODSS_INTA0_CFG_IMAP	0x4B60900000	0x4B60907FFF	32 KB
NAVSS0_VIRT_ALIAS_6_MODSS_INTA1_CFG_IMAP	0x4B60908000	0x4B6090FFFF	32 KB
NAVSS0_VIRT_ALIAS_6_UDMASS_INTA0_CFG_IMAP	0x4B60940000	0x4B6097FFFF	256 KB
NAVSS0_VIRT_ALIAS_6_NAV_DDR0_VIRTID_CFG_MMRS	0x4B60A02000	0x4B60A020FF	256 B
NAVSS0_VIRT_ALIAS_6_NAV_DDR1_VIRTID_CFG_MMRS	0x4B60A03000	0x4B60A030FF	256 B
NAVSS0_VIRT_ALIAS_6_UDMAP0_CFG_TCHAN	0x4B60B00000	0x4B60B1FFFF	128 KB
NAVSS0_VIRT_ALIAS_6_UDMAP0_CFG_RCHAN	0x4B60C00000	0x4B60C0FFFF	64 KB
NAVSS0_VIRT_ALIAS_6_UDMAP0_CFG_RFLOW	0x4B60D00000	0x4B60D07FFF	32 KB
NAVSS0_VIRT_ALIAS_6_SPINLOCK0_CFG	0x4B60E00000	0x4B60E07FFF	32 KB
NAVSS0_VIRT_ALIAS_6_TIMERMGR0_CFG	0x4B60E80000	0x4B60E801FF	512 B
NAVSS0_VIRT_ALIAS_6_TIMERMGR1_CFG	0x4B60E81000	0x4B60E811FF	512 B
NAVSS0_VIRT_ALIAS_6_TIMERMGR0_CFG_OES	0x4B60F00000	0x4B60F00FFF	4 KB
NAVSS0_VIRT_ALIAS_6_TIMERMGR1_CFG_OES	0x4B60F01000	0x4B60F01FFF	4 KB
NAVSS0_VIRT_ALIAS_6_ECCAGGR0	0x4B61000000	0x4B610003FF	1 KB
NAVSS0_VIRT_ALIAS_6_UDMASS_ECCAGGR_CFG	0x4B61001000	0x4B610013FF	1 KB
NAVSS0_VIRT_ALIAS_6_VIRTSS_ECCAGGR_CFG	0x4B61002000	0x4B610023FF	1 KB
NAVSS0_VIRT_ALIAS_6_PAT0_CFG_MMRS	0x4B61010000	0x4B610100FF	256 B
NAVSS0_VIRT_ALIAS_6_PAT1_CFG_MMRS	0x4B61011000	0x4B610110FF	256 B
NAVSS0_VIRT_ALIAS_6_PAT2_CFG_MMRS	0x4B61012000	0x4B610120FF	256 B
NAVSS0_VIRT_ALIAS_6_PAT3_CFG_MMRS	0x4B61013000	0x4B610130FF	256 B
NAVSS0_VIRT_ALIAS_6_PAT4_CFG_MMRS	0x4B61014000	0x4B610140FF	256 B
NAVSS0_VIRT_ALIAS_6_UDMASS_INTA0_CFG_GCNTCFG	0x4B61040000	0x4B61043FFF	16 KB
NAVSS0_VIRT_ALIAS_6_RINGACC0_CFG	0x4B61080000	0x4B610BFFFF	256 KB
NAVSS0_VIRT_ALIAS_6_REGS0_CFG_MMRS	0x4B610C0000	0x4B610C00FF	256 B
NAVSS0_VIRT_ALIAS_6_CPTS0_S_VBUSP_CPTS_VBUSP	0x4B610D0000	0x4B610D03FF	1 KB
NAVSS0_VIRT_ALIAS_6_INTR0_CFG_INTR_ROUTER_CFG	0x4B610E0000	0x4B610E3FFF	16 KB
NAVSS0_VIRT_ALIAS_6_UDMASS_INTA0_CFG_L2G	0x4B61100000	0x4B61100FFF	4 KB
NAVSS0_VIRT_ALIAS_6_UDMASS_INTA0_CFG_MCAST	0x4B61110000	0x4B61113FFF	16 KB
NAVSS0_VIRT_ALIAS_6_PROXY0_CFG_BUF_CFG_GCFG	0x4B61120000	0x4B611200FF	256 B

**Table 2-2. NAVSS0 Alias Memory Map (continued)**

Region Name	Start Address	End Address	Size
NAVSS0_VIRT_ALIAS_6_PROXY0_CFG_BUFRAM_SLV_RAM	0x4B61130000	0x4B61133FFF	16 KB
NAVSS0_VIRT_ALIAS_6_SEC_PROXY0_CFG_MMRS	0x4B61140000	0x4B611400FF	256 B
NAVSS0_VIRT_ALIAS_6_UDMAP0_CFG_GCFCG	0x4B61150000	0x4B611500FF	256 B
NAVSS0_VIRT_ALIAS_6_RINGACCO_CFG_GCFCG	0x4B61160000	0x4B611603FF	1 KB
NAVSS0_VIRT_ALIAS_6_PSILSS_LOCAL_CFG_MMRS	0x4B61170000	0x4B61170FFF	4 KB
NAVSS0_VIRT_ALIAS_6_PSILSS_GLOBAL_CFG_MMRS	0x4B61180000	0x4B61180FFF	4 KB
NAVSS0_VIRT_ALIAS_6_PSILSS_TR_CFG_MMRS	0x4B61190000	0x4B61190FFF	4 KB
NAVSS0_VIRT_ALIAS_6_MCRC0_S_CFG_MCRC64	0x4B61F70000	0x4B61F70FFF	4 KB
NAVSS0_VIRT_ALIAS_6_PSILCFG0_CFG_PROXY	0x4B61F78000	0x4B61F781FF	512 B
NAVSS0_VIRT_ALIAS_6_MAILBOX0_CFG_REGS0	0x4B61F80000	0x4B61F801FF	512 B
NAVSS0_VIRT_ALIAS_6_MAILBOX0_CFG_REGS1	0x4B61F81000	0x4B61F811FF	512 B
NAVSS0_VIRT_ALIAS_6_MAILBOX0_CFG_REGS2	0x4B61F82000	0x4B61F821FF	512 B
NAVSS0_VIRT_ALIAS_6_MAILBOX0_CFG_REGS3	0x4B61F83000	0x4B61F831FF	512 B
NAVSS0_VIRT_ALIAS_6_MAILBOX0_CFG_REGS4	0x4B61F84000	0x4B61F841FF	512 B
NAVSS0_VIRT_ALIAS_6_MAILBOX0_CFG_REGS5	0x4B61F85000	0x4B61F851FF	512 B
NAVSS0_VIRT_ALIAS_6_MAILBOX0_CFG_REGS6	0x4B61F86000	0x4B61F861FF	512 B
NAVSS0_VIRT_ALIAS_6_MAILBOX0_CFG_REGS7	0x4B61F87000	0x4B61F871FF	512 B
NAVSS0_VIRT_ALIAS_6_MAILBOX0_CFG_REGS8	0x4B61F88000	0x4B61F881FF	512 B
NAVSS0_VIRT_ALIAS_6_MAILBOX0_CFG_REGS9	0x4B61F89000	0x4B61F891FF	512 B
NAVSS0_VIRT_ALIAS_6_MAILBOX0_CFG_REGS10	0x4B61F8A000	0x4B61F8A1FF	512 B
NAVSS0_VIRT_ALIAS_6_MAILBOX0_CFG_REGS11	0x4B61F8B000	0x4B61F8B1FF	512 B
NAVSS0_VIRT_ALIAS_6_RINGACCO_CFG_MON	0x4B62000000	0x4B6201FFFF	128 KB
NAVSS0_VIRT_ALIAS_6_TIMERMGR0_CFG_TIMERS	0x4B62200000	0x4B6223FFFF	256 KB
NAVSS0_VIRT_ALIAS_6_TIMERMGR1_CFG_TIMERS	0x4B62240000	0x4B6227FFFF	256 KB
NAVSS0_VIRT_ALIAS_6_SEC_PROXY0_CFG_RT	0x4B62400000	0x4B624FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_6_SEC_PROXY0_CFG_SCFCG	0x4B62800000	0x4B628FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_6_SEC_PROXY0_SRC_TARGET_DATA	0x4B62C00000	0x4B62CFFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_6_PROXY0_SRC_TARGET0_DATA	0x4B63000000	0x4B6303FFFF	256 KB
NAVSS0_VIRT_ALIAS_6_PROXY0_CFG_BUF_CFG	0x4B63400000	0x4B6343FFFF	256 KB
NAVSS0_VIRT_ALIAS_6_UDMASS_INTA0_CFG_GCINTRTI	0x4B63800000	0x4B639FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_6_MODSS_INTA0_CFG_INTR	0x4B63C00000	0x4B63C3FFFF	256 KB
NAVSS0_VIRT_ALIAS_6_MODSS_INTA1_CFG_INTR	0x4B63C40000	0x4B63C7FFFF	256 KB
NAVSS0_VIRT_ALIAS_6_UDMASS_INTA0_CFG_INTR	0x4B63D00000	0x4B63DFFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_6_UDMAP0_CFG_RCHANRT	0x4B64000000	0x4B640FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_6_UDMAP0_CFG_TCHANRT	0x4B65000000	0x4B651FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_6_PAT0_CFG_SCRATCH	0x4B66200000	0x4B6620FFFFFF	64 KB
NAVSS0_VIRT_ALIAS_6_PAT1_CFG_SCRATCH	0x4B66210000	0x4B6621FFFF	64 KB
NAVSS0_VIRT_ALIAS_6_PAT2_CFG_SCRATCH	0x4B66220000	0x4B6622FFFF	64 KB
NAVSS0_VIRT_ALIAS_6_PAT3_CFG_SCRATCH	0x4B66230000	0x4B66231FFF	8 KB
NAVSS0_VIRT_ALIAS_6_PAT4_CFG_SCRATCH	0x4B66240000	0x4B66241FFF	8 KB
NAVSS0_VIRT_ALIAS_6_PAT0_CFG_TABLE	0x4B66400000	0x4B6643FFFF	256 KB
NAVSS0_VIRT_ALIAS_6_PAT1_CFG_TABLE	0x4B66440000	0x4B6647FFFF	256 KB
NAVSS0_VIRT_ALIAS_6_PAT2_CFG_TABLE	0x4B66480000	0x4B664BFFFF	256 KB
NAVSS0_VIRT_ALIAS_6_PAT3_CFG_TABLE	0x4B664C0000	0x4B664C7FFF	32 KB
NAVSS0_VIRT_ALIAS_6_PAT4_CFG_TABLE	0x4B66500000	0x4B66507FFF	32 KB



**Table 2-2. NAVSS0 Alias Memory Map (continued)**

Region Name	Start Address	End Address	Size
NAVSS0_VIRT_ALIAS_6_TCU_CFG	0x4B66600000	0x4B667FFFFF	2 MB
NAVSS0_VIRT_ALIAS_6_RINGACC0_SRC_FIFOS	0x4B68000000	0x4B683FFFFF	4 MB
NAVSS0_VIRT_ALIAS_6_RINGACC0_CFG_RT	0x4B6C000000	0x4B6C3FFFFF	4 MB
NAVSS0_VIRT_ALIAS_7_MODSS_INTA0_CFG	0x4B70800000	0x4B7080001F	32 B
NAVSS0_VIRT_ALIAS_7_MODSS_INTA1_CFG	0x4B70801000	0x4B7080101F	32 B
NAVSS0_VIRT_ALIAS_7_UDMASS_INTA0_CFG	0x4B70802000	0x4B7080201F	32 B
NAVSS0_VIRT_ALIAS_7_MODSS_INTA0_CFG_IMAP	0x4B70900000	0x4B70907FFF	32 KB
NAVSS0_VIRT_ALIAS_7_MODSS_INTA1_CFG_IMAP	0x4B70908000	0x4B7090FFFF	32 KB
NAVSS0_VIRT_ALIAS_7_UDMASS_INTA0_CFG_IMAP	0x4B70940000	0x4B7097FFFF	256 KB
NAVSS0_VIRT_ALIAS_7_NAV_DDR0_VIRTID_CFG_MMRS	0x4B70A02000	0x4B70A020FF	256 B
NAVSS0_VIRT_ALIAS_7_NAV_DDR1_VIRTID_CFG_MMRS	0x4B70A03000	0x4B70A030FF	256 B
NAVSS0_VIRT_ALIAS_7_UDMAP0_CFG_TCHAN	0x4B70B00000	0x4B70B1FFFF	128 KB
NAVSS0_VIRT_ALIAS_7_UDMAP0_CFG_RCHAN	0x4B70C00000	0x4B70C0FFFF	64 KB
NAVSS0_VIRT_ALIAS_7_UDMAP0_CFG_RFLOW	0x4B70D00000	0x4B70D07FFF	32 KB
NAVSS0_VIRT_ALIAS_7_SPINLOCK0_CFG	0x4B70E00000	0x4B70E07FFF	32 KB
NAVSS0_VIRT_ALIAS_7_TIMERMGR0_CFG	0x4B70E80000	0x4B70E801FF	512 B
NAVSS0_VIRT_ALIAS_7_TIMERMGR1_CFG	0x4B70E81000	0x4B70E811FF	512 B
NAVSS0_VIRT_ALIAS_7_TIMERMGR0_CFG_OES	0x4B70F00000	0x4B70F00FFF	4 KB
NAVSS0_VIRT_ALIAS_7_TIMERMGR1_CFG_OES	0x4B70F01000	0x4B70F01FFF	4 KB
NAVSS0_VIRT_ALIAS_7_ECCAGGR0	0x4B71000000	0x4B710003FF	1 KB
NAVSS0_VIRT_ALIAS_7_UDMASS_ECCAGGR_CFG	0x4B71001000	0x4B710013FF	1 KB
NAVSS0_VIRT_ALIAS_7_VIRTSS_ECCAGGR_CFG	0x4B71002000	0x4B710023FF	1 KB
NAVSS0_VIRT_ALIAS_7_PAT0_CFG_MMRS	0x4B71010000	0x4B710100FF	256 B
NAVSS0_VIRT_ALIAS_7_PAT1_CFG_MMRS	0x4B71011000	0x4B710110FF	256 B
NAVSS0_VIRT_ALIAS_7_PAT2_CFG_MMRS	0x4B71012000	0x4B710120FF	256 B
NAVSS0_VIRT_ALIAS_7_PAT3_CFG_MMRS	0x4B71013000	0x4B710130FF	256 B
NAVSS0_VIRT_ALIAS_7_PAT4_CFG_MMRS	0x4B71014000	0x4B710140FF	256 B
NAVSS0_VIRT_ALIAS_7_UDMASS_INTA0_CFG_GCNTCFG	0x4B71040000	0x4B71043FFF	16 KB
NAVSS0_VIRT_ALIAS_7_RINGACC0_CFG	0x4B71080000	0x4B710BFFFF	256 KB
NAVSS0_VIRT_ALIAS_7_REGS0_CFG_MMRS	0x4B710C0000	0x4B710C00FF	256 B
NAVSS0_VIRT_ALIAS_7_CPTS0_S_VBUSP_CPTS_VBUSP	0x4B710D0000	0x4B710D03FF	1 KB
NAVSS0_VIRT_ALIAS_7_INTR0_CFG_INTR_ROUTER_CFG	0x4B710E0000	0x4B710E3FFF	16 KB
NAVSS0_VIRT_ALIAS_7_UDMASS_INTA0_CFG_L2G	0x4B71100000	0x4B71100FFF	4 KB
NAVSS0_VIRT_ALIAS_7_UDMASS_INTA0_CFG_MCAST	0x4B71110000	0x4B71113FFF	16 KB
NAVSS0_VIRT_ALIAS_7_PROXY0_CFG_BUF_CFG_GCFG	0x4B71120000	0x4B711200FF	256 B
NAVSS0_VIRT_ALIAS_7_PROXY0_CFG_BUFRAM_SLV_RAM	0x4B71130000	0x4B71133FFF	16 KB
NAVSS0_VIRT_ALIAS_7_SEC_PROXY0_CFG_MMRS	0x4B71140000	0x4B711400FF	256 B
NAVSS0_VIRT_ALIAS_7_UDMAP0_CFG_GCFG	0x4B71150000	0x4B711500FF	256 B
NAVSS0_VIRT_ALIAS_7_RINGACC0_CFG_GCFG	0x4B71160000	0x4B711603FF	1 KB
NAVSS0_VIRT_ALIAS_7_PSILSS_LOCAL_CFG_MMRS	0x4B71170000	0x4B71170FFF	4 KB
NAVSS0_VIRT_ALIAS_7_PSILSS_GLOBAL_CFG_MMRS	0x4B71180000	0x4B71180FFF	4 KB
NAVSS0_VIRT_ALIAS_7_PSILSS_TR_CFG_MMRS	0x4B71190000	0x4B71190FFF	4 KB
NAVSS0_VIRT_ALIAS_7_MCRC0_S_CFG_MCRC64	0x4B71F70000	0x4B71F70FFF	4 KB
NAVSS0_VIRT_ALIAS_7_PSILCFG0_CFG_PROXY	0x4B71F78000	0x4B71F781FF	512 B
NAVSS0_VIRT_ALIAS_7_MAILBOX0_CFG_REGS0	0x4B71F80000	0x4B71F801FF	512 B

**Table 2-2. NAVSS0 Alias Memory Map (continued)**

Region Name	Start Address	End Address	Size
NAVSS0_VIRT_ALIAS_7_MAILBOX0_CFG_REGS1	0x4B71F81000	0x4B71F811FF	512 B
NAVSS0_VIRT_ALIAS_7_MAILBOX0_CFG_REGS2	0x4B71F82000	0x4B71F821FF	512 B
NAVSS0_VIRT_ALIAS_7_MAILBOX0_CFG_REGS3	0x4B71F83000	0x4B71F831FF	512 B
NAVSS0_VIRT_ALIAS_7_MAILBOX0_CFG_REGS4	0x4B71F84000	0x4B71F841FF	512 B
NAVSS0_VIRT_ALIAS_7_MAILBOX0_CFG_REGS5	0x4B71F85000	0x4B71F851FF	512 B
NAVSS0_VIRT_ALIAS_7_MAILBOX0_CFG_REGS6	0x4B71F86000	0x4B71F861FF	512 B
NAVSS0_VIRT_ALIAS_7_MAILBOX0_CFG_REGS7	0x4B71F87000	0x4B71F871FF	512 B
NAVSS0_VIRT_ALIAS_7_MAILBOX0_CFG_REGS8	0x4B71F88000	0x4B71F881FF	512 B
NAVSS0_VIRT_ALIAS_7_MAILBOX0_CFG_REGS9	0x4B71F89000	0x4B71F891FF	512 B
NAVSS0_VIRT_ALIAS_7_MAILBOX0_CFG_REGS10	0x4B71F8A000	0x4B71F8A1FF	512 B
NAVSS0_VIRT_ALIAS_7_MAILBOX0_CFG_REGS11	0x4B71F8B000	0x4B71F8B1FF	512 B
NAVSS0_VIRT_ALIAS_7_RINGACC0_CFG_MON	0x4B72000000	0x4B7201FFFF	128 KB
NAVSS0_VIRT_ALIAS_7_TIMERMGR0_CFG_TIMERS	0x4B72200000	0x4B7223FFFF	256 KB
NAVSS0_VIRT_ALIAS_7_TIMERMGR1_CFG_TIMERS	0x4B72240000	0x4B7227FFFF	256 KB
NAVSS0_VIRT_ALIAS_7_SEC_PROXY0_CFG_RT	0x4B72400000	0x4B724FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_7_SEC_PROXY0_CFG_SCFG	0x4B72800000	0x4B728FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_7_SEC_PROXY0_SRC_TARGET_DATA	0x4B72C00000	0x4B72CFFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_7_PROXY0_SRC_TARGET0_DATA	0x4B73000000	0x4B7303FFFF	256 KB
NAVSS0_VIRT_ALIAS_7_PROXY0_CFG_BUF_CFG	0x4B73400000	0x4B7343FFFF	256 KB
NAVSS0_VIRT_ALIAS_7_UDMASS_INTA0_CFG_GCINTRTI	0x4B73800000	0x4B739FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_7_MODSS_INTA0_CFG_INTR	0x4B73C00000	0x4B73C3FFFF	256 KB
NAVSS0_VIRT_ALIAS_7_MODSS_INTA1_CFG_INTR	0x4B73C40000	0x4B73C7FFFF	256 KB
NAVSS0_VIRT_ALIAS_7_UDMASS_INTA0_CFG_INTR	0x4B73D00000	0x4B73DFFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_7_UDMAP0_CFG_RCHANRT	0x4B74000000	0x4B740FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_7_UDMAP0_CFG_TCHANRT	0x4B75000000	0x4B751FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_7_PAT0_CFG_SCRATCH	0x4B76200000	0x4B7620FFFF	64 KB
NAVSS0_VIRT_ALIAS_7_PAT1_CFG_SCRATCH	0x4B76210000	0x4B7621FFFF	64 KB
NAVSS0_VIRT_ALIAS_7_PAT2_CFG_SCRATCH	0x4B76220000	0x4B7622FFFF	64 KB
NAVSS0_VIRT_ALIAS_7_PAT3_CFG_SCRATCH	0x4B76230000	0x4B76231FFF	8 KB
NAVSS0_VIRT_ALIAS_7_PAT4_CFG_SCRATCH	0x4B76240000	0x4B76241FFF	8 KB
NAVSS0_VIRT_ALIAS_7_PAT0_CFG_TABLE	0x4B76400000	0x4B7643FFFF	256 KB
NAVSS0_VIRT_ALIAS_7_PAT1_CFG_TABLE	0x4B76440000	0x4B7647FFFF	256 KB
NAVSS0_VIRT_ALIAS_7_PAT2_CFG_TABLE	0x4B76480000	0x4B764BFFFF	256 KB
NAVSS0_VIRT_ALIAS_7_PAT3_CFG_TABLE	0x4B764C0000	0x4B764C7FFF	32 KB
NAVSS0_VIRT_ALIAS_7_PAT4_CFG_TABLE	0x4B76500000	0x4B76507FFF	32 KB
NAVSS0_VIRT_ALIAS_7_TCU_CFG	0x4B76600000	0x4B767FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_7_RINGACC0_SRC_FIFOS	0x4B78000000	0x4B783FFFFFFF	4 MB
NAVSS0_VIRT_ALIAS_7_RINGACC0_CFG_RT	0x4B7C000000	0x4B7C3FFFFFFF	4 MB
NAVSS0_VIRT_ALIAS_8_MODSS_INTA0_CFG	0x4B80800000	0x4B8080001F	32 B
NAVSS0_VIRT_ALIAS_8_MODSS_INTA1_CFG	0x4B80801000	0x4B8080101F	32 B
NAVSS0_VIRT_ALIAS_8_UDMASS_INTA0_CFG	0x4B80802000	0x4B8080201F	32 B
NAVSS0_VIRT_ALIAS_8_MODSS_INTA0_CFG_IMAP	0x4B80900000	0x4B80907FFF	32 KB
NAVSS0_VIRT_ALIAS_8_MODSS_INTA1_CFG_IMAP	0x4B80908000	0x4B8090FFFFFF	32 KB
NAVSS0_VIRT_ALIAS_8_UDMASS_INTA0_CFG_IMAP	0x4B80940000	0x4B8097FFFF	256 KB
NAVSS0_VIRT_ALIAS_8_NAV_DDR0_VIRTID_CFG_MMRS	0x4B80A02000	0x4B80A020FF	256 B

**Table 2-2. NAVSS0 Alias Memory Map (continued)**

Region Name	Start Address	End Address	Size
NAVSS0_VIRT_ALIAS_8_NAV_DDR1_VIRTID_CFG_MMRS	0x4B80A03000	0x4B80A030FF	256 B
NAVSS0_VIRT_ALIAS_8_UDMAP0_CFG_TCHAN	0x4B80B00000	0x4B80B1FFFF	128 KB
NAVSS0_VIRT_ALIAS_8_UDMAP0_CFG_RCHAN	0x4B80C00000	0x4B80C0FFFF	64 KB
NAVSS0_VIRT_ALIAS_8_UDMAP0_CFG_RFLOW	0x4B80D00000	0x4B80D07FFF	32 KB
NAVSS0_VIRT_ALIAS_8_SPINLOCK0_CFG	0x4B80E00000	0x4B80E07FFF	32 KB
NAVSS0_VIRT_ALIAS_8_TIMERMGR0_CFG	0x4B80E80000	0x4B80E801FF	512 B
NAVSS0_VIRT_ALIAS_8_TIMERMGR1_CFG	0x4B80E81000	0x4B80E811FF	512 B
NAVSS0_VIRT_ALIAS_8_TIMERMGR0_CFG_OES	0x4B80F00000	0x4B80F00FFF	4 KB
NAVSS0_VIRT_ALIAS_8_TIMERMGR1_CFG_OES	0x4B80F01000	0x4B80F01FFF	4 KB
NAVSS0_VIRT_ALIAS_8_ECCAGGR0	0x4B81000000	0x4B810003FF	1 KB
NAVSS0_VIRT_ALIAS_8_UDMASS_ECCAGGR_CFG	0x4B81001000	0x4B810013FF	1 KB
NAVSS0_VIRT_ALIAS_8_VIRTSS_ECCAGGR_CFG	0x4B81002000	0x4B810023FF	1 KB
NAVSS0_VIRT_ALIAS_8_PAT0_CFG_MMRS	0x4B81010000	0x4B810100FF	256 B
NAVSS0_VIRT_ALIAS_8_PAT1_CFG_MMRS	0x4B81011000	0x4B810110FF	256 B
NAVSS0_VIRT_ALIAS_8_PAT2_CFG_MMRS	0x4B81012000	0x4B810120FF	256 B
NAVSS0_VIRT_ALIAS_8_PAT3_CFG_MMRS	0x4B81013000	0x4B810130FF	256 B
NAVSS0_VIRT_ALIAS_8_PAT4_CFG_MMRS	0x4B81014000	0x4B810140FF	256 B
NAVSS0_VIRT_ALIAS_8_UDMASS_INTA0_CFG_GCNTCFG	0x4B81040000	0x4B81043FFF	16 KB
NAVSS0_VIRT_ALIAS_8_RINGACC0_CFG	0x4B81080000	0x4B810BFFFF	256 KB
NAVSS0_VIRT_ALIAS_8_REGS0_CFG_MMRS	0x4B810C0000	0x4B810C00FF	256 B
NAVSS0_VIRT_ALIAS_8_CPTS0_S_VBUSP_CPTS_VBUSP	0x4B810D0000	0x4B810D03FF	1 KB
NAVSS0_VIRT_ALIAS_8_INTR0_CFG_INTR_ROUTER_CFG	0x4B810E0000	0x4B810E3FFF	16 KB
NAVSS0_VIRT_ALIAS_8_UDMASS_INTA0_CFG_L2G	0x4B81100000	0x4B81100FFF	4 KB
NAVSS0_VIRT_ALIAS_8_UDMASS_INTA0_CFG_MCAST	0x4B81110000	0x4B81113FFF	16 KB
NAVSS0_VIRT_ALIAS_8_PROXY0_CFG_BUF_CFG_GCFG	0x4B81120000	0x4B811200FF	256 B
NAVSS0_VIRT_ALIAS_8_PROXY0_CFG_BUFRAM_SLV_RAM	0x4B81130000	0x4B81133FFF	16 KB
NAVSS0_VIRT_ALIAS_8_SEC_PROXY0_CFG_MMRS	0x4B81140000	0x4B811400FF	256 B
NAVSS0_VIRT_ALIAS_8_UDMAP0_CFG_GCFG	0x4B81150000	0x4B811500FF	256 B
NAVSS0_VIRT_ALIAS_8_RINGACC0_CFG_GCFG	0x4B81160000	0x4B811603FF	1 KB
NAVSS0_VIRT_ALIAS_8_PSILSS_LOCAL_CFG_MMRS	0x4B81170000	0x4B81170FFF	4 KB
NAVSS0_VIRT_ALIAS_8_PSILSS_GLOBAL_CFG_MMRS	0x4B81180000	0x4B81180FFF	4 KB
NAVSS0_VIRT_ALIAS_8_PSILSS_TR_CFG_MMRS	0x4B81190000	0x4B81190FFF	4 KB
NAVSS0_VIRT_ALIAS_8_MCRC0_S_CFG_MCRC64	0x4B81F70000	0x4B81F70FFF	4 KB
NAVSS0_VIRT_ALIAS_8_PSILCFG0_CFG_PROXY	0x4B81F78000	0x4B81F781FF	512 B
NAVSS0_VIRT_ALIAS_8_MAILBOX0_CFG_REGS0	0x4B81F80000	0x4B81F801FF	512 B
NAVSS0_VIRT_ALIAS_8_MAILBOX0_CFG_REGS1	0x4B81F81000	0x4B81F811FF	512 B
NAVSS0_VIRT_ALIAS_8_MAILBOX0_CFG_REGS2	0x4B81F82000	0x4B81F821FF	512 B
NAVSS0_VIRT_ALIAS_8_MAILBOX0_CFG_REGS3	0x4B81F83000	0x4B81F831FF	512 B
NAVSS0_VIRT_ALIAS_8_MAILBOX0_CFG_REGS4	0x4B81F84000	0x4B81F841FF	512 B
NAVSS0_VIRT_ALIAS_8_MAILBOX0_CFG_REGS5	0x4B81F85000	0x4B81F851FF	512 B
NAVSS0_VIRT_ALIAS_8_MAILBOX0_CFG_REGS6	0x4B81F86000	0x4B81F861FF	512 B
NAVSS0_VIRT_ALIAS_8_MAILBOX0_CFG_REGS7	0x4B81F87000	0x4B81F871FF	512 B
NAVSS0_VIRT_ALIAS_8_MAILBOX0_CFG_REGS8	0x4B81F88000	0x4B81F881FF	512 B
NAVSS0_VIRT_ALIAS_8_MAILBOX0_CFG_REGS9	0x4B81F89000	0x4B81F891FF	512 B
NAVSS0_VIRT_ALIAS_8_MAILBOX0_CFG_REGS10	0x4B81F8A000	0x4B81F8A1FF	512 B

**Table 2-2. NAVSS0 Alias Memory Map (continued)**

Region Name	Start Address	End Address	Size
NAVSS0_VIRT_ALIAS_8_MAILBOX0_CFG_REGS11	0x4B81F8B000	0x4B81F8B1FF	512 B
NAVSS0_VIRT_ALIAS_8_RINGACC0_CFG_MON	0x4B82000000	0x4B8201FFFF	128 KB
NAVSS0_VIRT_ALIAS_8_TIMERMGR0_CFG_TIMERS	0x4B82200000	0x4B8223FFFF	256 KB
NAVSS0_VIRT_ALIAS_8_TIMERMGR1_CFG_TIMERS	0x4B82240000	0x4B8227FFFF	256 KB
NAVSS0_VIRT_ALIAS_8_SEC_PROXY0_CFG_RT	0x4B82400000	0x4B824FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_8_SEC_PROXY0_CFG_SCFG	0x4B82800000	0x4B828FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_8_SEC_PROXY0_SRC_TARGET_DATA	0x4B82C00000	0x4B82CFFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_8_PROXY0_SRC_TARGET0_DATA	0x4B83000000	0x4B8303FFFF	256 KB
NAVSS0_VIRT_ALIAS_8_PROXY0_CFG_BUF_CFG	0x4B83400000	0x4B8343FFFF	256 KB
NAVSS0_VIRT_ALIAS_8_UDMASS_INTA0_CFG_GCINTRTI	0x4B83800000	0x4B839FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_8_MODSS_INTA0_CFG_INTR	0x4B83C00000	0x4B83C3FFFF	256 KB
NAVSS0_VIRT_ALIAS_8_MODSS_INTA1_CFG_INTR	0x4B83C40000	0x4B83C7FFFF	256 KB
NAVSS0_VIRT_ALIAS_8_UDMASS_INTA0_CFG_INTR	0x4B83D00000	0x4B83DFFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_8_UDMAP0_CFG_RCHANRT	0x4B84000000	0x4B840FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_8_UDMAP0_CFG_TCHANRT	0x4B85000000	0x4B851FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_8_PAT0_CFG_SCRATCH	0x4B86200000	0x4B8620FFFF	64 KB
NAVSS0_VIRT_ALIAS_8_PAT1_CFG_SCRATCH	0x4B86210000	0x4B8621FFFF	64 KB
NAVSS0_VIRT_ALIAS_8_PAT2_CFG_SCRATCH	0x4B86220000	0x4B8622FFFF	64 KB
NAVSS0_VIRT_ALIAS_8_PAT3_CFG_SCRATCH	0x4B86230000	0x4B86231FFF	8 KB
NAVSS0_VIRT_ALIAS_8_PAT4_CFG_SCRATCH	0x4B86240000	0x4B86241FFF	8 KB
NAVSS0_VIRT_ALIAS_8_PAT0_CFG_TABLE	0x4B86400000	0x4B8643FFFF	256 KB
NAVSS0_VIRT_ALIAS_8_PAT1_CFG_TABLE	0x4B86440000	0x4B8647FFFF	256 KB
NAVSS0_VIRT_ALIAS_8_PAT2_CFG_TABLE	0x4B86480000	0x4B864BFFFF	256 KB
NAVSS0_VIRT_ALIAS_8_PAT3_CFG_TABLE	0x4B864C0000	0x4B864C7FFF	32 KB
NAVSS0_VIRT_ALIAS_8_PAT4_CFG_TABLE	0x4B86500000	0x4B86507FFF	32 KB
NAVSS0_VIRT_ALIAS_8_TCU_CFG	0x4B86600000	0x4B867FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_8_RINGACC0_SRC_FIFOS	0x4B88000000	0x4B883FFFFFFF	4 MB
NAVSS0_VIRT_ALIAS_8_RINGACC0_CFG_RT	0x4B8C000000	0x4B8C3FFFFFFF	4 MB
NAVSS0_VIRT_ALIAS_9_MODSS_INTA0_CFG	0x4B90800000	0x4B9080001F	32 B
NAVSS0_VIRT_ALIAS_9_MODSS_INTA1_CFG	0x4B90801000	0x4B9080101F	32 B
NAVSS0_VIRT_ALIAS_9_UDMASS_INTA0_CFG	0x4B90802000	0x4B9080201F	32 B
NAVSS0_VIRT_ALIAS_9_MODSS_INTA0_CFG_IMAP	0x4B90900000	0x4B90907FFF	32 KB
NAVSS0_VIRT_ALIAS_9_MODSS_INTA1_CFG_IMAP	0x4B90908000	0x4B9090FFFFFF	32 KB
NAVSS0_VIRT_ALIAS_9_UDMASS_INTA0_CFG_IMAP	0x4B90940000	0x4B9097FFFF	256 KB
NAVSS0_VIRT_ALIAS_9_NAV_DDR0_VIRTID_CFG_MMRS	0x4B90A02000	0x4B90A020FF	256 B
NAVSS0_VIRT_ALIAS_9_NAV_DDR1_VIRTID_CFG_MMRS	0x4B90A03000	0x4B90A030FF	256 B
NAVSS0_VIRT_ALIAS_9_UDMAP0_CFG_TCHAN	0x4B90B00000	0x4B90B1FFFF	128 KB
NAVSS0_VIRT_ALIAS_9_UDMAP0_CFG_RCHAN	0x4B90C00000	0x4B90C0FFFF	64 KB
NAVSS0_VIRT_ALIAS_9_UDMAP0_CFG_RFLOW	0x4B90D00000	0x4B90D07FFF	32 KB
NAVSS0_VIRT_ALIAS_9_SPINLOCK0_CFG	0x4B90E00000	0x4B90E07FFF	32 KB
NAVSS0_VIRT_ALIAS_9_TIMERMGR0_CFG	0x4B90E80000	0x4B90E801FF	512 B
NAVSS0_VIRT_ALIAS_9_TIMERMGR1_CFG	0x4B90E81000	0x4B90E811FF	512 B
NAVSS0_VIRT_ALIAS_9_TIMERMGR0_CFG_OES	0x4B90F00000	0x4B90F00FFF	4 KB
NAVSS0_VIRT_ALIAS_9_TIMERMGR1_CFG_OES	0x4B90F01000	0x4B90F01FFF	4 KB
NAVSS0_VIRT_ALIAS_9_ECCAGGR0	0x4B91000000	0x4B910003FF	1 KB

**Table 2-2. NAVSS0 Alias Memory Map (continued)**

Region Name	Start Address	End Address	Size
NAVSS0_VIRT_ALIAS_9_UDMASS_ECCAGGR_CFG	0x4B91001000	0x4B910013FF	1 KB
NAVSS0_VIRT_ALIAS_9_VIRTSS_ECCAGGR_CFG	0x4B91002000	0x4B910023FF	1 KB
NAVSS0_VIRT_ALIAS_9_PAT0_CFG_MMRS	0x4B91010000	0x4B910100FF	256 B
NAVSS0_VIRT_ALIAS_9_PAT1_CFG_MMRS	0x4B91011000	0x4B910110FF	256 B
NAVSS0_VIRT_ALIAS_9_PAT2_CFG_MMRS	0x4B91012000	0x4B910120FF	256 B
NAVSS0_VIRT_ALIAS_9_PAT3_CFG_MMRS	0x4B91013000	0x4B910130FF	256 B
NAVSS0_VIRT_ALIAS_9_PAT4_CFG_MMRS	0x4B91014000	0x4B910140FF	256 B
NAVSS0_VIRT_ALIAS_9_UDMASS_INTA0_CFG_GCNTCFG	0x4B91040000	0x4B91043FFF	16 KB
NAVSS0_VIRT_ALIAS_9_RINGACC0_CFG	0x4B91080000	0x4B910BFFFF	256 KB
NAVSS0_VIRT_ALIAS_9_REGS0_CFG_MMRS	0x4B910C0000	0x4B910C00FF	256 B
NAVSS0_VIRT_ALIAS_9_CPTS0_S_VBUSP_CPTS_VBUSP	0x4B910D0000	0x4B910D03FF	1 KB
NAVSS0_VIRT_ALIAS_9_INTR0_CFG_INTR_ROUTER_CFG	0x4B910E0000	0x4B910E3FFF	16 KB
NAVSS0_VIRT_ALIAS_9_UDMASS_INTA0_CFG_L2G	0x4B91100000	0x4B91100FFF	4 KB
NAVSS0_VIRT_ALIAS_9_UDMASS_INTA0_CFG_MCAST	0x4B91110000	0x4B91113FFF	16 KB
NAVSS0_VIRT_ALIAS_9_PROXY0_CFG_BUF_CFG_GCFG	0x4B91120000	0x4B911200FF	256 B
NAVSS0_VIRT_ALIAS_9_PROXY0_CFG_BUFRAM_SLV_RAM	0x4B91130000	0x4B91133FFF	16 KB
NAVSS0_VIRT_ALIAS_9_SEC_PROXY0_CFG_MMRS	0x4B91140000	0x4B911400FF	256 B
NAVSS0_VIRT_ALIAS_9_UDMAP0_CFG_GCFG	0x4B91150000	0x4B911500FF	256 B
NAVSS0_VIRT_ALIAS_9_RINGACC0_CFG_GCFG	0x4B91160000	0x4B911603FF	1 KB
NAVSS0_VIRT_ALIAS_9_PSILSS_LOCAL_CFG_MMRS	0x4B91170000	0x4B91170FFF	4 KB
NAVSS0_VIRT_ALIAS_9_PSILSS_GLOBAL_CFG_MMRS	0x4B91180000	0x4B91180FFF	4 KB
NAVSS0_VIRT_ALIAS_9_PSILSS_TR_CFG_MMRS	0x4B91190000	0x4B91190FFF	4 KB
NAVSS0_VIRT_ALIAS_9_MCRC0_S_CFG_MCRC64	0x4B91F70000	0x4B91F70FFF	4 KB
NAVSS0_VIRT_ALIAS_9_PSILCFG0_CFG_PROXY	0x4B91F78000	0x4B91F781FF	512 B
NAVSS0_VIRT_ALIAS_9_MAILBOX0_CFG_REGS0	0x4B91F80000	0x4B91F801FF	512 B
NAVSS0_VIRT_ALIAS_9_MAILBOX0_CFG_REGS1	0x4B91F81000	0x4B91F811FF	512 B
NAVSS0_VIRT_ALIAS_9_MAILBOX0_CFG_REGS2	0x4B91F82000	0x4B91F821FF	512 B
NAVSS0_VIRT_ALIAS_9_MAILBOX0_CFG_REGS3	0x4B91F83000	0x4B91F831FF	512 B
NAVSS0_VIRT_ALIAS_9_MAILBOX0_CFG_REGS4	0x4B91F84000	0x4B91F841FF	512 B
NAVSS0_VIRT_ALIAS_9_MAILBOX0_CFG_REGS5	0x4B91F85000	0x4B91F851FF	512 B
NAVSS0_VIRT_ALIAS_9_MAILBOX0_CFG_REGS6	0x4B91F86000	0x4B91F861FF	512 B
NAVSS0_VIRT_ALIAS_9_MAILBOX0_CFG_REGS7	0x4B91F87000	0x4B91F871FF	512 B
NAVSS0_VIRT_ALIAS_9_MAILBOX0_CFG_REGS8	0x4B91F88000	0x4B91F881FF	512 B
NAVSS0_VIRT_ALIAS_9_MAILBOX0_CFG_REGS9	0x4B91F89000	0x4B91F891FF	512 B
NAVSS0_VIRT_ALIAS_9_MAILBOX0_CFG_REGS10	0x4B91F8A000	0x4B91F8A1FF	512 B
NAVSS0_VIRT_ALIAS_9_MAILBOX0_CFG_REGS11	0x4B91F8B000	0x4B91F8B1FF	512 B
NAVSS0_VIRT_ALIAS_9_RINGACC0_CFG_MON	0x4B92000000	0x4B9201FFFF	128 KB
NAVSS0_VIRT_ALIAS_9_TIMERMGR0_CFG_TIMERS	0x4B92200000	0x4B9223FFFF	256 KB
NAVSS0_VIRT_ALIAS_9_TIMERMGR1_CFG_TIMERS	0x4B92240000	0x4B9227FFFF	256 KB
NAVSS0_VIRT_ALIAS_9_SEC_PROXY0_CFG_RT	0x4B92400000	0x4B924FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_9_SEC_PROXY0_CFG_SCFG	0x4B92800000	0x4B928FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_9_SEC_PROXY0_SRC_TARGET_DATA	0x4B92C00000	0x4B92CFFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_9_PROXY0_SRC_TARGET0_DATA	0x4B93000000	0x4B9303FFFFFF	256 KB
NAVSS0_VIRT_ALIAS_9_PROXY0_CFG_BUF_CFG	0x4B93400000	0x4B9343FFFFFF	256 KB
NAVSS0_VIRT_ALIAS_9_UDMASS_INTA0_CFG_GCNTRTI	0x4B93800000	0x4B939FFFFFFF	2 MB



**Table 2-2. NAVSS0 Alias Memory Map (continued)**

Region Name	Start Address	End Address	Size
NAVSS0_VIRT_ALIAS_9_MODSS_INTA0_CFG_INTR	0x4B93C00000	0x4B93C3FFFF	256 KB
NAVSS0_VIRT_ALIAS_9_MODSS_INTA1_CFG_INTR	0x4B93C40000	0x4B93C7FFFF	256 KB
NAVSS0_VIRT_ALIAS_9_UDMASS_INTA0_CFG_INTR	0x4B93D00000	0x4B93DFFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_9_UDMAP0_CFG_RCHANRT	0x4B94000000	0x4B940FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_9_UDMAP0_CFG_TCHANRT	0x4B95000000	0x4B951FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_9_PAT0_CFG_SCRATCH	0x4B96200000	0x4B9620FFFF	64 KB
NAVSS0_VIRT_ALIAS_9_PAT1_CFG_SCRATCH	0x4B96210000	0x4B9621FFFF	64 KB
NAVSS0_VIRT_ALIAS_9_PAT2_CFG_SCRATCH	0x4B96220000	0x4B9622FFFF	64 KB
NAVSS0_VIRT_ALIAS_9_PAT3_CFG_SCRATCH	0x4B96230000	0x4B96231FFF	8 KB
NAVSS0_VIRT_ALIAS_9_PAT4_CFG_SCRATCH	0x4B96240000	0x4B96241FFF	8 KB
NAVSS0_VIRT_ALIAS_9_PAT0_CFG_TABLE	0x4B96400000	0x4B9643FFFF	256 KB
NAVSS0_VIRT_ALIAS_9_PAT1_CFG_TABLE	0x4B96440000	0x4B9647FFFF	256 KB
NAVSS0_VIRT_ALIAS_9_PAT2_CFG_TABLE	0x4B96480000	0x4B964BFFFF	256 KB
NAVSS0_VIRT_ALIAS_9_PAT3_CFG_TABLE	0x4B964C0000	0x4B964C7FFF	32 KB
NAVSS0_VIRT_ALIAS_9_PAT4_CFG_TABLE	0x4B96500000	0x4B96507FFF	32 KB
NAVSS0_VIRT_ALIAS_9_TCU_CFG	0x4B96600000	0x4B967FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_9_RINGACC0_SRC_FIFOS	0x4B98000000	0x4B983FFFFFFF	4 MB
NAVSS0_VIRT_ALIAS_9_RINGACC0_CFG_RT	0x4B9C000000	0x4B9C3FFFFFFF	4 MB
NAVSS0_VIRT_ALIAS_10_MODSS_INTA0_CFG	0x4BA0800000	0x4BA080001F	32 B
NAVSS0_VIRT_ALIAS_10_MODSS_INTA1_CFG	0x4BA0801000	0x4BA080101F	32 B
NAVSS0_VIRT_ALIAS_10_UDMASS_INTA0_CFG	0x4BA0802000	0x4BA080201F	32 B
NAVSS0_VIRT_ALIAS_10_MODSS_INTA0_CFG_IMAP	0x4BA0900000	0x4BA0907FFF	32 KB
NAVSS0_VIRT_ALIAS_10_MODSS_INTA1_CFG_IMAP	0x4BA0908000	0x4BA090FFFFFF	32 KB
NAVSS0_VIRT_ALIAS_10_UDMASS_INTA0_CFG_IMAP	0x4BA0940000	0x4BA097FFFF	256 KB
NAVSS0_VIRT_ALIAS_10_NAV_DDR0_VIRTID_CFG_MMRS	0x4BA0A02000	0x4BA0A020FF	256 B
NAVSS0_VIRT_ALIAS_10_NAV_DDR1_VIRTID_CFG_MMRS	0x4BA0A03000	0x4BA0A030FF	256 B
NAVSS0_VIRT_ALIAS_10_UDMAP0_CFG_TCHAN	0x4BA0B00000	0x4BA0B1FFFF	128 KB
NAVSS0_VIRT_ALIAS_10_UDMAP0_CFG_RCHAN	0x4BA0C00000	0x4BA0C0FFFF	64 KB
NAVSS0_VIRT_ALIAS_10_UDMAP0_CFG_RFLOW	0x4BA0D00000	0x4BA0D07FFF	32 KB
NAVSS0_VIRT_ALIAS_10_SPINLOCK0_CFG	0x4BA0E00000	0x4BA0E07FFF	32 KB
NAVSS0_VIRT_ALIAS_10_TIMERMGR0_CFG	0x4BA0E80000	0x4BA0E801FF	512 B
NAVSS0_VIRT_ALIAS_10_TIMERMGR1_CFG	0x4BA0E81000	0x4BA0E811FF	512 B
NAVSS0_VIRT_ALIAS_10_TIMERMGR0_CFG_OES	0x4BA0F00000	0x4BA0F00FFF	4 KB
NAVSS0_VIRT_ALIAS_10_TIMERMGR1_CFG_OES	0x4BA0F01000	0x4BA0F01FFF	4 KB
NAVSS0_VIRT_ALIAS_10_ECCAGGR0	0x4BA1000000	0x4BA10003FF	1 KB
NAVSS0_VIRT_ALIAS_10_UDMASS_ECCAGGR_CFG	0x4BA1001000	0x4BA10013FF	1 KB
NAVSS0_VIRT_ALIAS_10_VIRTSS_ECCAGGR_CFG	0x4BA1002000	0x4BA10023FF	1 KB
NAVSS0_VIRT_ALIAS_10_PAT0_CFG_MMRS	0x4BA1010000	0x4BA10100FF	256 B
NAVSS0_VIRT_ALIAS_10_PAT1_CFG_MMRS	0x4BA1011000	0x4BA10110FF	256 B
NAVSS0_VIRT_ALIAS_10_PAT2_CFG_MMRS	0x4BA1012000	0x4BA10120FF	256 B
NAVSS0_VIRT_ALIAS_10_PAT3_CFG_MMRS	0x4BA1013000	0x4BA10130FF	256 B
NAVSS0_VIRT_ALIAS_10_PAT4_CFG_MMRS	0x4BA1014000	0x4BA10140FF	256 B
NAVSS0_VIRT_ALIAS_10_UDMASS_INTA0_CFG_GCNTCFG	0x4BA1040000	0x4BA1043FFF	16 KB
NAVSS0_VIRT_ALIAS_10_RINGACC0_CFG	0x4BA1080000	0x4BA10BFFFFFF	256 KB
NAVSS0_VIRT_ALIAS_10_REGS0_CFG_MMRS	0x4BA10C0000	0x4BA10C00FF	256 B

**Table 2-2. NAVSS0 Alias Memory Map (continued)**

Region Name	Start Address	End Address	Size
NAVSS0_VIRT_ALIAS_10_CPTS0_S_VBUSP_CPTS_VBUSP	0x4BA10D0000	0x4BA10D03FF	1 KB
NAVSS0_VIRT_ALIAS_10_INTR0_CFG_INTR_ROUTER_CFG	0x4BA10E0000	0x4BA10E3FFF	16 KB
NAVSS0_VIRT_ALIAS_10_UDMASS_INTA0_CFG_L2G	0x4BA1100000	0x4BA1100FFF	4 KB
NAVSS0_VIRT_ALIAS_10_UDMASS_INTA0_CFG_MCAST	0x4BA1110000	0x4BA1113FFF	16 KB
NAVSS0_VIRT_ALIAS_10_PROXY0_CFG_BUF_CFG_GCFG	0x4BA1120000	0x4BA11200FF	256 B
NAVSS0_VIRT_ALIAS_10_PROXY0_CFG_BUFRAM_SLV_RAM	0x4BA1130000	0x4BA1133FFF	16 KB
NAVSS0_VIRT_ALIAS_10_SEC_PROXY0_CFG_MMRS	0x4BA1140000	0x4BA11400FF	256 B
NAVSS0_VIRT_ALIAS_10_UDMAP0_CFG_GCFG	0x4BA1150000	0x4BA11500FF	256 B
NAVSS0_VIRT_ALIAS_10_RINGACC0_CFG_GCFG	0x4BA1160000	0x4BA11603FF	1 KB
NAVSS0_VIRT_ALIAS_10_PSILSS_LOCAL_CFG_MMRS	0x4BA1170000	0x4BA1170FFF	4 KB
NAVSS0_VIRT_ALIAS_10_PSILSS_GLOBAL_CFG_MMRS	0x4BA1180000	0x4BA1180FFF	4 KB
NAVSS0_VIRT_ALIAS_10_PSILSS_TR_CFG_MMRS	0x4BA1190000	0x4BA1190FFF	4 KB
NAVSS0_VIRT_ALIAS_10_MCRC0_S_CFG_MCRC64	0x4BA1F70000	0x4BA1F70FFF	4 KB
NAVSS0_VIRT_ALIAS_10_PSILCFG0_CFG_PROXY	0x4BA1F78000	0x4BA1F781FF	512 B
NAVSS0_VIRT_ALIAS_10_MAILBOX0_CFG_REGS0	0x4BA1F80000	0x4BA1F801FF	512 B
NAVSS0_VIRT_ALIAS_10_MAILBOX0_CFG_REGS1	0x4BA1F81000	0x4BA1F811FF	512 B
NAVSS0_VIRT_ALIAS_10_MAILBOX0_CFG_REGS2	0x4BA1F82000	0x4BA1F821FF	512 B
NAVSS0_VIRT_ALIAS_10_MAILBOX0_CFG_REGS3	0x4BA1F83000	0x4BA1F831FF	512 B
NAVSS0_VIRT_ALIAS_10_MAILBOX0_CFG_REGS4	0x4BA1F84000	0x4BA1F841FF	512 B
NAVSS0_VIRT_ALIAS_10_MAILBOX0_CFG_REGS5	0x4BA1F85000	0x4BA1F851FF	512 B
NAVSS0_VIRT_ALIAS_10_MAILBOX0_CFG_REGS6	0x4BA1F86000	0x4BA1F861FF	512 B
NAVSS0_VIRT_ALIAS_10_MAILBOX0_CFG_REGS7	0x4BA1F87000	0x4BA1F871FF	512 B
NAVSS0_VIRT_ALIAS_10_MAILBOX0_CFG_REGS8	0x4BA1F88000	0x4BA1F881FF	512 B
NAVSS0_VIRT_ALIAS_10_MAILBOX0_CFG_REGS9	0x4BA1F89000	0x4BA1F891FF	512 B
NAVSS0_VIRT_ALIAS_10_MAILBOX0_CFG_REGS10	0x4BA1F8A000	0x4BA1F8A1FF	512 B
NAVSS0_VIRT_ALIAS_10_MAILBOX0_CFG_REGS11	0x4BA1F8B000	0x4BA1F8B1FF	512 B
NAVSS0_VIRT_ALIAS_10_RINGACC0_CFG_MON	0x4BA2000000	0x4BA201FFFF	128 KB
NAVSS0_VIRT_ALIAS_10_TIMERMGR0_CFG_TIMERS	0x4BA2200000	0x4BA223FFFF	256 KB
NAVSS0_VIRT_ALIAS_10_TIMERMGR1_CFG_TIMERS	0x4BA2240000	0x4BA227FFFF	256 KB
NAVSS0_VIRT_ALIAS_10_SEC_PROXY0_CFG_RT	0x4BA2400000	0x4BA24FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_10_SEC_PROXY0_CFG_SCFG	0x4BA2800000	0x4BA28FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_10_SEC_PROXY0_SRC_TARGET_DATA	0x4BA2C00000	0x4BA2CFFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_10_PROXY0_SRC_TARGET0_DATA	0x4BA3000000	0x4BA303FFFF	256 KB
NAVSS0_VIRT_ALIAS_10_PROXY0_CFG_BUF_CFG	0x4BA3400000	0x4BA343FFFF	256 KB
NAVSS0_VIRT_ALIAS_10_UDMASS_INTA0_CFG_GCINTRTI	0x4BA3800000	0x4BA39FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_10_MODSS_INTA0_CFG_INTR	0x4BA3C00000	0x4BA3C3FFFF	256 KB
NAVSS0_VIRT_ALIAS_10_MODSS_INTA1_CFG_INTR	0x4BA3C40000	0x4BA3C7FFFF	256 KB
NAVSS0_VIRT_ALIAS_10_UDMASS_INTA0_CFG_INTR	0x4BA3D00000	0x4BA3DFFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_10_UDMAP0_CFG_RCHANRT	0x4BA4000000	0x4BA40FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_10_UDMAP0_CFG_TCHANRT	0x4BA5000000	0x4BA51FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_10_PAT0_CFG_SCRATCH	0x4BA6200000	0x4BA620FFFF	64 KB
NAVSS0_VIRT_ALIAS_10_PAT1_CFG_SCRATCH	0x4BA6210000	0x4BA621FFFF	64 KB
NAVSS0_VIRT_ALIAS_10_PAT2_CFG_SCRATCH	0x4BA6220000	0x4BA622FFFF	64 KB
NAVSS0_VIRT_ALIAS_10_PAT3_CFG_SCRATCH	0x4BA6230000	0x4BA6231FFF	8 KB
NAVSS0_VIRT_ALIAS_10_PAT4_CFG_SCRATCH	0x4BA6240000	0x4BA6241FFF	8 KB

**Table 2-2. NAVSS0 Alias Memory Map (continued)**

Region Name	Start Address	End Address	Size
NAVSS0_VIRT_ALIAS_10_PAT0_CFG_TABLE	0x4BA6400000	0x4BA643FFFF	256 KB
NAVSS0_VIRT_ALIAS_10_PAT1_CFG_TABLE	0x4BA6440000	0x4BA647FFFF	256 KB
NAVSS0_VIRT_ALIAS_10_PAT2_CFG_TABLE	0x4BA6480000	0x4BA64BFFFF	256 KB
NAVSS0_VIRT_ALIAS_10_PAT3_CFG_TABLE	0x4BA64C0000	0x4BA64C7FFF	32 KB
NAVSS0_VIRT_ALIAS_10_PAT4_CFG_TABLE	0x4BA6500000	0x4BA6507FFF	32 KB
NAVSS0_VIRT_ALIAS_10_TCU_CFG	0x4BA6600000	0x4BA67FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_10_RINGACC0_SRC_FIFOS	0x4BA8000000	0x4BA83FFFFFFF	4 MB
NAVSS0_VIRT_ALIAS_10_RINGACC0_CFG_RT	0x4BAC000000	0x4BAC3FFFFFFF	4 MB
NAVSS0_VIRT_ALIAS_11_MODSS_INTA0_CFG	0x4BB0800000	0x4BB080001F	32 B
NAVSS0_VIRT_ALIAS_11_MODSS_INTA1_CFG	0x4BB0801000	0x4BB080101F	32 B
NAVSS0_VIRT_ALIAS_11_UDMASS_INTA0_CFG	0x4BB0802000	0x4BB080201F	32 B
NAVSS0_VIRT_ALIAS_11_MODSS_INTA0_CFG_IMAP	0x4BB0900000	0x4BB0907FFF	32 KB
NAVSS0_VIRT_ALIAS_11_MODSS_INTA1_CFG_IMAP	0x4BB0908000	0x4BB090FFFFFF	32 KB
NAVSS0_VIRT_ALIAS_11_UDMASS_INTA0_CFG_IMAP	0x4BB0940000	0x4BB097FFFFFF	256 KB
NAVSS0_VIRT_ALIAS_11_NAV_DDR0_VIRTID_CFG_MMRS	0x4BB0A02000	0x4BB0A020FF	256 B
NAVSS0_VIRT_ALIAS_11_NAV_DDR1_VIRTID_CFG_MMRS	0x4BB0A03000	0x4BB0A030FF	256 B
NAVSS0_VIRT_ALIAS_11_UDMAP0_CFG_TCHAN	0x4BB0B00000	0x4BB0B1FFFF	128 KB
NAVSS0_VIRT_ALIAS_11_UDMAP0_CFG_RCHAN	0x4BB0C00000	0x4BB0C0FFFF	64 KB
NAVSS0_VIRT_ALIAS_11_UDMAP0_CFG_RFLOW	0x4BB0D00000	0x4BB0D07FFF	32 KB
NAVSS0_VIRT_ALIAS_11_SPINLOCK0_CFG	0x4BB0E00000	0x4BB0E07FFF	32 KB
NAVSS0_VIRT_ALIAS_11_TIMERMGR0_CFG	0x4BB0E80000	0x4BB0E801FF	512 B
NAVSS0_VIRT_ALIAS_11_TIMERMGR1_CFG	0x4BB0E81000	0x4BB0E811FF	512 B
NAVSS0_VIRT_ALIAS_11_TIMERMGR0_CFG_OES	0x4BB0F00000	0x4BB0F00FFF	4 KB
NAVSS0_VIRT_ALIAS_11_TIMERMGR1_CFG_OES	0x4BB0F01000	0x4BB0F01FFF	4 KB
NAVSS0_VIRT_ALIAS_11_ECCAGGR0	0x4BB1000000	0x4BB10003FF	1 KB
NAVSS0_VIRT_ALIAS_11_UDMASS_ECCAGGR_CFG	0x4BB1001000	0x4BB10013FF	1 KB
NAVSS0_VIRT_ALIAS_11_VIRTSS_ECCAGGR_CFG	0x4BB1002000	0x4BB10023FF	1 KB
NAVSS0_VIRT_ALIAS_11_PAT0_CFG_MMRS	0x4BB1010000	0x4BB10100FF	256 B
NAVSS0_VIRT_ALIAS_11_PAT1_CFG_MMRS	0x4BB1011000	0x4BB10110FF	256 B
NAVSS0_VIRT_ALIAS_11_PAT2_CFG_MMRS	0x4BB1012000	0x4BB10120FF	256 B
NAVSS0_VIRT_ALIAS_11_PAT3_CFG_MMRS	0x4BB1013000	0x4BB10130FF	256 B
NAVSS0_VIRT_ALIAS_11_PAT4_CFG_MMRS	0x4BB1014000	0x4BB10140FF	256 B
NAVSS0_VIRT_ALIAS_11_UDMASS_INTA0_CFG_GCNTCFG	0x4BB1040000	0x4BB1043FFF	16 KB
NAVSS0_VIRT_ALIAS_11_RINGACC0_CFG	0x4BB1080000	0x4BB10BFFFF	256 KB
NAVSS0_VIRT_ALIAS_11_REGS0_CFG_MMRS	0x4BB10C0000	0x4BB10C00FF	256 B
NAVSS0_VIRT_ALIAS_11_CPTS0_S_VBUSP_CPTS_VBUSP	0x4BB10D0000	0x4BB10D03FF	1 KB
NAVSS0_VIRT_ALIAS_11_INTR0_CFG_INTR_ROUTER_CFG	0x4BB10E0000	0x4BB10E3FFF	16 KB
NAVSS0_VIRT_ALIAS_11_UDMASS_INTA0_CFG_L2G	0x4BB1100000	0x4BB1100FFF	4 KB
NAVSS0_VIRT_ALIAS_11_UDMASS_INTA0_CFG_MCAST	0x4BB1110000	0x4BB1113FFF	16 KB
NAVSS0_VIRT_ALIAS_11_PROXY0_CFG_BUF_CFG_GCFG	0x4BB1120000	0x4BB11200FF	256 B
NAVSS0_VIRT_ALIAS_11_PROXY0_CFG_BUFRAM_SLV_RAM	0x4BB1130000	0x4BB1133FFF	16 KB
NAVSS0_VIRT_ALIAS_11_SEC_PROXY0_CFG_MMRS	0x4BB1140000	0x4BB11400FF	256 B
NAVSS0_VIRT_ALIAS_11_UDMAP0_CFG_GCFG	0x4BB1150000	0x4BB11500FF	256 B
NAVSS0_VIRT_ALIAS_11_RINGACC0_CFG_GCFG	0x4BB1160000	0x4BB11603FF	1 KB
NAVSS0_VIRT_ALIAS_11_PSILSS_LOCAL_CFG_MMRS	0x4BB1170000	0x4BB1170FFF	4 KB

**Table 2-2. NAVSS0 Alias Memory Map (continued)**

Region Name	Start Address	End Address	Size
NAVSS0_VIRT_ALIAS_11_PSILSS_GLOBAL_CFG_MMRS	0x4BB1180000	0x4BB1180FFF	4 KB
NAVSS0_VIRT_ALIAS_11_PSILSS_TR_CFG_MMRS	0x4BB1190000	0x4BB1190FFF	4 KB
NAVSS0_VIRT_ALIAS_11_MCRC0_S_CFG_MCRC64	0x4BB1F70000	0x4BB1F70FFF	4 KB
NAVSS0_VIRT_ALIAS_11_PSILCFG0_CFG_PROXY	0x4BB1F78000	0x4BB1F781FF	512 B
NAVSS0_VIRT_ALIAS_11_MAILBOX0_CFG_REGS0	0x4BB1F80000	0x4BB1F801FF	512 B
NAVSS0_VIRT_ALIAS_11_MAILBOX0_CFG_REGS1	0x4BB1F81000	0x4BB1F811FF	512 B
NAVSS0_VIRT_ALIAS_11_MAILBOX0_CFG_REGS2	0x4BB1F82000	0x4BB1F821FF	512 B
NAVSS0_VIRT_ALIAS_11_MAILBOX0_CFG_REGS3	0x4BB1F83000	0x4BB1F831FF	512 B
NAVSS0_VIRT_ALIAS_11_MAILBOX0_CFG_REGS4	0x4BB1F84000	0x4BB1F841FF	512 B
NAVSS0_VIRT_ALIAS_11_MAILBOX0_CFG_REGS5	0x4BB1F85000	0x4BB1F851FF	512 B
NAVSS0_VIRT_ALIAS_11_MAILBOX0_CFG_REGS6	0x4BB1F86000	0x4BB1F861FF	512 B
NAVSS0_VIRT_ALIAS_11_MAILBOX0_CFG_REGS7	0x4BB1F87000	0x4BB1F871FF	512 B
NAVSS0_VIRT_ALIAS_11_MAILBOX0_CFG_REGS8	0x4BB1F88000	0x4BB1F881FF	512 B
NAVSS0_VIRT_ALIAS_11_MAILBOX0_CFG_REGS9	0x4BB1F89000	0x4BB1F891FF	512 B
NAVSS0_VIRT_ALIAS_11_MAILBOX0_CFG_REGS10	0x4BB1F8A000	0x4BB1F8A1FF	512 B
NAVSS0_VIRT_ALIAS_11_MAILBOX0_CFG_REGS11	0x4BB1F8B000	0x4BB1F8B1FF	512 B
NAVSS0_VIRT_ALIAS_11_RINGACC0_CFG_MON	0x4BB2000000	0x4BB201FFFF	128 KB
NAVSS0_VIRT_ALIAS_11_TIMERMGR0_CFG_TIMERS	0x4BB2200000	0x4BB223FFFF	256 KB
NAVSS0_VIRT_ALIAS_11_TIMERMGR1_CFG_TIMERS	0x4BB2240000	0x4BB227FFFF	256 KB
NAVSS0_VIRT_ALIAS_11_SEC_PROXY0_CFG_RT	0x4BB2400000	0x4BB24FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_11_SEC_PROXY0_CFG_SCFG	0x4BB2800000	0x4BB28FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_11_SEC_PROXY0_SRC_TARGET_DATA	0x4BB2C00000	0x4BB2CFFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_11_PROXY0_SRC_TARGET0_DATA	0x4BB3000000	0x4BB303FFFF	256 KB
NAVSS0_VIRT_ALIAS_11_PROXY0_CFG_BUF_CFG	0x4BB3400000	0x4BB343FFFF	256 KB
NAVSS0_VIRT_ALIAS_11_UDMASS_INTA0_CFG_GCINTRTI	0x4BB3800000	0x4BB39FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_11_MODSS_INTA0_CFG_INTR	0x4BB3C00000	0x4BB3C3FFFF	256 KB
NAVSS0_VIRT_ALIAS_11_MODSS_INTA1_CFG_INTR	0x4BB3C40000	0x4BB3C7FFFF	256 KB
NAVSS0_VIRT_ALIAS_11_UDMASS_INTA0_CFG_INTR	0x4BB3D00000	0x4BB3DFFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_11_UDMAP0_CFG_RCHANRT	0x4BB4000000	0x4BB40FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_11_UDMAP0_CFG_TCHANRT	0x4BB5000000	0x4BB51FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_11_PAT0_CFG_SCRATCH	0x4BB6200000	0x4BB620FFFF	64 KB
NAVSS0_VIRT_ALIAS_11_PAT1_CFG_SCRATCH	0x4BB6210000	0x4BB621FFFF	64 KB
NAVSS0_VIRT_ALIAS_11_PAT2_CFG_SCRATCH	0x4BB6220000	0x4BB622FFFF	64 KB
NAVSS0_VIRT_ALIAS_11_PAT3_CFG_SCRATCH	0x4BB6230000	0x4BB6231FFF	8 KB
NAVSS0_VIRT_ALIAS_11_PAT4_CFG_SCRATCH	0x4BB6240000	0x4BB6241FFF	8 KB
NAVSS0_VIRT_ALIAS_11_PAT0_CFG_TABLE	0x4BB6400000	0x4BB643FFFF	256 KB
NAVSS0_VIRT_ALIAS_11_PAT1_CFG_TABLE	0x4BB6440000	0x4BB647FFFF	256 KB
NAVSS0_VIRT_ALIAS_11_PAT2_CFG_TABLE	0x4BB6480000	0x4BB64BFFFF	256 KB
NAVSS0_VIRT_ALIAS_11_PAT3_CFG_TABLE	0x4BB64C0000	0x4BB64C7FFF	32 KB
NAVSS0_VIRT_ALIAS_11_PAT4_CFG_TABLE	0x4BB6500000	0x4BB6507FFF	32 KB
NAVSS0_VIRT_ALIAS_11_TCU_CFG	0x4BB6600000	0x4BB67FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_11_RINGACC0_SRC_FIFOS	0x4BB8000000	0x4BB83FFFFFFF	4 MB
NAVSS0_VIRT_ALIAS_11_RINGACC0_CFG_RT	0x4BBC000000	0x4BBC3FFFFFFF	4 MB
NAVSS0_VIRT_ALIAS_12_MODSS_INTA0_CFG	0x4BC0800000	0x4BC080001F	32 B
NAVSS0_VIRT_ALIAS_12_MODSS_INTA1_CFG	0x4BC0801000	0x4BC080101F	32 B

**Table 2-2. NAVSS0 Alias Memory Map (continued)**

Region Name	Start Address	End Address	Size
NAVSS0_VIRT_ALIAS_12_UDMASS_INTA0_CFG	0x4BC0802000	0x4BC080201F	32 B
NAVSS0_VIRT_ALIAS_12_MODSS_INTA0_CFG_IMAP	0x4BC0900000	0x4BC0907FFF	32 KB
NAVSS0_VIRT_ALIAS_12_MODSS_INTA1_CFG_IMAP	0x4BC0908000	0x4BC090FFFF	32 KB
NAVSS0_VIRT_ALIAS_12_UDMASS_INTA0_CFG_IMAP	0x4BC0940000	0x4BC097FFFF	256 KB
NAVSS0_VIRT_ALIAS_12_NAV_DDR0_VIRTID_CFG_MMRS	0x4BC0A02000	0x4BC0A020FF	256 B
NAVSS0_VIRT_ALIAS_12_NAV_DDR1_VIRTID_CFG_MMRS	0x4BC0A03000	0x4BC0A030FF	256 B
NAVSS0_VIRT_ALIAS_12_UDMAP0_CFG_TCHAN	0x4BC0B00000	0x4BC0B1FFFF	128 KB
NAVSS0_VIRT_ALIAS_12_UDMAP0_CFG_RCHAN	0x4BC0C00000	0x4BC0C0FFFF	64 KB
NAVSS0_VIRT_ALIAS_12_UDMAP0_CFG_RFLOW	0x4BC0D00000	0x4BC0D07FFF	32 KB
NAVSS0_VIRT_ALIAS_12_SPINLOCK0_CFG	0x4BC0E00000	0x4BC0E07FFF	32 KB
NAVSS0_VIRT_ALIAS_12_TIMERMGR0_CFG	0x4BC0E80000	0x4BC0E801FF	512 B
NAVSS0_VIRT_ALIAS_12_TIMERMGR1_CFG	0x4BC0E81000	0x4BC0E811FF	512 B
NAVSS0_VIRT_ALIAS_12_TIMERMGR0_CFG_OES	0x4BC0F00000	0x4BC0F00FFF	4 KB
NAVSS0_VIRT_ALIAS_12_TIMERMGR1_CFG_OES	0x4BC0F01000	0x4BC0F01FFF	4 KB
NAVSS0_VIRT_ALIAS_12_ECCAGGR0	0x4BC1000000	0x4BC10003FF	1 KB
NAVSS0_VIRT_ALIAS_12_UDMASS_ECCAGGR_CFG	0x4BC1001000	0x4BC10013FF	1 KB
NAVSS0_VIRT_ALIAS_12_VIRTSS_ECCAGGR_CFG	0x4BC1002000	0x4BC10023FF	1 KB
NAVSS0_VIRT_ALIAS_12_PAT0_CFG_MMRS	0x4BC1010000	0x4BC10100FF	256 B
NAVSS0_VIRT_ALIAS_12_PAT1_CFG_MMRS	0x4BC1011000	0x4BC10110FF	256 B
NAVSS0_VIRT_ALIAS_12_PAT2_CFG_MMRS	0x4BC1012000	0x4BC10120FF	256 B
NAVSS0_VIRT_ALIAS_12_PAT3_CFG_MMRS	0x4BC1013000	0x4BC10130FF	256 B
NAVSS0_VIRT_ALIAS_12_PAT4_CFG_MMRS	0x4BC1014000	0x4BC10140FF	256 B
NAVSS0_VIRT_ALIAS_12_UDMASS_INTA0_CFG_GCNTCFG	0x4BC1040000	0x4BC1043FFF	16 KB
NAVSS0_VIRT_ALIAS_12_RINGACC0_CFG	0x4BC1080000	0x4BC10BFFFF	256 KB
NAVSS0_VIRT_ALIAS_12_REG0_CFG_MMRS	0x4BC10C0000	0x4BC10C00FF	256 B
NAVSS0_VIRT_ALIAS_12_CPTS0_S_VBUSP_CPTS_VBUSP	0x4BC10D0000	0x4BC10D03FF	1 KB
NAVSS0_VIRT_ALIAS_12_INTR0_CFG_INTR_ROUTER_CFG	0x4BC10E0000	0x4BC10E3FFF	16 KB
NAVSS0_VIRT_ALIAS_12_UDMASS_INTA0_CFG_L2G	0x4BC1100000	0x4BC1100FFF	4 KB
NAVSS0_VIRT_ALIAS_12_UDMASS_INTA0_CFG_MCAST	0x4BC1110000	0x4BC1113FFF	16 KB
NAVSS0_VIRT_ALIAS_12_PROXY0_CFG_BUF_CFG_GCFG	0x4BC1120000	0x4BC11200FF	256 B
NAVSS0_VIRT_ALIAS_12_PROXY0_CFG_BUFRAM_SLV_RAM	0x4BC1130000	0x4BC1133FFF	16 KB
NAVSS0_VIRT_ALIAS_12_SEC_PROXY0_CFG_MMRS	0x4BC1140000	0x4BC11400FF	256 B
NAVSS0_VIRT_ALIAS_12_UDMAP0_CFG_GCFG	0x4BC1150000	0x4BC11500FF	256 B
NAVSS0_VIRT_ALIAS_12_RINGACC0_CFG_GCFG	0x4BC1160000	0x4BC11603FF	1 KB
NAVSS0_VIRT_ALIAS_12_PSILSS_LOCAL_CFG_MMRS	0x4BC1170000	0x4BC1170FFF	4 KB
NAVSS0_VIRT_ALIAS_12_PSILSS_GLOBAL_CFG_MMRS	0x4BC1180000	0x4BC1180FFF	4 KB
NAVSS0_VIRT_ALIAS_12_PSILSS_TR_CFG_MMRS	0x4BC1190000	0x4BC1190FFF	4 KB
NAVSS0_VIRT_ALIAS_12_MCRC0_S_CFG_MCRC64	0x4BC1F70000	0x4BC1F70FFF	4 KB
NAVSS0_VIRT_ALIAS_12_PSILCFG0_CFG_PROXY	0x4BC1F78000	0x4BC1F781FF	512 B
NAVSS0_VIRT_ALIAS_12_MAILBOX0_CFG_REGS0	0x4BC1F80000	0x4BC1F801FF	512 B
NAVSS0_VIRT_ALIAS_12_MAILBOX0_CFG_REGS1	0x4BC1F81000	0x4BC1F811FF	512 B
NAVSS0_VIRT_ALIAS_12_MAILBOX0_CFG_REGS2	0x4BC1F82000	0x4BC1F821FF	512 B
NAVSS0_VIRT_ALIAS_12_MAILBOX0_CFG_REGS3	0x4BC1F83000	0x4BC1F831FF	512 B
NAVSS0_VIRT_ALIAS_12_MAILBOX0_CFG_REGS4	0x4BC1F84000	0x4BC1F841FF	512 B
NAVSS0_VIRT_ALIAS_12_MAILBOX0_CFG_REGS5	0x4BC1F85000	0x4BC1F851FF	512 B



**Table 2-2. NAVSS0 Alias Memory Map (continued)**

Region Name	Start Address	End Address	Size
NAVSS0_VIRT_ALIAS_12_MAILBOX0_CFG_REGS6	0x4BC1F86000	0x4BC1F861FF	512 B
NAVSS0_VIRT_ALIAS_12_MAILBOX0_CFG_REGS7	0x4BC1F87000	0x4BC1F871FF	512 B
NAVSS0_VIRT_ALIAS_12_MAILBOX0_CFG_REGS8	0x4BC1F88000	0x4BC1F881FF	512 B
NAVSS0_VIRT_ALIAS_12_MAILBOX0_CFG_REGS9	0x4BC1F89000	0x4BC1F891FF	512 B
NAVSS0_VIRT_ALIAS_12_MAILBOX0_CFG_REGS10	0x4BC1F8A000	0x4BC1F8A1FF	512 B
NAVSS0_VIRT_ALIAS_12_MAILBOX0_CFG_REGS11	0x4BC1F8B000	0x4BC1F8B1FF	512 B
NAVSS0_VIRT_ALIAS_12_RINGACC0_CFG_MON	0x4BC2000000	0x4BC201FFFF	128 KB
NAVSS0_VIRT_ALIAS_12_TIMERMGR0_CFG_TIMERS	0x4BC2200000	0x4BC223FFFF	256 KB
NAVSS0_VIRT_ALIAS_12_TIMERMGR1_CFG_TIMERS	0x4BC2240000	0x4BC227FFFF	256 KB
NAVSS0_VIRT_ALIAS_12_SEC_PROXY0_CFG_RT	0x4BC2400000	0x4BC24FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_12_SEC_PROXY0_CFG_SCFG	0x4BC2800000	0x4BC28FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_12_SEC_PROXY0_SRC_TARGET_DATA	0x4BC2C00000	0x4BC2CFFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_12_PROXY0_SRC_TARGET0_DATA	0x4BC3000000	0x4BC303FFFF	256 KB
NAVSS0_VIRT_ALIAS_12_PROXY0_CFG_BUF_CFG	0x4BC3400000	0x4BC343FFFF	256 KB
NAVSS0_VIRT_ALIAS_12_UDMASS_INTA0_CFG_GCINTRTI	0x4BC3800000	0x4BC39FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_12_MODSS_INTA0_CFG_INTR	0x4BC3C00000	0x4BC3C3FFFF	256 KB
NAVSS0_VIRT_ALIAS_12_MODSS_INTA1_CFG_INTR	0x4BC3C40000	0x4BC3C7FFFF	256 KB
NAVSS0_VIRT_ALIAS_12_UDMASS_INTA0_CFG_INTR	0x4BC3D00000	0x4BC3DFFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_12_UDMAP0_CFG_RCHANRT	0x4BC4000000	0x4BC40FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_12_UDMAP0_CFG_TCHANRT	0x4BC5000000	0x4BC51FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_12_PAT0_CFG_SCRATCH	0x4BC6200000	0x4BC620FFFF	64 KB
NAVSS0_VIRT_ALIAS_12_PAT1_CFG_SCRATCH	0x4BC6210000	0x4BC621FFFF	64 KB
NAVSS0_VIRT_ALIAS_12_PAT2_CFG_SCRATCH	0x4BC6220000	0x4BC622FFFF	64 KB
NAVSS0_VIRT_ALIAS_12_PAT3_CFG_SCRATCH	0x4BC6230000	0x4BC6231FFF	8 KB
NAVSS0_VIRT_ALIAS_12_PAT4_CFG_SCRATCH	0x4BC6240000	0x4BC6241FFF	8 KB
NAVSS0_VIRT_ALIAS_12_PAT0_CFG_TABLE	0x4BC6400000	0x4BC643FFFF	256 KB
NAVSS0_VIRT_ALIAS_12_PAT1_CFG_TABLE	0x4BC6440000	0x4BC647FFFF	256 KB
NAVSS0_VIRT_ALIAS_12_PAT2_CFG_TABLE	0x4BC6480000	0x4BC64BFFFF	256 KB
NAVSS0_VIRT_ALIAS_12_PAT3_CFG_TABLE	0x4BC64C0000	0x4BC64C7FFF	32 KB
NAVSS0_VIRT_ALIAS_12_PAT4_CFG_TABLE	0x4BC6500000	0x4BC6507FFF	32 KB
NAVSS0_VIRT_ALIAS_12_TCU_CFG	0x4BC6600000	0x4BC67FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_12_RINGACC0_SRC_FIFOS	0x4BC8000000	0x4BC83FFFFFFF	4 MB
NAVSS0_VIRT_ALIAS_12_RINGACC0_CFG_RT	0x4BCC000000	0x4BCC3FFFFFFF	4 MB
NAVSS0_VIRT_ALIAS_13_MODSS_INTA0_CFG	0x4BD0800000	0x4BD080001F	32 B
NAVSS0_VIRT_ALIAS_13_MODSS_INTA1_CFG	0x4BD0801000	0x4BD080101F	32 B
NAVSS0_VIRT_ALIAS_13_UDMASS_INTA0_CFG	0x4BD0802000	0x4BD080201F	32 B
NAVSS0_VIRT_ALIAS_13_MODSS_INTA0_CFG_IMAP	0x4BD0900000	0x4BD0907FFF	32 KB
NAVSS0_VIRT_ALIAS_13_MODSS_INTA1_CFG_IMAP	0x4BD0908000	0x4BD090FFFF	32 KB
NAVSS0_VIRT_ALIAS_13_UDMASS_INTA0_CFG_IMAP	0x4BD0940000	0x4BD097FFFF	256 KB
NAVSS0_VIRT_ALIAS_13_NAV_DDR0_VIRTID_CFG_MMRS	0x4BD0A02000	0x4BD0A020FF	256 B
NAVSS0_VIRT_ALIAS_13_NAV_DDR1_VIRTID_CFG_MMRS	0x4BD0A03000	0x4BD0A030FF	256 B
NAVSS0_VIRT_ALIAS_13_UDMAP0_CFG_TCHAN	0x4BD0B00000	0x4BD0B1FFFF	128 KB
NAVSS0_VIRT_ALIAS_13_UDMAP0_CFG_RCHAN	0x4BD0C00000	0x4BD0C0FFFF	64 KB
NAVSS0_VIRT_ALIAS_13_UDMAP0_CFG_RFLOW	0x4BD0D00000	0x4BD0D07FFF	32 KB
NAVSS0_VIRT_ALIAS_13_SPINLOCK0_CFG	0x4BD0E00000	0x4BD0E07FFF	32 KB

**Table 2-2. NAVSS0 Alias Memory Map (continued)**

Region Name	Start Address	End Address	Size
NAVSS0_VIRT_ALIAS_13_TIMERMGR0_CFG	0x4BD0E80000	0x4BD0E801FF	512 B
NAVSS0_VIRT_ALIAS_13_TIMERMGR1_CFG	0x4BD0E81000	0x4BD0E811FF	512 B
NAVSS0_VIRT_ALIAS_13_TIMERMGR0_CFG_OES	0x4BD0F00000	0x4BD0F00FFF	4 KB
NAVSS0_VIRT_ALIAS_13_TIMERMGR1_CFG_OES	0x4BD0F01000	0x4BD0F01FFF	4 KB
NAVSS0_VIRT_ALIAS_13_ECCAGGR0	0x4BD1000000	0x4BD10003FF	1 KB
NAVSS0_VIRT_ALIAS_13_UDMASS_ECCAGGR_CFG	0x4BD1001000	0x4BD10013FF	1 KB
NAVSS0_VIRT_ALIAS_13_VIRTSS_ECCAGGR_CFG	0x4BD1002000	0x4BD10023FF	1 KB
NAVSS0_VIRT_ALIAS_13_PAT0_CFG_MMRS	0x4BD1010000	0x4BD10100FF	256 B
NAVSS0_VIRT_ALIAS_13_PAT1_CFG_MMRS	0x4BD1011000	0x4BD10110FF	256 B
NAVSS0_VIRT_ALIAS_13_PAT2_CFG_MMRS	0x4BD1012000	0x4BD10120FF	256 B
NAVSS0_VIRT_ALIAS_13_PAT3_CFG_MMRS	0x4BD1013000	0x4BD10130FF	256 B
NAVSS0_VIRT_ALIAS_13_PAT4_CFG_MMRS	0x4BD1014000	0x4BD10140FF	256 B
NAVSS0_VIRT_ALIAS_13_UDMASS_INTA0_CFG_GCNTCFG	0x4BD1040000	0x4BD1043FFF	16 KB
NAVSS0_VIRT_ALIAS_13_RINGACC0_CFG	0x4BD1080000	0x4BD10BFFFF	256 KB
NAVSS0_VIRT_ALIAS_13_REGS0_CFG_MMRS	0x4BD10C0000	0x4BD10C00FF	256 B
NAVSS0_VIRT_ALIAS_13_CPTS0_S_VBUSP_CPTS_VBUSP	0x4BD10D0000	0x4BD10D03FF	1 KB
NAVSS0_VIRT_ALIAS_13_INTR0_CFG_INTR_ROUTER_CFG	0x4BD10E0000	0x4BD10E3FFF	16 KB
NAVSS0_VIRT_ALIAS_13_UDMASS_INTA0_CFG_L2G	0x4BD1100000	0x4BD1100FFF	4 KB
NAVSS0_VIRT_ALIAS_13_UDMASS_INTA0_CFG_MCAST	0x4BD1110000	0x4BD1113FFF	16 KB
NAVSS0_VIRT_ALIAS_13_PROXY0_CFG_BUF_CFG_GCFG	0x4BD1120000	0x4BD11200FF	256 B
NAVSS0_VIRT_ALIAS_13_PROXY0_CFG_BUFRAM_SLV_RAM	0x4BD1130000	0x4BD1133FFF	16 KB
NAVSS0_VIRT_ALIAS_13_SEC_PROXY0_CFG_MMRS	0x4BD1140000	0x4BD11400FF	256 B
NAVSS0_VIRT_ALIAS_13_UDMAP0_CFG_GCFG	0x4BD1150000	0x4BD11500FF	256 B
NAVSS0_VIRT_ALIAS_13_RINGACC0_CFG_GCFG	0x4BD1160000	0x4BD11603FF	1 KB
NAVSS0_VIRT_ALIAS_13_PSILSS_LOCAL_CFG_MMRS	0x4BD1170000	0x4BD1170FFF	4 KB
NAVSS0_VIRT_ALIAS_13_PSILSS_GLOBAL_CFG_MMRS	0x4BD1180000	0x4BD1180FFF	4 KB
NAVSS0_VIRT_ALIAS_13_PSILSS_TR_CFG_MMRS	0x4BD1190000	0x4BD1190FFF	4 KB
NAVSS0_VIRT_ALIAS_13_MCRC0_S_CFG_MCRC64	0x4BD1F70000	0x4BD1F70FFF	4 KB
NAVSS0_VIRT_ALIAS_13_PSILCFG0_CFG_PROXY	0x4BD1F78000	0x4BD1F781FF	512 B
NAVSS0_VIRT_ALIAS_13_MAILBOX0_CFG_REGS0	0x4BD1F80000	0x4BD1F801FF	512 B
NAVSS0_VIRT_ALIAS_13_MAILBOX0_CFG_REGS1	0x4BD1F81000	0x4BD1F811FF	512 B
NAVSS0_VIRT_ALIAS_13_MAILBOX0_CFG_REGS2	0x4BD1F82000	0x4BD1F821FF	512 B
NAVSS0_VIRT_ALIAS_13_MAILBOX0_CFG_REGS3	0x4BD1F83000	0x4BD1F831FF	512 B
NAVSS0_VIRT_ALIAS_13_MAILBOX0_CFG_REGS4	0x4BD1F84000	0x4BD1F841FF	512 B
NAVSS0_VIRT_ALIAS_13_MAILBOX0_CFG_REGS5	0x4BD1F85000	0x4BD1F851FF	512 B
NAVSS0_VIRT_ALIAS_13_MAILBOX0_CFG_REGS6	0x4BD1F86000	0x4BD1F861FF	512 B
NAVSS0_VIRT_ALIAS_13_MAILBOX0_CFG_REGS7	0x4BD1F87000	0x4BD1F871FF	512 B
NAVSS0_VIRT_ALIAS_13_MAILBOX0_CFG_REGS8	0x4BD1F88000	0x4BD1F881FF	512 B
NAVSS0_VIRT_ALIAS_13_MAILBOX0_CFG_REGS9	0x4BD1F89000	0x4BD1F891FF	512 B
NAVSS0_VIRT_ALIAS_13_MAILBOX0_CFG_REGS10	0x4BD1F8A000	0x4BD1F8A1FF	512 B
NAVSS0_VIRT_ALIAS_13_MAILBOX0_CFG_REGS11	0x4BD1F8B000	0x4BD1F8B1FF	512 B
NAVSS0_VIRT_ALIAS_13_RINGACC0_CFG_MON	0x4BD2000000	0x4BD201FFFF	128 KB
NAVSS0_VIRT_ALIAS_13_TIMERMGR0_CFG_TIMERS	0x4BD2200000	0x4BD223FFFF	256 KB
NAVSS0_VIRT_ALIAS_13_TIMERMGR1_CFG_TIMERS	0x4BD2240000	0x4BD227FFFF	256 KB
NAVSS0_VIRT_ALIAS_13_SEC_PROXY0_CFG_RT	0x4BD2400000	0x4BD24FFFFFFF	1 MB

**Table 2-2. NAVSS0 Alias Memory Map (continued)**

Region Name	Start Address	End Address	Size
NAVSS0_VIRT_ALIAS_13_SEC_PROXY0_CFG_SCFG	0x4BD2800000	0x4BD28FFFFF	1 MB
NAVSS0_VIRT_ALIAS_13_SEC_PROXY0_SRC_TARGET_DATA	0x4BD2C00000	0x4BD2CFFFFF	1 MB
NAVSS0_VIRT_ALIAS_13_PROXY0_SRC_TARGET0_DATA	0x4BD3000000	0x4BD303FFFF	256 KB
NAVSS0_VIRT_ALIAS_13_PROXY0_CFG_BUF_CFG	0x4BD3400000	0x4BD343FFFF	256 KB
NAVSS0_VIRT_ALIAS_13_UDMASS_INTA0_CFG_GCINTRTI	0x4BD3800000	0x4BD39FFFFF	2 MB
NAVSS0_VIRT_ALIAS_13_MODSS_INTA0_CFG_INTR	0x4BD3C00000	0x4BD3C3FFFF	256 KB
NAVSS0_VIRT_ALIAS_13_MODSS_INTA1_CFG_INTR	0x4BD3C40000	0x4BD3C7FFFF	256 KB
NAVSS0_VIRT_ALIAS_13_UDMASS_INTA0_CFG_INTR	0x4BD3D00000	0x4BD3DFFFFF	1 MB
NAVSS0_VIRT_ALIAS_13_UDMAP0_CFG_RCHANRT	0x4BD4000000	0x4BD40FFFFF	1 MB
NAVSS0_VIRT_ALIAS_13_UDMAP0_CFG_TCHANRT	0x4BD5000000	0x4BD51FFFFF	2 MB
NAVSS0_VIRT_ALIAS_13_PAT0_CFG_SCRATCH	0x4BD6200000	0x4BD620FFFF	64 KB
NAVSS0_VIRT_ALIAS_13_PAT1_CFG_SCRATCH	0x4BD6210000	0x4BD621FFFF	64 KB
NAVSS0_VIRT_ALIAS_13_PAT2_CFG_SCRATCH	0x4BD6220000	0x4BD622FFFF	64 KB
NAVSS0_VIRT_ALIAS_13_PAT3_CFG_SCRATCH	0x4BD6230000	0x4BD6231FFF	8 KB
NAVSS0_VIRT_ALIAS_13_PAT4_CFG_SCRATCH	0x4BD6240000	0x4BD6241FFF	8 KB
NAVSS0_VIRT_ALIAS_13_PAT0_CFG_TABLE	0x4BD6400000	0x4BD643FFFF	256 KB
NAVSS0_VIRT_ALIAS_13_PAT1_CFG_TABLE	0x4BD6440000	0x4BD647FFFF	256 KB
NAVSS0_VIRT_ALIAS_13_PAT2_CFG_TABLE	0x4BD6480000	0x4BD64BFFFF	256 KB
NAVSS0_VIRT_ALIAS_13_PAT3_CFG_TABLE	0x4BD64C0000	0x4BD64C7FFF	32 KB
NAVSS0_VIRT_ALIAS_13_PAT4_CFG_TABLE	0x4BD6500000	0x4BD6507FFF	32 KB
NAVSS0_VIRT_ALIAS_13_TCU_CFG	0x4BD6600000	0x4BD67FFFFF	2 MB
NAVSS0_VIRT_ALIAS_13_RINGACC0_SRC_FIFOS	0x4BD8000000	0x4BD83FFFFF	4 MB
NAVSS0_VIRT_ALIAS_13_RINGACC0_CFG_RT	0x4BDC000000	0x4BDC3FFFFF	4 MB
NAVSS0_VIRT_ALIAS_14_MODSS_INTA0_CFG	0x4BE0800000	0x4BE080001F	32 B
NAVSS0_VIRT_ALIAS_14_MODSS_INTA1_CFG	0x4BE0801000	0x4BE080101F	32 B
NAVSS0_VIRT_ALIAS_14_UDMASS_INTA0_CFG	0x4BE0802000	0x4BE080201F	32 B
NAVSS0_VIRT_ALIAS_14_MODSS_INTA0_CFG_IMAP	0x4BE0900000	0x4BE0907FFF	32 KB
NAVSS0_VIRT_ALIAS_14_MODSS_INTA1_CFG_IMAP	0x4BE0908000	0x4BE090FFFF	32 KB
NAVSS0_VIRT_ALIAS_14_UDMASS_INTA0_CFG_IMAP	0x4BE0940000	0x4BE097FFFF	256 KB
NAVSS0_VIRT_ALIAS_14_NAV_DDR0_VIRTID_CFG_MMRS	0x4BE0A02000	0x4BE0A020FF	256 B
NAVSS0_VIRT_ALIAS_14_NAV_DDR1_VIRTID_CFG_MMRS	0x4BE0A03000	0x4BE0A030FF	256 B
NAVSS0_VIRT_ALIAS_14_UDMAP0_CFG_TCHAN	0x4BE0B00000	0x4BE0B1FFFF	128 KB
NAVSS0_VIRT_ALIAS_14_UDMAP0_CFG_RCHAN	0x4BE0C00000	0x4BE0C0FFFF	64 KB
NAVSS0_VIRT_ALIAS_14_UDMAP0_CFG_RFLOW	0x4BE0D00000	0x4BE0D07FFF	32 KB
NAVSS0_VIRT_ALIAS_14_SPINLOCK0_CFG	0x4BE0E00000	0x4BE0E07FFF	32 KB
NAVSS0_VIRT_ALIAS_14_TIMERMGR0_CFG	0x4BE0E80000	0x4BE0E801FF	512 B
NAVSS0_VIRT_ALIAS_14_TIMERMGR1_CFG	0x4BE0E81000	0x4BE0E811FF	512 B
NAVSS0_VIRT_ALIAS_14_TIMERMGR0_CFG_OES	0x4BE0F00000	0x4BE0F00FFF	4 KB
NAVSS0_VIRT_ALIAS_14_TIMERMGR1_CFG_OES	0x4BE0F01000	0x4BE0F01FFF	4 KB
NAVSS0_VIRT_ALIAS_14_ECCAGGR0	0x4BE1000000	0x4BE10003FF	1 KB
NAVSS0_VIRT_ALIAS_14_UDMASS_ECCAGGR_CFG	0x4BE1001000	0x4BE10013FF	1 KB
NAVSS0_VIRT_ALIAS_14_VIRTSS_ECCAGGR_CFG	0x4BE1002000	0x4BE10023FF	1 KB
NAVSS0_VIRT_ALIAS_14_PAT0_CFG_MMRS	0x4BE1010000	0x4BE10100FF	256 B
NAVSS0_VIRT_ALIAS_14_PAT1_CFG_MMRS	0x4BE1011000	0x4BE10110FF	256 B
NAVSS0_VIRT_ALIAS_14_PAT2_CFG_MMRS	0x4BE1012000	0x4BE10120FF	256 B

**Table 2-2. NAVSS0 Alias Memory Map (continued)**

Region Name	Start Address	End Address	Size
NAVSS0_VIRT_ALIAS_14_PAT3_CFG_MMRS	0x4BE1013000	0x4BE10130FF	256 B
NAVSS0_VIRT_ALIAS_14_PAT4_CFG_MMRS	0x4BE1014000	0x4BE10140FF	256 B
NAVSS0_VIRT_ALIAS_14_UDMASS_INTA0_CFG_GCNTCFG	0x4BE1040000	0x4BE1043FFF	16 KB
NAVSS0_VIRT_ALIAS_14_RINGACC0_CFG	0x4BE1080000	0x4BE10BFFFF	256 KB
NAVSS0_VIRT_ALIAS_14_REGS0_CFG_MMRS	0x4BE10C0000	0x4BE10C00FF	256 B
NAVSS0_VIRT_ALIAS_14_CPTS0_S_VBUSP_CPTS_VBUSP	0x4BE10D0000	0x4BE10D03FF	1 KB
NAVSS0_VIRT_ALIAS_14_INTR0_CFG_INTR_ROUTER_CFG	0x4BE10E0000	0x4BE10E3FFF	16 KB
NAVSS0_VIRT_ALIAS_14_UDMASS_INTA0_CFG_L2G	0x4BE1100000	0x4BE1100FFF	4 KB
NAVSS0_VIRT_ALIAS_14_UDMASS_INTA0_CFG_MCAST	0x4BE1110000	0x4BE1113FFF	16 KB
NAVSS0_VIRT_ALIAS_14_PROXY0_CFG_BUF_CFG_GCFG	0x4BE1120000	0x4BE11200FF	256 B
NAVSS0_VIRT_ALIAS_14_PROXY0_CFG_BUFRAM_SLV_RAM	0x4BE1130000	0x4BE1133FFF	16 KB
NAVSS0_VIRT_ALIAS_14_SEC_PROXY0_CFG_MMRS	0x4BE1140000	0x4BE11400FF	256 B
NAVSS0_VIRT_ALIAS_14_UDMAP0_CFG_GCFG	0x4BE1150000	0x4BE11500FF	256 B
NAVSS0_VIRT_ALIAS_14_RINGACC0_CFG_GCFG	0x4BE1160000	0x4BE11603FF	1 KB
NAVSS0_VIRT_ALIAS_14_PSILSS_LOCAL_CFG_MMRS	0x4BE1170000	0x4BE1170FFF	4 KB
NAVSS0_VIRT_ALIAS_14_PSILSS_GLOBAL_CFG_MMRS	0x4BE1180000	0x4BE1180FFF	4 KB
NAVSS0_VIRT_ALIAS_14_PSILSS_TR_CFG_MMRS	0x4BE1190000	0x4BE1190FFF	4 KB
NAVSS0_VIRT_ALIAS_14_MCRC0_S_CFG_MCRC64	0x4BE1F70000	0x4BE1F70FFF	4 KB
NAVSS0_VIRT_ALIAS_14_PSILCFG0_CFG_PROXY	0x4BE1F78000	0x4BE1F781FF	512 B
NAVSS0_VIRT_ALIAS_14_MAILBOX0_CFG_REGS0	0x4BE1F80000	0x4BE1F801FF	512 B
NAVSS0_VIRT_ALIAS_14_MAILBOX0_CFG_REGS1	0x4BE1F81000	0x4BE1F811FF	512 B
NAVSS0_VIRT_ALIAS_14_MAILBOX0_CFG_REGS2	0x4BE1F82000	0x4BE1F821FF	512 B
NAVSS0_VIRT_ALIAS_14_MAILBOX0_CFG_REGS3	0x4BE1F83000	0x4BE1F831FF	512 B
NAVSS0_VIRT_ALIAS_14_MAILBOX0_CFG_REGS4	0x4BE1F84000	0x4BE1F841FF	512 B
NAVSS0_VIRT_ALIAS_14_MAILBOX0_CFG_REGS5	0x4BE1F85000	0x4BE1F851FF	512 B
NAVSS0_VIRT_ALIAS_14_MAILBOX0_CFG_REGS6	0x4BE1F86000	0x4BE1F861FF	512 B
NAVSS0_VIRT_ALIAS_14_MAILBOX0_CFG_REGS7	0x4BE1F87000	0x4BE1F871FF	512 B
NAVSS0_VIRT_ALIAS_14_MAILBOX0_CFG_REGS8	0x4BE1F88000	0x4BE1F881FF	512 B
NAVSS0_VIRT_ALIAS_14_MAILBOX0_CFG_REGS9	0x4BE1F89000	0x4BE1F891FF	512 B
NAVSS0_VIRT_ALIAS_14_MAILBOX0_CFG_REGS10	0x4BE1F8A000	0x4BE1F8A1FF	512 B
NAVSS0_VIRT_ALIAS_14_MAILBOX0_CFG_REGS11	0x4BE1F8B000	0x4BE1F8B1FF	512 B
NAVSS0_VIRT_ALIAS_14_RINGACC0_CFG_MON	0x4BE2000000	0x4BE201FFFF	128 KB
NAVSS0_VIRT_ALIAS_14_TIMERMGR0_CFG_TIMERS	0x4BE2200000	0x4BE223FFFF	256 KB
NAVSS0_VIRT_ALIAS_14_TIMERMGR1_CFG_TIMERS	0x4BE2240000	0x4BE227FFFF	256 KB
NAVSS0_VIRT_ALIAS_14_SEC_PROXY0_CFG_RT	0x4BE2400000	0x4BE24FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_14_SEC_PROXY0_CFG_SCFG	0x4BE2800000	0x4BE28FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_14_SEC_PROXY0_SRC_TARGET_DATA	0x4BE2C00000	0x4BE2CFFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_14_PROXY0_SRC_TARGET0_DATA	0x4BE3000000	0x4BE303FFFF	256 KB
NAVSS0_VIRT_ALIAS_14_PROXY0_CFG_BUF_CFG	0x4BE3400000	0x4BE343FFFF	256 KB
NAVSS0_VIRT_ALIAS_14_UDMASS_INTA0_CFG_GCINTRTI	0x4BE3800000	0x4BE39FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_14_MODSS_INTA0_CFG_INTR	0x4BE3C00000	0x4BE3C3FFFF	256 KB
NAVSS0_VIRT_ALIAS_14_MODSS_INTA1_CFG_INTR	0x4BE3C40000	0x4BE3C7FFFF	256 KB
NAVSS0_VIRT_ALIAS_14_UDMASS_INTA0_CFG_INTR	0x4BE3D00000	0x4BE3DFFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_14_UDMAP0_CFG_RCHANRT	0x4BE4000000	0x4BE40FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_14_UDMAP0_CFG_TCHANRT	0x4BE5000000	0x4BE51FFFFFFF	2 MB

**Table 2-2. NAVSS0 Alias Memory Map (continued)**

Region Name	Start Address	End Address	Size
NAVSS0_VIRT_ALIAS_14_PAT0_CFG_SCRATCH	0x4BE6200000	0x4BE620FFFF	64 KB
NAVSS0_VIRT_ALIAS_14_PAT1_CFG_SCRATCH	0x4BE6210000	0x4BE621FFFF	64 KB
NAVSS0_VIRT_ALIAS_14_PAT2_CFG_SCRATCH	0x4BE6220000	0x4BE622FFFF	64 KB
NAVSS0_VIRT_ALIAS_14_PAT3_CFG_SCRATCH	0x4BE6230000	0x4BE6231FFF	8 KB
NAVSS0_VIRT_ALIAS_14_PAT4_CFG_SCRATCH	0x4BE6240000	0x4BE6241FFF	8 KB
NAVSS0_VIRT_ALIAS_14_PAT0_CFG_TABLE	0x4BE6400000	0x4BE643FFFF	256 KB
NAVSS0_VIRT_ALIAS_14_PAT1_CFG_TABLE	0x4BE6440000	0x4BE647FFFF	256 KB
NAVSS0_VIRT_ALIAS_14_PAT2_CFG_TABLE	0x4BE6480000	0x4BE64BFFFF	256 KB
NAVSS0_VIRT_ALIAS_14_PAT3_CFG_TABLE	0x4BE64C0000	0x4BE64C7FFF	32 KB
NAVSS0_VIRT_ALIAS_14_PAT4_CFG_TABLE	0x4BE6500000	0x4BE6507FFF	32 KB
NAVSS0_VIRT_ALIAS_14_TCU_CFG	0x4BE6600000	0x4BE67FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_14_RINGACC0_SRC_FIFOS	0x4BE8000000	0x4BE83FFFFFFF	4 MB
NAVSS0_VIRT_ALIAS_14_RINGACC0_CFG_RT	0x4BEC000000	0x4BEC3FFFFFFF	4 MB
NAVSS0_VIRT_ALIAS_15_MODSS_INTA0_CFG	0x4BF0800000	0x4BF080001F	32 B
NAVSS0_VIRT_ALIAS_15_MODSS_INTA1_CFG	0x4BF0801000	0x4BF080101F	32 B
NAVSS0_VIRT_ALIAS_15_UDMASS_INTA0_CFG	0x4BF0802000	0x4BF080201F	32 B
NAVSS0_VIRT_ALIAS_15_MODSS_INTA0_CFG_IMAP	0x4BF0900000	0x4BF0907FFF	32 KB
NAVSS0_VIRT_ALIAS_15_MODSS_INTA1_CFG_IMAP	0x4BF0908000	0x4BF090FFFFFF	32 KB
NAVSS0_VIRT_ALIAS_15_UDMASS_INTA0_CFG_IMAP	0x4BF0940000	0x4BF097FFFFFF	256 KB
NAVSS0_VIRT_ALIAS_15_NAV_DDR0_VIRTID_CFG_MMRS	0x4BF0A02000	0x4BF0A020FF	256 B
NAVSS0_VIRT_ALIAS_15_NAV_DDR1_VIRTID_CFG_MMRS	0x4BF0A03000	0x4BF0A030FF	256 B
NAVSS0_VIRT_ALIAS_15_UDMAP0_CFG_TCHAN	0x4BF0B00000	0x4BF0B1FFFF	128 KB
NAVSS0_VIRT_ALIAS_15_UDMAP0_CFG_RCHAN	0x4BF0C00000	0x4BF0C0FFFFFF	64 KB
NAVSS0_VIRT_ALIAS_15_UDMAP0_CFG_RFLOW	0x4BF0D00000	0x4BF0D07FFF	32 KB
NAVSS0_VIRT_ALIAS_15_SPINLOCK0_CFG	0x4BF0E00000	0x4BF0E07FFF	32 KB
NAVSS0_VIRT_ALIAS_15_TIMERMGR0_CFG	0x4BF0E80000	0x4BF0E801FF	512 B
NAVSS0_VIRT_ALIAS_15_TIMERMGR1_CFG	0x4BF0E81000	0x4BF0E811FF	512 B
NAVSS0_VIRT_ALIAS_15_TIMERMGR0_CFG_OES	0x4BF0F00000	0x4BF0F00FFF	4 KB
NAVSS0_VIRT_ALIAS_15_TIMERMGR1_CFG_OES	0x4BF0F01000	0x4BF0F01FFF	4 KB
NAVSS0_VIRT_ALIAS_15_ECCAGGR0	0x4BF1000000	0x4BF10003FF	1 KB
NAVSS0_VIRT_ALIAS_15_UDMASS_ECCAGGR_CFG	0x4BF1001000	0x4BF10013FF	1 KB
NAVSS0_VIRT_ALIAS_15_VIRTSS_ECCAGGR_CFG	0x4BF1002000	0x4BF10023FF	1 KB
NAVSS0_VIRT_ALIAS_15_PAT0_CFG_MMRS	0x4BF1010000	0x4BF10100FF	256 B
NAVSS0_VIRT_ALIAS_15_PAT1_CFG_MMRS	0x4BF1011000	0x4BF10110FF	256 B
NAVSS0_VIRT_ALIAS_15_PAT2_CFG_MMRS	0x4BF1012000	0x4BF10120FF	256 B
NAVSS0_VIRT_ALIAS_15_PAT3_CFG_MMRS	0x4BF1013000	0x4BF10130FF	256 B
NAVSS0_VIRT_ALIAS_15_PAT4_CFG_MMRS	0x4BF1014000	0x4BF10140FF	256 B
NAVSS0_VIRT_ALIAS_15_UDMASS_INTA0_CFG_GCNTCFG	0x4BF1040000	0x4BF1043FFF	16 KB
NAVSS0_VIRT_ALIAS_15_RINGACC0_CFG	0x4BF1080000	0x4BF10BFFFFFF	256 KB
NAVSS0_VIRT_ALIAS_15_REGS0_CFG_MMRS	0x4BF10C0000	0x4BF10C00FF	256 B
NAVSS0_VIRT_ALIAS_15_CPTS0_S_VBUSP_CPTS_VBUSP	0x4BF10D0000	0x4BF10D03FF	1 KB
NAVSS0_VIRT_ALIAS_15_INTR0_CFG_INTR_ROUTER_CFG	0x4BF10E0000	0x4BF10E3FFF	16 KB
NAVSS0_VIRT_ALIAS_15_UDMASS_INTA0_CFG_L2G	0x4BF1100000	0x4BF1100FFF	4 KB
NAVSS0_VIRT_ALIAS_15_UDMASS_INTA0_CFG_MCAST	0x4BF1110000	0x4BF1113FFF	16 KB
NAVSS0_VIRT_ALIAS_15_PROXY0_CFG_BUF_CFG_GCFG	0x4BF1120000	0x4BF11200FF	256 B



**Table 2-2. NAVSS0 Alias Memory Map (continued)**

Region Name	Start Address	End Address	Size
NAVSS0_VIRT_ALIAS_15_PROXY0_CFG_BUFRAM_SLV_RAM	0x4BF1130000	0x4BF1133FFF	16 KB
NAVSS0_VIRT_ALIAS_15_SEC_PROXY0_CFG_MMRS	0x4BF1140000	0x4BF11400FF	256 B
NAVSS0_VIRT_ALIAS_15_UDMAP0_CFG_GCFG	0x4BF1150000	0x4BF11500FF	256 B
NAVSS0_VIRT_ALIAS_15_RINGACC0_CFG_GCFG	0x4BF1160000	0x4BF11603FF	1 KB
NAVSS0_VIRT_ALIAS_15_PSILSS_LOCAL_CFG_MMRS	0x4BF1170000	0x4BF1170FFF	4 KB
NAVSS0_VIRT_ALIAS_15_PSILSS_GLOBAL_CFG_MMRS	0x4BF1180000	0x4BF1180FFF	4 KB
NAVSS0_VIRT_ALIAS_15_PSILSS_TR_CFG_MMRS	0x4BF1190000	0x4BF1190FFF	4 KB
NAVSS0_VIRT_ALIAS_15_MCRC0_S_CFG_MCRC64	0x4BF1F70000	0x4BF1F70FFF	4 KB
NAVSS0_VIRT_ALIAS_15_PSILCFG0_CFG_PROXY	0x4BF1F78000	0x4BF1F781FF	512 B
NAVSS0_VIRT_ALIAS_15_MAILBOX0_CFG_REGS0	0x4BF1F80000	0x4BF1F801FF	512 B
NAVSS0_VIRT_ALIAS_15_MAILBOX0_CFG_REGS1	0x4BF1F81000	0x4BF1F811FF	512 B
NAVSS0_VIRT_ALIAS_15_MAILBOX0_CFG_REGS2	0x4BF1F82000	0x4BF1F821FF	512 B
NAVSS0_VIRT_ALIAS_15_MAILBOX0_CFG_REGS3	0x4BF1F83000	0x4BF1F831FF	512 B
NAVSS0_VIRT_ALIAS_15_MAILBOX0_CFG_REGS4	0x4BF1F84000	0x4BF1F841FF	512 B
NAVSS0_VIRT_ALIAS_15_MAILBOX0_CFG_REGS5	0x4BF1F85000	0x4BF1F851FF	512 B
NAVSS0_VIRT_ALIAS_15_MAILBOX0_CFG_REGS6	0x4BF1F86000	0x4BF1F861FF	512 B
NAVSS0_VIRT_ALIAS_15_MAILBOX0_CFG_REGS7	0x4BF1F87000	0x4BF1F871FF	512 B
NAVSS0_VIRT_ALIAS_15_MAILBOX0_CFG_REGS8	0x4BF1F88000	0x4BF1F881FF	512 B
NAVSS0_VIRT_ALIAS_15_MAILBOX0_CFG_REGS9	0x4BF1F89000	0x4BF1F891FF	512 B
NAVSS0_VIRT_ALIAS_15_MAILBOX0_CFG_REGS10	0x4BF1F8A000	0x4BF1F8A1FF	512 B
NAVSS0_VIRT_ALIAS_15_MAILBOX0_CFG_REGS11	0x4BF1F8B000	0x4BF1F8B1FF	512 B
NAVSS0_VIRT_ALIAS_15_RINGACC0_CFG_MON	0x4BF2000000	0x4BF201FFFF	128 KB
NAVSS0_VIRT_ALIAS_15_TIMERMGR0_CFG_TIMERS	0x4BF2200000	0x4BF223FFFF	256 KB
NAVSS0_VIRT_ALIAS_15_TIMERMGR1_CFG_TIMERS	0x4BF2240000	0x4BF227FFFF	256 KB
NAVSS0_VIRT_ALIAS_15_SEC_PROXY0_CFG_RT	0x4BF2400000	0x4BF24FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_15_SEC_PROXY0_CFG_SCFG	0x4BF2800000	0x4BF28FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_15_SEC_PROXY0_SRC_TARGET_DATA	0x4BF2C00000	0x4BF2CFFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_15_PROXY0_SRC_TARGET0_DATA	0x4BF3000000	0x4BF303FFFF	256 KB
NAVSS0_VIRT_ALIAS_15_PROXY0_CFG_BUF_CFG	0x4BF3400000	0x4BF343FFFF	256 KB
NAVSS0_VIRT_ALIAS_15_UDMASS_INTA0_CFG_GCINTRTI	0x4BF3800000	0x4BF39FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_15_MODSS_INTA0_CFG_INTR	0x4BF3C00000	0x4BF3C3FFFF	256 KB
NAVSS0_VIRT_ALIAS_15_MODSS_INTA1_CFG_INTR	0x4BF3C40000	0x4BF3C7FFFF	256 KB
NAVSS0_VIRT_ALIAS_15_UDMASS_INTA0_CFG_INTR	0x4BF3D00000	0x4BF3DFFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_15_UDMAP0_CFG_RCHANRT	0x4BF4000000	0x4BF40FFFFFFF	1 MB
NAVSS0_VIRT_ALIAS_15_UDMAP0_CFG_TCHANRT	0x4BF5000000	0x4BF51FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_15_PAT0_CFG_SCRATCH	0x4BF6200000	0x4BF620FFFF	64 KB
NAVSS0_VIRT_ALIAS_15_PAT1_CFG_SCRATCH	0x4BF6210000	0x4BF621FFFF	64 KB
NAVSS0_VIRT_ALIAS_15_PAT2_CFG_SCRATCH	0x4BF6220000	0x4BF622FFFF	64 KB
NAVSS0_VIRT_ALIAS_15_PAT3_CFG_SCRATCH	0x4BF6230000	0x4BF6231FFF	8 KB
NAVSS0_VIRT_ALIAS_15_PAT4_CFG_SCRATCH	0x4BF6240000	0x4BF6241FFF	8 KB
NAVSS0_VIRT_ALIAS_15_PAT0_CFG_TABLE	0x4BF6400000	0x4BF643FFFF	256 KB
NAVSS0_VIRT_ALIAS_15_PAT1_CFG_TABLE	0x4BF6440000	0x4BF647FFFF	256 KB
NAVSS0_VIRT_ALIAS_15_PAT2_CFG_TABLE	0x4BF6480000	0x4BF64BFFFF	256 KB
NAVSS0_VIRT_ALIAS_15_PAT3_CFG_TABLE	0x4BF64C0000	0x4BF64C7FFF	32 KB
NAVSS0_VIRT_ALIAS_15_PAT4_CFG_TABLE	0x4BF6500000	0x4BF6507FFF	32 KB

**Table 2-2. NAVSS0 Alias Memory Map (continued)**

Region Name	Start Address	End Address	Size
NAVSS0_VIRT_ALIAS_15_TCU_CFG	0x4BF6600000	0x4BF67FFFFFFF	2 MB
NAVSS0_VIRT_ALIAS_15_RINGACC0_SRC_FIFOS	0x4BF8000000	0x4BF83FFFFFFF	4 MB
NAVSS0_VIRT_ALIAS_15_RINGACC0_CFG_RT	0x4BFC000000	0x4BFC3FFFFFFF	4 MB

---

**Note**

NAVSS0\_DDR0 and NAVSS0\_DDR1 are used for load balancing and real time/ non-real time traffic separation. Max addressable DDR memory across the two paths is 8 GB.

---

## 2.2 MCU Domain Memory Map

Table 2-3 shows the MCU domain memory map.

### Note

The memory locations not shown in Table 2-3 are either unallocated or reserved and not used. Accesses to these locations are not recommended and should be avoided.

**Table 2-3. MCU Domain Memory Map**

Region Name	Start Address	End Address	Size
MCU_NAVSS0_ECCAGGR0	0x0028380000	0x00283803FF	1 KB
MCU_NAVSS0_UDMASS_ECCAGGR0	0x0028381000	0x00283813FF	1 KB
MCU_NAVSS0_UDMASS_INTA0_CFG	0x00283C0000	0x00283C001F	32 B
MCU_NAVSS0_UDMASS_UDMAP0_CFG_RFLOW	0x0028400000	0x0028401FFF	8 KB
MCU_NAVSS0_UDMASS_RINGACC0_CFG	0x0028440000	0x002847FFFF	256 KB
MCU_NAVSS0_UDMASS_INTA0_GCNT	0x0028480000	0x0028481FFF	8 KB
MCU_NAVSS0_UDMASS_UDMAP0_TCHAN	0x00284A0000	0x00284A3FFF	16 KB
MCU_NAVSS0_UDMASS_UDMAP0_RCHAN	0x00284C0000	0x00284C3FFF	16 KB
MCU_NAVSS0_CFG	0x0028520000	0x00285200FF	256 B
MCU_NAVSS0_INTR0_CFG	0x0028540000	0x00285407FF	2 KB
MCU_NAVSS0_UDMASS_INTA0_IMAP	0x0028560000	0x002856FFFF	64 KB
MCU_NAVSS0_UDMASS_INTA0_I2G	0x0028570000	0x00285701FF	512 B
MCU_NAVSS0_UDMASS_INTA0_MCAST	0x0028580000	0x0028580FFF	4 KB
MCU_NAVSS0_PROXY_CFG_GCFG	0x0028590000	0x00285900FF	256 B
MCU_NAVSS0_PROXY_CFG_BUF	0x00285A0000	0x00285A3FFF	16 KB
MCU_NAVSS0_SEC_PROXY0_CFG	0x00285B0000	0x00285B00FF	256 B
MCU_NAVSS0_UDMASS_UDMAP0_CFG_GCFG	0x00285C0000	0x00285C00FF	256 B
MCU_NAVSS0_UDMASS_RINGACC0_CFG_GCFG	0x00285D0000	0x00285D03FF	1 KB
MCU_NAVSS0_UDMASS_PSISS0_CFG_MMRS	0x00285E0000	0x00285E0FFF	4 KB
MCU_NAVSS0_MCRC	0x002A264000	0x002A264FFF	4 KB
MCU_NAVSS0_UDMASS_PSISS_CFG0_PROXY	0x002A268000	0x002A2681FF	512 B
MCU_NAVSS0_UDMASS_RINGACC0_CFG_MON	0x002A280000	0x002A29FFFF	128 KB
MCU_NAVSS0_SEC_PROXY0_CFG_RT	0x002A380000	0x002A3FFFFFFF	512 KB
MCU_NAVSS0_SEC_PROXY0_CFG_SCFG	0x002A400000	0x002A47FFFF	512 KB
MCU_NAVSS0_SEC_PROXY0_TARGET_DATA	0x002A480000	0x002A4FFFFFFF	512 KB
MCU_NAVSS0_PROXY0_TARGET0_DATA	0x002A500000	0x002A53FFFF	256 KB
MCU_NAVSS0_PROXY0_BUF_CFG	0x002A580000	0x002A5BFFFF	256 KB
MCU_NAVSS0_UDMASS_INTA0_GCNTRTI	0x002A600000	0x002A6FFFFFFF	1 MB
MCU_NAVSS0_UDMASS_INTA0_INTR	0x002A700000	0x002A7FFFFFFF	1 MB
MCU_NAVSS0_UDMASS_UDMAP_RCHANRT	0x002A800000	0x002A83FFFF	256 KB
MCU_NAVSS0_UDMASS_UDMAP_TCHANRT	0x002AA00000	0x002AA3FFFF	256 KB
MCU_NAVSS0_UDMASS_RINGACC0_FIFOS	0x002B000000	0x002B3FFFFFFF	4 MB
MCU_NAVSS0_UDMASS_RINGACC0_CFG_RT	0x002B800000	0x002BBFFFFFFF	4 MB
MCU_R5FSS0_CORE0_ECC_AGGR	0x0040080000	0x00400803FF	1 KB
MCU_R5FSS0_ECC_AGGR	0x00400C0000	0x00400C03FF	1 KB
MCU_R5FSS0_COMPARE_CFG	0x00400F0000	0x00400F00FF	256 B
MCU_DCC0	0x0040100000	0x004010003F	64 B
MCU_DCC1	0x0040110000	0x004011003F	64 B

**Table 2-3. MCU Domain Memory Map (continued)**

Region Name	Start Address	End Address	Size
MCU_DCC2	0x0040120000	0x004012003F	64 B
MCU_ADC0	0x0040200000	0x00402003FF	1 KB
MCU_ADC0_FIFO	0x0040208000	0x00402083FF	1 KB
MCU_ADC1	0x0040210000	0x00402103FF	1 KB
MCU_ADC1_FIFO	0x0040218000	0x00402183FF	1 KB
MCU_PSRAM0_RAM	0x0040280000	0x00402801FF	512 B
MCU_MCSPI0_CFG	0x0040300000	0x00403003FF	1 KB
MCU_MCSPI1_CFG	0x0040310000	0x00403103FF	1 KB
MCU_MCSPI2_CFG	0x0040320000	0x00403203FF	1 KB
MCU_TIMER0_CFG	0x0040400000	0x00404003FF	1 KB
MCU_TIMER1_CFG	0x0040410000	0x00404103FF	1 KB
MCU_TIMER2_CFG	0x0040420000	0x00404203FF	1 KB
MCU_TIMER3_CFG	0x0040430000	0x00404303FF	1 KB
MCU_TIMER4_CFG	0x0040440000	0x00404403FF	1 KB
MCU_TIMER5_CFG	0x0040450000	0x00404503FF	1 KB
MCU_TIMER6_CFG	0x0040460000	0x00404603FF	1 KB
MCU_TIMER7_CFG	0x0040470000	0x00404703FF	1 KB
MCU_TIMER8_CFG	0x0040480000	0x00404803FF	1 KB
MCU_TIMER9_CFG	0x0040490000	0x00404903FF	1 KB
MCU_MCAN0_MSGMEM_RAM	0x0040500000	0x0040507FFF	32 KB
MCU_MCAN0_SS	0x0040520000	0x00405200FF	256 B
MCU_MCAN0_CFG	0x0040528000	0x00405281FF	512 B
MCU_MCAN1_MSGMEM_RAM	0x0040540000	0x0040547FFF	32 KB
MCU_MCAN1_SS	0x0040560000	0x00405600FF	256 B
MCU_MCAN1_CFG	0x0040568000	0x00405681FF	512 B
MCU_RTIO_CFG	0x0040600000	0x00406000FF	256 B
MCU_RTII_CFG	0x0040610000	0x00406100FF	256 B
MCU_MCAN0_ECC_AGGR	0x0040700000	0x00407003FF	1 KB
MCU_MCAN1_ECC_AGGR	0x0040701000	0x00407013FF	1 KB
MCU_ADC0_ECC_REGS	0x0040707000	0x00407073FF	1 KB
MCU_ADC1_ECC_REGS	0x0040708000	0x00407083FF	1 KB
MCU_CPSW0_ECC	0x0040709000	0x00407093FF	1 KB
MCU_MSRAM_1MB0_ECC_AGGR_REGS	0x004070B000	0x004070B3FF	1 KB
MCU_SA2_UL0_ECC_AGGR	0x004070C000	0x004070C3FF	1 KB
MCU_I3C0_P_ECC_AGGR_CFG	0x0040720000	0x00407203FF	1 KB
MCU_I3C0_S_ECC_AGGR_CFG	0x0040721000	0x00407213FF	1 KB
MCU_I3C1_P_ECC_AGGR_CFG	0x0040722000	0x00407223FF	1 KB
MCU_I3C1_S_ECC_AGGR_CFG	0x0040723000	0x00407233FF	1 KB
MCU_TIMEOUT_64B2_CFG	0x0040730000	0x00407303FF	1 KB
MCU_TIMEOUT_INFRA0_CFG	0x0040731000	0x00407313FF	1 KB
MCU_TIMEOUT_FW1_CFG	0x0040732000	0x00407323FF	1 KB
MCU_ESM0_CFG	0x0040800000	0x0040800FFF	4 KB
MCU_SA2_UL0	0x0040900000	0x0040900FFF	4 KB
MCU_SA2_UL0_MMRA	0x0040901000	0x00409011FF	512 B
MCU_SA2_UL0_EIP_76	0x0040910000	0x004091007F	128 B

**Table 2-3. MCU Domain Memory Map (continued)**

Region Name	Start Address	End Address	Size
MCU_SA2_UL0_EIP_29T2	0x0040920000	0x004092FFFF	64 KB
MCU_UART0	0x0040A00000	0x0040A001FF	512 B
MCU_I2C0_CFG	0x0040B00000	0x0040B000FF	256 B
MCU_I2C1_CFG	0x0040B10000	0x0040B100FF	256 B
MCU_I3C0_MMR_MMRVBP	0x0040B80000	0x0040B801FF	512 B
MCU_I3C0_VBP2APB_WRAP_CORE_VBP_MIPI_I3C_MST	0x0040B88000	0x0040B883FF	1 KB
MCU_I3C1_MMR_MMRVBP	0x0040B90000	0x0040B901FF	512 B
MCU_I3C1_VBP2APB_WRAP_CORE_VBP_MIPI_I3C_MST	0x0040B98000	0x0040B983FF	1 KB
MCU_EFUSE0	0x0040C00000	0x0040C000FF	256 B
MCU_PLL0_CFG	0x0040D00000	0x0040D03FFF	16 KB
MCU_PBI0	0x0040E00000	0x0040E003FF	1 KB
MCU_PBI1	0x0040E10000	0x0040E103FF	1 KB
MCU_CTRL_MMR0_CFG0	0x0040F00000	0x0040F1FFFF	128 KB
MCU_R5FSS0_CORE0_ATCM	0x0041000000	0x0041007FFF	32 KB
MCU_R5FSS0_CORE0_BTCM	0x0041010000	0x0041017FFF	32 KB
MCU_R5FSS0_CORE1_ATCM <sup>(1)</sup>	0x0041400000	0x0041407FFF	32 KB
MCU_R5FSS0_CORE1_BTCM <sup>(1)</sup>	0x0041410000	0x0041417FFF	32 KB
MCU_ROM0	0x0041800000	0x004183FFFF	256 KB
MCU_MSRAM_1MB0_RAM	0x0041C00000	0x0041CFFFFFFF	1 MB
MCU_CBASS0_FW	0x0045100000	0x004517FFFF	512 KB
MCU_NAVSS0_MODSS_DMSC_FW	0x0045600000	0x004563FFFF	256 KB
MCU_SEC_MMR0_DBG_CTRL	0x0045950000	0x00459503FF	1 KB
MCU_SEC_MMR0_CFG0	0x0045A50000	0x0045A503FF	1 KB
MCU_NAVSS0_MODSS_DMSC_GLB	0x0045B04000	0x0045B043FF	1 KB
MCU_CBASS0_GLB	0x0045B06000	0x0045B063FF	1 KB
MCU_CPSW0_NUSS	0x0046000000	0x00461FFFFFFF	2 MB
MCU_FSS0_CFG	0x0047000000	0x00470000FF	256 B
MCU_FSS0_FSAS_CFG	0x0047010000	0x00470100FF	256 B
MCU_FSS0_HPB_SS_CFG	0x0047030000	0x00470300FF	256 B
MCU_FSS0_HPB_CTRL	0x0047034000	0x00470340FF	256 B
MCU_FSS0_OSPI0_CTRL	0x0047040000	0x00470400FF	256 B
MCU_FSS0_OSPI0_SS_CFG	0x0047044000	0x00470441FF	512 B
MCU_FSS0_OSPI1_CTRL	0x0047050000	0x00470500FF	256 B
MCU_FSS0_OSPI1_SS_CFG	0x0047054000	0x00470541FF	512 B
MCU_FSS0_HPB_ECC_AGGR	0x0047060000	0x00470603FF	1 KB
MCU_FSS0_OSPI1_ECC_AGGR	0x0047064000	0x00470643FF	1 KB
MCU_FSS0_OSPI0_ECC_AGGR	0x0047068000	0x00470683FF	1 KB
MCU_CBASS0_ERR	0x0047100000	0x00471003FF	1 KB
MCU_CBASS_DEBUG0_ERR	0x0047104000	0x00471043FF	1 KB
MCU_CBASS_FW0_ERR	0x0047108000	0x00471083FF	1 KB
MCU_ECC_AGGR0_ECC_AGGR	0x0047200000	0x00472003FF	1 KB
MCU_FSS0_DAT_REG1	0x0050000000	0x0057FFFFFFF	128 MB
MCU_FSS0_OSPI1_R1	0x0058000000	0x005FFFFFFF	128 MB
MCU_FSS0_DAT_REG0	0x0400000000	0x04FFFFFFF	4 GB
MCU_FSS0_DAT_REG3	0x0500000000	0x05FFFFFFF	4 GB



**Table 2-3. MCU Domain Memory Map (continued)**

Region Name	Start Address	End Address	Size
MCU_FSS0_OSPI1_R0	0x0600000000	0x06FFFFFFF	4 GB
MCU_FSS0_OSPI1_R3	0x0700000000	0x07FFFFFFF	4 GB
MCU_CPT2_AGGR0_MMR	0x4C3E000000	0x4C3E0000FF	256 B
MCU_CPT2_AGGR0_MEM0	0x4C3E020000	0x4C3E020FFF	4 KB
MCU_CPT2_AGGR0_MEM1	0x4C3E021000	0x4C3E021FFF	4 KB
MCU_CPT2_AGGR0_MEM2	0x4C3E022000	0x4C3E022FFF	4 KB
MCU_CPT2_AGGR0_MEM3	0x4C3E023000	0x4C3E023FFF	4 KB
MCU_CPT2_AGGR0_MEM4	0x4C3E024000	0x4C3E024FFF	4 KB
MCU_CPT2_AGGR0_MEM5	0x4C3E025000	0x4C3E025FFF	4 KB
MCU_CPT2_AGGR0_MEM6	0x4C3E026000	0x4C3E026FFF	4 KB
MCU_CPT2_AGGR0_MEM7	0x4C3E027000	0x4C3E027FFF	4 KB
MCU_CPT2_AGGR0_MEM8	0x4C3E028000	0x4C3E028FFF	4 KB
MCU_CPT2_AGGR0_MEM9	0x4C3E029000	0x4C3E029FFF	4 KB
MCU_CPT2_AGGR0_MEM10	0x4C3E02A000	0x4C3E02AFFF	4 KB
MCU_CPT2_AGGR0_MEM11	0x4C3E02B000	0x4C3E02BFFF	4 KB
MCU_CPT2_AGGR0_MEM12	0x4C3E02C000	0x4C3E02CFFF	4 KB
MCU_CPT2_AGGR0_MEM13	0x4C3E02D000	0x4C3E02DFFF	4 KB
MCU_CPT2_AGGR0_MEM14	0x4C3E02E000	0x4C3E02EFFF	4 KB
MCU_CPT2_AGGR0_MEM15	0x4C3E02F000	0x4C3E02FFFF	4 KB
MCU_CPT2_AGGR0_MEM16	0x4C3E030000	0x4C3E030FFF	4 KB
MCU_CPT2_AGGR0_MEM17	0x4C3E031000	0x4C3E031FFF	4 KB
MCU_CPT2_AGGR0_MEM18	0x4C3E032000	0x4C3E032FFF	4 KB
MCU_CPT2_AGGR0_MEM19	0x4C3E033000	0x4C3E033FFF	4 KB
MCU_CPT2_AGGR0_MEM20	0x4C3E034000	0x4C3E034FFF	4 KB
MCU_CPT2_AGGR0_MEM21	0x4C3E035000	0x4C3E035FFF	4 KB
MCU_CPT2_AGGR0_MEM22	0x4C3E036000	0x4C3E036FFF	4 KB
MCU_CPT2_AGGR0_MEM23	0x4C3E037000	0x4C3E037FFF	4 KB
MCU_CPT2_AGGR0_MEM24	0x4C3E038000	0x4C3E038FFF	4 KB
MCU_CPT2_AGGR0_MEM25	0x4C3E039000	0x4C3E039FFF	4 KB
MCU_CPT2_AGGR0_MEM26	0x4C3E03A000	0x4C3E03AFFF	4 KB
MCU_CPT2_AGGR0_MEM27	0x4C3E03B000	0x4C3E03BFFF	4 KB
MCU_CPT2_AGGR0_MEM28	0x4C3E03C000	0x4C3E03CFFF	4 KB
MCU_CPT2_AGGR0_MEM29	0x4C3E03D000	0x4C3E03DFFF	4 KB
MCU_CPT2_AGGR0_MEM30	0x4C3E03E000	0x4C3E03EFFF	4 KB
MCU_CPT2_AGGR0_MEM31	0x4C3E03F000	0x4C3E03FFFF	4 KB
MCU_R5FSS0_CORE0_ICACHE	0x5400000000	0x54007FFFFFF	8 MB
MCU_R5FSS0_CORE0_DCACHE	0x5400800000	0x5400FFFFFFF	8 MB
MCU_R5FSS0_CORE1_ICACHE	0x5401000000	0x54017FFFFFF	8 MB
MCU_R5FSS0_CORE1_DCACHE	0x5401800000	0x5401FFFFFFF	8 MB

- (1) These regions are used when MCU\_R5FSS0 works in split mode. For more information about split and lockstep modes, see [Section 6.3.3.2 MCU Cortex-R5F Core](#). For more information about ATCM and BTCM, see [Section 6.3.3.2.2 Tightly-Coupled Memories \(TCMs\)](#).

## 2.3 WKUP Domain Memory Map

Table 2-4 shows the WKUP domain memory map.

### Note

The memory locations not shown in Table 2-4 are either unallocated or reserved and not used. Accesses to these locations are not recommended and should be avoided.

**Table 2-4. WKUP Domain Memory Map**

Region Name	Start Address	End Address	Size
WKUP_PSC0	0x0042000000	0x0042000FFF	4 KB
WKUP_PLLCTRL0	0x0042010000	0x00420101FF	512 B
WKUP_VTM0_MMR_VBUSP_CFG1	0x0042040000	0x00420403FF	1 KB
WKUP_VTM0_MMR_VBUSP_CFG2	0x0042050000	0x00420503FF	1 KB
WKUP_DDPA0	0x0042060000	0x00420603FF	1 KB
WKUP_ESM0_CFG	0x0042080000	0x0042080FFF	4 KB
WKUP_GPIO1	0x0042100000	0x00421000FF	256 B
WKUP_GPIO0	0x0042110000	0x00421100FF	256 B
WKUP_I2C0_CFG	0x0042120000	0x00421200FF	256 B
WKUP_GPIOMUX_INTRTR0_CFG	0x0042200000	0x00422003FF	1 KB
WKUP_UART0	0x0042300000	0x00423001FF	512 B
WKUP_CBASS0_ERR	0x0042400000	0x00424003FF	1 KB
WKUP_CBASS_FW0_ERR	0x0042404000	0x00424043FF	1 KB
WKUP_ECC_AGGR0_ECC_AGGR	0x0042410000	0x00424103FF	1 KB
WKUP_VTM0_ECCAGGR_CFG	0x0042810000	0x00428103FF	1 KB
WKUP_TIMEOUT_INFRA0_CFG	0x0042900000	0x00429003FF	1 KB
WKUP_CTRL_MMR0_CFG0	0x0043000000	0x004301FFFF	128 KB
WKUP_CBASS0_FW	0x0045020000	0x004502FFFF	64 KB
WKUP_CBASS0_GLB	0x0045B02000	0x0045B023FF	1 KB

## 2.4 Processors View Memory Map

Table 2-5 through Table 2-8 show the processors view memory maps.

### Note

The memory locations not shown in Table 2-5 through Table 2-8 are either unallocated or reserved and not used. Accesses to these locations are not recommended and should be avoided.

**Table 2-5. R5FSS0/1 Memory Map**

Region Name	Start Address	End Address	Size
ARMSS_ATCM	0x00000000	0x00007FFF	32 KB
ARMSS_RAT_REGION0	0x00008000	0x01FFFFFF	32736 KB
ARMSS_NON_RAT_SOC_REGION0	0x02000000	0x02FFFFFF	16 MB
ARMSS_RAT_REGION1	0x03000000	0x0BFFFFFF	144 MB
ARMSS_NON_RAT_SOC_REGION1	0x0C000000	0x0FF7FFFF	65024 KB
ARMSS_VIC_CFG	0x0FF80000	0x0FF83FFF	16 KB
ARMSS_NON_RAT_SOC_REGION2	0x0FF84000	0x0FF8FFFF	48 KB
ARMSS_RAT_CFG	0x0FF90000	0x0FF90FFF	4 KB
ARMSS_NON_RAT_SOC_REGION3	0x0FF91000	0x0FFFFFFF	444 KB
ARMSS_RAT_REGION2	0x10000000	0x4100FFFF	802880 KB
ARMSS_BTCM	0x41010000	0x41017FFF	32 KB
ARMSS_RAT_REGION3	0x41018000	0x7FFFFFFF	1032096 KB
ARMSS_RAT_REGION4	0x80000000	0xFFFFFFFF	2 GB

**Table 2-6. C66SS0/1 Memory Map**

Region Name	Start Address	End Address	Size
C66_COREPAC_L2	0x00800000	0x00847FFF	288 KB
C66_COREPAC_L1P	0x00E00000	0x00E07FFF	32 KB
C66_COREPAC_L1D	0x00F00000	0x00F07FFF	32 KB
C66_COREPAC_ICFG	0x01000000	0x01BFFFFFFF	12 MB
C66_COREPAC_EXTERNAL_CFG	0x01C00000	0x025FFFFFFF	10 MB
C66_COREPAC_C66_RATCFG	0x07FF0000	0x07FFFFFFF	64 KB
C66_COREPAC_XMC_MMR	0x08000000	0x0800FFFF	64 KB
C66_COREPAC_XMC EDI	0x08010000	0x0801FFFF	64 KB
C66_COREPAC_RAT_REGION	0x20000000	0xFFFFFFFF	3584 MB

**Table 2-7. PRU\_ICSSG0/1 Memory Map**

Region Name	Start Address	End Address	Size
PRU_ICSSG_RAT_REGION0	0x0040000	0xFFFFFFFF	4194048 KB
PRU_ICSSG0_DRAM0_SLV_RAM	0xB000000	0xB001FFF	8 KB
PRU_ICSSG0_DRAM1_SLV_RAM	0xB002000	0xB003FFF	8 KB
PRU_ICSSG0_PR1_RTU0_PR1_RTU0_IRAM_RAM	0xB004000	0xB005FFF	8 KB
PRU_ICSSG0_PR1_RTU1_PR1_RTU1_IRAM_RAM	0xB006000	0xB007FFF	8 KB
PRU_ICSSG0_RAT_SLICE0_CFG	0xB008000	0xB008FFF	4 KB
PRU_ICSSG0_RAT_SLICE1_CFG	0xB009000	0xB009FFF	4 KB
PRU_ICSSG0_RAM_SLV_RAM	0xB010000	0xB01FFFF	64 KB
PRU_ICSSG0_PR1_ICSS_INTC_INTC_SLV	0xB020000	0xB021FFF	8 KB
PRU_ICSSG0_PR1_PDSP0_IRAM	0xB022000	0xB0220FF	256 B
PRU_ICSSG0_PR1_PDSP0_IRAM_DEBUG	0xB022400	0xB0224FF	256 B

**Table 2-7. PRU\_ICSSG0/1 Memory Map (continued)**

Region Name	Start Address	End Address	Size
PRU_ICSSG0_PR1_RTU0_PR1_RTU0_IRAM	0xB023000	0xB0230FF	256 B
PRU_ICSSG0_PR1_RTU0_PR1_RTU0_IRAM_DEBUG	0xB023400	0xB0234FF	256 B
PRU_ICSSG0_PR1_RTU1_PR1_RTU1_IRAM	0xB023800	0xB0238FF	256 B
PRU_ICSSG0_PR1_RTU1_PR1_RTU1_IRAM_DEBUG	0xB023C00	0xB023CFF	256 B
PRU_ICSSG0_PR1_PDSP1_IRAM	0xB024000	0xB0240FF	256 B
PRU_ICSSG0_PR1_PDSP1_IRAM_DEBUG	0xB024400	0xB0244FF	256 B
PRU_ICSSG0_PR1_PROT_SLV	0xB024C00	0xB024CFF	256 B
PRU_ICSSG0_PR1_CFG_SLV	0xB026000	0xB0261FF	512 B
PRU_ICSSG0_PA_STAT_WRAP_PA_SLV_QSTAT	0xB027000	0xB027FFF	4 KB
PRU_ICSSG0_PR1_ICSS_UART_UART_SLV	0xB028000	0xB02803F	64 B
PRU_ICSSG0_PR1_TASKS_MGR_PRU0_PR1_TASKS_MGR_PRU0_MMR	0xB02A000	0xB02A0FF	256 B
PRU_ICSSG0_PR1_TASKS_MGR_RTU0_PR1_TASKS_MGR_RTU0_MMR	0xB02A100	0xB02A1FF	256 B
PRU_ICSSG0_PR1_TASKS_MGR_PRU1_PR1_TASKS_MGR_PRU1_MMR	0xB02A200	0xB02A2FF	256 B
PRU_ICSSG0_PR1_TASKS_MGR_RTU1_PR1_TASKS_MGR_RTU1_MMR	0xB02A300	0xB02A3FF	256 B
PRU_ICSSG0_PA_STAT_WRAP_PA_SLV_CSTAT	0xB02C000	0xB02CFFF	4 KB
PRU_ICSSG0_IEP0	0xB02E000	0xB02EFFF	4 KB
PRU_ICSSG0_IEP1	0xB02F000	0xB02FFFF	4 KB
PRU_ICSSG0_PR1_ICSS_ECAP0_ECAP_SLV	0xB030000	0xB0300FF	256 B
PRU_ICSSG0_PR1_MII_RT_PR1_MII_RT_CFG	0xB032000	0xB0320FF	256 B
PRU_ICSSG0_PR1_MII_RT_PR1_SGMII0_CFG_SGMII0	0xB032100	0xB0321FF	256 B
PRU_ICSSG0_PR1_MII_RT_PR1_SGMII1_CFG_SGMII1	0xB032200	0xB0322FF	256 B
PRU_ICSSG0_PR1_MDIO_V1P7_MDIO	0xB032400	0xB0324FF	256 B
PRU_ICSSG0_PR1_MII_RT_PR1_MII_RT_G_CFG_REGS_G	0xB033000	0xB033FFF	4 KB
PRU_ICSSG0_PR1_PDSP0_IRAM_RAM	0xB034000	0xB037FFF	16 KB
PRU_ICSSG0_PR1_PDSP1_IRAM_RAM	0xB038000	0xB03BFFF	16 KB
PRU_ICSSG0_PA_STAT_WRAP_PA_SLV	0xB03C000	0xB03C0FF	256 B

**Table 2-8. MCU\_R5FSS0 Memory Map**

Region Name	Start Address	End Address	Size
MCU_ARMSS_ATCM	0x00000000	0x00007FFF	32 KB
MCU_ARMSS_RAT_REGION0	0x00008000	0x40007FFF	1 GB
MCU_ARMSS_NON_RAT_SOC_REGION0	0x40008000	0x40F87FFF	15872 KB
MCU_ARMSS_VIC_CFG	0x40F80000	0x40F83FFF	16 KB
MCU_ARMSS_NON_RAT_SOC_REGION1	0x40F84000	0x40F8FFFF	48 KB
MCU_ARMSS_RAT_CFG	0x40F90000	0x40F90FFF	4 KB
MCU_ARMSS_NON_RAT_SOC_REGION2	0x40F91000	0x40FFFFFF	444 KB
MCU_ARMSS_RAT_REGION1	0x41000000	0x4100FFFF	64 KB
MCU_ARMSS_BTCM	0x41010000	0x41017FFF	32 KB
MCU_ARMSS_RAT_REGION3	0x41018000	0x7FFFFFFF	1032096 KB
MCU_ARMSS_RAT_REGION4	0x80000000	0xFFFFFFFF	2 GB

## 2.5 Region-based Address Translation

The SoC memory map view shown in [Section 2.1](#), [Section 2.2](#) and [Section 2.3](#) is used by the processors in COMPUTE\_CLUSTER0 but there are processors which support only 32-bit addressing and cannot access higher address ranges. For these processors RAT modules are integrated to remap the 32-bit addresses to

48-bit addresses allowing higher address range accesses. The "RAT\_REGION" regions shown in [Table 2-5](#) through [Table 2-8](#) are dedicated to RAT remapping.

For more information about the RAT module, see [Section 8.4 Region-based Address Translation \(RAT\) Module](#).



## Chapter 3

# System Interconnect

---



The following sections describe the device system interconnect.

<b>3.1 System Interconnect Overview.....</b>	<b>182</b>
<b>3.2 System Interconnect Integration.....</b>	<b>184</b>
<b>3.3 System Interconnect Functional Description.....</b>	<b>189</b>

### 3.1 System Interconnect Overview

All modules and subsystems in the device communicate with each other through the system interconnect for any memory map accesses. It is partitioned into the following sections:

- CBASS0 interconnect
- INFRA\_CBASS0 interconnect
- MCU\_CBASS0 interconnect
- WKUP\_CBASS0 interconnect

These interconnects are used for data transfers and configuration. They are composed by switch fabrics enabling fast internal data movement. They also provide low-latency and concurrent data transfers between master and slave peripherals.

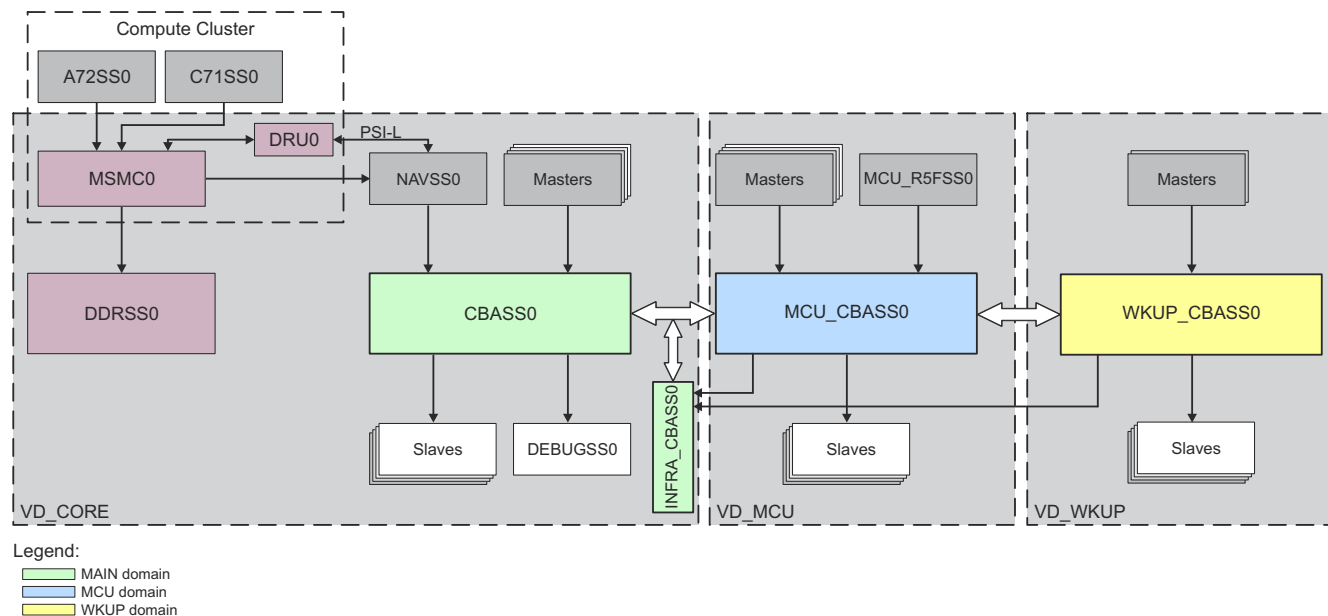
The CBASS0 is further partitioned in the following sub-interconnects:

- CBASS\_HC2\_0
- CBASS\_AC0
- CBASS\_DATADEBUG0
- CBASS\_HC0
- CBASS\_CSI0
- CBASS\_HC\_CFG0
- CBASS\_MCASP\_G0\_0
- CBASS\_MCASP\_G1\_0
- CBASS\_RC0
- CBASS\_RC\_CFG0
- CBASS\_AASRC0
- CBASS\_IPPHY0

Most of the device modules are connected to the CBASS0 interconnect that is located in VD\_CORE. These modules, INFRA\_CBASS0 and CBASS0 are part of the device MAIN domain. MCU\_CBASS0 resides in VD\_MCU and connects all modules from the MCU domain. WKUP\_CBASS0 is located in VD\_WKUP and is for modules in the WKUP domain. INFRA\_CBASS0 contains almost all infrastructure and internal diagnostic components and also peripherals not power managed in the MAIN domain.

[Figure 3-1](#) shows the device system interconnect. All modules and subsystems can be classified into two categories: masters and slaves. The masters are capable of initiating read and write transfers in the system. The

slaves on the other hand depend on the masters to perform transfers to and from them. They cannot generate read/write requests but can respond to these requests generating interrupts or DMA requests.



intconn-001

**Figure 3-1. Device System Interconnect Overview**

## 3.2 System Interconnect Integration

This section describes system interconnect integration in the device, including information about clocks, resets, and hardware requests.

### 3.2.1 Interconnect Integration in WKUP Domain

Table 3-1 through Table 3-3 summarize the integration of WKUP\_CBASS0 in device WKUP domain.

**Table 3-1. WKUP\_CBASS0 Integration Attributes**

Module Instance	Attributes		
	Power Sleep Controller	Power Domain	Module Domain
WKUP_CBASS0	WKUP_PSC0	PD0	LPSC0
WKUP_CBASS_FW0	WKUP_PSC0	PD0	LPSC0

**Table 3-2. WKUP\_CBASS0 Clocks and Resets**

Clocks			
Module Instance	Source Clock Signal	Source	Description
WKUP_CBASS0	MCU_SYSCLK0/3	WKUP_PLLCTRL0	WKUP_CBASS0 clocks
	MCU_SYSCLK0/6	WKUP_PLLCTRL0	
	MCU_SYSCLK0/12	WKUP_PLLCTRL0	
WKUP_CBASS_FW0	MCU_PLL_CLKOUT/3	MCU_PLL0	Clock for all WKUP_CBASS0 firewalls
Resets			
Module Instance	Source Reset Signal	Source	Description
WKUP_CBASS0	MOD_G_RST	LPSC0	WKUP_CBASS0 reset
WKUP_CBASS_FW0	MOD_G_RST	LPSC0	Reset for all WKUP_CBASS0 firewalls

**Table 3-3. WKUP\_CBASS0 Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
WKUP_CBASS0	WKUP_COMMON_ERR_INTR	GIC500_SPI_IN_952	GIC500	WKUP CBASS null endpoint error interrupt	Level
		R5FSS0_INTRTR0_IN_156	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_156	R5FSS1_INTRTR0		
		MCU_R5FSS0_CORE0_INTR_IN_157	MCU_R5FSS0_CORE0		
		MCU_R5FSS0_CORE1_INTR_IN_157	MCU_R5FSS0_CORE1		

**Table 3-3. WKUP\_CBASS0 Hardware Requests (continued)**

WKUP_FW_COMMON_ERR_INTR	GIC500_SPI_IN_953	GIC500	WKUP FW	CBASS null endpoint error interrupt	Level
	WKUP_DMSC0_INTR_IN_3	WKUP_DMSC0			
	R5FSS0_INTRTR0_IN_154	R5FSS0_INTRTR0			
	R5FSS1_INTRTR0_IN_154	R5FSS1_INTRTR0			
	MCU_R5FSS0_CORE0_INTR_IN_156	MCU_R5FSS0_CORE0			
	MCU_R5FSS0_CORE1_INTR_IN_156	MCU_R5FSS0_CORE1			
DMA Events					
Module Instance	Module DMA Event	Destination DMA Event Input	Destination	Description	Type
WKUP_CBASS0	-	-	-	-	-

### Note

For more information on the power, reset and clock management, see the corresponding sections within [Chapter 5, Device Configuration](#).

For more information on the device interrupt controllers, see [Section 9.2, Interrupt Controllers](#).

### 3.2.2 Interconnect Integration in MCU Domain

[Table 3-4](#) through [Table 3-6](#) summarize the integration of MCU\_CBASS0 in device MCU domain.

**Table 3-4. MCU\_CBASS0 Integration Attributes**

Module Instance	Attributes		
	Power Sleep Controller	Power Domain	Module Domain
MCU_CBASS0	WKUP_PSC0	PD0	LPSC0
MCU_CBASS_FW0	WKUP_PSC0	PD0	LPSC0

**Table 3-5. MCU\_CBASS0 Clocks and Resets**

Clocks			
Module Instance	Source Clock Signal	Source	Description
MCU_CBASS0	MCU_SYCLK0/3	WKUP_PLLCTRL0	MCU_CBASS0 clocks
	MCU_SYCLK0/6	WKUP_PLLCTRL0	
	MCU_SYCLK0/12	WKUP_PLLCTRL0	
MCU_CBASS_FW0	MCU_SYCLK0/3 or MCU_SYCLK0/6	WKUP_PLLCTRL0	Clocks for all MCU_CBASS0 firewalls
Resets			
Module Instance	Source Reset Signal	Source	Description
MCU_CBASS0	MOD_G_RST	LPSC0	MCU_CBASS0 reset

**Table 3-5. MCU\_CBASS0 Clocks and Resets (continued)**

MCU_CBASS_FW0	MOD_G_RST	LPSC0	Reset for all MCU_CBASS0 firewalls
---------------	-----------	-------	------------------------------------

**Table 3-6. MCU\_CBASS0 Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCU_CBASS0	MCU_COMMON_ERR_INTR	GIC500_SPI_IN_920	GIC500	MCU CBASS null endpoint error interrupt	Level
		WKUP_DMSC0_INTR_IN_37	WKUP_DMSC0		
		R5FSS0_INTRTR0_IN_152	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_152	R5FSS1_INTRTR0		
		MCU_R5FSS0_CORE0_INTR_IN_150	MCU_R5FSS0_CORE0		
		MCU_R5FSS0_CORE1_INTR_IN_150	MCU_R5FSS0_CORE1		
	MCU_FW_COMMON_ERR_INTR	GIC500_SPI_IN_953	GIC500	MCU FW CBASS null endpoint error interrupt	Level
		WKUP_DMSC0_INTR_IN_3	WKUP_DMSC0		
		R5FSS0_INTRTR0_IN_154	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_154	R5FSS1_INTRTR0		
		MCU_R5FSS0_CORE0_INTR_IN_156	MCU_R5FSS0_CORE0		
		MCU_R5FSS0_CORE1_INTR_IN_156	MCU_R5FSS0_CORE1		
DMA Events					
Module Instance	Module DMA Event	Destination DMA Event Input	Destination	Description	Type
MCU_CBASS0	-	-	-	-	-

### Note

For more information on the power, reset and clock management, see the corresponding sections within [Chapter 5, Device Configuration](#).

For more information on the device interrupt controllers, see [Section 9.2, Interrupt Controllers](#).

### 3.2.3 Interconnect Integration in MAIN Domain

[Table 3-7](#) through [Table 3-9](#) summarize the integration of CBASS0 in device MAIN domain.

**Table 3-7. CBASS0 Integration Attributes**

Module Instance	Attributes		
	Power Sleep Controller	Power Domain	Module Domain
CBASS0	PSC0	PD0	LPSC0
INFRA_CBASS0	PSC0	PD0	LPSC2



**Table 3-7. CBASS0 Integration Attributes (continued)**

CBASS_FW0	PSC0	PD0	LPSC0
-----------	------	-----	-------

**Table 3-8. CBASS0 Clocks and Resets**

Clocks			
Module Instance	Source Clock Signal	Source	Description
CBASS0	MAIN_SYSCLK0	PLLCTRL0	CBASS0 and INFRA_CBASS0 clocks
INFRA_CBASS0	MAIN_SYSCLK0/2	PLLCTRL0	
	MAIN_SYSCLK0/4	PLLCTRL0	
CBASS_FW0	MAIN_SYSCLK0 or MAIN_SYSCLK0/2	PLLCTRL0	Clocks for all CBASS0 firewalls
Resets			
Module Instance	Source Reset Signal	Source	Description
CBASS0	MOD_G_RST	LPSC0	CBASS0 reset
INFRA_CBASS0	MOD_G_RST	LPSC2	INFRA_CBASS0 reset
CBASS_FW0	MOD_G_RST	LPSC0	Reset for all CBASS0 firewalls

**Table 3-9. CBASS0 Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
CBASS_CSI0	DEFAULT_ERR_INTR	GIC500_SPI_IN_793	GIC500	CBASS_CSI0 null endpoint error interrupt	Level
		WKUP_DMSC0_INTR_IN_37	WKUP_DMSC0		
		MAIN2MCU_LVL_INTRTR0_IN_156	MAIN2MCU_LVL_INTRTR0		
CBASS_HC0	DEFAULT_ERR_INTR	GIC500_SPI_IN_793	GIC500	CBASS_HC0 null endpoint error interrupt	Level
		WKUP_DMSC0_INTR_IN_37	WKUP_DMSC0		
		MAIN2MCU_LVL_INTRTR0_IN_156	MAIN2MCU_LVL_INTRTR0		
CBASS_HC_CFG0	DEFAULT_ERR_INTR	GIC500_SPI_IN_793	GIC500	CBASS_HC_CFG0 null endpoint error interrupt	Level
		WKUP_DMSC0_INTR_IN_37	WKUP_DMSC0		
		MAIN2MCU_LVL_INTRTR0_IN_156	MAIN2MCU_LVL_INTRTR0		
CBASS_HC2_0	DEFAULT_ERR_INTR	GIC500_SPI_IN_793	GIC500	CBASS_HC2_0 null endpoint error interrupt	Level
		WKUP_DMSC0_INTR_IN_37	WKUP_DMSC0		
		MAIN2MCU_LVL_INTRTR0_IN_156	MAIN2MCU_LVL_INTRTR0		
CBASS_IPPHY0	DEFAULT_ERR_INTR	GIC500_SPI_IN_793	GIC500	CBASS_IPPHY0 null endpoint error interrupt	Level
		WKUP_DMSC0_INTR_IN_37	WKUP_DMSC0		
		MAIN2MCU_LVL_INTRTR0_IN_156	MAIN2MCU_LVL_INTRTR0		

**Table 3-9. CBASS0 Hardware Requests (continued)**

CBASS_MCASP_G0_0	DEFAULT_ERR_INTR	GIC500_SPI_IN_793	GIC500	CBASS_MCASP_G0_0 null endpoint error interrupt	Level
		WKUP_DMSC0_INTR_IN_37	WKUP_DMSC0		
		MAIN2MCU_LVL_INTRTR0_IN_156	MAIN2MCU_LVL_INTRTR0		
CBASS_MCASP_G1_0	DEFAULT_ERR_INTR	GIC500_SPI_IN_793	GIC500	CBASS_MCASP_G1_0 null endpoint error interrupt	Level
		WKUP_DMSC0_INTR_IN_37	WKUP_DMSC0		
		MAIN2MCU_LVL_INTRTR0_IN_156	MAIN2MCU_LVL_INTRTR0		
CBASS_RC0	DEFAULT_ERR_INTR	GIC500_SPI_IN_793	GIC500	CBASS_RC0 null endpoint error interrupt	Level
		WKUP_DMSC0_INTR_IN_37	WKUP_DMSC0		
		MAIN2MCU_LVL_INTRTR0_IN_156	MAIN2MCU_LVL_INTRTR0		
CBASS_RC_CFG0	DEFAULT_ERR_INTR	GIC500_SPI_IN_793	GIC500	CBASS_RC_CFG0 null endpoint error interrupt	Level
		WKUP_DMSC0_INTR_IN_37	WKUP_DMSC0		
		MAIN2MCU_LVL_INTRTR0_IN_156	MAIN2MCU_LVL_INTRTR0		
CBASS_FW0	DEFAULT_ERR_INTR	GIC500_SPI_IN_953	GIC500	MAIN FW CBASS null endpoint error interrupt	Level
		WKUP_DMSC0_INTR_IN_3	WKUP_DMSC0		
		R5FSS0_INTRTR0_IN_154	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_154	R5FSS1_INTRTR0		
		MCU_R5FSS0_CORE0_INTR_IN_156	MCU_R5FSS0_CORE0		
		MCU_R5FSS0_CORE1_INTR_IN_156	MCU_R5FSS0_CORE1		
INFRA_CBASS0	CBASS_INFRA0_DEFAULT_ERR_INTR_0	GIC500_SPI_IN_791	GIC500	MAIN INFRA CBASS null endpoint error interrupt	Level
		WKUP_DMSC0_INTR_IN_37	WKUP_DMSC0		
		R5FSS0_INTRTR0_IN_324	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_324	R5FSS1_INTRTR0		
		MAIN2MCU_LVL_INTRTR0_IN_167	MAIN2MCU_LVL_INTRTR0		

**DMA Events**

Module Instance	Module DMA Event	Destination DMA Event Input	Destination	Description	Type
CBASS0	-	-	-	-	-

**Note**

For more information on the power, reset and clock management, see the corresponding sections within [Chapter 5, Device Configuration](#).

For more information on the device interrupt controllers, see [Section 9.2, Interrupt Controllers](#).

### 3.3 System Interconnect Functional Description

#### 3.3.1 Master-Slave Connections

Table 3-10 through Table 3-19 list the master and slave end point connections on the CBASS0. A cell may contain one of the following:

- Y – There is a connection between this master and that slave
- N – There is no connection between this master and that slave.

**Table 3-10. CBASS0 Connectivity Matrix (Part 1)**

Masters	Slaves on CBASS0																											
	ATL0	COMPUTE_CLUSTER0	CPW0	CSIRX0	CSIRX1	CSITX0	STM0	DEBUGSS0_CFG	DMPAC0_CFG	DMPAC0_MEM	TIMER0	TIMER1	TIMER5	TIMER10	TIMER11	TIMER12	TIMER13	TIMER14	TIMER15	TIMER16	TIMER17	TIMER18	TIMER19	TIMER2	TIMER3	TIMER4	TIMER6	
WKUP_DMSC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
WKUP_DMSC0_FWMGR_CFG	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_R5FSS0_CORE0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
MCU_R5FSS0_CORE1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
MCU_NAVSS0_SRC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
MCU_PDMA_ADC0_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G20_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G20_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
DEBUGSS0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
C66SS0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
C66SS1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
VPAC0_DATA0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
VPAC0_DATA1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
VPAC0_LDC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
DMPAC0_DATA	Y	Y	Y	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
GPU0_M0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
GPU0_M1_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
GPU0_M1_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
GPU0_M0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
R5FSS0_CORE0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
R5FSS0_CORE1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
R5FSS1_CORE0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
R5FSS1_CORE1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	

**Table 3-10. CBASS0 Connectivity Matrix (Part 1) (continued)**

	Slaves on CBASS0																											
NAVSS0_HI	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
NAVSS0_LO	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MMCSD0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MMCSD0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DEBUGSS0_RD	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
DEBUGSS0_WR	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PRU_ICSSG0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PRU_ICSSG1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE0_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE0_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE0_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE0_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MMCSD1_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MMCSD1_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MMCSD2_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MMCSD2_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
ENCODER0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
ENCODER0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
VPFE0_DMA	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DECODER0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DECODER0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MLBSS0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
USB3SS0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
USB3SS0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
USB3SS1_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
USB3SS1_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
COMPUTE_CLUSTER0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
COMPUTE_CLUSTER0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
UFS0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
UFS0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DSS0_DMA	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DSS0_FBDC	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_DEBUG_CCMCU0_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

**Table 3-10. CBASS0 Connectivity Matrix (Part 1) (continued)**

	Slaves on CBASS0																											
PDMA_DEBUG_C660_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G20_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G20_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G30_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G30_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G20_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G20_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_AASRC0_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_AASRC0_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

**Table 3-11. CBASS0 Connectivity Matrix (Part 2)**

	Slaves on CBASS0																											
Masters	TIMER 7	TIMER 8	TIMER 9	ECAP 0	ECAP 1	ECAP 2	EPWM 0	HRPWM 0	EPWM 1	HRPWM 1	EPWM 2	HRPWM 2	EPWM 3	HRPWM 3	EPWM 4	HRPWM 4	EPWM 5	HRPWM 5	ELM 0	MMCS D 0	MMCS D 1	MMCS D 2	EQEP 0	EQEP 1	EQEP 2	GP M C 0	I3 C 0	
																				CFG	CFG	CFG						
WKUP_DMSC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
WKUP_DMSC0_FWMGR_CFG	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_R5FSS0_CORE0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
MCU_R5FSS0_CORE1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
MCU_NAVSS0_SRC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
MCU_PDMA_ADC0_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G20_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G20_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
DEBUGSS0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
C66SS0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	

[illegible]



**Table 3-11. CBASS0 Connectivity Matrix (Part 2) (continued)**

	Slaves on CBASS0																											
DECODER0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MLBSS0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
USB3SS0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
USB3SS0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
USB3SS1_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
USB3SS1_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
COMPUTE_CLUSTER0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
COMPUTE_CLUSTER0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
UFS0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
UFS0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DSS0_DMA	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DSS0_FBDC	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_DEBUG_CCMCU0_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_DEBUG_C660_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G20_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G20_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G30_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G30_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G20_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G20_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_AASRC0_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_AASRC0_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

**Table 3-12. CBASS0 Connectivity Matrix (Part 3)**

Masters	Slaves on CBASS0																											
	P R U _ C S S G 0	P R U _ C S S G 1	C 6 6 S S 0 _ D E B U G _ C E L L	C 6 6 S S 1 _ D E B U G _ C E L L	C B A S S _ A C C 0 _ E R R	C B A S S _ R C C 0 _ E R R	C B A S S _ H C C 0 _ E R R	C B A S S _ H C _ C F G 0 _ E R R	C B A S S _ A A S R C 0 _ E R R	C B A S S _ C S I 0 _ E R R	C B A S S _ D A T A D E B U G 0 _ E R R	C B A S S _ P H Y 0 _ E R R	C B A S S _ M C A S P _ G 0 _ 0 _ E R R	C B A S S _ M C A S P _ G 1 _ 0 _ E R R	C B A S S _ R C _ C F G 0 _ E R R	C B A S S _ A C _ C F G 0 _ E R R	C B A S S _ H C 2 _ 0 _ E R R	A A S R C 0 _ C F G	A A S R C 0 _ D A T A	C 6 6 S S 0 _ S R C	C 6 6 S S 0 _ C F G	C 6 6 S S 1 _ S R C	C 6 6 S S 1 _ C F G	C C _ D E B U G _ C E L L 0	D E C O D E R 0	D P H Y _ R X 0		
WKUP_DMSC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
WKUP_DMSC0_FWMGR_CFG	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_R5FSS0_CORE0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
MCU_R5FSS0_CORE1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
MCU_NAVSS0_SRC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
MCU_PDMA_ADC0_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G20_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G20_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
DEBUGSS0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
C66SS0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	
C66SS1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	
VPAC0_DATA0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
VPAC0_DATA1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
VPAC0_LDC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
DMPAC0_DATA	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
GPU0_M0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
GPU0_M1_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
GPU0_M1_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
GPU0_M0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
R5FSS0_CORE0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
R5FSS0_CORE1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
R5FSS1_CORE0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
R5FSS1_CORE1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
NAVSS0_HI	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
NAVSS0_LO	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
MMCSD0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MMCSD0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	

**Table 3-12. CBASS0 Connectivity Matrix (Part 3) (continued)**

	Slaves on CBASS0																											
DEBUGSS0_RD	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
DEBUGSS0_WR	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PRU_ICSSG0	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PRU_ICSSG1	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE0_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE0_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE0_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE0_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MMCSD1_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MMCSD1_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MMCSD2_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MMCSD2_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
ENCODER0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
ENCODER0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
VPFE0_DMA	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DECODER0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DECODER0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MLBSS0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
USB3SS0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
USB3SS0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
USB3SS1_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
USB3SS1_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
COMPUTE_CLUSTER0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
COMPUTE_CLUSTER0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
UFS0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
UFS0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DSS0_DMA	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DSS0_FBDC	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_DEBUG_CCMCU0_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N
PDMA_DEBUG_C660_MEMR0	N	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

**Table 3-12. CBASS0 Connectivity Matrix (Part 3) (continued)**

	Slaves on CBASS0																											
PDMA_MCASP_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G20_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G20_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G30_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G30_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G20_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G20_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_AASRC0_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N	N	N	N
PDMA_AASRC0_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N	N	N	N	N

**Table 3-13. CBASS0 Connectivity Matrix (Part 4)**

	Slaves on CBASS0																											
Masters	D P H Y  R X 1	D S S 0  I N S T 0  C F G	D S I 0  C F G	E D P 0  C F G	E N C O D E R 0	M A I N _ D E B U G  C E L L 0	M A I N _ D E B U G  C E L L 1	P B I S T  _A C  _D E C O D E R  _C F G	P B I S T  _A C  _D E M P A C  _C F G	P B I S T  _E D P D S I  _C F G	P B I S T  _A C  _E N C O D E R  _C F G	P B I S T  _V P A C  _C F G	P B I S T  _H C 0  _C F G	P B I S T  _I N F R A 0  _C F G	P B I S T  _R C  _N A V S S  _C F G	P B I S T  _R C  _N B  _C F G	P B I S T  _R C  _R 5 F S S 0  _C F G	P B I S T  _R C  _R 5 F S S 1  _C F G	M C A N 0	M C A N 1	M C A N 2	M C A N 3	M C A N 4	M C A N 5	M C A N 6	M C A N 7		
WKUP_DMSC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
WKUP_DMSC0_FWMGR_CFG	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MCU_R5FSS0_CORE0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
MCU_R5FSS0_CORE1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
MCU_NAVSS0_SRC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
MCU_PDMA_ADC0_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G20_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G20_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	

[illegible]

**Table 3-13. CBASS0 Connectivity Matrix (Part 4) (continued)**

	Slaves on CBASS0																											
VPFE0_DMA	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DECODER0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DECODER0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MLBSS0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
USB3SS0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
USB3SS0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
USB3SS1_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
USB3SS1_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
COMPUTE_CLUSTER0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
COMPUTE_CLUSTER0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
UFS0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
UFS0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DSS0_DMA	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DSS0_FBDC	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_DEBUG_CCMCU0_MEMR0	N	N	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_DEBUG_C660_MEMR0	N	N	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	Y	N	N	N	N
PDMA_MISC_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	Y	N	N	N	N
PDMA_MISC_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	Y	N	N	N
PDMA_MISC_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	Y	N	N	N
PDMA_MISC_G20_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	Y	N	N	N
PDMA_MISC_G20_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	Y	N	N	N
PDMA_MISC_G30_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	Y	N	N
PDMA_MISC_G30_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	Y	N	N
PDMA_USART_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G20_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G20_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_AASRC0_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_AASRC0_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N



	Slaves on CBASS0																											
Masters	MCAN8	MCAN9	MCAN10	MCAN11	MCAN12	MCAN13	MCASP0_CFG	MCASP0_DMA	MCASP1_CFG	MCASP1_DMA	MCASP10_CFG	MCASP10_DMA	MCASP11_CFG	MCASP11_DMA	MCASP12_CFG	MCASP12_DMA	MCASP13_CFG	MCASP13_DMA	MCASP14_CFG	MCASP14_DMA	MCASP15_CFG	MCASP15_DMA	MCASP16_CFG	MCASP16_DMA	MCASP17_CFG	MCASP17_DMA	MCASP18_CFG	MCASP18_DMA
WKUP_DMSC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
WKUP_DMSC0_FWMGR_CFG	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_R5FSS0_CORE0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
MCU_R5FSS0_CORE1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
MCU_NAVSS0_SRC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
MCU_PDMA_ADC0_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G20_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G20_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
DEBUGSS0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
C66SS0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
C66SS1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
VPAC0_DATA0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
VPAC0_DATA1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
VPAC0_LDC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
DMPAC0_DATA	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
GPU0_M0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
GPU0_M1_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
GPU0_M1_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
GPU0_M0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
R5FSS0_CORE0	Y	Y	Y	Y</																								

**Table 3-14. CBASS0 Connectivity Matrix (Part 5) (continued)**

	Slaves on CBASS0																											
PCIE0_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE0_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MMCSD1_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MMCSD1_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MMCSD2_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MMCSD2_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
ENCODER0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
ENCODER0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
VPFE0_DMA	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DECODER0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DECODER0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MLBSS0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
USB3SS0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
USB3SS0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
USB3SS1_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
USB3SS1_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
COMPUTE_CLUSTER0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
COMPUTE_CLUSTER0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
UFS0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
UFS0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DSS0_DMA	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DSS0_FBDC	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_DEBUG_CCMCU0_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_DEBUG_C660_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G00_MEMR0	N	N	N	N	N	N	Y	Y	Y	Y	N	N	N	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G00_MEMW0	N	N	N	N	N	N	Y	Y	Y	Y	N	N	N	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	Y	Y	Y	Y	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PDMA_MCASP_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	Y	Y	Y	Y	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PDMA_MISC_G00_MEMR0	Y	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G00_MEMW0	Y	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G10_MEMR0	N	Y	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G10_MEMW0	N	Y	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G20_MEMR0	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

**Table 3-14. CBASS0 Connectivity Matrix (Part 5) (continued)**

	Slaves on CBASS0																											
PDMA_MISC_G20_MEMW0	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G30_MEMR0	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G30_MEMW0	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G20_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G20_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_AASRC0_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_AASRC0_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

**Table 3-15. CBASS0 Connectivity Matrix (Part 6)**

	Slaves on CBASS0																											
Masters	M C A S P 8  D M A	M C A S P 9  C F G	M C A S P 9  D M A	M L B S S 0  C F G	I2 C 0	I2 C 1	I2 C 2	I2 C 3	I2 C 4	I2 C 5	I2 C 6	N A V S S 0  D D R	N A V S S 0  S R A M	N A V S S 0  N B S S  C F G	P C I E 0  C F G	P C I E 0  H P	P C I E 0  L P	P C I E 1  C F G	P C I E 1  H P	P C I E 1  L P	P C I E 2  C F G	P C I E 2  H P	P C I E 2  L P	P C I E 3  C F G	P C I E 3  H P	P C I E 3  L P	P D M A  A S R C  P S I L O  C F G	
WKUP_DMSC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
WKUP_DMSC0_FWMGR_CFG	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MCU_R5FSS0_CORE0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MCU_R5FSS0_CORE1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MCU_NAVSS0_SRC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MCU_PDMA_ADC0_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MCU_PDMA_MISC_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MCU_PDMA_MISC_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MCU_PDMA_MISC_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MCU_PDMA_MISC_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MCU_PDMA_MISC_G20_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MCU_PDMA_MISC_G20_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DEBUGSS0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
C66SS0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
C66SS1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
VPAC0_DATA0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
VPAC0_DATA1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
VPAC0_LDC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

**Table 3-15. CBASS0 Connectivity Matrix (Part 6) (continued)**

	Slaves on CBASS0																												
DMPAC0_DATA	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
GPU0_M0_WR	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
GPU0_M1_WR	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
GPU0_M1_RD	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
GPU0_M0_RD	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
R5FSS0_CORE0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
R5FSS0_CORE1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
R5FSS1_CORE0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
R5FSS1_CORE1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
NAVSS0_HI	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
NAVSS0_LO	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
MMCSD0_WR	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MMCSD0_RD	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DEBUGSS0_RD	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
DEBUGSS0_WR	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
PRU_ICSSG0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
PRU_ICSSG1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
PCIE0_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
PCIE0_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
PCIE0_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
PCIE0_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
PCIE1_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	
PCIE1_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	
PCIE1_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	
PCIE1_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	
PCIE2_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y	Y	Y	Y	Y	
PCIE2_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y	Y	Y	Y	Y	
PCIE2_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y	Y	Y	Y	Y	
PCIE2_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y	Y	Y	Y	Y	
PCIE3_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y	Y	
PCIE3_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y	
PCIE3_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y	
PCIE3_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y	
MMCSD1_WR	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MMCSD1_RD	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MMCSD2_WR	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MMCSD2_RD	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
ENCODER0_RD	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
ENCODER0_WR	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
VPFE0_DMA	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DECODER0_RD	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DECODER0_WR	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MLBSS0	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
USB3SS0_RD	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
USB3SS0_WR	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

[illegible][illegible]

**Table 3-16. CBASS0 Connectivity Matrix (Part 7) (continued)**

	Slaves on CBASS0																			
MCU_R5FSS0_CORE0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MCU_R5FSS0_CORE1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MCU_NAVSS0_SRC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MCU_PDMA_ADC0_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MCU_PDMA_MISC_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MCU_PDMA_MISC_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MCU_PDMA_MISC_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MCU_PDMA_MISC_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MCU_PDMA_MISC_G20_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MCU_PDMA_MISC_G20_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DEBUGSS0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
C66SS0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
C66SS1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
VPAC0_DATA0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
VPAC0_DATA1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
VPAC0_LDC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
DMPAC0_DATA	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
GPU0_M0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
GPU0_M1_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
GPU0_M1_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
GPU0_M0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
R5FSS0_CORE0	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
R5FSS0_CORE1	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
R5FSS1_CORE0	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
R5FSS1_CORE1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
NAVSS0_HI	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
NAVSS0_LO	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MMCSD0_WR	N	N	N	N	N	Y	N	Y	N	Y	N	Y	N	N	N	N	N	N	N	N
MMCSD0_RD	N	N	N	N	N	Y	N	Y	N	Y	N	Y	N	N	N	N	N	N	N	N
DEBUGSS0_RD	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
DEBUGSS0_WR	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PRU_ICSSG0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PRU_ICSSG1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE0_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE0_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE0_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE0_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y



**Table 3-16. CBASS0 Connectivity Matrix (Part 7) (continued)**

	Slaves on CBASS0																			
PCIE3_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MMCSD1_WR	N	N	N	N	N	Y	N	Y	N	Y	N	Y	N	N	N	N	N	N	N	N
MMCSD1_RD	N	N	N	N	N	Y	N	Y	N	Y	N	Y	N	N	N	N	N	N	N	N
MMCSD2_WR	N	N	N	N	N	Y	N	Y	N	Y	N	Y	N	N	N	N	N	N	N	N
MMCSD2_RD	N	N	N	N	N	Y	N	Y	N	Y	N	Y	N	N	N	N	N	N	N	N
ENCODER0_RD	N	N	N	N	N	Y	N	Y	N	Y	N	Y	N	N	N	N	N	N	N	N
ENCODER0_WR	N	N	N	N	N	Y	N	Y	N	Y	N	Y	N	N	N	N	N	N	N	N
VPFE0_DMA	N	N	N	N	N	Y	N	Y	N	Y	N	Y	N	N	N	N	N	N	N	N
DECODER0_RD	N	N	N	N	N	Y	N	Y	N	Y	N	Y	N	N	N	N	N	N	N	N
DECODER0_WR	N	N	N	N	N	Y	N	Y	N	Y	N	Y	N	N	N	N	N	N	N	N
MLBSS0	N	N	N	N	N	Y	N	Y	N	Y	N	Y	N	N	N	N	N	N	N	N
USB3SS0_RD	N	N	N	N	N	Y	N	Y	N	Y	N	Y	N	N	N	N	N	N	N	N
USB3SS0_WR	N	N	N	N	N	Y	N	Y	N	Y	N	Y	N	N	N	N	N	N	N	N
USB3SS1_RD	N	N	N	N	N	Y	N	Y	N	Y	N	Y	N	N	N	N	N	N	N	N
USB3SS1_WR	N	N	N	N	N	Y	N	Y	N	Y	N	Y	N	N	N	N	N	N	N	N
COMPUTE_CLUSTER0_RD	N	N	N	N	N	Y	N	Y	N	Y	N	Y	N	N	N	N	N	N	N	N
COMPUTE_CLUSTER0_WR	N	N	N	N	N	Y	N	Y	N	Y	N	Y	N	N	N	N	N	N	N	N
UFS0_RD	N	N	N	N	N	Y	N	Y	N	Y	N	Y	N	N	N	N	N	N	N	N
UFS0_WR	N	N	N	N	N	Y	N	Y	N	Y	N	Y	N	N	N	N	N	N	N	N
DSS0_DMA	N	N	N	N	N	Y	N	Y	N	Y	N	Y	N	N	N	N	N	N	N	N
DSS0_FBDC	N	N	N	N	N	Y	N	Y	N	Y	N	Y	N	N	N	N	N	N	N	N
PDMA_DEBUG_CCMCU0_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_DEBUG_C660_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G20_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G20_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G30_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G30_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G20_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G20_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_AASRC0_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

**Table 3-16. CBASS0 Connectivity Matrix (Part 7) (continued)**

	Slaves on CBASS0																	
PDMA_AASRC0_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

**Table 3-17. CBASS0 Connectivity Matrix (Part 8)**

Masters	Slaves on CBASS0																									
	SPI0	SPI1	SPI2	SPI3	SPI4	SPI5	SPI6	SPI7	UFS0_CFG	UART0	UART1	UART2	UART3	UART4	UART5	UART6	UART7	UART8	UART9	USB3SS0_CORE	USB3SS0_USBPHY	USB3SS1_CORE	USB3SS1_USBPHY	VPAC0_CFG	VPAC0_MEM	
WKUP_DMSC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
WKUP_DMSC0_FWMGR_CFG	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_R5FSS0_CORE0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
MCU_R5FSS0_CORE1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
MCU_NAVSS0_SRC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
MCU_PDMA_ADC0_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G20_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G20_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
DEBUGSS0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
C66SS0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
C66SS1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
VPAC0_DATA0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	
VPAC0_DATA1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	
VPAC0_LDC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	
DMPAC0_DATA	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
GPU0_M0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
GPU0_M1_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
GPU0_M1_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
GPU0_M0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
R5FSS0_CORE0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
R5FSS0_CORE1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
R5FSS1_CORE0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
R5FSS1_CORE1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
NAVSS0_HI	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
NAVSS0_LO	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
MMCS0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MMCS0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	

**Table 3-17. CBASS0 Connectivity Matrix (Part 8) (continued)**

	Slaves on CBASS0																							
DEBUGSS0_RD	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
DEBUGSS0_WR	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PRU_ICSSG0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PRU_ICSSG1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE0_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE0_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE0_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE0_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MMCSD1_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MMCSD1_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MMCSD2_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MMCSD2_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
ENCODER0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
ENCODER0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
VPFE0_DMA	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DECODER0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DECODER0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MLBSS0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
USB3SS0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N	N
USB3SS0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N	N
USB3SS1_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N
USB3SS1_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N
COMPUTE_CLUSTER0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
COMPUTE_CLUSTER0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
UFS0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
UFS0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DSS0_DMA	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DSS0_FBDC	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_DEBUG_CCMCU0_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_DEBUG_C660_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

**Table 3-17. CBASS0 Connectivity Matrix (Part 8) (continued)**

	Slaves on CBASS0																											
PDMA_MCASP_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G00_MEMR0	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G00_MEMW0	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G10_MEMR0	N	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G10_MEMW0	N	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G20_MEMR0	N	N	N	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G20_MEMW0	N	N	N	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G30_MEMR0	N	N	N	N	N	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G30_MEMW0	N	N	N	N	N	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G00_MEMR0	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G00_MEMW0	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G20_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N	N
PDMA_USART_G20_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N	N
PDMA_AASRC0_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_AASRC0_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

**Table 3-18. CBASS0 Connectivity Matrix (Part 9)**

	Slaves on CBASS0																											
Masters	V P F E 0 _ R A T C F G	V P F E 0 _ C F G	D P H Y _ T X 0	D C C 4	D C C 0	D C C 1	D C C 2	D C C 3	D C C 5	D C C 6	D C C 7	E S M 0 _ C F G	G P I O 0	G P I O 1	G T C 0	C 6 S S 0 _ I N T R T R 0 _ C F G	C 6 S S 0 _ I N T R T R 1 _ C F G	C M P E V T _ I N T R T R 0 _ C F G	C T R L _ M R 0	F W _ C B A S S 0 _ E R R	G P I O M U X _ I N T R T R 0 _ C F G	I N F R A _ C B A S S 0 _ E R R	I N F R A _ E C C _ A G G R 0 _ C F G	P L L 0 _ C F G	P S C 0	M A I N 2 M C U _ L V L _ I N T R T R 0 _ C F G	M A I N 2 M C U _ P L S _ I N T R T R 0 _ C F G	
WKUP_DMSC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
WKUP_DMSC0_FWMGR_CFG	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MCU_R5FSS0_CORE0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MCU_R5FSS0_CORE1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MCU_NAVSS0_SRC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MCU_PDMA_ADC0_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MCU_PDMA_MISC_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MCU_PDMA_MISC_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MCU_PDMA_MISC_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MCU_PDMA MISC G10 MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

**Table 3-18. CBASS0 Connectivity Matrix (Part 9) (continued)**

	Slaves on CBASS0																											
MCU_PDMA_MISC_G20_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MCU_PDMA_MISC_G20_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DEBUGSS0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
C66SS0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
C66SS1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
VPAC0_DATA0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
VPAC0_DATA1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
VPAC0_LDC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
DMPAC0_DATA	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
GPU0_M0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
GPU0_M1_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
GPU0_M1_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
GPU0_M0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
R5FSS0_CORE0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
R5FSS0_CORE1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
R5FSS1_CORE0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
R5FSS1_CORE1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
NAVSS0_HI	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
NAVSS0_LO	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MMCSD0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MMCSD0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DEBUGSS0_RD	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
DEBUGSS0_WR	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PRU_ICSSG0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PRU_ICSSG1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE0_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE0_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE0_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE0_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MMCSD1_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MMCSD1_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MMCSD2_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MMCSD2_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

**Table 3-18. CBASS0 Connectivity Matrix (Part 9) (continued)**

	Slaves on CBASS0																											
ENCODER0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
ENCODER0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
VPFE0_DMA	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DECODER0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DECODER0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MLBSS0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
USB3SS0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
USB3SS0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
USB3SS1_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
USB3SS1_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
COMPUTE_CLUSTER0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
COMPUTE_CLUSTER0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
UFS0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
UFS0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DSS0_DMA	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DSS0_FBD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_DEBUG_CCMCU0_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_DEBUG_C660_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G20_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G20_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G30_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G30_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G20_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G20_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_AASRC0_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_AASRC0_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N



**Table 3-19. CBASS0 Connectivity Matrix (Part 10)**

Masters	Slaves on CBASS0																								
	TIMESYNC_INTERRUPT_CFG	EFUSE0	PLLCTRL0	PSRAM_ECC0_ECC_AGG_R	PSRAM_ECC0	PSRAM2_ECC0_ECC_AGG_R	PSRAM2_ECC0	SERDES0	SERDES1	SERDES2	SERDES3	AC6_ECC_AGG_R_CFG	HC5_ECC_AGG_R_CFG	RC4_ECC_AGG_R_CFG	RC50_ECC_AGG_R_CFG	RC50_1_ECC_AGG_R_CFG	RC51_0_ECC_AGG_R_CFG	RC51_1_ECC_AGG_R_CFG	RCNAVSS10_ECC_AGG_R_CFG	RCNAVSS11_ECC_AGG_R_CFG	GPU0_CFG	GPU0_PBS_CFG	MSRAM16_KX256_CFG	MSRAM16_KX256_CFG	
WKUP_DMSC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
WKUP_DMSC0_FWMGR_CFG	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_R5FSS0_CORE0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
MCU_R5FSS0_CORE1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
MCU_NAVSS0_SRC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
MCU_PDMA_ADC0_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G20_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G20_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
DEBUGSS0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
C66SS0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
C66SS1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
VPAC0_DATA0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
VPAC0_DATA1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
VPAC0_LDC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
DMPAC0_DATA	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
GPU0_M0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	Y	
GPU0_M1_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	Y	
GPU0_M1_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	Y	
GPU0_M0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	Y	
R5FSS0_CORE0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
R5FSS0_CORE1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
R5FSS1_CORE0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
R5FSS1_CORE1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
NAVSS0_HI	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
NAVSS0_LO	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
MMCSD0_WR	N	N	N	N	Y	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	Y	

212 J721E DRA829/TDA4VM/AM68P Processors Silicon Revision 1.1 Texas Instruments Families of Products

**Table 3-19. CBASS0 Connectivity Matrix (Part 10) (continued)**

	Slaves on CBASS0																							
PDMA_MCASP_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G20_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G20_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G30_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G30_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G20_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G20_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_AASRC0_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_AASRC0_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Table 3-20 and Table 3-21 list the master and slave end point connections on the MCU\_CBASS0. A cell may contain one of the following:

- Y – There is a connection between this master and that slave
- N – There is no connection between this master and that slave.

**Table 3-20. MCU\_CBASS0 Connectivity Matrix (Part 1)**

	Slaves on MCU_CBASS0																											
Masters	M C U A D C 0 C F G	M C U A D C 0 D M A	M C U A D C 1 C F G	M C U A D C 1 D M A	M C U C P S W 0	M C U C C C 0	M C U C C C 1	M C U C C C 2	M C U T I M E R 0	M C U T I M E R 1	M C U T I M E R 2	M C U T I M E R 3	M C U T I M E R 4	M C U T I M E R 5	M C U T I M E R 6	M C U T I M E R 7	M C U T I M E R 8	M C U T I M E R 9	M C U E S S M 0 C F G	M C U F S S 0 C F G	M C U F S S 0 S 0	M C U F S S 0 S 1	M C U 3 C 0	M C U 3 C 1	M C U C B A S S 0 E R R	M C U C T R L M M R 0		
WKUP_DMSC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
WKUP_DMSC0_FWMGR_CFG	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_R5FSS0_CORE0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
MCU_R5FSS0_CORE1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
MCU_NAVSS0_SRC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
MCU_PDMA_ADC0_MEMR0	Y	Y	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
MCU_PDMA_MISC_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	

**Table 3-20. MCU\_CBASS0 Connectivity Matrix (Part 1) (continued)**

	Slaves on MCU_CBASS0																											
MCU_PDMA_MISC_G20_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MCU_PDMA_MISC_G20_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
DEBUGSS0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
C66SS0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
C66SS1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
VPAC0_DATA0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
VPAC0_DATA1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
VPAC0_LDC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
DMPAC0_DATA	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
GPU0_M0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
GPU0_M1_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
GPU0_M1_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
GPU0_M0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
R5FSS0_CORE0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
R5FSS0_CORE1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
R5FSS1_CORE0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
R5FSS1_CORE1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
NAVSS0_HI	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
NAVSS0_LO	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MMCSD0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N	N
MMCSD0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N	N
DEBUGSS0_RD	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
DEBUGSS0_WR	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PRU_ICSSG0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PRU_ICSSG1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE0_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE0_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE0_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE0_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MMCSD1_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N	N
MMCSD1_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N	N
MMCSD2_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N	N
MMCSD2_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N	N

**Table 3-20. MCU\_CBASS0 Connectivity Matrix (Part 1) (continued)**

[illegible]

**Table 3-21. MCU\_CBASS0 Connectivity Matrix (Part 2)**

Masters	Slaves on MCU_CBASS0																								
	MCU_FWC_BASS0_ERR	MCU_PLL_CFG	MCU_ECC_AGGREG_CFG	MCU_EFUSE0	MCU_MCAN0	MCU_MCAN1	MCU_I2C0	MCU_I2C1	MCU_MSRAM1_MBO_CFG	MCU_MSRAM1_MBO	MCU_NAVSS0_DST0	MCU_PSRAM0	MCU_PSRAM0	MCU_R5FSS0_CORE0_CFG	MCU_R5FSS0_CORE0	MCU_R5FSS0_CORE1_CFG	MCU_R5FSS0_CORE1	MCU_RTI0	MCU_RTI1	MCU_SPI0	MCU_SPI1	MCU_SPI2	MCU_UART0	MCU_PBI_ST0_CFG	MCU_PBI_ST5_CFG
WKUP_DMSC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
WKUP_DMSC0_FWMGR_CFG	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MCU_R5FSS0_CORE0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MCU_R5FSS0_CORE1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y
MCU_NAVSS0_SRC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MCU_PDMA_ADC0_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MCU_PDMA_MISC_G00_MEMW0	N	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N	N
MCU_PDMA_MISC_G00_MEMR0	N	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N	N
MCU_PDMA_MISC_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N
MCU_PDMA_MISC_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N
MCU_PDMA_MISC_G20_MEMW0	N	N	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N
MCU_PDMA_MISC_G20_MEMR0	N	N	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N
DEBUGSS0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
C66SS0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
C66SS1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
VPAC0_DATA0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
VPAC0_DATA1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
VPAC0_LDC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
DMPAC0_DATA	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
GPU0_M0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
GPU0_M1_WR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
GPU0_M1_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
GPU0_M0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
R5FSS0_CORE0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
R5FSS0_CORE1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
R5FSS1_CORE0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
R5FSS1_CORE1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
NAVSS0_HI	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
NAVSS0_LO	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MMCSD0_WR	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	Y	N	Y	N	N	N	N	N	N	N	N
MMCSD0_RD	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	Y	N	Y	N	N	N	N	N	N	N	N
DEBUGSS0_RD	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y



**Table 3-21. MCU\_CBASS0 Connectivity Matrix (Part 2) (continued)**

	Slaves on MCU_CBASS0																							
DEBUGSS0_WR	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PRU_ICSSG0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PRU_ICSSG1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE0_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE0_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE0_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE0_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MMCSD1_WR	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	Y	N	Y	N	N	N	N	N	N	N
MMCSD1_RD	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	Y	N	Y	N	N	N	N	N	N	N
MMCSD2_WR	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	Y	N	Y	N	N	N	N	N	N	N
MMCSD2_RD	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	Y	N	Y	N	N	N	N	N	N	N
ENCODER0_RD	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	Y	N	Y	N	N	N	N	N	N	N
ENCODER0_WR	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	Y	N	Y	N	N	N	N	N	N	N
VPFE0_DMA	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	Y	N	Y	N	N	N	N	N	N	N
DECODER0_RD	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	Y	N	Y	N	N	N	N	N	N	N
DECODER0_WR	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	Y	N	Y	N	N	N	N	N	N	N
MLBSS0	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	Y	N	Y	N	N	N	N	N	N	N
USB3SS0_RD	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	Y	N	Y	N	N	N	N	N	N	N
USB3SS0_WR	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	Y	N	Y	N	N	N	N	N	N	N
USB3SS1_RD	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	Y	N	Y	N	N	N	N	N	N	N
USB3SS1_WR	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	Y	N	Y	N	N	N	N	N	N	N
COMPUTE_CLUSTER0_RD	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	Y	N	Y	N	N	N	N	N	N	N
COMPUTE_CLUSTER0_WR	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	Y	N	Y	N	N	N	N	N	N	N
UFS0_RD	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	Y	N	Y	N	N	N	N	N	N	N
UFS0_WR	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	Y	N	Y	N	N	N	N	N	N	N
DSS0_DMA	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	Y	N	Y	N	N	N	N	N	N	N
DSS0_FBDC	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	Y	N	Y	N	N	N	N	N	N	N
PDMA_DEBUG_CCMCU0_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_DEBUG_C660_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

**Table 3-21. MCU\_CBASS0 Connectivity Matrix (Part 2) (continued)**

	Slaves on MCU_CBASS0																							
PDMA_MISC_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G20_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G20_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G30_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G30_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G20_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G20_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_AASRC0_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_AASRC0_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Table 3-22 lists the master and slave end point connections on the WKUP\_CBASS0. A cell may contain one of the following:

- Y – There is a connection between this master and that slave
- N – There is no connection between this master and that slave.

**Table 3-22. WKUP\_CBASS0 Connectivity Matrix**

	Slaves on WKUP_CBASS0												
Masters	WKUP_PS C0	WKUP_DM SC0	WKUP_ES M0_C FG	WKUP_GPI O0	WKUP_CB AS S0_ER R	WKUP_EC C_A GG R0_C FG	WKUP_CT RL MM R0	WKUP_FW CB AS S0_ER R	WKUP_GPI OM UX_INT RT R0_C FG	WKUP_PLL CT RL0	WKUP_VT M0	WKUP_I2C 0	WKUP_UA RT0
WKUP_DMSC0	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
WKUP_DMSC0_FWMGR_CFG	N	N	N	N	N	N	N	N	N	N	N	N	N
MCU_R5FSS0_CORE0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MCU_R5FSS0_CORE1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MCU_NAVSS0_SRC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MCU_PDMA_ADC0_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N
MCU_PDMA_MISC_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N
MCU_PDMA_MISC_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N
MCU_PDMA_MISC_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N
MCU_PDMA_MISC_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N
MCU_PDMA_MISC_G20_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N
MCU_PDMA_MISC_G20_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N
DEBUGSS0	N	N	N	N	N	N	N	N	N	N	N	N	N
C66SS0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
C66SS1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

**Table 3-22. WKUP\_CBASS0 Connectivity Matrix (continued)**

	Slaves on WKUP_CBASS0												
VPAC0_DATA0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
VPAC0_DATA1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
VPAC0_LDC0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
DMPAC0_DATA	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
GPU0_M0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N
GPU0_M1_WR	N	N	N	N	N	N	N	N	N	N	N	N	N
GPU0_M1_RD	N	N	N	N	N	N	N	N	N	N	N	N	N
GPU0_M0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N
R5FSS0_CORE0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
R5FSS0_CORE1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
R5FSS1_CORE0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
R5FSS1_CORE1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
NAVSS0_HI	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
NAVSS0_LO	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MMCSD0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N
MMCSD0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N
DEBUGSS0_RD	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
DEBUGSS0_WR	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PRU_ICSSG0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PRU_ICSSG1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE0_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE0_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE0_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE0_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE1_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE2_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_RD_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_WR_HP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_RD_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
PCIE3_WR_LP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MMCSD1_WR	N	N	N	N	N	N	N	N	N	N	N	N	N
MMCSD1_RD	N	N	N	N	N	N	N	N	N	N	N	N	N
MMCSD2_WR	N	N	N	N	N	N	N	N	N	N	N	N	N
MMCSD2_RD	N	N	N	N	N	N	N	N	N	N	N	N	N
ENCODER0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N
ENCODER0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N
VPFE0_DMA	N	N	N	N	N	N	N	N	N	N	N	N	N
DECODER0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N
DECODER0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N

**Table 3-22. WKUP\_CBASS0 Connectivity Matrix (continued)**

	Slaves on WKUP_CBASS0												
MLBSS0	N	N	N	N	N	N	N	N	N	N	N	N	N
USB3SS0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N
USB3SS0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N
USB3SS1_RD	N	N	N	N	N	N	N	N	N	N	N	N	N
USB3SS1_WR	N	N	N	N	N	N	N	N	N	N	N	N	N
COMPUTE_CLUSTER0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N
COMPUTE_CLUSTER0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N
UFS0_RD	N	N	N	N	N	N	N	N	N	N	N	N	N
UFS0_WR	N	N	N	N	N	N	N	N	N	N	N	N	N
DSS0_DMA	N	N	N	N	N	N	N	N	N	N	N	N	N
DSS0_FBDC	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_DEBUG_CCMCU0_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_DEBUG_C660_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MCASP_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G20_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G20_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G30_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_MISC_G30_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G00_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G00_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G10_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G10_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G20_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_USART_G20_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_AASRC0_MEMR0	N	N	N	N	N	N	N	N	N	N	N	N	N
PDMA_AASRC0_MEMW0	N	N	N	N	N	N	N	N	N	N	N	N	N

Table 3-23 shows the limitations of programming.

**Table 3-23. Limitations of Programming**

Credential	Can program when the interface is idle	Can program when the interface is not idle
Virtid in the CBASS QoS endpoints	Y	N
Orderid in the CBASS QoS endpoints	Y	N
Priority, epriority and QoS in CBASS QoS endpoints	Y	Y
**	**	**
Virtid in the ring	Y	Y
Orderid in the ring	Y	N
Atype in the UDMA channel	Y	N

**Table 3-23. Limitations of Programming (continued)**

Credential	Can program when the interface is idle	Can program when the interface is not idle
**	**	**
Virtid in the PCIe wrapper	Y	Y
Atype in the PCIe wrapper	Y	Y
**	**	**
Virtid in the VIRTID aliases	Y	N
**	**	**

### 3.3.2 Quality of Service (QoS)

The device interconnect provides two mechanisms to achieve quality of service (QoS): parallel routing based on order ID and arbitration based on priority. The order ID has two main usages: to order execution of transactions and parallel paths selection. The parallel paths are defined as multiple paths between master and slave. If two or more slaves share same physical memory map assignment, it is considered parallel paths.

The CBASS\_MAP\_i[7-4] ORDERID bit field is for transactions for each master and each channel of the master. This allows different mapping not only for each master, but also for each channel of the master, if multiple channels are supported. The CBASS\_MAP\_i[7-4] ORDERID bit field specifies the initial order ID value. The ORDERIDx bit fields in CBASS\_GRP\_MAP1\_j and CBASS\_GRP\_MAP2\_j registers take the initial order ID value and result in a final order ID value, which is nothing more than a 4-bit identifier used for path routing. The final order ID is used to differentiate the paths to a slave and thus allowing load balancing of the traffic over each path.

The interconnect performs traffic routing and switching from a number of masters to a number of slaves. The masters send transactions. The interconnect decodes which slave is being accessed based on special signals and memory map and routes that transaction to the slave. The interconnect also arbitrates for a slave when there are multiple transactions being requested simultaneously. The arbitration is based on transaction priority. Once a transaction is selected, the interconnect sends the transaction to the slave. The transaction priority is controlled per master and per channel of the master through the CBASS\_MAP\_i[14-12] EPRIORITY bit field. The priority setting for each master can be done by any processor. Increasing the priority level for certain master can help the transaction from that master to have advantage on winning arbitration, therefore improving the latency and bandwidth for the transactions from that master.

Since the order ID is controlled from configuration registers, the user can re-partition the traffic among parallel paths to achieve better load balance adjusted for specific use cases. As order ID has routing implication for both command and return data, its configuration must not be changed during run time. Otherwise, the return data may be routed to wrong path and cause system hang. To avoid this, the system must be put into idle state with no traffic on the interconnect before reprogramming procedure.

There are many QoS register blocks each of which is master related only. [Table 3-24](#) through [Table 3-37](#) show the mapping between masters and QoS block physical addresses. "N/A" in column "**QoS Block Physical Address**" means that the corresponding master does not have QoS registers at device level. Local QoS registers may still be available.

#### Note

Route ID is not related to QoS. For route ID description, see [Section 3.3.3](#).

#### Note

There are no master modules on the INFRA\_CBASS0 interconnect.

**Table 3-24. CBASS\_AASRC0 Master Modules Attributes**

Masters	Route ID	Virtualization	QoS Block Physical Address
PDMA_AASRC0_MEMW0	560	Y	N/A

**Table 3-24. CBASS\_AASRC0 Master Modules Attributes (continued)**

Masters	Route ID	Virtualization	QoS Block Physical Address
PDMA_AASRC0_MEMR0	561	Y	N/A

**Table 3-25. CBASS\_AC0 Master Modules Attributes**

Masters	Route ID	Virtualization	QoS Block Physical Address
DMPAC0_DATA	2568-2569	Y	0x45DC0000
DECODER0_RD	2584	Y	0x45DC0400
DECODER0_WR	2585	Y	0x45DC0800
ENCODER0_RD	2576	Y	0x45DC0C00
ENCODER0_WR	2577	Y	0x45DC1000
VPAC0_DATA0	2562-2563	Y	0x45DC1400
VPAC0_DATA1	2564-2565	Y	0x45DC1800
VPAC0_LDC0	2560	Y	0x45DC1C00
DSS0_DMA	2592	Y	0x45DC2000
DSS0_FBDC	2593	Y	0x45DC2400
GPU0_M0_RD	2601	Y	0x45DC5000
GPU0_M0_WR	2602	Y	0x45DC5800
GPU0_M1_RD	2600	Y	0x45DC6000
GPU0_M1_WR	2603	Y	0x45DC6800
NAVSS0_AC_SLV0	0-2559 3072-4095	Y	N/A

**Table 3-26. CBASS\_CSIO Master Modules Attributes**

Masters	Route ID	Virtualization	QoS Block Physical Address
There are no master ports of modules or subsystems on this sub-interconnect	N/A	N/A	N/A

**Table 3-27. CBASS\_DATADEBUG0 Master Modules Attributes**

Masters	Route ID	Virtualization	QoS Block Physical Address
DEBUGSS0_RD	350	N	0x45DA0000
DEBUGSS0_WR	351	N	0x45DA0400
PDMA_DEBUG_CCMCU0_MEMR0	624	Y	N/A
PDMA_DEBUG_C660_MEMR0	625	Y	N/A

**Table 3-28. CBASS\_HC\_CFG0 Master Modules Attributes**

Masters	Route ID	Virtualization	QoS Block Physical Address
There are no master ports of modules or subsystems on this sub-interconnect	N/A	N/A	N/A

**Table 3-29. CBASS\_HC0 Master Modules Attributes**

Masters	Route ID	Virtualization	QoS Block Physical Address
PCIE0_RD_HP	352	Y	0x45D90000
PCIE0_RD_LP	354	Y	0x45D90400
PCIE0_WR_HP	353	Y	0x45D90800
PCIE0_WR_LP	355	Y	0x45D90C00
PCIE1_RD_HP	356	Y	0x45D91000
PCIE1_RD_LP	358	Y	0x45D91400
PCIE1_WR_HP	357	Y	0x45D91800
PCIE1_WR_LP	359	Y	0x45D91C00



**Table 3-29. CBASS\_HC0 Master Modules Attributes (continued)**

Masters	Route ID	Virtualization	QoS Block Physical Address
PCIE2_RD_HP	360	Y	0x45D92000
PCIE2_RD_LP	362	Y	0x45D92400
PCIE2_WR_HP	361	Y	0x45D92800
PCIE2_WR_LP	363	Y	0x45D92C00
PCIE3_RD_HP	364	Y	0x45D93000
PCIE3_RD_LP	366	Y	0x45D93400
PCIE3_WR_HP	365	Y	0x45D93800
PCIE3_WR_LP	367	Y	0x45D93C00

**Table 3-30. CBASS\_HC2\_0 Master Modules Attributes**

Masters	Route ID	Virtualization	QoS Block Physical Address
USB3SS0_RD	304	Y	0x45D98000
USB3SS0_WR	305	Y	0x45D98400
USB3SS1_RD	306	Y	0x45D98800
USB3SS1_WR	307	Y	0x45D98C00
MLBSS0	552	Y	0x45D99C00
MMCSD0_RD	257	Y	0x45D9A000
MMCSD0_WR	256	Y	0x45D9A400
UFS0_RD	274	Y	0x45D9B000
UFS0_WR	275	Y	0x45D9B400

**Table 3-31. CBASS\_IPPHY0 Master Modules Attributes**

Masters	Route ID	Virtualization	QoS Block Physical Address
PDMA_MISC_G00_MEMW0	608	Y	N/A
PDMA_MISC_G00_MEMR0	609	Y	N/A
PDMA_MISC_G10_MEMW0	610	Y	N/A
PDMA_MISC_G10_MEMR0	611	Y	N/A
PDMA_MISC_G20_MEMW0	612	Y	N/A
PDMA_MISC_G20_MEMR0	613	Y	N/A
PDMA_MISC_G30_MEMW0	614	Y	N/A
PDMA_MISC_G30_MEMR0	615	Y	N/A
PDMA_USART_G00_MEMW0	616	Y	N/A
PDMA_USART_G00_MEMR0	617	Y	N/A
PDMA_USART_G10_MEMW0	618	Y	N/A
PDMA_USART_G10_MEMR0	619	Y	N/A
PDMA_USART_G20_MEMW0	620	Y	N/A
PDMA_USART_G20_MEMR0	621	Y	N/A

**Table 3-32. CBASS\_MCASP\_G0\_0 Master Modules Attributes**

Masters	Route ID	Virtualization	QoS Block Physical Address
PDMA_MCASP_G00_MEMW0	624	Y	N/A
PDMA_MCASP_G00_MEMR0	625	Y	N/A

**Table 3-33. CBASS\_MCASP\_G1\_0 Master Modules Attributes**

Masters	Route ID	Virtualization	QoS Block Physical Address
PDMA_MCASP_G10_MEMW0	626	Y	N/A

**Table 3-33. CBASS\_MCASP\_G1\_0 Master Modules Attributes (continued)**

Masters	Route ID	Virtualization	QoS Block Physical Address
PDMA_MCASP_G10_MEMR0	627	Y	N/A

**Table 3-34. CBASS\_RC\_CFG0 Master Modules Attributes**

Masters	Route ID	Virtualization	QoS Block Physical Address
R5FSS0_CORE0_PER0	524	Y	0x45DA4000
R5FSS0_CORE1_PER0	532	Y	0x45DA4400
R5FSS1_CORE0_PER0	540	Y	0x45DA4800
R5FSS1_CORE1_PER0	548	Y	0x45DA4C00

**Table 3-35. CBASS\_RC0 Master Modules Attributes**

Masters	Route ID	Virtualization	QoS Block Physical Address
PRU_ICSSG0	384-415	N	0x45D80000
PRU_ICSSG1	416-447	N	0x45D80400
C66SS0_MDMA	576	Y	0x45D81000
C66SS1_MDMA	578	Y	0x45D81400
MMCSD1_RD	259	Y	0x45D82000
MMCSD1_WR	258	Y	0x45D82400
MMCSD2_RD	261	Y	0x45D82800
MMCSD2_WR	260	Y	0x45D82C00
R5FSS0_CORE0_MEM_RD	520	Y	0x45D84000
R5FSS0_CORE1_MEM_RD	528	Y	0x45D84400
R5FSS0_CORE0_MEM_WR	521	Y	0x45D84800
R5FSS0_CORE1_MEM_WR	529	Y	0x45D84C00
R5FSS1_CORE0_MEM_RD	536	Y	0x45D85000
R5FSS1_CORE1_MEM_RD	544	Y	0x45D85400
R5FSS1_CORE0_MEM_WR	537	Y	0x45D85800
R5FSS1_CORE1_MEM_WR	545	Y	0x45D85C00
COMPUTE_CLUSTER0_RD	302	Y	0x45D86000
COMPUTE_CLUSTER0_WR	303	Y	0x45D86400
C66SS0_CFG	577	N	0x45D87000
C66SS1_CFG	579	N	0x45D87400
R5FSS0_CORE0_PER1_RD	522	Y	0x45D89800
R5FSS0_CORE1_PER1_RD	530	Y	0x45D89C00
R5FSS0_CORE0_PER1_WR	523	Y	0x45D8A000
R5FSS0_CORE1_PER1_WR	531	Y	0x45D8A400
R5FSS1_CORE0_PER1_RD	538	Y	0x45D8A800
R5FSS1_CORE1_PER1_RD	546	Y	0x45D8AC00
R5FSS1_CORE0_PER1_WR	539	Y	0x45D8B000
R5FSS1_CORE1_PER1_WR	547	Y	0x45D8B400
VPFE0_DMA	512	Y	0x45D8C000
NAVSS0_HI	0-255 2560-3071	Y	N/A
NAVSS0_LO	0-255 2560-3071	Y	N/A

**Table 3-36. MCU\_CBASS0 Master Modules Attributes**

Masters	Route ID	Virtualization	QoS Block Physical Address
MCU_R5FSS0_CORE0_MEM_RD	4089	Y	0x45D10000
MCU_R5FSS0_CORE0_MEM_WR	4088	Y	0x45D10400
MCU_R5FSS0_CORE0_PER0	4090	Y	0x45D10800
MCU_R5FSS0_CORE1_MEM_RD	4085	Y	0x45D11000
MCU_R5FSS0_CORE1_MEM_WR	4084	Y	0x45D11400
MCU_R5FSS0_CORE1_PER0	4091	Y	0x45D11800
MCU_PDMA_MISC_G00_MEMW0	4075	Y	N/A
MCU_PDMA_MISC_G00_MEMR0	4076	Y	N/A
MCU_PDMA_MISC_G10_MEMW0	4077	Y	N/A
MCU_PDMA_MISC_G10_MEMR0	4078	Y	N/A
MCU_PDMA_MISC_G20_MEMW0	4079	Y	N/A
MCU_PDMA_MISC_G20_MEMR0	4080	Y	N/A
MCU_PDMA_ADC0_MEMR0	4081	Y	N/A
MCU_NAVSS0_SRC0	3584-3711	Y	N/A

**Table 3-37. WKUP\_CBASS0 Master Modules Attributes**

Masters	Route ID	Virtualization	QoS Block Physical Address
WKUP_DMSC0	4095	N	N/A

The interconnect inside NAVSS0 uses order ID to provide multiple (at least two) parallel paths to DDR and a separate set of multiple (at least two) parallel paths to SRAM. NAVSS0 also provides multiple (at least two) parallel paths for the DMA traffic to SoC level, which can provide isolated DMA traffic paths. For more information, see *Navigator Subsystem (NAVSS)*.

The MSMC does not use order ID for routing purpose to provide separate physical paths for transaction. Instead, it provides two threads to isolate two classes of transactions: thread 0 and thread 2. The arbitration between these two threads is based on credits, and thread 2 has priority over thread 0 when both threads have credits available for transfer. In addition, there is bandwidth management scheme based on transaction priority and bandwidth starvation prevention mechanism. For more information, see *Multicore Shared Memory Controller (MSMC)*.

By default all masters send transactions with order ID = 0. All transactions with the same order ID execute in order if they are sent to the same slave or going through a common bridge. On the SoC level interconnect, the order ID is used to partition the transactions to MSMC and DDR data space into parallel routing paths. Further, the north bridge inside NAVSS0 provides multiple parallel paths to the compute cluster. Each path is separated by order ID value. All read commands towards MSMC and DDR sharing the same order ID and sharing the same master path from SoC side (including NAVSS0) provide read return data in order back to the master port. The write response can be returned out of order for the same order ID value. Therefore, programming order ID has implications on overall system performance as well as achieving QoS for certain class of traffic. Multiple configurations are needed to make sure that the QoS goal is met.

For any write to address range 0x4500\_0000 to 0x45FF\_FFFF, it is recommended to read back the value after the write to make sure the write landed. The registers in this regions are mainly for firewall configuration, ISC/DMA credential configuration, QoS MMR configuration.

On QoS MMR configuration, in order to configure certain transaction to be the highest priority, the priority field needs to be set to zero for that transaction. In order to make sure that priority field is indeed to set to zero, the following sequence should be followed:

1. Read the priority field, if it is non-zero value, follow a write to overwrite priority field to be 0x0.

2. If read back priority field is zero, write to QoS MMR register to set priority field to be 0x7. Read back the same priority field. If the read back value is 0x7, write to QoS MMR register to set priority field to be 0x0. If the read back value is 0x0, it means this register is not implemented and priority setting is not working.

### **3.3.3 Route ID**

The route ID is used by the interconnect to route return status and data back to the transaction initiator. Each master has one or multiple unique route IDs assigned to it. The route ID has no other usage. There are no software controls for the route ID.

### 3.3.4 Initiator-Side Security Controls and Firewalls

Firewalls (FW) and Initiator-side Security Controls (ISC) are important interconnect components that enable hardware isolation for freedom from interference or security uses. ISCs enable every transaction to be identified and tagged to precisely assign the source ID. Firewalls are downstream (target) components that provide the ability to filtering transactions based on the sideband information.

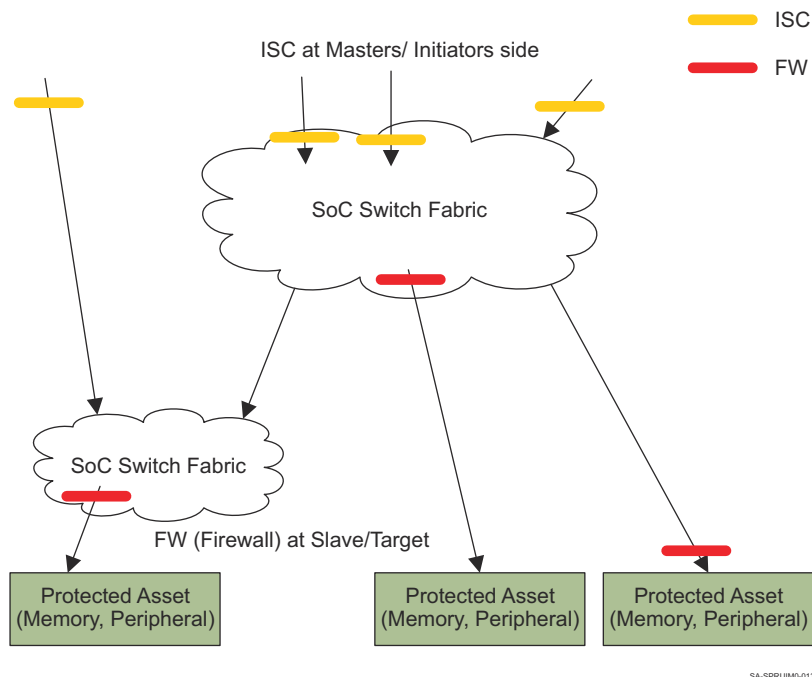
The device protection depends on firewalls. They are used to protect data and configuration spaces by managing the accesses to these memory regions. There are two types of firewalls - region based and channelized. There aren't channelized firewalls on system level. Only NAVSS0, MCU\_NAVSS0 and DRU0 have channelized firewalls for the various DMA channels. Only region based firewalls are available on system level. See [Section 3.3.4.2.2.1](#) and [Section 3.3.4.2.3.1](#) for description of the region based and channelized firewalls.

Almost all slaves have a firewall right before the transaction reaches them. There are few exceptions such as WKUP\_DMSC0, VPAC0, DMPAC0, NAVSS0 and MCU\_NAVSS0 slave ports. These subsystems contain local interconnect with own firewalls inside the subsystem itself. The firewalls inside compute cluster (for A72SS0, C71SS0 and DRU) are also an exception. They are put on the master instead on the slave port side. To enable access for that master port to the slaves these master side firewalls must individually be programmed. Note that there is also a second firewall for C71SS0 that is put on the slave port side.

On devices supporting secure boot and secure DMSC, these components allow the SoC to support multi-tier security and provide segregation of secure and non-secure worlds. All configuration of ISCs and FWs are under exclusive control of DMSC, using a dedicated interconnect.

The number of ISC and Firewall blocks and placement of these blocks are based on the topology of each device. ISC and Firewall blocks are placed in host modules (AXI to VBUSM.C Bridge); or part of the interconnect (for example: CBASS).

Figure 3-2 presents a generic view of ISCs and Firewalls in SoC.



**Figure 3-2. ISC and Firewall in SoC**

### 3.3.4.1 Initiator-Side Security Controls (ISC)

Initiator-Side Security Control (ISC) modules are hosted at the initiator side where the transactions are sourced. ISC controls the security sideband attributes that are applied to outgoing transactions and have ability to override security values under exclusive control of DMSC.

ISC Capabilities:

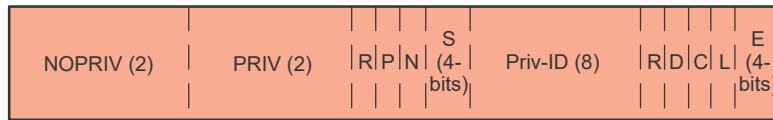
- Attach Priv-ID to each bus transaction.
- Priv-ID is a source ID attached to identify the source of a transaction in the system.
- By default, each initiator in the system has a unique Priv-ID.
- Multiple modules can be assigned the same Priv-ID to form logical groups.
- Assert, De-assert or pass-through secure bit.
- Assert, De-assert or pass through Priv-bits.

ISCs are connected to the dedicated security VBUSP interconnect and can be exclusively configured by DMSC. ISC optionally can support region and channel number based security control, where, based on incoming channel or address, the associated security controls, are applied.

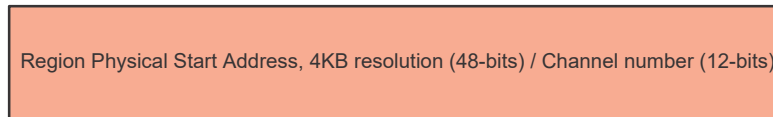


Figure 3-3 presents ISC config registers per region.

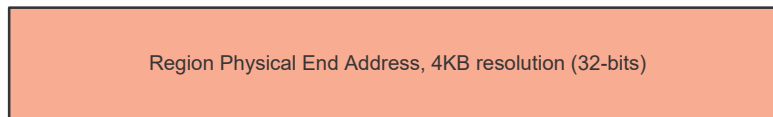
ISC Control Regs **per** region



E = Enable Region (0xA = Enabled)  
L = Lock Config  
C = Use Channel number, not address  
D = Default region indicator  
S = Make outgoing transaction secure (0xA = Enabled)  
N = Make outgoing transaction Non-secure  
R = Reserved  
P = Do not replace Priv-ID, pass through PrivID  
NOPRIV = Clear Outgoing PRIV Attribute  
PRIV = Set Outgoing PRIV Attribute



Lower 12-bits valid in case of channel number



Not valid in case of channel number

**Figure 3-3. ISC Config Registers per Region**

Priv-ID is used to identify the logical source of transaction and is attached by ISC. The Priv-IDs are allocated based on subsystem as part of platform definition.

#### Note

See Appendix Spreadsheet for ISC and Priv-ID details.

#### 3.3.4.1.1 Special System Level Priv-ID

This section describes Special Priv-IDs that must not be assigned to any ISC. These Priv-ID when encountered by various blocks like firewall have special meaning.

Table 3-38 presents Special Priv-IDs.

**Table 3-38. Special Priv-IDs**

Special Priv-IDs	Hex Value	Decimal Value
System Reserved Priv-ID	0xC1, 0xC2, 0xC4, 0xC6, 0xC7	193, 194, 196, 198, 199
Wild card Priv ID	0xC3	195
Block Priv ID	0xC5	197
DMA Reserved Priv-ID	0xC0	192

#### 3.3.4.1.2 Priv ID and ISC Assignment

**Table 3-39. Priv ID and ISC Assignment for DRA829/TDA4VM**

Masters	Reset Priv ID	ISC Regions <sup>(1)</sup>	ISC Block Physical Address
WKUP_DMSC0			
M3 I bus	202	1	0x45808000
M3 D bus	202	1	0x45808400
M3 S bus	202	1	0x45808800
WKUP_CBASS0			
WKUP_DMSC0	N/A	N/A	N/A
MCU_CBASS0			
MCU_R5FSS0_CORE0_MEM_RD	96	4	0x45810000
MCU_R5FSS0_CORE0_MEM_WR	96	4	0x45810400

**Table 3-39. Priv ID and ISC Assignment for DRA829/TDA4VM (continued)**

Masters	Reset Priv ID	ISC Regions <sup>(1)</sup>	ISC Block Physical Address
MCU_R5FSS0_CORE0_PER0	96	4	0x45810800
MCU_R5FSS0_CORE1_MEM_RD	97	4	0x45811000
MCU_R5FSS0_CORE1_MEM_WR	97	4	0x45811400
MCU_R5FSS0_CORE1_PER0	97	4	0x45811800
MCU_SA2_UL0	152	4	0x45813000
MCU_NAVSS0			
MCU_NAVSS0_PROXY0	80	1	0x45818000
MCU_NAVSS0_SEC_PROXY0	81	1	0x45818400
MCU_NAVSS0_UDMASS_RINGACC0	82	1	0x45818800
NAVSS0			
NAVSS0_PROXY0	64	1	0x45840000
NAVSS0_SEC_PROXY0	65	1	0x45840400
NAVSS0_VIRTSS_TCU0_RD	255	1	0x45850000
NAVSS0_VIRTSS_TCU0_WR	255	1	0x45850400
NAVSS0_UDMASS_RINGACC0	74	1	0x45870000
DMPAC0			
SDE	208	ch	0x45859000
DOF_RD	209	ch	0x45859400
DOF	210	ch	0x45859800
FOCO_0	211	ch	0x45859C00
FOCO_1	212	ch	0x4585A000
CBASS_RC0			
ICSS_G0_PR1	136	10	0x45880000
ICSS_G1_PR1	136	10	0x45880400
C66X_CORE0_MDMA	220	4	0x45881000
C66X_CORE1_MDMA	221	4	0x45881400
EMMCSD4SS0_RD	129	1	0x45882000
EMMCSD4SS0_WR	129	1	0x45882400
EMMCSD4SS1_RD	129	1	0x45882800
EMMCSD4SS1_WR	129	1	0x45882C00
R5FSS0_CORE0_MEM_RD	212	4	0x45884000
R5FSS0_CORE1_MEM_RD	213	4	0x45884400
R5FSS0_CORE0_MEM_WR	212	4	0x45884800
R5FSS0_CORE1_MEM_WR	213	4	0x45884C00
R5FSS1_CORE0_MEM_RD	214	4	0x45885000
R5FSS1_CORE1_MEM_RD	215	4	0x45885400
R5FSS1_CORE0_MEM_WR	214	4	0x45885800
R5FSS1_CORE1_MEM_WR	215	4	0x45885C00
COMPUTE_CLUSTER0_RD	154	1	0x45886000
COMPUTE_CLUSTER0_WR	154	1	0x45886400
C66X_CORE0_CFG	220	4	0x45887000
C66X_CORE1_CFG	221	4	0x45887400
SA2_UL0	152	4	0x45888800
R5FSS0_CORE0_PER1_RD	212	4	0x45889800
R5FSS0_CORE1_PER1_RD	213	4	0x45889C00

**Table 3-39. Priv ID and ISC Assignment for DRA829/TDA4VM (continued)**

Masters	Reset Priv ID	ISC Regions <sup>(1)</sup>	ISC Block Physical Address
R5FSS0_CORE0_PER1_WR	212	4	0x4588A000
R5FSS0_CORE1_PER1_WR	213	4	0x4588A400
R5FSS1_CORE0_PER1_RD	214	4	0x4588A800
R5FSS1_CORE1_PER1_RD	215	4	0x4588AC00
R5FSS1_CORE0_PER1_WR	214	4	0x4588B000
R5FSS1_CORE1_PER1_WR	215	4	0x4588B400
VPFE0_DMA	189	1	0x4588C000
CBASS_HC0			
PCIE0_RD_HP	178	4	0x45890000
PCIE0_RD_LP	179	4	0x45890400
PCIE0_WR_HP	178	4	0x45890800
PCIE0_WR_LP	179	4	0x45890C00
PCIE1_RD_HP	180	4	0x45891000
PCIE1_RD_LP	181	4	0x45891400
PCIE1_WR_HP	180	4	0x45891800
PCIE1_WR_LP	181	4	0x45891C00
PCIE2_RD_HP	182	4	0x45892000
PCIE2_RD_LP	183	4	0x45892400
PCIE2_WR_HP	182	4	0x45892800
PCIE2_WR_LP	183	4	0x45892C00
PCIE3_RD_HP	184	4	0x45893000
PCIE3_RD_LP	185	4	0x45893400
PCIE3_WR_HP	184	4	0x45893800
PCIE3_WR_LP	185	4	0x45893C00
CBASS_HC2_0			
USB3SS0_RD	155	8	0x45898000
USB3SS0_WR	155	8	0x45898400
USB3SS1_RD	155	8	0x45898800
USB3SS1_WR	155	8	0x45898C00
USB3SS2_RD	155	8	0x45899000
USB3SS2_WR	155	8	0x45899400
MLBSS0	128	1	0x45899C00
EMMC8SS0_RD	128	1	0x4589A000
EMMC8SS0_WR	128	1	0x4589A400
UFS_HCI0_RD	128	1	0x4589B000
UFS_HCI0_WR	128	1	0x4589B400
UFS_HCI1_RD	128	1	0x4589B800
UFS_HCI1_WR	128	1	0x4589BC00
CBASS_DATADEBUG0			
DEBUGSS0_RD	177	1	0x458A0000
DEBUGSS0_WR	177	1	0x458A0400
CBASS_RC_CFG0			
R5FSS0_CORE0_PER0	212	4	0x458A4000
R5FSS0_CORE1_PER0	213	4	0x458A4400
R5FSS1_CORE0_PER0	214	4	0x458A4800

**Table 3-39. Priv ID and ISC Assignment for DRA829/TDA4VM (continued)**

Masters	Reset Priv ID	ISC Regions <sup>(1)</sup>	ISC Block Physical Address
R5FSS1_CORE1_PER0	215	4	0x458A4C00
CBASS_AC0			
DECODER0_RD	186	2	0x458C0400
DECODER0_WR	186	2	0x458C0800
ENCODER0_RD	190	5	0x458C0C00
ENCODER0_WR	190	5	0x458C1000
VPAC0_LDC0	220	3	0x458C1C00
DSS0_DMA	173	10	0x458C2000
DSS0_FBDC	173	10	0x458C2400
GPU0_M0_RD	187	48	0x458C5000
GPU0_M0_WR	187	48	0x458C5800
GPU0_M1_RD	187	48	0x458C6000
GPU0_M1_WR	187	48	0x458C6800
CBASS_CSI0			
There are no master ports of modules or subsystems on this sub-interconnect	N/A	N/A	N/A
CBASS_HC_CFG0			
There are no master ports of modules or subsystems on this sub-interconnect	N/A	N/A	N/A
CBASS_MCASP_G0_0			
PDMA_MCASP_G00_MEMW0	N/A	N/A	N/A
PDMA_MCASP_G00_MEMR0	N/A	N/A	N/A
CBASS_MCASP_G1_0			
PDMA_MCASP_G10_MEMW0	N/A	N/A	N/A
PDMA_MCASP_G10_MEMR0			
CBASS_AASRC0			
PDMA_AASRC0_MEMW0	N/A	N/A	N/A
PDMA_AASRC0_MEMR0	N/A	N/A	N/A
CBASS_IPPHY0			
PDMA_MISC_G00_MEMW0	N/A	N/A	N/A
PDMA_MISC_G00_MEMR0	N/A	N/A	N/A
PDMA_MISC_G10_MEMW0	N/A	N/A	N/A
PDMA_MISC_G10_MEMR0	N/A	N/A	N/A
PDMA_MISC_G20_MEMW0	N/A	N/A	N/A
PDMA_MISC_G20_MEMR0	N/A	N/A	N/A
PDMA_MISC_G30_MEMW0	N/A	N/A	N/A
PDMA_MISC_G30_MEMR0	N/A	N/A	N/A
PDMA_USART_G00_MEMW0	N/A	N/A	N/A
PDMA_USART_G00_MEMR0	N/A	N/A	N/A
PDMA_USART_G10_MEMW0	N/A	N/A	N/A
PDMA_USART_G10_MEMR0	N/A	N/A	N/A
PDMA_USART_G20_MEMW0	N/A	N/A	N/A
PDMA_USART_G20_MEMR0	N/A	N/A	N/A

(1) Value "ch" means one ISC setting per one channel as this is a channelized ISC.

### 3.3.4.2 Firewalls (FW)

Firewalls play an important role in implementing overall SoC security by providing a means to allow or restrict access to device resources to any given master entity or Secure/Non-Secure/Priv/User world. Firewalls are placed at various data path points throughout the SoC to control access to protected asset (Peripheral, memory and so forth).

Firewalls ensure that assets can be protected and are only accessible by allowed master and in selected operation mode (Secure, Non-Secure, Priv, User, write, read and so forth). In case of firewall violations, the transaction is dropped and the appropriate violation code is registered with associated parameters.

There are three types of firewalls, Peripheral firewalls, Memory firewalls (referred as one group **Region Based Firewall** in the device-specific TRM) and Channelized firewalls.

Each Firewall module in the SoC has a unique Firewall ID that allows software to decode the source of each violation. The Firewall IDs are allocated based on subsystem as part of platform definition. The full SoC view of Firewall IDs for various Firewall modules are captured in each device spec.

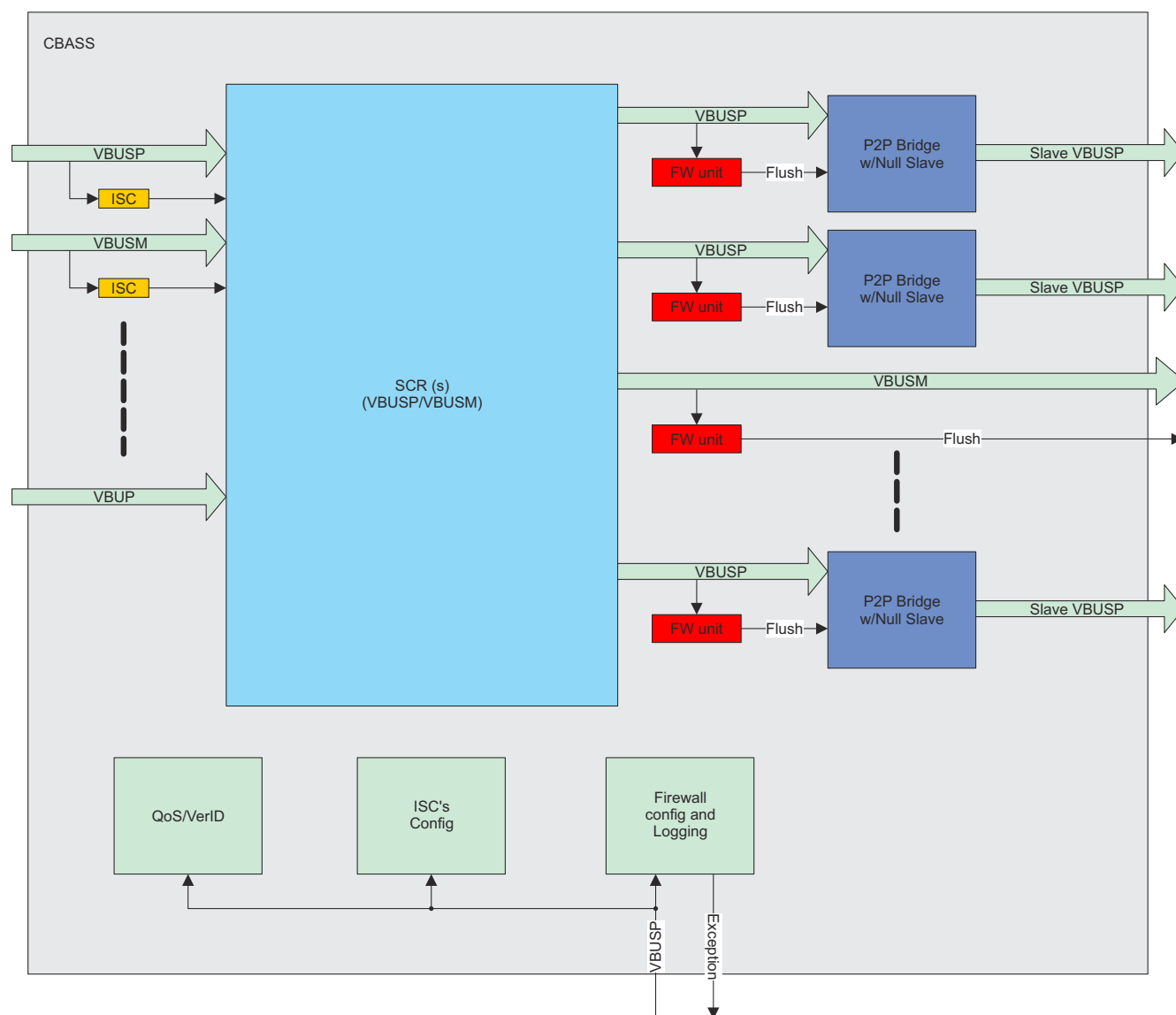
#### 3.3.4.2.1 Peripheral Firewalls (FW)

Peripheral firewall modules are placed in front of peripherals like UART, SPI and so forth. They protect access to the peripheral registers/memory and typically have firewall regions as number of regions supported by module that is being protected.

All Peripheral firewalls support 3 Priv-ID slots that allow multiple masters/initiators to access protected peripherals.

The peripheral firewall is configured using dedicated VBUSP port to CBASS that connects to DMSC private VBUSP interconnect.

[Figure 3-4](#) presents Peripheral Firewall.



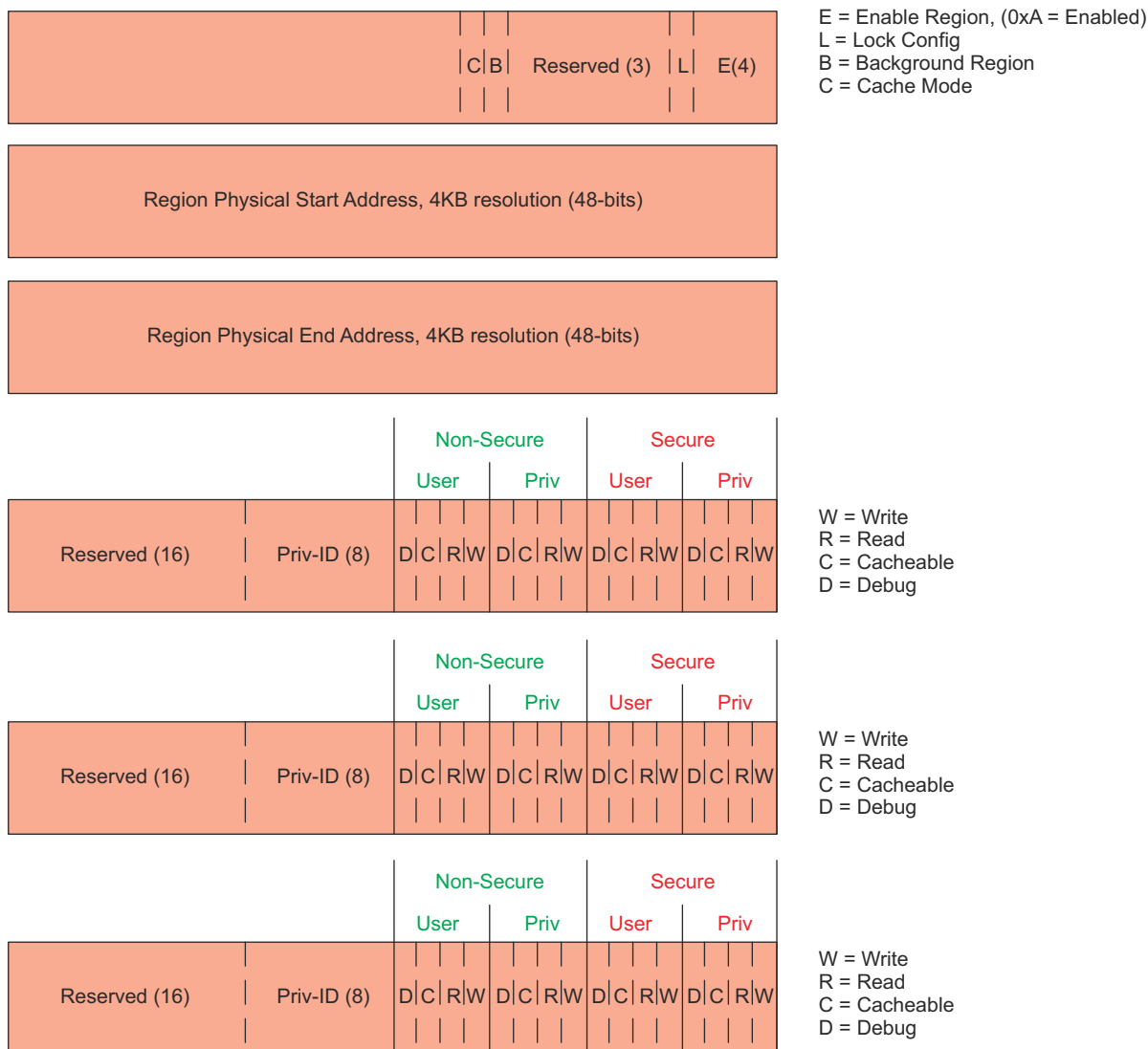
SA-SPRUI1D-015

**Figure 3-4. Peripheral Firewall**

Each region is defined by start and end physical address and associated permission/control register as shown in [Figure 3-5](#).



Firewall Control Regs **per** region



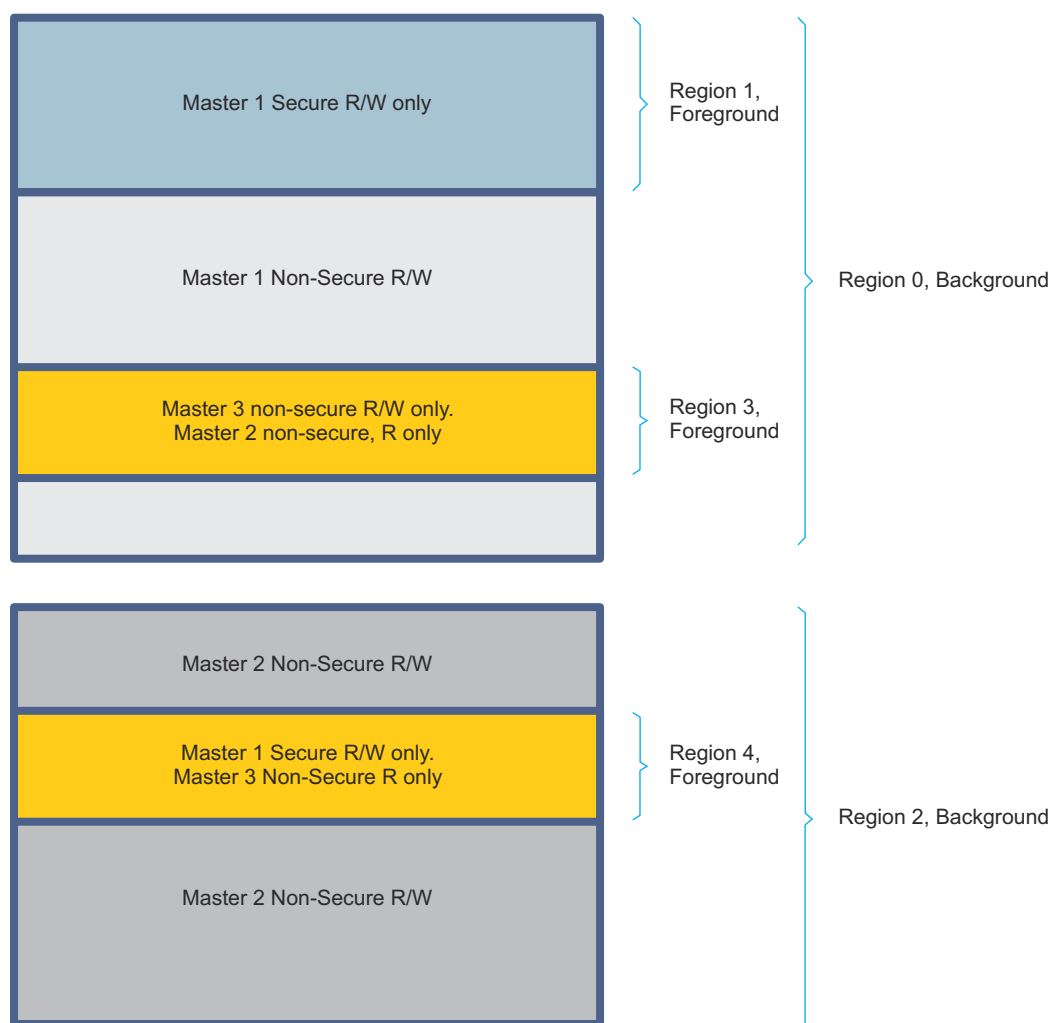
SA-SPRUI00-016

**Figure 3-5. Peripheral Firewall Config Registers per Regions**

A Peripheral's Firewall module can have one or more regions. In case there is more than 1 region, the registers are duplicated for each region.

If multiple regions are supported, a firewall configuration can be defined as either a Background or Foreground region depending on the setting of the Background (B) bit. Foreground regions can overlap background regions and their Firewall settings take precedence over the Background settings when there is overlap. However, background regions cannot overlap each other and foreground regions cannot overlap each other. A firewall error will result in case a memory transaction is made to the improper overlap regions.

Figure 3-6 presents peripherals firewall regions.



SA-SPRUID-017

**Figure 3-6. Peripherals Firewall Regions**

In the above case, Region 0 and Region 2 are background regions (background bit set in control register). Region 1, Region 3 and Region 4 are **not** background (foreground) regions. In case the incoming transaction hits in address in Region 1, the permissions of Region 1 are applied to filter incoming transaction, thereby completely ignoring the permission of the background Region 0.

Table 3-40 through Table 3-54 show the initiator and target firewalls, firewall IDs, physical base address of the corresponding firewall, number of firewall regions, and areas covered by the firewall. The firewall ID uniquely identifies each firewall. In case of firewall violation this ID is logged and can be read through the CBASS\_EXCEPTION\_LOGGING\_HEADER0[23-8] SRC\_ID field.

**Table 3-40. CBASS\_AASRC0 Slave Modules Attributes**

Slaves	Firewall ID	Firewall Regions	Firewall Physical Address
AASRC0_CFG	2608	1	0x4528C000
AASRC0_DATA	2609	2	0x4528C400
CBASS_AASRC0_ERR	2623	1	0x4528FC00

**Table 3-41. CBASS\_AC0 Slave Modules Attributes**

Slaves	Firewall ID	Firewall Regions	Firewall Physical Address
There are no slave ports of modules or subsystems on this sub-interconnect	N/A	N/A	N/A

**Table 3-42. CBASS\_CSI0 Slave Modules Attributes**

Slaves	Firewall ID	Firewall Regions	Firewall Physical Address
CSI_RX0	2432	4	0x45260000
CSI_RX1	2433	4	0x45260400
DPHY_RX0	2436	2	0x45261000
DPHY_RX1	2437	2	0x45261400
CSI_TX0	2440	4	0x45262000
DPHY_TX0	2442	1	0x45262800
CSI_PSILO_CFG	2446	1	0x45263800
CBASS_CSI0_ERR	2463	1	0x45267C00

**Table 3-43. CBASS\_DATADEBUG0 Slave Modules Attributes**

Slaves	Firewall ID	Firewall Regions	Firewall Physical Address
STM0	2464	2	0x45268000
DEBUGSS0_CFG	2465	1	0x45268400
MAIN_DEBUG_CELL0	2466	1	0x45268800
CC_DEBUG_CELL0	2467	1	0x45268C00
MAIN_DEBUG_CELL1	2468	1	0x45269000
C66SS0_DEBUG_CELL	2469	1	0x45269400
C66SS1_DEBUG_CELL	2470	1	0x45269800
PDMA_DEBUG_PSILO_CFG	2472	1	0x4526A000
CBASS_DATADEBUG0_ERR	2495	1	0x4526FC00

**Table 3-44. CBASS\_HC\_CFG0 Slave Modules Attributes**

Slaves	Firewall ID	Firewall Regions	Firewall Physical Address
PCIE0_CFG	2560	8	0x45280000
PCIE1_CFG	2561	8	0x45280400
PCIE2_CFG	2562	8	0x45280800
PCIE3_CFG	2563	8	0x45280C00
USB3SS0_CORE	2568	4	0x45282000
USB3SS0_USB2PHY	2569	1	0x45282400
USB3SS1_CORE	2570	4	0x45282800
USB3SS1_USB2PHY	2571	1	0x45282C00
MMCSD0_CFG	2576	4	0x45284000
MLBSS0_CFG	2578	4	0x45284800
UFS0_CFG	2580	4	0x45285000
HC5_ECC_AGGR_CFG	2583	1	0x45285C00
SERDES0	2584	1	0x45286000
SERDES1	2585	1	0x45286400
SERDES2	2586	1	0x45286800
SERDES3	2587	1	0x45286C00
PBIST_HC0_CFG	2588	1	0x45287000
CBASS_HC2_0_ERR	2589	1	0x45287400
CBASS_HC0_ERR	2590	1	0x45287800

**Table 3-44. CBASS\_HC\_CFG0 Slave Modules Attributes (continued)**

Slaves	Firewall ID	Firewall Regions	Firewall Physical Address
CBASS_HC_CFG0_ERR	2591	1	0x45287C00

**Table 3-45. CBASS\_HC0 Slave Modules Attributes**

Slaves	Firewall ID	Firewall Regions	Firewall Physical Address
PCIE0_HP	2528	24	0x45278000
PCIE0_LP	2529	24	0x45278400
PCIE1_HP	2530	24	0x45278800
PCIE1_LP	2531	24	0x45278C00
PCIE2_HP	2532	24	0x45279000
PCIE2_LP	2533	24	0x45279400
PCIE3_HP	2534	24	0x45279800
PCIE3_LP	2535	24	0x45279C00

**Table 3-46. CBASS\_HC2\_0 Slave Modules Attributes**

Slaves	Firewall ID	Firewall Regions	Firewall Physical Address
There are no slave ports of modules or subsystems on this sub-interconnect	N/A	N/A	N/A

**Table 3-47. CBASS\_IPPHY0 Slave Modules Attributes**

Slaves	Firewall ID	Firewall Regions	Firewall Physical Address
EPWM0	2048	1	0x45200000
HRPWM0	2049	1	0x45200400
EPWM1	2050	1	0x45200800
HRPWM1	2051	1	0x45200C00
EPWM2	2052	1	0x45201000
HRPWM2	2053	1	0x45201400
EPWM3	2054	1	0x45201800
HRPWM3	2055	1	0x45201C00
EPWM4	2056	1	0x45202000
HRPWM4	2057	1	0x45202400
EPWM5	2058	1	0x45202800
HRPWM5	2059	1	0x45202C00
EQEP0	2064	1	0x45204000
EQEP1	2065	1	0x45204400
EQEP2	2066	1	0x45204800
ECAP0	2068	1	0x45205000
ECAP1	2069	1	0x45205400
ECAP2	2070	1	0x45205800
I2C0	2072	1	0x45206000
I2C1	2073	1	0x45206400
I2C2	2074	1	0x45206800
I2C3	2075	1	0x45206C00
I2C4	2076	1	0x45207000
I2C5	2077	1	0x45207400
I2C6	2078	1	0x45207800
ATL0	2080	1	0x45208000
I3C0	2082	4	0x45208800

**Table 3-47. CBASS\_IPPHY0 Slave Modules Attributes (continued)**

Slaves	Firewall ID	Firewall Regions	Firewall Physical Address
RTI0	2086	1	0x45209800
RTI1	2087	1	0x45209C00
RTI28	2094	1	0x4520B800
RTI29	2095	1	0x4520BC00
RTI30	2096	1	0x4520C000
RTI31	2097	1	0x4520C400
RTI16	2100	1	0x4520D000
RTI24	2104	1	0x4520E000
RTI25	2105	1	0x4520E400
RTI15	2108	1	0x4520F000
TIMER0	2112	1	0x45210000
TIMER1	2113	1	0x45210400
TIMER2	2114	1	0x45210800
TIMER3	2115	1	0x45210C00
TIMER4	2116	1	0x45211000
TIMER5	2117	1	0x45211400
TIMER6	2118	1	0x45211800
TIMER7	2119	1	0x45211C00
TIMER8	2120	1	0x45212000
TIMER9	2121	1	0x45212400
TIMER10	2122	1	0x45212800
TIMER11	2123	1	0x45212C00
TIMER12	2124	1	0x45213000
TIMER13	2125	1	0x45213400
TIMER14	2126	1	0x45213800
TIMER15	2127	1	0x45213C00
TIMER16	2128	1	0x45214000
TIMER17	2129	1	0x45214400
TIMER18	2130	1	0x45214800
TIMER19	2131	1	0x45214C00
SPI0	2136	1	0x45216000
SPI1	2137	1	0x45216400
SPI2	2138	1	0x45216800
SPI3	2139	1	0x45216C00
SPI4	2140	1	0x45217000
SPI5	2141	1	0x45217400
SPI6	2142	1	0x45217800
SPI7	2143	1	0x45217C00
UART0	2148	1	0x45219000
UART1	2149	1	0x45219400
UART2	2150	1	0x45219800
UART3	2151	1	0x45219C00
UART4	2152	1	0x4521A000
UART5	2153	1	0x4521A400
UART6	2154	1	0x4521A800

**Table 3-47. CBASS\_IPPHY0 Slave Modules Attributes (continued)**

Slaves	Firewall ID	Firewall Regions	Firewall Physical Address
UART7	2155	1	0x4521AC00
UART8	2156	1	0x4521B000
UART9	2157	1	0x4521B400
MCAN0	2160	4	0x4521C000
MCAN1	2161	4	0x4521C400
MCAN2	2162	4	0x4521C800
MCAN3	2163	4	0x4521CC00
MCAN4	2164	4	0x4521D000
MCAN5	2165	4	0x4521D400
MCAN6	2166	4	0x4521D800
MCAN7	2167	4	0x4521DC00
MCAN8	2168	4	0x4521E000
MCAN9	2169	4	0x4521E400
MCAN10	2170	4	0x4521E800
MCAN11	2171	4	0x4521EC00
MCAN12	2172	4	0x4521F000
MCAN13	2173	4	0x4521F400
PDMA_MISC_PSIL0_CFG	2184	1	0x45222000
PDMA_USART_PSIL0_CFG	2185	1	0x45222400
CBASS_IPPHY0_ERR	2303	1	0x4523FC00

**Table 3-48. CBASS\_MCASP\_G0\_0 Slave Modules Attributes**

Slaves	Firewall ID	Firewall Regions	Firewall Physical Address
MCASP0_CFG	2592	1	0x45288000
MCASP0_DMA	2593	1	0x45288400
MCASP1_CFG	2594	1	0x45288800
MCASP1_DMA	2595	1	0x45288C00
MCASP2_CFG	2596	1	0x45289000
MCASP2_DMA	2597	1	0x45289400
PDMA_AASRC_PSIL0_CFG	2606	1	0x4528B800
CBASS_MCASP_G0_0_ERR	2607	1	0x4528BC00

**Table 3-49. CBASS\_MCASP\_G1\_0 Slave Modules Attributes**

Slaves	Firewall ID	Firewall Regions	Firewall Physical Address
MCASP3_CFG	2496	1	0x45270000
MCASP3_DMA	2497	1	0x45270400
MCASP4_CFG	2498	1	0x45270800
MCASP4_DMA	2499	1	0x45270C00
MCASP5_CFG	2500	1	0x45271000
MCASP5_DMA	2501	1	0x45271400
MCASP6_CFG	2502	1	0x45271800
MCASP6_DMA	2503	1	0x45271C00
MCASP7_CFG	2504	1	0x45272000
MCASP7_DMA	2505	1	0x45272400
MCASP8_CFG	2506	1	0x45272800
MCASP8_DMA	2507	1	0x45272C00



**Table 3-49. CBASS\_MCASP\_G1\_0 Slave Modules Attributes (continued)**

Slaves	Firewall ID	Firewall Regions	Firewall Physical Address
MCASP9_CFG	2508	1	0x45273000
MCASP9_DMA	2509	1	0x45273400
MCASP10_CFG	2510	1	0x45273800
MCASP10_DMA	2511	1	0x45273C00
MCASP11_CFG	2512	1	0x45274000
MCASP11_DMA	2513	1	0x45274400
CBASS_MCASP_G1_0_ERR	2527	1	0x45277C00

**Table 3-50. CBASS\_RC\_CFG0 Slave Modules Attributes**

Slaves	Firewall ID	Firewall Regions	Firewall Physical Address
R5FSS0_CORE0_CFG	2368	2	0x45250000
R5FSS0_CORE1_CFG	2369	1	0x45250400
R5FSS1_CORE0_CFG	2370	2	0x45250800
R5FSS1_CORE1_CFG	2371	1	0x45250C00
C66SS0_CFG	2376	1	0x45252000
C66SS1_CFG	2377	1	0x45252400
MMCSD1_CFG	2380	4	0x45253000
MMCSD2_CFG	2381	4	0x45253400
ELM0	2384	1	0x45254000
PRU_ICSSG0	2386	8	0x45254800
PRU_ICSSG1	2387	8	0x45254C00
CPSW0	2390	2	0x45255800
VPFE0_RATCFG	2394	1	0x45256800
VPFE0_CFG	2395	1	0x45256C00
NAVSS0_NBSS_CFG	2398	7	0x45257800
COMPUTE_CLUSTER0	2400	16	0x45258000
RCNAVSS11_ECC_AGGR_CFG	2408	1	0x4525A000
RC4_ECC_AGGR_CFG	2409	1	0x4525A400
RC_R500_ECC_AGGR_CFG	2410	1	0x4525A800
RC_R511_ECC_AGGR_CFG	2411	1	0x4525AC00
RCNAVSS10_ECC_AGGR_CFG	2415	1	0x4525BC00
PBIST_RC_NAVSS_CFG	2416	1	0x4525C000
PBIST_RC_NB_CFG	2417	1	0x4525C400
PBIST_RC_R5FSS0_CFG	2418	1	0x4525C800
PBIST_RC_R5FSS1_CFG	2419	1	0x4525CC00
CBASS_RC0_ERR	2430	1	0x4525F800
CBASS_RC_CFG0_ERR	2431	1	0x4525FC00

**Table 3-51. CBASS\_RC0 Slave Modules Attributes**

Slaves	Firewall ID	Firewall Regions	Firewall Physical Address
C66SS0_SRC	2304	8	0x45240000
C66SS1_SRC	2305	8	0x45240400
GPMC0	2310	8	0x45241800
R5FSS0_CORE0	2314	4	0x45242800
R5FSS0_CORE1	2315	4	0x45242C00
R5FSS1_CORE0	2316	4	0x45243000

**Table 3-51. CBASS\_RC0 Slave Modules Attributes (continued)**

Slaves	Firewall ID	Firewall Regions	Firewall Physical Address
R5FSS1_CORE1	2317	4	0x45243400

**Table 3-52. INFRA\_CBASS0 Slave Modules Attributes**

Slaves	Firewall ID	Firewall Regions	Firewall Physical Address
PSRAM2KECC0	1	1	0x45000400
PSRAM2KECC0_ECCAGGR	2	1	0x45000800
PSC0	5	1	0x45001400
PLLCTRL0	6	1	0x45001800
GTC0	7	4	0x45001C00
PLL0_CFG	8	26	0x45002000
CTRL_MMR0	9	16	0x45002400
EFUSE0	11	1	0x45002C00
PBIST_INFRA0_CFG	12	1	0x45003000
GPIO0	16	1	0x45004000
GPIO1	17	1	0x45004400
GPIO2	18	1	0x45004800
GPIO3	19	1	0x45004C00
GPIO4	20	1	0x45005000
GPIO5	21	1	0x45005400
GPIO6	22	1	0x45005800
GPIO7	23	1	0x45005C00
ESM0_CFG	24	1	0x45006000
DCC0	32	1	0x45008000
DCC1	33	1	0x45008400
DCC2	34	1	0x45008800
DCC3	35	1	0x45008C00
DCC4	36	1	0x45009000
DCC5	37	1	0x45009400
DCC6	38	1	0x45009800
DCC7	39	1	0x45009C00
DCC8	40	1	0x4500A000
DCC9	41	1	0x4500A400
DCC10	42	1	0x4500A800
DCC11	43	1	0x4500AC00
DCC12	44	1	0x4500B000
GPIOMUX_INTRTR0_CFG	56	1	0x4500E000
MAIN2MCU_LVL_INTRTR0_CFG	57	1	0x4500E400
MAIN2MCU_PLS_INTRTR0_CFG	58	1	0x4500E800
CMPEVT_INTRTR0_CFG	59	1	0x4500EC00
TIMESYNC_INTRTR0_CFG	60	1	0x4500F000
MAIN_MCU1_INTROUTER0	62	1	0x4500F800
MAIN_MCU0_INTROUTER0	63	1	0x4500FC00
C66SS0_INTRTR0_CFG	64	1	0x45010000
C66SS0_INTRTR0_CFG	65	1	0x45010400
FW_CBASS0_ERR	73	1	0x45012400

**Table 3-52. INFRA\_CBASS0 Slave Modules Attributes (continued)**

Slaves	Firewall ID	Firewall Regions	Firewall Physical Address
INFRA_CBASS0_ERR	74	1	0x45012800
INFRA_ECC_AGGR0_CFG	81	1	0x45014400
PSRAMECC0	84	1	0x45015000
PSRAMECC0_ECCAGGR	85	1	0x45015400

**Table 3-53. MCU\_CBASS0 Slave Modules Attributes**

Slaves	Firewall ID	Firewall Regions	Firewall Physical Address
MCU_R5FSS0_CORE0	1024	4	0x45100000
MCU_R5FSS0_CORE0_CFG	1025	2	0x45100400
MCU_RTIO	1026	1	0x45100800
MCU_R5FSS0_CORE1	1028	4	0x45101000
MCU_R5FSS0_CORE1_CFG	1029	1	0x45101400
MCU_RT11	1030	1	0x45101800
MCU_FSS0_CFG	1032	12	0x45102000
MCU_FSS0_S1	1033	8	0x45102400
MCU_FSS0_S0	1036	8	0x45103000
MCU_PSRAM0	1048	1	0x45106000
MCU_MSRAM_1MB0	1050	8	0x45106800
MCU_MSRAM_1MB0_CFG	1051	1	0x45106C00
MCU_PSRAM0	1052	1	0x45107000
MCU_TIMER0	1056	1	0x45108000
MCU_TIMER1	1057	1	0x45108400
MCU_TIMER2	1058	1	0x45108800
MCU_TIMER3	1059	1	0x45108C00
MCU_TIMER4	1060	1	0x45109000
MCU_TIMER5	1061	1	0x45109400
MCU_TIMER6	1062	1	0x45109800
MCU_TIMER7	1063	1	0x45109C00
MCU_TIMER8	1064	1	0x4510A000
MCU_TIMER9	1065	1	0x4510A400
MCU_SPI0	1072	1	0x4510C000
MCU_SPI1	1073	1	0x4510C400
MCU_SPI2	1074	1	0x4510C800
MCU_DCC0	1088	1	0x45110000
MCU_DCC1	1089	1	0x45110400
MCU_DCC2	1090	1	0x45110800
MCU_ADC0_DMA	1104	1	0x45114000
MCU_ADC0_CFG	1105	2	0x45114400
MCU_ADC1_DMA	1106	1	0x45114800
MCU_ADC1_CFG	1107	2	0x45114C00
MCU_UART0	1120	1	0x45118000
MCU_I2C0	1152	1	0x45120000
MCU_I2C1	1153	1	0x45120400
MCU_I3C0	1160	4	0x45122000
MCU_I3C1	1161	4	0x45122400

**Table 3-53. MCU\_CBASS0 Slave Modules Attributes (continued)**

Slaves	Firewall ID	Firewall Regions	Firewall Physical Address
MCU_ESM0_CFG	1168	1	0x45124000
MCU_MCAN0	1184	4	0x45128000
MCU_MCAN1	1185	4	0x45128400
MCU_CTRL_MMR0	1200	8	0x4512C000
MCU_PLL0_CFG	1201	3	0x4512C400
MCU_EFUSE0	1208	1	0x4512E000
MCU_PBI0_CFG	1212	1	0x4512F000
MCU_PBI0_R5FSS0_CFG	1213	1	0x4512F400
MCU_CPSW0	1220	2	0x45131000
MCU_CBASS0_ERR	1244	1	0x45137000
MCU_FW_CBASS0_ERR	1246	1	0x45137800
MCU_ECC_AGGR0_CFG	1253	1	0x45139400

**Table 3-54. WKUP\_CBASS0 Slave Modules Attributes**

Slaves	Firewall ID	Firewall Regions	Firewall Physical Address
WKUP_PSC0	129	1	0x45020400
WKUP_PLLCTRL0	130	1	0x45020800
WKUP_CTRL_MMR0	131	16	0x45020C00
WKUP_GPIO0	132	1	0x45021000
WKUP_ESM0_CFG	133	1	0x45021400
WKUP_VTM0	135	3	0x45021C00
WKUP_GPIO1	137	1	0x45022400
WKUP_I2C0	144	1	0x45024000
WKUP_UART0	160	1	0x45028000
WKUP_GPIOMUX_INTRTR0_CFG	168	1	0x4502A000
WKUP_CBASS0_ERR	176	1	0x4502C000
WKUP_FW_CBASS0_ERR	177	1	0x4502C400
WKUP_ECC_AGGR0_CFG	178	1	0x4502C800

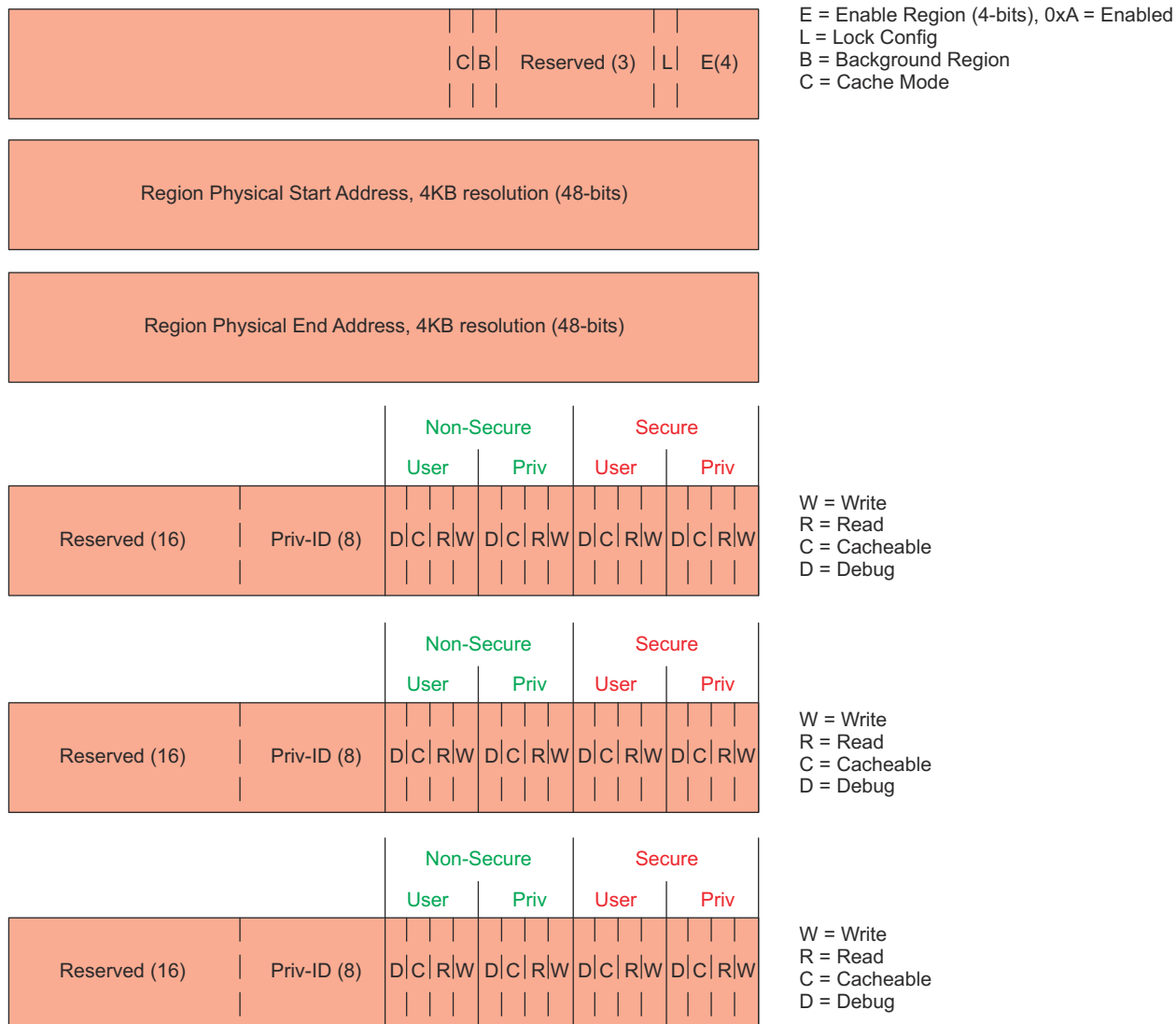
### 3.3.4.2.2 Memory or Region-based Firewalls

Memory/Data firewalls are designed to protect Memory (SRAM/DDR and so forth) and data regions (GPMC and so forth). Memory/Data firewalls have a defined firewall region count so that the memory can be partitioned into multiple firewall regions.

The architecture of Memory/Data firewalls is similar to Peripheral Firewalls, however, the Memory/Data firewall can have either 1 or 3 Priv-ID slots per region, with associated permissions. The number of Priv-ID slots is determined based on placement of firewall block. Each region in this type of firewall is defined by a physical start and end address.

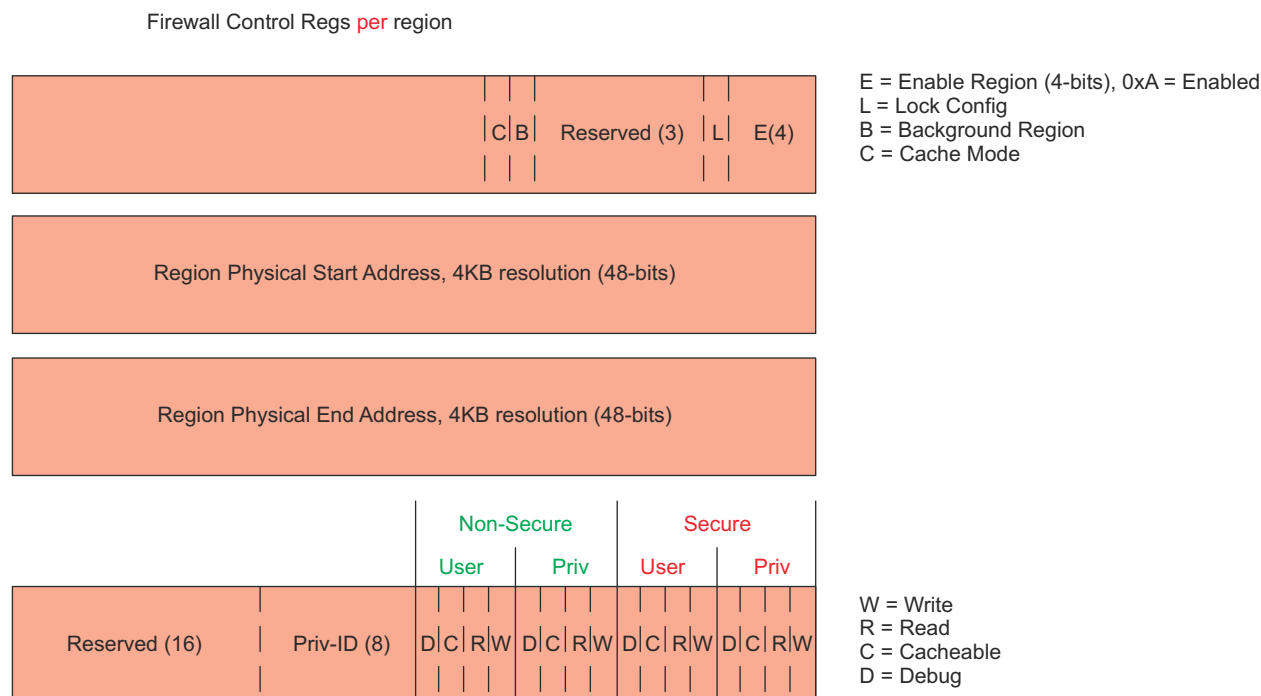
Figure 3-7 presents Memory/Data firewall config registers with 3 Priv-ID slots per region.

## Firewall Control Regs **per** region



### Figure 3-7. Memory/Data Firewall Config Registers with 3 Priv-ID slot per Region

Figure 3-8 presents Memory/Data firewall config registers with 1 Priv-ID slot per region.



SA-SPRUI1D-019

**Figure 3-8. Memory/Data Firewall Config Registers with 1 Priv-ID slot per Region**

The Memory/Data firewall is configured using dedicated VBUSP port to CBASS/AXI to VBUSM.C Bridge/DRU that connects to DMSC private VBUSP interconnect.

Memory/Data Firewalls also support background regions as in peripheral firewalls.

### 3.3.4.2.2.1 Region Based Firewall Functional Description

The region based firewall provides a memory region based protection and isolation mechanism. For each defined memory region, it checks the transaction attributes such as secure vs non-secure, debug vs non-debug, supervisor vs user mode, read vs write and so on. The transaction is dropped, if it does not match the configuration and appropriate violation code is registered with offending parameters.

Host modules (for example, AXI2VBUSMC bridge or CBASSes) provide transaction attributes that are checked by the firewalls, which then determine if the transaction should be blocked or passed. The firewall implements filtering algorithm that compares these host incoming transaction parameters against the region policy to give block indication to the host. Firewall region registers configure the filtering mechanism which includes setting region address range, region permission registers and region control register. There are also firewall exception registers used for violation reporting and logging.

The region based firewall supports multiple regions. Each one is defined by address range and associated access permission. The minimum memory region size is 4KB. The firewall concurrently checks the incoming transaction against all enabled regions looking for violations.

Each firewall is associated with the following registers:

- CBASS\_FW\_REGION\_i\_CONTROL



- CBASS\_FW\_REGION\_i\_PERMISSION\_0 to CBASS\_FW\_REGION\_i\_PERMISSION\_2
- CBASS\_FW\_REGION\_i\_START\_ADDRESS\_L and CBASS\_FW\_REGION\_i\_START\_ADDRESS\_H
- CBASS\_FW\_REGION\_i\_END\_ADDRESS\_L and CBASS\_FW\_REGION\_i\_END\_ADDRESS\_H

Each region is defined by start and end physical address and associated permission and control registers. A firewall can have 1-24 regions. In case there is more than 1 region, then these registers are duplicated for all regions. Setting the CBASS\_FW\_REGION\_i\_CONTROL[3-0] ENABLE field to 0xA enables the region and makes firewall check active for this region. Setting to 0x1 the CBASS\_FW\_REGION\_i\_CONTROL[8] BACKGROUND bit indicates to the firewall that the region is background region. Setting to 0x1 the CBASS\_FW\_REGION\_i\_CONTROL[9] CACHE\_MODE bit ignores cacheable check, so that it cannot fail. In this case the access check is performed based on the READ and WRITE bits in CBASS\_FW\_REGION\_i\_PERMISSION\_x. Clearing CACHE\_MODE enables the cacheable check, so that cacheable transactions are allowed only when the corresponding cacheable permission bit (CBASS\_FW\_REGION\_i\_PERMISSION\_x [y\_CACHEABLE]) is set. If the CBASS\_FW\_REGION\_i\_CONTROL[4] LOCK bit is set to 0x1, then the region configuration cannot be changed at all. This is one-time change. This bit is typically used for primary master to consume and lock its resources and then pass firewall control to secondary master.

In case two regions overlap, the CBASS\_FW\_REGION\_i\_CONTROL[8] BACKGROUND bit is used to select the appropriate permission to be used. The region whose BACKGROUND bit is 0x0 (foreground) takes precedence and its permissions are taken into effect. The background region is ignored.

#### Note

There can be only one background region per firewall. Foreground regions can have overlapping addresses only with the background region.

It is a software mistake to have overlapping regions with the BACKGROUND bit set to the same value (either background or foreground). Software must be careful to not configure two overlapping regions with the BACKGROUND bit set to the same value.

The firewall also checks if the transaction crosses a 4KB boundary. If so, the transaction is blocked regardless of any matching regions.

When a region is set to be debugable through the corresponding CBASS\_FW\_REGION\_i\_PERMISSION\_x [y\_DEBUG] bit, the firewall ignores the read and write checks for any debug transactions, so that it cannot fail. This allows debug breakpoints to be written out to code even in read-only regions.

When a region is set to be cacheable through the corresponding CBASS\_FW\_REGION\_i\_PERMISSION\_x [y\_CACHEABLE] bit and CACHE\_MODE = 0x0, the firewall ignores the read and write checks for any transaction (cacheable or not), so that it cannot fail due to these checks. This allows a write allocated cache to read a cache line even in write-only regions. Due to caches not protecting user and supervisor data from each other, the firewall allows cacheable access to the region when either the user cacheable permission or the supervisor cacheable permission is set.

The firewall notifies the host that the transaction is blocked in case of the following violation conditions:

- All regions are disabled.
- Incoming address does not hit any region.
- Non-secure incoming read transaction attempting to access configured secure-only write/read region.
- Non-secure incoming write transaction attempting to access configured secure-only write/read region.
- Secure incoming read transaction attempting to access configured non-secure-only write/read region.
- Secure incoming write transaction attempting to access configured non-secure region.
- Incoming secure write transaction attempting to access configured secure read region. This check is ignored if the region is configured as cacheable.
- Incoming secure read transaction attempting to access configured secure write region. This check is ignored if the region is configured as cacheable.

- Incoming non-secure write transaction attempting to access configured non-secure read region. This check is ignored if the region is configured as cacheable.
- Incoming non-secure read transaction attempting to access configured non-secure write region. This check is ignored if the region is configured as cacheable.
- Incoming non-secure debug access to non-debugable secure region.
- Incoming non-secure debug access to non-debugable non-secure region.
- Incoming secure debug access to non-debugable secure region.
- Incoming secure debug access to non-debugable non-secure region.
- Incoming cacheable transaction attempting to access non-cacheable configured region.

All violation parameters caused the exception are logged in the firewall exception registers described in *Firewall Exception Registers*. Table 3-55 shows the mapping between the register fields and violation parameters. If the CBASS\_EXCEPTION\_LOGGING\_CONTROL[0] DISABLE\_F bit is set to 0x1, logging is disabled.

If violation occurs, the corresponding firewall notifies WKUP\_DMSC0 driving high a dedicated signal. The notification (that is, the signal) can be masked by setting to 0x1 the CBASS\_EXCEPTION\_LOGGING\_CONTROL[1] DISABLE\_PEND bit. The firewall exception notification signal gets automatically cleared when the CBASS\_EXCEPTION\_LOGGING\_DATA3 register is read. That signal can be manually set via the CBASS\_EXCEPTION\_PEND\_SET[0] PEND\_SET bit and cleared via the CBASS\_EXCEPTION\_PEND\_CLEAR[0] PEND\_CLR bit. Reading one of these two bits returns the status of the signal (that is, violation occurred or not).

**Table 3-55. Firewall Violation Parameters**

Field	Value	Description
CBASS_EXCEPTION_LOGGING_HEADER0[31-24] TYPE_F	0x1	Exception type for firewall violation. This is fixed for the SoC firewalls and is used by software to detect that this corresponding violation comes from a firewall.
CBASS_EXCEPTION_LOGGING_HEADER0[23-8] SRC_ID	0x-	Firewall ID. Unique for each firewall. This is used to identify the exact firewall that issued violation so that software detects the precise source.
CBASS_EXCEPTION_LOGGING_HEADER0[7-0] DEST_ID	0x-	Destination ID of node where the firewall violation has be to routed.
CBASS_EXCEPTION_LOGGING_HEADER1[31-24] GROUP	0x0	Exception group. This is used to group exceptions to category. All firewall exceptions are in one group.
CBASS_EXCEPTION_LOGGING_HEADER1[23-16] CODE	Exception code:	
	0x0	Reserved.
	0x1	No region enabled.
	0x2	Incoming address does not hit any active region and transaction is dropped.
	0x3	Reserved.
	0x4	Cacheable error. A cacheable transaction attempting to read/write non-cached marked region.
	0x5	Debug error. A debug transaction attempting to read/write a non-allowed debug region.
	0x6	Read error. A Read transaction attempting to read from non-allowed read region.
	0x7	Write error. A Write transaction attempting to write from non-allowed write region.
	0x8	4KB crossing error. A transaction attempting to cross a 4KB boundary.
CBASS_EXCEPTION_LOGGING_DATA0[31-0] ADDR_L	0x-	Lower 32 address bits (31:0) of the incoming transaction
CBASS_EXCEPTION_LOGGING_DATA1[15-0] ADDR_H	0x-	Upper 16 address bits (47:32) of the incoming transaction

**Table 3-55. Firewall Violation Parameters (continued)**

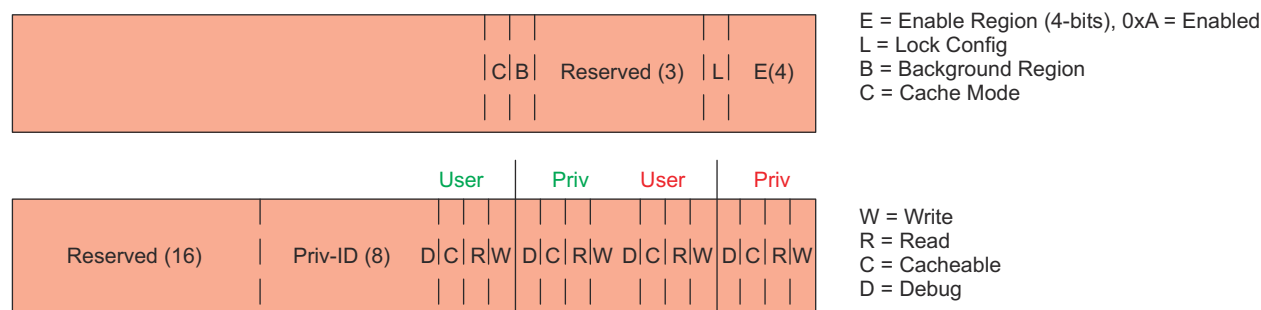
Field	Value	Description
CBASS_EXCEPTION_LOGGING_DATA2[7-0] PRIV_ID	Not used	Incoming transaction parameters
CBASS_EXCEPTION_LOGGING_DATA2[8] SECURE	0x-	
CBASS_EXCEPTION_LOGGING_DATA2[9] PRIV	0x-	
CBASS_EXCEPTION_LOGGING_DATA2[10] CACHEABLE	0x-	
CBASS_EXCEPTION_LOGGING_DATA2[11] DEBUG	0x-	
CBASS_EXCEPTION_LOGGING_DATA2[12] READ	0x-	
CBASS_EXCEPTION_LOGGING_DATA2[13] WRITE	0x-	
CBASS_EXCEPTION_LOGGING_DATA2[27-16] ROUTEID	0x-	
CBASS_EXCEPTION_LOGGING_DATA3[9-0] BYTECNT	0x-	Byte count of the incoming transaction

### 3.3.4.2.3 Channelized Firewalls

A Channelized firewall is designed to protect modules that have logical channels (Ring Accelerator and so forth, for example). These firewalls operate at the resolution of a channel and have no association with physical addresses. Each channel is defined as a logical control entity.

In case of channelized firewalls, the number of regions are less than or equal to number of logical channels and each typical channel/region is very small memory space (typically in Bytes, like 16 bytes). A channel is owned by one processing entity, hence only 1 Priv-ID slot is provided for each Channelized firewall along with associated permissions.

## Firewall Control Regs **per** region



SA-SPR11M0-020

The Channelized firewall is configured using dedicated VBUSP port to CBASS/Bridges that connects to DMSC private VBUSP interconnect.

## Note

For more information about the interconnect firewalls, see *Interconnect Firewalls*.

The channelized firewall protects an address space that consists of multiple channels where each channel needs its own permissions. This is in contrast to the region based firewall which implements a number of regions that can protect a programmed address range. The channelized firewall extends this to a larger number of ranges that can be protected, but each range is no longer programmable and is instead fixed to a particular channel address range. This allows each channel to be owned and protected individually. The channels within same memory region have same size. The system can allocate each channel to a particular owner or owner groups with certain permissions and guarantee that others cannot access that channel. This protection is useful for either accessing data resources, or control for each resource. The channelized firewall provides an efficient way to implement access protection and isolation requiring finer granularity than the region based firewall.

Same as the region based firewall, the channelized firewall may also support multiple regions. Each region contains a number of channels that need to be protected individually and has size in bytes equal to the channel's data to protect. This allows multiple regions so that the firewall can protect a module containing multiple sets of registers that are each channelized and need separate protections, such as data access as well as control setup.

The transaction addresses are compared against region addresses to identify which region is being accessed. Then the offset within the region is used to identify the particular channel being accessed. If a valid channelized region is not decoded during the firewall check, then the transaction is passed through unmodified because the module may have other regions which are not channelized, but accesses to them are already protected by a region based firewall so the channelized one should not block them.

The permission check is performed just like for a region based firewall. The channelized firewall checks for user or supervisor privileged levels. It also checks for transactions that cross important boundaries and gives errors if violated. The first check is for a transaction crossing a 4KB address boundary. This is illegal on the bus, so the firewall checks for compliance. The second check is if the transaction crosses a channel boundary. The channelized firewall blocks a transaction accessing multiple channels in the same burst, as it cannot check the permission for the entire range of channels. The channelized firewall also supports error logging when a transaction fails the checks. It sends information about the type of error and the transaction that caused it.

The channelized firewall is associated with the following registers:

- \*\_CH\_i\_CONTROL, where "i" can be from 0 to 63 and denotes the channel. For example, see *Data Routing Unit (DRU)*. This register is equivalent to the CBASS\_FW\_REGION\_i\_CONTROL register of the region based firewall.
- \*\_CH\_i\_PERMISSION\_x, where "i" can be from 0 to 63 and denotes the channel and "x" can be 0 to 2. For example, see *Data Routing Unit (DRU)*. This register is equivalent to the CBASS\_FW\_REGION\_i\_PERMISSION\_x register of the region based firewall.

#### 3.3.4.2.3.2 Null Error Reporting

The interconnect supports capturing null slave errors for reporting. A null slave error occurs when a transaction does not address any slaves based on the address map, resulting in the transaction returning error status. The null errors are captured into registers and an interrupt is generated. There is only one set of reporting registers, so only one error can be captured until cleared by software. The report captures some details of the transaction, such as the Route ID to identify the master involved and the address accessed. See *Null Error Reporting Registers* for details about the null error reporting registers.

### 3.3.5 VBUSM\_TIMEOUT\_GASKET (MCU\_TIMEOUT\_64B2)

#### 3.3.5.1 Overview and Feature List

##### 3.3.5.1.1 Features Supported

- Tracks and times out VBUSM transactions
- Reports an error via interrupt
- Logs information about the transaction for debug
- Provides appropriate responses on read/status interface so issuing initiator may proceed
- Reports an error if an unexpected response arrives
- Software and Hardware flushing of outstanding transactions
- Number of outstanding write and read transactions:
- MCU FW to Main FW: num\_writes = 2, num\_reads = 2
- FSS: num\_writes = 32, num\_reads = 32
- MCU Island to RC: num\_writes = 9, num\_reads = 15
- MCU to MAIN\_INFRA: num\_writes = 2, num\_reads = 2

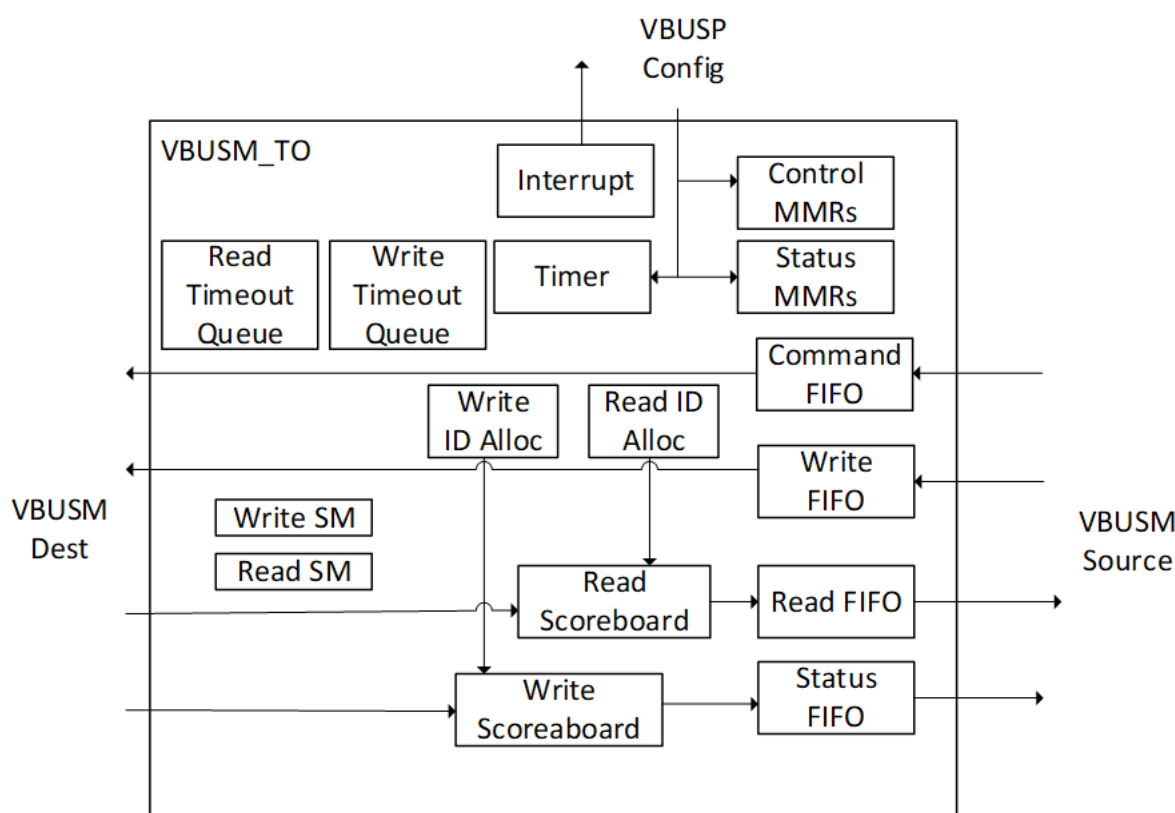
##### 3.3.5.1.2 Features Not Supported

- Protocol Checking
- Credit Mode
- Store and Forward
- Unaligned command and write data

#### 3.3.5.2 Functional Description

##### 3.3.5.2.1 Functional Operation

### 3.3.5.2.1.1 Overview



**Figure 3-10. VBUSM Block Diagram**

The VBUSM Timeout Gasket tracks transactions outstanding on a VBUSM 4.0 initiator interface. If a transaction has been outstanding for more than a programmed time, the transaction is considered timed-out (with the expectation that the target is hung and will not return a response). The gasket will then signal an error to the system and provide responses (with appropriate error status) so that the initiator can proceed and will not be hung as well.

### 3.3.5.2.1.2 FIFOs

There is a shallow FIFO on the source side for each of command, write data, write status, and read data to ease timing integration and reduce stalls.

### 3.3.5.2.1.3 ID Allocator

To track outstanding transactions, the gasket replaces the Command ID with a tag which it can then track. The ID Allocator allocates a free tag for each new transaction. If there are no free IDs to allocate, the command and write busses are stalled. Whenever a returned transaction is completed, the tag is freed.

### 3.3.5.2.1.4 Timer

There is a 30-bit free-running timer inside of the gasket. There is also a 2-bit eon bit. Whenever the timer is equal to or greater than the programmed timeout value (Timeout Value Register (Base Address + 0x14)), the timer is reset to 0 and the eon increments (when 3, the eon will roll-over to 0).

When a command is presented (the creq is asserted on the destination side) the command is given an eon value of the current eon minus 1 (for example, if the eon is currently 0, then the transaction is assigned an eon of 3). Thus each command has a minimum of 2 and a maximum of 3 eons before it times out. If the command is not accepted (cready does not assert) by the time the eon rolls around to the command eon, this is considered a Command Timeout. In this case:



- The creq will be de-asserted
- The command will be pushed onto the timeout queue
  - The gasket will return proper responses with a timeout status and appropriate interrupts and reporting as detailed in this spec
- The gasket will automatically transition to Software Flush mode:
  - If a command is not accepted, it is assumed that the target infrastructure is hung, at which point issuing more commands is of no use.
  - Software may disable software flush mode

Each transaction, once issued (cready is asserted), is assigned the current eon minus 1 (for example, if the eon is currently 0, then the transaction is assigned an eon of 3). Thus each transaction has a timeout of a minimum of 2 eons and a maximum of 3 (see Note below). If all of the responses for the transaction have not returned by the eon is equal to the transaction eon, this is considered a Transaction Timeout. In this case:

- The command will be pushed onto the timeout queue
  - The gasket returns proper responses with a timeout status, and appropriate interrupts and reporting as detailed in this spec.

The timer only increments when the gasket is enabled.

---

#### Note

To avoid violating bus protocol rules of not interrupt in-progress physical read bursts, if a read burst has been partially returned but an older command times out in the middle of this burst, that older command may be prevented from timing out until the current burst completes. Theoretically this may delay a read transaction timeout until up to 6 eons. However, this should be *extremely rare*, as a target is unlikely to return only part of a physical burst and then wait to complete that burst (this would be inefficient, as targets typically return a physical burst contiguously and split it in to logical bursts if more time is needed for subsequent data).

---

##### 3.3.5.2.1.5 Timeout Queue

There is a timeout queue each for reads and writes. Whenever a new transaction is submitted it is pushed onto the timeout queue with the correct eon (see [Section 3.3.5.2.1.4](#)). Whenever a transaction completes, it is popped off of the timeout queue. Whenever the head element of a timeout queue has an eon equal to the current eon, that transaction is considered timed-out. It is flushed by proxying back responses.

##### 3.3.5.2.1.6 Write Scoreboard

The write scoreboard keeps track of all write transactions currently in-flight through the gasket. The write scoreboard can keep track of <num\_writes> write transactions at a time (specified at configuration time, see [Section 3.3.5.5.1](#)). If a new write transaction comes in and the write scoreboard is full, the command bus (including all writes and reads) and the write data bus are stalled until a slot becomes available. The write scoreboard tracks the following information about all write commands that come through.

- OrderID
- RouteID
- CID
- Current Bytecnt
- Original Bytecnt
- Address
- Eon

When a write status is received on the status bus, the write scoreboard compares the tag (stored in the SID) to find the associated transaction. If the transaction matches one being tracked by the write scoreboard, then the status is passed to the egress FIFO and sbytecnt is subtracted from the current bytecnt. If the current bytecnt becomes 0, then the transaction is evicted from the write scoreboard and the tag is returned. If there are bytes left, then it is kept.

If the write status does not match a tracked transaction, then an Unexpected Response error is generated, the information about the response is logged (see [Section 3.3.5.2.1.12](#)), and the status is discarded (not forwarded to the egress FIFO). Probable reasons for an unexpected response are:

- A response to a transaction that has already been flushed due to timeout
- An error on the destination side

If the Timeout Queue is not empty and the next element in the queue is a write, the scoreboard issues a response so that the initiator is not hung:

1. Stall the write status interface until the following steps are done.
2. Issue a Transaction Timeout Interrupt.
3. Log the information about the transaction (See [Timeout Error Reporting](#)).
4. Issue a response to the write status egress FIFO.
  - a. sid/sorderid/srouteid set appropriately
  - b. sbytecnt set to the remaining outstanding bytes
  - c. sstatus set to 3'd3 (Timeout Error)
5. Evict the transaction from the write scoreboard.
6. Pop the item from the Timeout Queue.

### 3.3.5.2.1.7 Read Scoreboard

The read scoreboard keeps track of all read transactions currently in-flight through the gasket. The read scoreboard can keep track of <num\_reads> read transactions at a time (specified at configuration time, see [Section 3.3.5.5.1](#)). If a new read transaction comes in and the read scoreboard is full, the command bus (including all writes and reads) is stalled until a slot becomes available. The read scoreboard tracks the following information about all ad commands that come through.

- OrderID
- RouteID
- CID
- Current Bytecount
- Original Bytecnt
- Read Data Alignment
- Address
- Eon

When read data is received on the read data bus, the read scoreboard compares the tag (stored in the RID) to find the associated transaction. If the transaction matches one being tracked by the read scoreboard, then the data is passed to the read egress FIFO and rbytecnt is subtracted from the current bytecnt. If the current bytecnt becomes 0, then the transaction is evicted from the read scoreboard and the tag is returned. If there are bytes left, then it is kept.

If the read data does not match a tracked transaction, then an Unexpected Response error is generated, the information about the response is logged (see [Section 3.3.5.2.1.12](#)), and the read data is discarded (not forwarded to the egress FIFO). Probable reasons for an unexpected response are:

- A response to a transaction that has already been flushed due to timeout
- An error on the destination side

If the Timeout Queue is not empty and the next element in the queue is a read, the read scoreboard issues responses so that the initiator is not hung:

1. Stall the read data interface until the following steps are done.
2. Issue a Transaction Timeout Interrupt.
3. Log the information about the transaction (see [Section 3.3.5.2.1.10](#)).
4. Issue a series of read responses to the read data egress FIFO until the read is complete.
  - a. RID/RORDERID/RROUTEID set appropriately
  - b. RBYTECNT set appropriately

- c. RSTATUS set to 3'd3 (Timeout Error)
- d. RDATA set to 0
- 5. Evict the transaction from the read scoreboard.
- 6. Pop the item from the Timeout Queue.

#### 3.3.5.2.1.8 Flush Mode

In addition to flushing individual commands that timeout, there are two ways to force the flushing of all commands (flush mode): software and hardware. Software flush is initiated by writing to the Flush Register (Base Address + 0x10). Hardware flush is initiated whenever the flush input is asserted.

When in flush mode:

- All currently outstanding transactions will be flushed as per the descriptions in [Section 3.3.5.2.1.6](#) and [Section 3.3.5.2.1.7](#).
- All new incoming commands are not forwarded to the destination, but they are put into the appropriate scoreboard, pushed into the Timeout Queue, and subsequently flushed.
- All incoming responses from the destination side are discarded.

In addition, if a command times out before it is ever accepted on the destination side of the bridge, the gasket automatically enters Software flush mode (Flush Register (Base Address + 0x10) flush field set to 4'b1111). Software may use this MMR to subsequently turn flush mode off.

#### 3.3.5.2.1.9 Flushing

When a transaction is flushed (because it has timed out or the gasket is in flush mode), the gasket proxies back transactions to the initiator. When the transaction is completely flushed, it is popped from the relevant scoreboard. If, after a transaction has started (or completed) flushing, a response comes back for that transaction, it triggers an Unexpected Response.

If a transaction has been flushed, the ID is re-allocated to a new transaction, then a response comes for the old, timed-out transaction, it can cause faults in the system. For this reason, the timeout value should be sufficiently large that, if reached, it is expected that a response for the original transaction never returns.

For writes, the gasket sends a single write status transaction for all of the remaining bytes in the transaction. The status will have sstatus set to 0x3 (Timeout Error).

For reads, the gasket sends however many read data phases are required to fulfill the remaining bytes of the command. The data will be all 0 and the rstatus set to 0x3 (Timeout Error).

#### 3.3.5.2.1.10 Timeout Error Reporting

If a transaction times out and a Transaction Timeout Interrupt is asserted (and there are no currently outstanding Interrupts) then the following registers are loaded with the information about the transaction that timed out:

- Dir/orderid
- Routeid/id
- Original Bytecnt / Current Bytecnt
- Address0
- Address1

The Timeout Error Info Register (Base Address + 0x30) register keeps track of how many Transaction Timeout Interrupts have occurred since the last one was serviced. When a Transaction Timeout Interrupt occurs, the value is incremented (until saturated). If a Transaction Timeout Interrupt occurs and there is already one pending, then the counter is incremented by one, but the reporting information for the new transaction is lost.

#### 3.3.5.2.1.11 Command Timeout Error Reporting

If a transaction times out not because all of the responses didn't come back but because the command was never accepted in the first place (ready does not come back), this is a Command Timeout. A command timeout ultimately leads to two interrupts, one when the command times out and is pushed onto the timeout queue, and a second when the transaction times out from the queue. When the transaction timeout happens, the same information is recorded for reporting purposes.

### 3.3.5.2.1.12 Unexpected Response Reporting

If an Unexpected Response Interrupt is asserted (and there are no currently outstanding interrupts), then the following registers are loaded with information about the unexpected response:

- Dir/orderid
- Routeid/cid
- Bytecnt

The Unexpected Response Info Register (Base Address + 0x34) register keeps track of how many Unexpected Transaction Interrupts have occurred since the last one was serviced. Whenever an Unexpected Transaction Interrupt occurs, the value is incremented (until saturated). If an Unexpected Transaction Interrupt occurs and there is already one pending, then the counter is incremented by one, but the reporting information for the new transaction is lost.

### 3.3.5.2.1.13 Latency and Stalls

The gasket inserts 1 cycle of latency in each forward direction (1 on command, 1 on write data) and 2 on responses (2 on read data, 2 on write status). In addition, the following stall conditions may occur.

- If a write command comes in and there are no free slots in the write scoreboard, the entire command bus and write data bus stalls.
- If a read command comes in and there are no free slots in the read scoreboard, the entire command bus stalls.
- If a write transaction is being timed out, then the write status bus stalls while the write scoreboard flushes the command.
- If a read transaction is being timed out, then the read data bus stalls while the read scoreboard flushes the command.

### 3.3.5.2.1.14 Bypass

The gasket is disabled by default. When disabled (see Enable Register (Base Address + 0x0C)), the gasket operates in bypass mode and just passes all commands, data, and responses through untouched (though the latency is still added and the CID is still remapped so that order can be preserved in the case where the gasket is enabled or disabled with transactions in flight). In Bypass mode, there are no timed out transactions and thus no timeout interrupts. However, if an unexpected response is received, the unexpected response logic will still work as normal (including issuing an interrupt).

### 3.3.5.2.1.15 Safety

The gasket supports the following features for functional safety:

- CBA safety signaling on source and destination interfaces (if enabled with <safe\_bus> parameter)
- CBA safety signaling on configuration interface (if enabled with <safe> parameter)
- ECC protection on scoreboard RAMs (if enabled with <safe> parameter)
- Parity protection on scoreboard flops (if enabled with <safe> parameter)
- Parity protection or multi-bit field protection on configuration MMRs (if enabled with <safe> parameter)
- Parity protection on the free-running timer (if enabled with <safe> parameter)

An uncorrected error reported on any of the protected fields indicates a potentially dangerous fault, and the system integrator should take appropriate action (such as transition to a safe state).

A fault in the free-running timer may cause errors by prematurely timing out transactions, but it may also be recoverable by taking the following steps.

1. Disable the gasket.
2. Wait for scoreboards to become empty. If they don't become empty for a long time, this may indicate that there are other issues in the system. The system integrator can use Software Flush, but additional intervention is probably required (such as transition to a safe state or system reset).
3. Write a 0 to the Timer Register (Base Address + 0x18) to reset the timer.
4. Enable the gasket.

### 3.3.5.3 Interrupt Conditions

Table 3-56 shows the interrupts generated by the module.

**Table 3-56. Interrupts**

Module Instance	Module Interrupt Signal	Destination Interrupt Name	Destination	Description	Type
MCU_TIMEOUT_64B2	MCU_TIMEOUT_64B2_TRANS_ERR_LVL_0	MCU_ESM0_LVL_IN_76	MCU_ESM0	MCU_TIMEOUT_64B2 interrupt request	Level

When an interrupt is received, software should read the Error Interrupt Enabled Status/Clear Register (Base Address + 0x24) to determine what type of interrupt it is.

#### 3.3.5.3.1 Transaction Error Interrupt

The Transaction Error Interrupt signals that the gasket has detected an error on a transaction. The gasket returns an appropriate response to the source side, so the source will not immediately violate a safety goal, but it is assumed that something uncorrected has happened in the system. The system must treat this as a dangerous event and react accordingly. The specific response is the responsibility of the system integrator, but possibilities are discussed below. There are three possible causes of the Transaction Error Interrupt: transaction timeout, unexpected response, and command timeout. Software should query the Error Interrupt Enabled Status/Clear Register (Base Address + 0x24) to determine the cause.

##### 3.3.5.3.1.1 Transaction Timeout

If all of the responses to a transaction have not been received within the time allotted (programmed by the Timeout Value Register (Base Address + 0x14)) or if a transaction was never accepted (Command Timeout) and placed on the timeout queue, then a transaction is considered to have timed out. The interrupt indicates that the gasket is flushing the transaction and that information has been logged about the transaction that timed out in the following registers:

- Error Transaction Valid/Dir/ID Register (Base Address + 0x38)
- Error Transaction RouteID/OrderID Register (Base Address + 0x3C)
- Error Transaction Bytecnt Register (Base Address + 0x40)
- Error Transaction Upper Address Register (Base Address + 0x44)
- Error Transaction Lower Address Register (Base Address + 0x48)

When servicing a Transaction Timeout Error, software should perform the following steps:

1. Read the Timeout Error Info Register (Base Address + 0x30) to determine how many transaction timeouts have occurred since last serviced.
2. Read the Error Transaction Valid/Dir/ID Register (Base Address + 0x38). If the valid field is 0 or the type field is Unexpected Response, then these registers do not contain any information about the timed out transaction, skip to step 4.
3. Read the information from the following registers:
  - Error Transaction RouteID/OrderID Register (Base Address + 0x3C)
  - Error Transaction Bytecnt Register (Base Address + 0x40)
  - Error Transaction Upper Address Register (Base Address + 0x44)
  - Error Transaction Lower Address Register (Base Address + 0x48)
  - Error Transaction Lower Address Register (Base Address + 0x48)
4. Write the number of timed out events serviced this ISR (read in step 1) to the Timeout Error Info Register (Base Address + 0x30).
5. Clear the interrupt by writing to the appropriate bit in the Error Interrupt Enabled Status/Clear Register (Base Address + 0x24).

If the write in step 4 does not decrement the value to 0 (because a new transaction has timed out), then the interrupt will re-issue after the write in step 5.

This section only describes how to service the interrupt. The system integrator must decide what actions to take based on this information. Suggestions include:

- Log the information for debug purposes
- Determine the target that is unresponsive and take appropriate action
  - Reset target
  - Reset main SoC
  - Reset whole device

### 3.3.5.3.1.2 Unexpected Response

If the gasket receives a response that it was not expecting then this interrupt is triggered. See [Section 3.3.5.2.1.12](#) for a discussion of unexpected responses. When this interrupt is triggered, information about the unexpected response will be in the following registers:

- Error Transaction Valid/Dir/ID Register (Base Address + 0x38)
- Error Transaction RouteID/OrderID Register (Base Address + 0x3C)
- Error Transaction Bytecnt Register (Base Address + 0x40)

When servicing an Unexpected Response Error, software should perform the following steps:

1. Read the Unexpected Response Info Register (Base Address + 0x34) to determine how many unexpected responses have occurred since last serviced.
2. Read the Error Transaction Valid/Dir/ID Register (Base Address + 0x38). If the valid field is 0 or the type field is Timeout Error, then these registers do not contain any information about the unexpected response, skip to step 4.
3. Read the information from the following registers:
  - Error Transaction RouteID/OrderID Register (Base Address + 0x3C)
  - Error Transaction Bytecnt Register (Base Address + 0x40)
4. Write the number of unexpected responses serviced this ISR (read in step 1) to the Unexpected Response Info Register (Base Address + 0x34).
5. Clear the interrupt by writing to the appropriate bit in the Error Interrupt Enabled Status/Clear Register (Base Address + 0x24).

If the write in step 4 does not decrement the value to 0 (because a new unexpected response has arrived), then the interrupt re-issues after the write in step 5.

This section only describes how to service the interrupt. The system integrator must decide what actions to take based on this information. Suggestions include:

- Log the information for debug purposes.
- Determine the target that is unresponsive and take appropriate action:
  - Reset target.
  - Reset main SoC.
  - Reset whole device.

### 3.3.5.3.1.3 Command Timeout

A Command Timeout happens when a command is presented on the destination side (creq) but never accepted (cready) within the allotted time. See [Section 3.3.5.2.1.11](#). When this happens, a Command Timeout interrupt is issued. When servicing a Command Timeout Error, software should perform the following step:

1. Clear the interrupt by writing to the appropriate bit in the Error Interrupt Enabled Status/Clear Register (Base Address + 0x24).

When a Command Timeout occurs, the gasket automatically transitions to Software Flush mode (see [Section 3.3.5.2.1.8](#)). The system can assume that the target infrastructure is hung and take appropriate action.

Suggested actions include:

- Transitioning to a safe state.
- Re-setting the target domain to eliminate the hang.
- Re-enable the gasket (disable software flush) and attempt further access.



### 3.3.5.4 Memory Map

**Table 3-57. Module Memory Map**

Address Offset	Register
0x00	Revision Register
0x04	Configuration Register
0x08	Info Register
0x0C	Enable Register
0x10	Flush Register
0x14	Timeout Value Register
0x18	Timer Register
0x1C	Reserved
0x20	Error Interrupt Raw Status/Set Register
0x24	Error Interrupt Enabled Status/Clear Register
0x28	Error Interrupt Mask Set Register
0x2C	Error Interrupt Mask Clear Register
0x30	Timeout Error Info Register
0x34	Unexpected Response Info Register
0x38	Error Transaction Valid/Dir/ID Register
0x3C	Error Transaction RouteID/OrderID Register
0x40	Error Transaction Bytecnt Register
0x44	Error Transaction Upper Address Register
0x48	Error Transaction Lower Address Register
0x4C	Error Transaction Timestamp Register

#### 3.3.5.4.1 Revision Register (Base Address + 0x00)

The Revision Register contains the major and minor revisions for the VBUSM Timeout Gasket.

Bits	Field	Type	Reset	Description
31:30	scheme	r	01	Always read as 01. Writes have no effect.
29:28	rsvd	r	10	Always read as 00.
27:16	func	r	0x606	Always read as the assigned func id.
15:11	rtl	r	0x4	RTL version of the module. The "R" value in the X.Y.R.Z as defined by K3.
10:8	major	r	0x1	Major revision of module. The "X" value in the X.Y.R.Z defined in K3.
7:6	custom	r	Hard coded	Special version.
5:0	minor	r	0x0	Minor revision of module. The "Y" value in the X.Y.R.Z defined in K3

#### 3.3.5.4.2 Configuration Register (Base Address + 0x04)

The Configuration Register contains information about the configuration of the gasket.

Bits	Field	Type	Reset	Description
31:24	Reserved	r	0x0	Reserved. Read as 0.
23:16	writes	r	<num_writes>	This is the total number of slots in the write scoreboard.
15:8	Reserved	r	0x0	Reserved. Read as 0.
7:0	reads	r	<num_reads>	This is the total number of slots in the read scoreboard

### 3.3.5.4.3 Info Register (Base Address + 0x08)

The Info Register contains information about the current status of the gasket.

Bits	Field	Type	Reset	Description
31:25	Reserved	r	0x0	Reserved. Read as 0.
24:16	cur_writes	r	0x0	This is the current number of occupied slots in the write scoreboard.
15:9	Reserved	r	0x0	Reserved. Read as 0.
8:0	cur_reads	r	0x0	This is the current number of occupied slots in the read scoreboard

### 3.3.5.4.4 Enable Register (Base Address + 0x0C)

The Enable Register contains the enable control of the gasket.

Bits	Field	Type	Reset	Description
31:4	Reserved	r	0x0	Reserved. Read as 0.
3:0	enable	r/w	0x0	This controls whether the gasket is enabled or not. See <a href="#">Section 3.3.5.2.1.14</a> . <ul style="list-style-type: none"> <li>4'b0000 – Disabled</li> <li>4'b0001-4'b1111 – Enabled</li> </ul>

### 3.3.5.4.5 Flush Register (Base Address + 0x10)

The Flush Register contains the flush control of the gasket.

Bits	Field	Type	Reset	Description
31	ext_flush	r	*	This indicates the value of the flush input synchronized to the gasket clock
31:4	Reserved	r	0x0	Reserved. Read as 0.
3:0	flush	r/w	0x0000	This is the software control for whether the gasket is in flush mode or not. See <a href="#">Section 3.3.5.2.1.8</a> . <ul style="list-style-type: none"> <li>4'b0000-4'b1110 – Normal Mode</li> <li>4'b1111 – Flush Mode</li> </ul> <p>Note that this field can be automatically set to 4'b1111 (Flush Mode) in the event of a Command Timeout (see <a href="#">Section 3.3.5.2.1.4</a>)</p>

### 3.3.5.4.6 Timeout Value Register (Base Address + 0x14)

The Timeout Value Register contains the timeout value for scoreboarded transactions.

Bits	Field	Type	Reset	Description
31:30	Reserved	r	0x0	Reserved. Read as 0.

Bits	Field	Type	Reset	Description
29:0	timeout	r/w	0x3FFF_FFFF	<p>This is the number of cycles in each eon (see <a href="#">Section 3.3.5.2.1.4</a>). Each transaction can be outstanding for a minimum of 2 and a maximum of 3 eons before it is considered timed out. Note that if this value is changed while there are outstanding transactions it may affect the timeout of those transactions.</p> <ul style="list-style-type: none"> <li>If the value is written to be greater than the current value, then all outstanding transactions may live longer based on the new timeout value.</li> <li>If the value is written to be lower than the current value, then all outstanding transactions may live shorter based on the new timeout value <ul style="list-style-type: none"> <li>If the new timeout value written is equal to or larger than the current timer value, the timer will be reset to 0 and the eon will flip. This may cause currently outstanding transactions to time out faster.</li> </ul> </li> </ul> <p>Note: This field can only be written if byte enables 3:0 are set. Byte writes will be ignored.</p>

#### 3.3.5.4.7 Timer Register (Base Address + 0x18)

The Timer Register contains the current value for free-running timer.

Bits	Field	Type	Reset	Description
31:30	eon	r	0x0	This is the current eon. It toggles each time the timer field is reset to 0. See <a href="#">Section 3.3.5.2.1.4</a> .
29:0	timer	r/wtc	0x0	<p>This is the current value of the free-running counter. It increments once per clock cycle if the gasket is enabled. When the value is equal to or greater than the timeout value in the Timeout Value Register (Base Address + 0x10), it will be reset to 0 and begin incrementing again. This value only increments when the gasket is enabled.</p> <p>If the gasket is disabled (Enable Register (Base Address + 0x08)) and a zero is written to this field, the timer will reset to 0 and be ready to increment when the gasket is re-enabled. The eon will also be reset to 0.</p> <p>If the gasket is enabled, writes have no effect. See <a href="#">Section 3.3.5.2.1.4</a>.</p>

#### 3.3.5.4.8 Error Interrupt Raw Status/Set Register (Base Address + 0x20)

The Error Interrupt Raw Status Register contains the raw status of the gasket interrupts

Bits	Field	Type	Reset	Description
31:3	Reserved	r	0x0	Reserved. Read as 0.

Bits	Field	Type	Reset	Description
2	cmd	r/w1ts	0x0	This field represents the raw status of the Command Timeout Error. Read 0 – No pending command timeout interrupt 1 – Pending command timeout interrupt Write 0 – No effect 1 – Set command timeout interrupt
1	unexp	r/w1ts	0x0	This field represents the raw status of the Unexpected Response Error. Read 0 – No pending unexpected response interrupt 1 – Pending unexpected response interrupt Write 0 – No effect 1 – Set unexpected response interrupt
0	timeout	r/w1ts	0x0	This field represents the raw status of the Timeout Error Interrupt. Read 0 – No pending timeout interrupt 1 – Pending timeout interrupt Write 0 – No effect 1 – Set timeout interrupt

### 3.3.5.4.9 Error Interrupt Enabled Status/Clear Register (Base Address + 0x24)

The Error Interrupt Raw Status Register contains the raw status of the gasket interrupts.

Bits	Field	Type	Reset	Description
31:3	Reserved	r	0x0	Reserved. Read as 0.
2	cmd	r/w1tc	0x0	This field represents the masked status of the Command Timeout Error. Read 0 – No pending command timeout interrupt 1 – Pending command timeout interrupt Write 0 – No effect 1 – Clear command timeout interrupt
1	unexp	r/w1tc	0x0	This field represents the masked status of the Unexpected Response Error. Read 0 – No pending unmasked unexpected response interrupt 1 – Pending unmasked unexpected response interrupt Write 0 – No effect 1 – Clear unexpected response interrupt

Bits	Field	Type	Reset	Description
0	timeout	r/w1tc	0x0	This field represents the masked status of the Timeout Error Interrupt. Read 0 – No pending unmasked timeout interrupt 1 – Pending unmasked timeout interrupt Write 0 – No effect 1 – Clear timeout interrupt

#### 3.3.5.4.10 Error Interrupt Mask Set Register (Base Address + 0x28)

The Error Interrupt Mask Set is used to enabled interrupts.

Bits	Field	Type	Reset	Description
31:3	Reserved	r	0x0	Reserved. Read as 0.
2	cmd	r/w1ts	0x1	This field is used to set the mask for the Command Timeout Error. Read 0 – Interrupt is disabled (mask cleared) 1 – Interrupt is enabled (mask set) Write 0 – No effect 1 – Enable interrupt (mask set)
1	unexp	r/w1ts	0x1	This field is used to The Error Interrupt Mask Set is used to enabled interrupts. set the mask for the Unexpected Response Error. Read 0 – Interrupt is disabled (mask cleared) 1 – Interrupt is enabled (mask set) Write 0 – No effect 1 – Enable interrupt (mask set)
0	timeout	r/w1ts	0x1	This field is used to set the mask for the Timeout Interrupt. Read 0 – Interrupt is disabled (mask cleared) 1 – Interrupt is enabled (mask set) Write 0 – No effect 1 – Enable interrupt (mask set)

#### 3.3.5.4.11 Error Interrupt Mask Clear Register (Base Address + 0x2C)

The Error Interrupt Mask Clear Register is used to disable interrupts.

Bits	Field	Type	Reset	Description
31:3	Reserved	r	0x0	Reserved. Read as 0.

Bits	Field	Type	Reset	Description
2	cmd	r/w1tc	0x1	This field is used to clear the mask for the Command Timeout Error. Read 0 – Interrupt is disabled (mask cleared) 1 – Interrupt is enabled (mask set) Write 0 – No effect 1 – Disable interrupt (mask clear)
1	unexp	r/w1tc	0x1	This field is used to clear the mask for the Unexpected Response Error. Read 0 – Interrupt is disabled (mask cleared) 1 – Interrupt is enabled (mask set) Write 0 – No effect 1 – Disable interrupt (mask clear)
0	timeout	r/w1tc	0x1	This field is used to clear the mask for the Timeout Interrupt. Read 0 – Interrupt is disabled (mask cleared) 1 – Interrupt is enabled (mask set) Write 0 – No effect 1 – Disable interrupt (mask clear)

### 3.3.5.4.12 Timeout Error Info Register (Base Address + 0x30)

The Timeout Error Info Register contains information about the number of Timeout errors that have occurred.

Bits	Field	Type	Reset	Description
31:2	Reserved	r	0x0	Reserved. Read as 0.
1:0	timeouts	r/wtd	0x0	This field contains information about how many transactions have timed out since the last one was serviced. Writing to this register decrements the contents by the value written. For instance, if the value is 2 and a 1 is written, the value will decrement to 1. If this field is non-0 when the interrupt is cleared by writing to the Error Interrupt Enabled Status/ Clear Register (Base Address + 0x24) the interrupt will be re-issued. The value saturates at 3, so if there are more than three pending it will be unknown exactly how many have occurred. If the value is saturated at 3 and a value of 3 is written, the value will turn to 0. Read 0 – No pending timeout interrupts 1 – One pending timeout interrupt 2 – Two pending timeout interrupts 3 – Three or more pending timeout interrupts



### 3.3.5.4.13 Unexpected Response Info Register (Base Address + 0x34)

The Unexpected Response Info Register contains information about the number of Timeout errors that have occurred.

Bits	Field	Type	Reset	Description
31:2	Reserved	r	0x0	Reserved. Read as 0.
1:0	unexps	r/wtd	0x0	<p>This field contains information about how many unexpected responses have been received since the last one was serviced. Writing to this register decrements the contents by the value written. For instance, if the value is 2 and a 1 is written, the value will decrement to 1. If this field is non-0 when the interrupt is cleared by writing to the Error Interrupt Enabled Status/Clear Register (Base Address + 0x24) the interrupt will be re-issued.</p> <p>The value saturates at 3, so if there are more than three pending it will be unknown exactly how many have occurred. If the value is saturated at 3 and a value of 3 is written, the value will turn to 0.</p> <p>Read</p> <p>0 – No pending unexpected response interrupts</p> <p>1 – One pending unexpected response interrupt</p> <p>2 – Two pending unexpected response interrupts</p> <p>3 – Three or more pending unexpected response interrupts</p>

### 3.3.5.4.14 Error Transaction Valid/Dir/RouteID Register (Base Address + 0x38)

This register contains the type, direction, and transaction ID of the transaction error that was captured.

Bits	Field	Type	Reset	Description
31:28	Reserved	r	0x0	Reserved. Read as 0.
27:16	rid	r	0x0	Route ID – This indicates the Route ID of the captured transaction
15:12	Reserved	r	0x0	Reserved. Read as 0.
11:8	oid	r	0x0	Order ID – This indicates the Order ID of the capture transaction
7:4	Reserved	r	0x0	Reserved. Read as 0.
2	dir	r	0x0	<p>Direction – This indicates whether the captured transaction was a read or a write</p> <p>0 – Write</p> <p>1 – Read</p>
1	type	r	0x0	<p>Type – This indicates what type of error these registers contain information about</p> <p>0 – Transaction Timeout</p> <p>1 – Unexpected Response</p>

Bits	Field	Type	Reset	Description
0	valid	r	0x0	<p>If this field is a 1, then the contents of this and the below registers is considered valid. I.e. it contains the information about the transaction that was captured. If this field is 0 then this and the other listed registers are not valid. This is because there is either no error pending, or the error pending did not have any captured information because there was an error pending in front of it that had already captured information.</p> <ul style="list-style-type: none"> <li>0 – Invalid</li> <li>1 – Valid</li> </ul>

### 3.3.5.4.15 Error Transaction Tag/CommandID Register (Base Address + 0x3C)

This register contains the routeid and orderid of the transaction error that was captured.

Bits	Field	Type	Reset	Description
31:28	Reserved	r	0x0	Reserved. Read as 0.
27:16	tag	r	0x0	Tag – This indicates the CID/RID/SID of the transaction on the destination side of the gasket. This is the tag assigned to the transaction. For an Unexpected Transaction Error, this field is whatever the sid/rid of the transaction was. For a Timeout Error, this is the tag (replacement CID) that was used.
15:12	Reserved	r	0x0	Reserved. Read as 0.
11:0	cid	r	0x0	Command ID – This indicates the original Command ID (SID/RID) of the command. This field is only valid on a Timeout Error, not on an Unexpected Transaction Error.

### 3.3.5.4.16 Error Transaction Bytecnt Register (Base Address + 0x40)

This register contains the byte count of the transaction error that was captured.

Bits	Field	Type	Reset	Description
31:26	Reserved	r	0x0	Reserved. Read as 0.
25:16	cbytecnt	r	0x0	Current Byte Count – If this is a timed out transaction, this is the number of bytes that were not returned as of the time the transaction timed out. If this is an unexpected response transaction, then this field is not applicable.
15:10	Reserved	r	0x0	Reserved. Read as 0.
9:0	obytecnt	r	0x0	Original Byte Count – If this is a timed out transaction, then this field represents the CBYTECNT value of the original command. If this is an unexpected response transaction, then this field contains the value of the bytecnt of the unexpected transaction (sbytecnt or rbytecnt).

### 3.3.5.4.17 Error Transaction Upper Address Register (Base Address + 0x44)

This register contains the upper address bits of the transaction error that was captured. If the address is 32-bits or smaller, this field will be all 0. *N* is the number of bits on the source/dest address bus minus 33.

Bits	Field	Type	Reset	Description
31:N	Reserved	r	0x0	Reserved. Read as 0.
N-1:0	addr_u	r	0x0	Upper Address bits – If the captured transaction was a Timeout Error, this field represents the lower address bits N-1:32] of the original transaction. If the error was an Unexpected Response error, then this field is not applicable.

#### 3.3.5.4.18 Error Transaction Lower Address Register (Base Address + 0x48)

This register contains the lower address bits of the transaction error that was captured.

Bits	Field	Type	Reset	Description
31:0	addr_l	r	0x0	Lower Address bits – If the captured transaction was a Timeout Error, this field represents the lower address bits [31:0] of the original transaction. If the error was an Unexpected Response error, then this field is not applicable.  If the address width is less than 32, then the bits above the address range will be read as 0.

#### 3.3.5.5 Integration Overview

##### 3.3.5.5.1 Parameterization Requirements

**Table 3-58. Module Parameters**

Parameter	Value
num_writes	This is the number of outstanding write transactions that the write scoreboard can handle at once. When the scoreboard is full, all subsequent commands will be stalled, so this value should be picked appropriately for throughput. See <a href="#">Section 3.3.5.2.1.6</a> . <ul style="list-style-type: none"> <li>1 – 1024 (Default 4)</li> </ul>
num_reads	This is the number of outstanding read transactions that the read scoreboard can handle at once. When the scoreboard is full, all subsequent commands will be stalled, so this value should be picked appropriately for throughput. See <a href="#">Section 3.3.5.2.1.7</a> . <ul style="list-style-type: none"> <li>1 – 1024 (Default 4)</li> </ul>
safe	Add CBA Safety Signaling on the config bus and internal flop protection. Include EDC controller and export sVbus interface(s) <ul style="list-style-type: none"> <li>0 – No Safety Signals</li> <li>1 – Safety Signals Included (Default)</li> </ul>
safe_bus	Add CBA Safety Signaling on the source and destination VBUSM interfaces <ul style="list-style-type: none"> <li>0 – No Safety Signals</li> <li>1 – Safety Signals Included (Default)</li> </ul>
src_name	This is the prefix for the signals on the source bus
dst_name	This is the prefix for the signals on the destination bus
osigs	Space separate list of optional signals on the CBA interfaces
data_width	Width of the data bus. Default 32
addr_width	Width of the address bus. Default 32
region_cnt	Number of CBA Regions. Default 1

**Table 3-58. Module Parameters (continued)**

Parameter	Value
pipe	Enable EDC Pipe Stage. Default 0 (off)

### 3.3.5.6 I/O Description

#### 3.3.5.6.1 Clockstop Idle

The gasket only returns clockstop idle when there are no outstanding transactions in any of the FIFOs or in the scoreboard. The free-running timer may still be running (but pauses if the clock pauses).

#### 3.3.5.6.2 Flush

The flush signal is expected to indicate that the destination side is down. It is synchronized internally.

#### 3.3.5.6.3 Module I/O

[Table 3-59](#) shows the interrupts from the module.

**Table 3-59. Interrupt Requests**

Module Instance	Module Interrupt Signal	Destination Interrupt Name	Destination	Description	Type
MCU_TIMEOUT_64B2	MCU_TIMEOUT_64B2_TRANS_ERR_LVL_0	MCU_ESM0_LVL_IN_76	MCU_ESM0	MCU_TIMEOUT_64B2 interrupt request	Level

If <safe> is enabled, the following sVbus signals in [Table 3-60](#) are present.

**Table 3-60. EDC Controller Interface**

Pin Name	Type	Function
edc_ctrl_strb	In	Strobe
edc_ctrl_txd	In	Transmit Data
edc_ctrl_rxd	Out	Receive Data
edc_ctrl_pend[1:0]	Out	Event Pending

### 3.3.5.7 User's Guide

#### 3.3.5.7.1 Programmer's Guide

##### 3.3.5.7.1.1 Initialization

Software should take the following steps when initializing the gasket.

1. Enable/Disable gasket as desired (Default enabled - Enable Register (Base Address + 0x08)).
2. Set the timeout value as desired (Default max - Timeout Value Register (Base Address + 0x10)).
3. Enable/Disable interrupts as desired (Default enabled - Error Interrupt Enabled Status/Clear Register (Base Address + 0x24)).

##### 3.3.5.7.1.2 Software Flush

If the system determines that it must flush all outstanding transactions (for instance, because the main SoC is in an error condition and is going to be reset), software may do this by writing to the Flush Register (Base Address + 0x0C). When all transactions are flushed, software should exit Flush mode. If the destination side is in reset, this should trigger hardware flush, keeping the gasket returning any transactions that arrive. The system should also use the CBA disconnect interface to keep transactions from going to the gasket when the destination side is taken down.



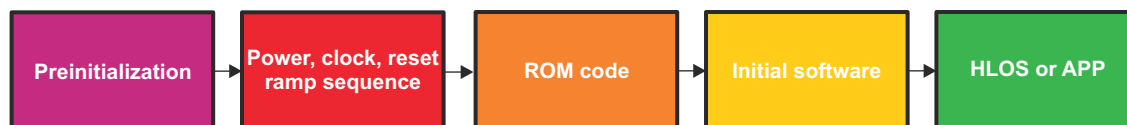
This chapter describes the steps for non-secure device initialization.

<b>4.1 Initialization Overview.....</b>	<b>270</b>
<b>4.2 Boot Process.....</b>	<b>273</b>
<b>4.3 Boot Mode Pins.....</b>	<b>279</b>
<b>4.4 Boot Parameter Tables.....</b>	<b>306</b>
<b>4.5 Boot Image Format.....</b>	<b>315</b>
<b>4.6 Boot Modes.....</b>	<b>319</b>
<b>4.7 Boot Memory Maps.....</b>	<b>325</b>

## 4.1 Initialization Overview

Figure 4-1 is an overview of the initialization process and its steps:

- **Preinitialization:** Power, clock, and control connections must be present, and the boot configuration pins must be held at the desired logical levels.
- **Power, clock, reset ramp sequence:** Specific sequence that is applied by the power-management chip(s)
- **ROM code:** Responsible for finding, for downloading, and for executing the initial software (SBL)
- **Initial software:** Software that loads, prepares, and passes control to application software or to the high-level operating system (HLOS)
- **High-Level Operating System** or bare-metal application which runs on main processor(s)



init-003

**Figure 4-1. Initialization Process**

The first two steps in the initialization process are hardware-oriented; however, they require an understanding of the process of configuring these system interface pins (balls on the device), which have software-configurable functionality. This configuration is an essential part of the chip configuration and is application-dependent. This chapter discusses these system-interface pins, the associated configuration registers, and memory structures that are vital to the correct initialization of the device.

### 4.1.1 ROM Code Overview

ROM bootloader (or ROM Code) is a software that resides in a on-chip read-only memory (ROM) to assist the customer in transferring and executing their application code. The device has two ROM codes operating in tandem – the MCU ROM code, and the DMSC ROM code.

In order to accommodate various system scenarios, the ROM Code supports several boot modes. These boot modes can be broadly classified as:

- Host boot modes
- Memory boot modes.

During a host boot, the device is configured to receive code from a host via the selected interface. Either the host writes the application code directly into internal memory or the ROM Code receives the application code on the selected interface and stores it in internal memory.

During a memory boot, the device transfers code from non-volatile memory to internal memory for execution.

In all boot modes, the entire boot operation can be partitioned into two sections:

1. Hardware initialization phase
2. Boot process.

During initialization, the ROM Code configures the device resources (PLLs, peripherals, pins) as needed to support the boot process. The resources used depend on the boot mode requirements.

During the boot process the boot image can be loaded into device memory and executed, or executed in place, depending on the boot peripheral. DMSC will perform code verification and allow, or forbid, the image execution.

Main configuration source for boot after power-up are the BOOTMODE pins sampled automatically after reset release and stored in device status registers. At ROM Code startup, these pin values are read from the registers to create the boot peripheral list and the boot configuration tables used later to initialize and startup the PLLs and boot peripherals.

The ROM Code also provides a multi-stage boot mechanism.



### 4.1.2 Bootloader Modes

Table 4-1 shows the boot modes supported by ROM code.

**Table 4-1. ROM Code Boot Modes**

Boot Mode	Boot Media/Host	Peripheral	Domain <sup>(1)</sup>	Can be a Backup Mode? <sup>(2)</sup>	Notes
No-boot/Dev-boot	No media or host	None	N/A	N	No boot or development boot – debug modes
OSPI	OSPI flash	MCU_FSS0_OSPI0	MCU	N	On OSPI port
QSPI	QSPI flash/EEPROM	MCU_FSS0_OSPI0 or MCU_FSS0_OSPI1	MCU	N	On OSPI port
SPI	SPI EEPROM	MCU_FSS0_OSPI0 or MCU_FSS0_OSPI1	MCU	Y <sup>(3)</sup>	On OSPI port
I2C	I <sup>2</sup> C EEPROM External host	MCU_I2C0	MCU	Y	I2C module is master I2C module is slave
MMCSD	MMC/SD card	MMCSD0 or MMCSD1	MAIN	Y <sup>(3)</sup>	Boot from User Data Area (UDA) or file system
eMMC	eMMC flash	MMCSD0 or MMCSD1	MAIN	N	Boot from boot partition with auto-fall back to file system
Ethernet	External host	MCU_CPSW0	MCU	Y <sup>(3)</sup>	In BOOTP mode. RGMII or RMII PHY
PCIe	External host	PCIE0 or PCIE1	MAIN	N	-
UART	External host	MCU_UART0 or WKUP_UART0	MCU	Y	XMODEM protocol
GPMC NOR	NOR flash	GPMC0	MAIN	N	-
USB	External host	USB0 or USB1	MAIN	Y <sup>(3)</sup>	USB device mode boot using DFU (device firmware upgrade)

(1) MCU peripherals are available for boot even when main domain is not powered.

(2) The peripheral can be selected also as a backup boot mode. A backup mode is tried if primary boot mode fails.

(3) When in normal boot flow (MCU Only = 0)

#### Note

Because different devices support different sets of peripherals, see the *device-specific Datasheet* to obtain the list of peripherals supported in your device.

### 4.1.3 Terminology

- **Boot Mode Pins:** Boot mode pins provide vital information to ROM code for boot. These pins must be properly set up before power ramp.
- **Bootstrap:** Initial software launched by the ROM code during the memory booting phase.
- **Boot Header:** Optional structure that precedes the initial software and allows the redefinition of the ROM code default settings.
- **Downloaded software:** Initial software downloaded into the internal static RAM (SRAM) by the ROM code during the peripheral booting phase.
- **eFuse:** A one-time programmable memory location usually set at the factory.
- **Flash loader:** Downloaded software launched by the ROM code during the preflashing stage. It also programs an image in external memories.
- **GP device:** General-purpose device (SoC) or a non-secure device.
- **Initial software:** Software executed by any of the ROM code mechanisms (memory booting or peripheral booting). Initial software is a generic term for bootstrap and downloaded software. This can be the SBL (secondary bootloader) responsible for loading an OS.
- **Memory booting:** ROM code mechanism that consists of executing initial software from external memory.

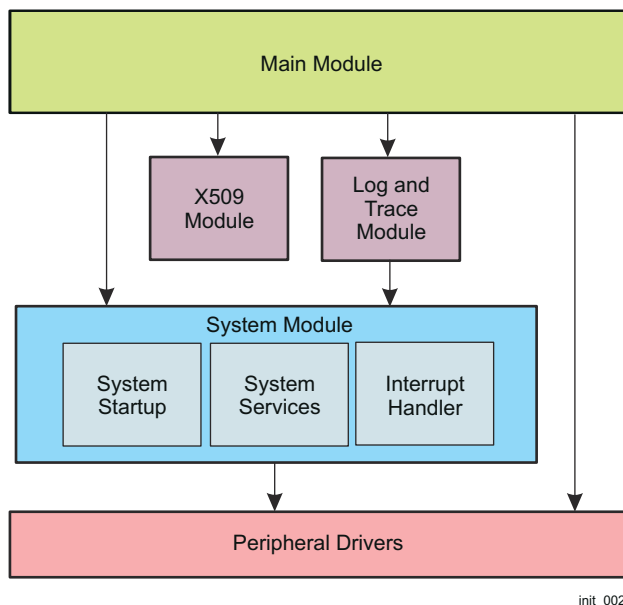
- **Master CPU:** The Arm® Cortex® CPU for which CPU-ID is 0. It configures the multicore platform and starts the ROM code to ensure device booting from a mass storage memory (memory booting) or a peripheral interface (peripheral booting).
- **Peripheral booting:** ROM code mechanism that consists of polling selected interfaces, downloading, and executing initial software (in this case, downloaded software) in the internal RAM.
- **Preflashing:** A specific case of peripheral booting where the ROM code mechanism is used to program the external flash memory.
- **ROM Code:** or ROM bootloader (RBL), the on-chip software in device ROM that executes first and implements booting.
- **ROM Code-controlled Boot Phase:** This phase covers the sequence operations from the time the platform releases the reset to the time first user- or customer-owned software starts execution. This phase is fully controlled by the device ROM code.
- **OCMC RAM memory:** On-chip RAM memory used by ROM code during boot and also for loading the booting image. ROM code is using MCU\_MSRAM0
- **Booting Parameter Table:** A logical structure stored in the on-chip RAM memory and contains information for the boot, such as the boot file name or an address to boot from.

## 4.2 Boot Process

### 4.2.1 MCU ROM Code Architecture

The MCU ROM code has the following components (see also [Figure 4-2](#)):

- Main
- Buffer manager
- X.509
- Log and Trace
- System
- Protocol
- Driver



**Figure 4-2. MCU ROM Code Architecture**

#### 4.2.1.1 Main Module

The Main module contains the top level execution loop. This loop repeats until a boot image has been received or directed to sleep by the DMSC. The main loop has three different execution sub-paths based on the boot peripheral.

- **Image Path** This path is used by the GPMC NOR, USB-DFU, PCIe, OSPI, and QSPI boot modes. In these cases the image data can be directly read by both, MCU and DMSC, in place.
- **Block Path** This path is used by the SPI, I2C, UART, eMMC, Ethernet and MMC/SD cards in raw mode. In this mode data is received from the peripheral in blocks. Blocks are accumulated in the boot buffer until a full X.509 certificate header has been received, at which point this full certificate and any subsequent blocks are passed to the DMSC as they arrive.
- **Filesystem Path** This path is used by the MMC/SD cards in filesystem mode. This mode executes exactly like in the block path, except that the boot image location is defined by a filesystem.

The main level is able to detect if a received boot image is in the correct format and reject non-conforming images.

#### 4.2.1.2 X509 Module

The X509 module parses the boot header. The boot header is an X.509 certificate as defined in [RFC5280](#). Extensions specific to boot are described in [Section 4.5.2, X.509 Certificate](#).

This module also includes an OID decoder as well as defines OID values as C #define constants for any values that can be used during boot.

#### 4.2.1.3 Buffer Manager Module

The buffer manager module is used to allocate buffers to hold boot data. The MCU code allocates a buffer when it is ready to read a block of data from the boot peripheral. Once the data is read the buffer ownership is passed to the DMSC. When the DMSC has completed processing the data in the buffer ownership is returned to the MCU, which then frees the buffer.

#### 4.2.1.4 Log and Trace Module

The Log and Trace modules are not integral to the boot process. Instead, they provide ability to record operation of the ROM code and track unexpected occurrences.

Log and trace function is currently TI internal.

#### 4.2.1.5 System Module

The system module provides services to other modules. These services are not directly related to boot drivers. The system module is the only module that operates in the supervisor mode of the MCU (a few services will run in user). The system module supports the following functions:

- Interrupt enable, disable, and service
- IPC
- Power
- Pinmux
- Clock
- Task switch

The main level is able to detect if a received boot image is in the correct format and reject non-conforming images.

#### 4.2.1.6 Protocol Module

The protocol modules provide implementation of high-level data transfer protocols. The BOOTP/TFTP and XMODEM protocols reside in this layer. These modules provide services as defined in well-known standards.

#### 4.2.1.7 Driver Module

The driver module implements the low level peripheral drivers.

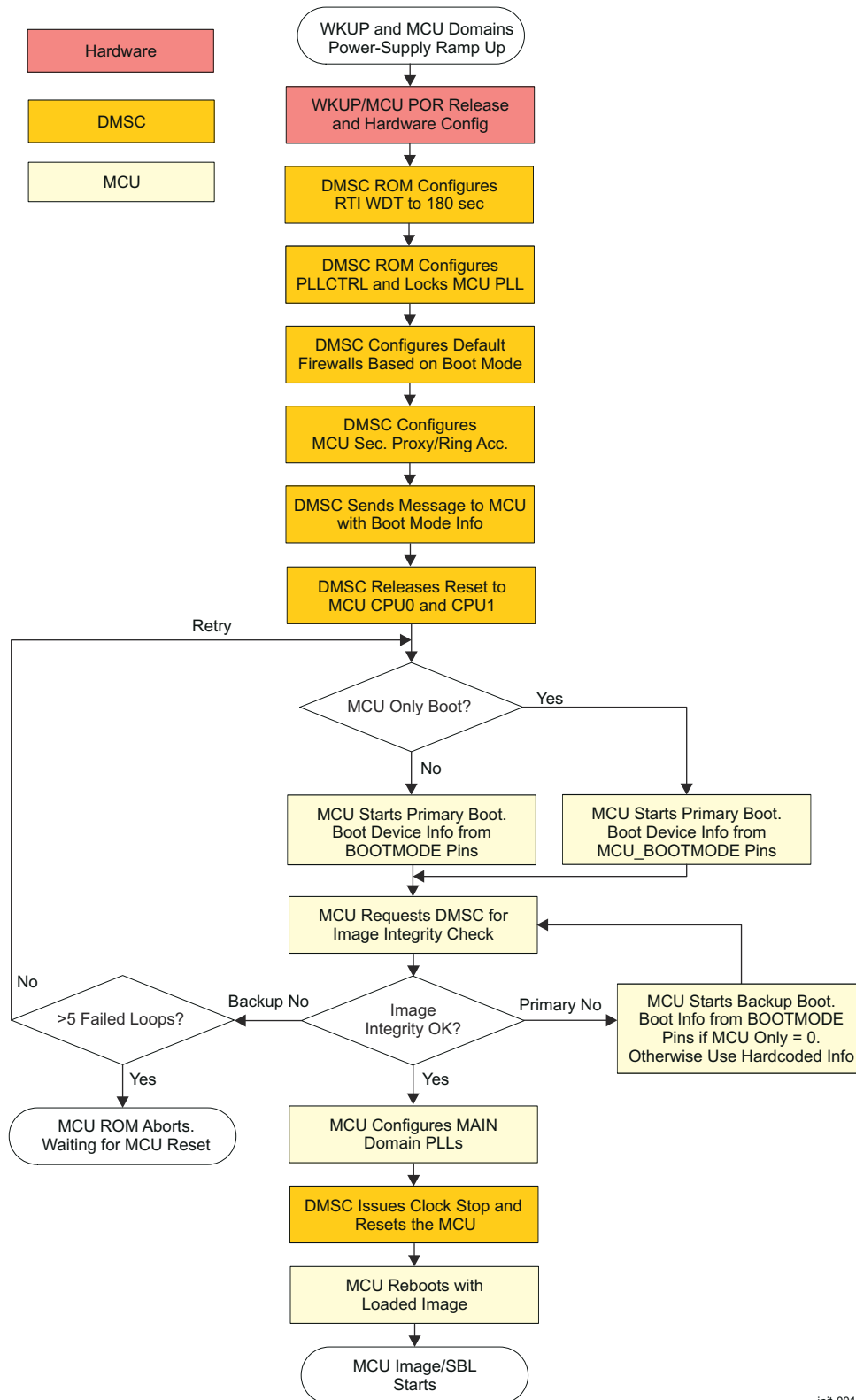
### 4.2.2 DMSC ROM Description

In a general-purpose (GP) device, DMSC ROM performs the following functions:

- Device management
- Configures the boot vectors (in BOOT\_CFG) and controls reset release of MCU core. That is, DMSC is the boot master of MCU core.
- IPC configuration via MCU\_NAVSS rings and Secure Proxy
- PLL configuration (MCU subsystem and SA2UL)
- X509 certificate parsing
- SA2UL configuration to SHA512 for image integrity checks
- Test flow support – Wait-In-Reset commands
- DMSC firmware loading

### 4.2.3 Boot Process Flow

The MCU boot process flow is shown in [Figure 4-3](#).



init-001

**Figure 4-3. Boot Process**

The values of (MCU\_)BOOTMODE pins are latched into the Device Status registers (CTRLMMR\_MAIN\_DEVSTAT and CTRLMMR\_WKUP\_DEVSTAT) by hardware as the device comes out of

global cold reset. When MCU Only boot was selected, only MCU\_BOOTMODE pins are taken into account and only MCUSS set of peripherals are available for boot. For more information how to set BOOTMODE and MCU\_BOOTMODE pins, see [Section 4.3, Boot Mode Pins](#).

The DMSC is the boot master of MCU. DMSC performs the necessary configurations and releases MCU's reset for CPU0 and CPU1 simultaneously even when in split mode.

---

#### Note

DMSC releases MCU's reset for CPU0 and CPU1 simultaneously regardless if MCU CPUs are in lockstep or split mode.

---

The MCU checks the boot mode pins and then configures the appropriate peripheral interface to get access to a boot image. A cursory check of the image is made, and the image is passed to DMSC. DMSC ROM then will perform code verification and route the boot image to the on-chip RAM. Once the image has been received, MCU enters a clean state and idles. DMSC ROM code will assert reset to the MCU, redirect the boot vector to the newly loaded image, and release the reset. This restarts the MCU with the MCU ROM code fully disconnected.

The MCU ROM code executes only on initial power up (POR). On subsequent (warm) resets, the reset vector base address will point to run-time loaded code (the SBL), and not ROM.

---

#### Note

DMSC ROM sets up a 3-minute watchdog timer (MCU\_RTI0) timeout. During this time, the MCU boot needs to get completed, otherwise a WDT reset will occur. Once the MCU image is loaded (SBL/SPL), DMSC ROM will restart the watchdog timer for additional 3 minutes upon entering the MCU SBL. The customer-provided MCU image needs to load and install the TI-provided SYSFW image into the DMSC, which will manage the watchdog timer during run time.

---



---

#### Note

The following system conditions must be met at POR to perform device boot:

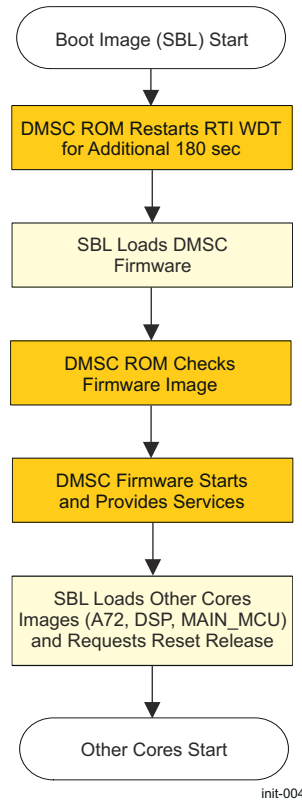
- USB cable plug must be inserted
- Ethernet PHY is powered up and out of reset
- The SD card cage must be powered before entering the SD card boot mode. A SD/MMC card with pre-loaded image must be inserted
- Memory devices must be up and ready (power ramped up and reset completed) at device startup:
  - eMMC
  - OSPI/QSPI flash
  - SPI or I<sup>2</sup>C EEPROM
  - NOR flash

Failing to meet these requirements may result in boot fail and performing a backup boot (if available for that mode).

---

[Figure 4-4](#) describes the external bootloader (SBL) typical tasks.





**Figure 4-4. External Bootloader Tasks**

Upon MCU reset and SBL execution start, DMSC ROM restarts the RTI watchdog timer for additional 180 seconds of timeout. During that time, SBL must load the DMSC firmware provided by TI otherwise a MCU reset will occur as a preventive measure against software misbehavior.

One of the SBL's main tasks is to load the DMSC firmware. Only after this task is performed, SBL can load the other processor (A72s, R5s, or optional DSPs) image and request a reset release from DMSC firmware for those cores.

#### 4.2.4 MCU Only vs Normal Boot

Users can select a special (MCU only) boot flow, which can boot the MCU, even if the main domain is unpowered. The MCU-only boot differences versus the normal boot are described below:

**Normal Boot** — Requires minimum of:

- VDDS\*\_MCU
- VDD\_MCU
- and VDD\_CORE, VDD\_CPU

voltage rails ON. Other voltage rails are optional.

Power domains for WKUP, MCUSS and Core domain are ON. Power domains required for the specified boot mode may be turned-on by the boot ROM code, if they are OFF by default.

**MCU-Only Boot** — Requires minimum of:

- VDDS\*\_MCU
- VDD\_MCU

voltage rails ON. Other voltage rails are optional. Power domains for WKUP and MCUSS are ON (only R5 is switchable). Core domain is unknown (may be powered or unpowered). Only limited set of boot modes involving

MCUSS peripherals are supported. Power domains required for the specified boot mode may be turned-on by the boot ROM code, if they are OFF by default

### 4.3 Boot Mode Pins

Boot Mode pins provide means to select the boot mode and options before the device is powered up. After every POR, they are the main source to populate the Boot Parameter Tables. See [Section 4.4, Boot Parameter Tables](#) for table list and description.

Boot mode pins can be divided into the following categories:

- **MCU\_BOOTMODE[02:00]** – Denote system clock frequency (WKUP\_HFOSC0) to ROM code for PLL configuration.
- **MCU\_BOOTMODE[05:03]** – Select the requested boot (primary) mode after POR, that is, the peripheral/memory to boot from. These are the only pins which direct boot on MCU Only boot mode
- **MCU\_BOOTMODE06** – MCU Only boot mode. In MCU Only boot, ROM code skips BOOTMODE pins and attempts boot only from MCU peripherals
- **MCU\_BOOTMODE[09:08]** – These pins can select tests to be performed on power-up (POST). POST runs in hardware, before the ROM code starts
- **BOOTMODE0** – This pin allows for additional primary boot modes to be selected when in non-MCU Only (normal) mode
- **BOOTMODE[3:1]** – Select the backup boot mode, that is, the peripheral/memory to boot from, if primary boot device failed.
- **BOOTMODE[6:4]** – These pins provide optional configurations for primary boot and are used in conjunction with the boot mode selected. See [Section 4.3.2.3](#) and the corresponding boot mode section.
- **BOOTMODE7** – This pin provides optional configurations for the backup boot devices. See [Section 4.3.2.4](#) and the corresponding boot mode section.

---

#### Note

It is user's responsibility to set the boot mode pins (via pullups or pulldowns, and jumpers/switches) depending on the desired boot scenario.

System board must provide means to switch bootmode pin settings easily. In a field design, some pins may be hardcoded to 0 or 1 but only after careful evaluation (for example, if using MCU Only pin = 1).

---

#### Note

CTRLMMR\_MAIN\_DEVSTAT register reflects the BOOTMODE pin values sampled at internal POR release.

CTRLMMR\_WKUP\_DEVSTAT register reflects the MCU\_BOOTMODE pin values sampled at internal POR release.

---

### 4.3.1 MCU\_BOOTMODE Pin Mapping

The MCU\_BOOTMODE pins are the only pins sampled in MCU-only boot mode. MCU\_BOOTMODE pin mapping is shown in [Table 4-2](#).

**Table 4-2. MCU\_BOOTMODE Pin Mapping**

9	8	7	6	5	4	3	2	1	0
POST Config		Reserved <sup>(1)</sup>	MCU Only	Primary Boot Mode A			PLL Config		

(1) Set to 0 (via pulldown to vss)

[Table 4-3](#) describes the MCU\_BOOTMODE pins that need to be set according to the system clock provided to the device.

The ROM Code will configure any PLLs required during the boot process. The ROM Code does not have the ability to select HFOSC1 (in main domain) during initial boot, however the selection can be done through the boot certificate (see [Section 4.5](#), *Boot Image Format*).

**Table 4-3. PLL Reference Clock Selection**

PLL Config Pins			Ref Clock (MHz)
MCU 2	MCU 1	MCU 0	
0	0	0	19.2
0	0	1	20
0	1	0	24
0	1	1	25
1	0	0	26
1	0	1	27 <sup>(2)</sup>
1	1	0	Reserved
1	1	1	No PLL setup <sup>(1)</sup>

(1) Only no-boot, I2C, and SPI modes are expected to function with PLL config = 7 (No PLL setup)

(2) USB DFU boot mode is not supported with the reference clock of 27 MHz

If MCU Only mode (boot from MCU domain only) needs to be selected, the MCU Only pin must be set to 1 (see [Table 4-4](#)).

**Table 4-4. MCU Only Configuration**

MCU Only Pin	MCU Only Selection
MCU 6	
0	Normal boot mode. Both MCU_BOOTMODE and BOOTMODE pins are read by ROM code
1	MCU Only boot mode. Only MCU_BOOTMODE pins are read by ROM code. Boot can proceed only from MCU peripherals

[Table 4-5](#) lists the boot modes (both primary and backup), available when MCU Only boot was selected. For normal boot modes, refer to [Table 4-8](#) and [Table 4-9](#)

**Table 4-5. Boot Mode Selection When MCU Only = 1**

Primary Boot Mode A Pins			Primary Boot Mode Selected	Backup Boot Mode Assigned <sup>(1)</sup>
MCU 5	MCU 4	MCU 3		
0	0	1	OSPI	UART
0	1	0	QSPI	UART
0	1	1	SPI	UART
1	0	0	Ethernet RGMII	I2C
1	0	1	Ethernet RMII	I2C
1	1	0	I2C	UART

**Table 4-5. Boot Mode Selection When MCU Only = 1 (continued)**

Primary Boot Mode A Pins			Primary Boot Mode Selected	Backup Boot Mode Assigned <sup>(1)</sup>
MCU 5	MCU 4	MCU 3		
1	1	1	UART	I2C

- (1) When MCU Only = 1, backup mode is assigned per primary mode. In general, non-flash primary boot modes are backed up with flash modes, and flash primary modes use non-flash for backup.

Power-On Self-Test sequence is executed as directed by pins shown in [Table 4-6](#). Dedicated eFuse bits may override the POST pins. See register CTRLMMR\_WKUP\_POST\_OPT in section *WKUP\_CTRL\_MMR0 Registers* and POST section in the Safety Manual for details.

**Table 4-6. POST Selection**

POST Config Pins		POST Sequence
MCU 9	MCU 8	
0	0	DMSC LBIST followed by MCU LBIST followed by PBIST <sup>(2)</sup>
0	1	DMSC LBIST and MCU LBIST in parallel followed by PBIST <sup>(2)</sup>
1	0	Reserved <sup>(2)</sup>
1	1	POST bypass <sup>(1)</sup>

- (1) eFuse may has been programmed so that these pins are overridden and a POST sequence is always executed (no bypass possible).  
(2) This sequence may has been replaced by another via eFuse settings. See register CTRLMMR\_WKUP\_POST\_OPT in section *WKUP\_CTRL\_MMR0 Registers* and POST section in SW Safety Manual, for details.

### 4.3.2 BOOTMODE Pin Mapping

In normal boot operation (MCU Only = 0), the ROM execution is directed also through the main boot mode pins. This provides more flexibility and more booting peripherals to boot from. The Main domain must be powered and functional when MCU Only = 0.

Main boot mode pins are shown in [Table 4-7](#).

**Table 4-7. BOOTMODE Pin Mapping**

7	6	5	4	3	2	1	0
Backup Boot Mode Config	Primary Boot Mode Config			Backup Boot Mode			Primary Boot Mode B

#### 4.3.2.1 Primary Boot Mode Selection

The primary boot mode is the first mode attempted after reset. [Table 4-8](#) lists all possible primary boot modes. Boot modes with Boot Mode B = 1 are available during normal boot only.

**Table 4-8. Primary Boot Mode Selection When MCU Only = 0**

Primary Boot Mode B Pin	Primary Boot Mode A Pins			Boot Mode Selected
0	MCU 5	MCU 4	MCU 3	
0	0	0	1	OSPI
0	0	1	0	QSPI
0	0	1	1	SPI
0	1	0	0	Ethernet RGMII
0	1	0	1	Ethernet RMII
0	1	1	0	I2C
0	1	1	1	UART
1	0	0	0	MMC/SD card
1	0	0	1	eMMC
1	0	1	0	USB
1	1	0	0	GPIC NOR
1	1	0	1	PCIe
1	1	1	0	Reserved
1	1	1	1	No-boot/Dev boot

#### 4.3.2.2 Backup Boot Mode Selection When MCU Only = 0

With MCU Only = 0, the backup boot mode is selected via pins within the main BOOTMODE map. [Table 4-9](#) lists all possible backup boot modes when MCU Only = 0. For backup boot modes when MCU Only = 1, please refer to [Table 4-5](#).

**Table 4-9. Backup Mode Selection (MCU Only=0)**

Backup Boot Mode Pins			Backup Boot Mode Selected
3	2	1	
0	0	0	None (no backup boot will be attempted)
0	0	1	USB – Device (DFU)
0	1	0	Reserved
0	1	1	UART
1	0	0	Ethernet
1	0	1	MMC/SD card
1	1	0	SPI



**Table 4-9. Backup Mode Selection (MCU Only=0) (continued)**

Backup Boot Mode Pins			Backup Boot Mode Selected
3	2	1	
1	1	1	I2C

#### 4.3.2.3 Primary Boot Mode Configuration

When in normal boot flow (MCU Only = 0), it is possible to modify certain peripheral settings, such as chip-select, bus speed, and others, depending on the peripheral selected.

The mapping of configuration pins per boot mode is shown in [Table 4-10](#). See the respective sections for details.

**Table 4-10. Primary Boot Mode Configuration**

Primary Boot Mode Config Pins			Primary Boot Mode B Pin	Primary Boot Mode A Pins			Primary Boot Mode
6	5	4	0	MCU 5	MCU 4	MCU 3	
Speed	Iclk	Csel	0	0	0	1	OSPI
Port	Iclk	Csel	0	0	1	0	QSPI
Port	Mode	Csel	0	0	1	1	SPI
Clkout	Delay	Link stat	0	1	0	0	Ethernet RGMII
Clkout	Clk src		0	1	0	1	Ethernet RMII
Bus reset	Mode	Addr	0	1	1	0	I2C
		Port	0	1	1	1	UART or No boot
Port	Bus width	Fs/raw	1	0	0	0	MMC/SD card
Port	Bus width	Voltage	1	0	0	1	eMMC
Port	Mode	Lane Swap	1	0	1	0	USB
AD Mux		Csel	1	1	0	0	GPMP NOR
Port	N lanes	sref	1	1	0	1	PCIe
			1	1	1	0	Reserved (PG1.0 only)
sfdp (PG1.1 only)	pincmd	mode	1	1	1	0	xSPI
Split	Arm/Thumb	No/Dev	1	1	1	1	No-boot/Dev boot

#### 4.3.2.4 Backup Boot Mode Configuration

When in normal boot flow (MCU Only = 0), it is possible to modify certain peripheral settings, such as chip-select, bus speed, and others depending on the peripheral selected.

The mapping of the backup configuration pin per boot mode selected is shown in [Table 4-11](#). See the respective sections for details.

**Table 4-11. Backup Boot Mode Configuration**

Backup Boot Mode Config Pin	Backup Boot Mode Pins			Backup Boot Mode
7	3	2	1	
	0	0	0	None (no backup boot will be attempted)
Port	0	0	1	USB – Device (DFU)
Port	0	1	0	Reserved
Port	0	1	1	UART
I/F	1	0	0	Ethernet
Port	1	0	1	MMC/SD card
Port	1	1	0	SPI
Nide	1	1	1	I2C

### 4.3.3 No-boot/Dev-boot Configuration

[Table 4-12](#) shows configuration pins assignment to functions when No-boot/Dev-boot was selected. In this mode, ROM code is bypassed and user is allowed to run his own code for development or debug.

MCU-only mode is not compatible with no-boot/dev-boot since the DebugSS is in the main domain.

**Table 4-12. No-boot/Dev-boot Configuration Fields**

BOOTMODE Pins	Field	Value	Description	MCU Only=1 Value
6	Split	0	Split/Lockstep configuration of MCU R5 cores derived from eFuse	N/A
		1	MCU R5 cores forced to split mode	
5	Arm/Thumb	0	ARM mode reset vectors	N/A
		1	Thumb mode reset vectors	
4	No/Dev	0	Development Boot	N/A
		1	No boot	

During the *Development boot* ( $BOOTMODE[4] = 0$ ), the DMSC ROM code will act as if boot of the primary image has completed, and the DMSC ROM will be waiting for a *Firmware load* message from MCU R5. Thus user can load a standard u-boot/SPL image to the R5 RAM. U-boot/SPL will then load the DMSC firmware and complete the full boot.

In *No-boot* ( $BOOTMODE[4] = 1$ ), both the DMSC and MCU R5 ROMs are bypassed and both CPUs are held in a dummy branch-to-self loop. No-boot is the most minimal device touch state by the ROM - only minimal hardware configurations are done and none of the PLLs is locked/configured. No-boot is suitable if user wants to load his own PLL, Pad config, and other basic settings.

#### 4.3.4 OSPI Boot Device Configuration

Octal SPI flash memories support various protocols, and OSPI boot mode of the device only supports a specific protocol defined below. *Additionally, if the flash memory is complaint with JEDEC xSPI standards JESD251 and JESD216D, refer to xSPI boot mode described in Section 4.3.5 (applicable only in PG1.1).* The OSPI protocol is described according to bit-width (1 or 8) and data rate (S or D for \*S\*ingle Data rate or \*D\*ouble Data rate) for the Command/Address/Data segments of the protocol. The OSPI boot mode supports 1S-1S-8S mode. The Command and Address issued are 8 bits and 24 bits, respectively. The Read Command issued for OSPI mode is 0x8b, followed by zero for address and 8 dummy cycles. The frequencies of operation is 33 MHz. The OSPI boot mode pin configuration and corresponding pin usage and mux configuration are shown below. This is the OSPI boot mode.

Table 4-13 shows configuration pins assignment to functions when boot mode is the Octal SPI.

**Table 4-13. OSPI Configuration Fields**

BOOTMODE Pins	Field	Value	Description	MCU Only=1 Value
6	Speed	0	33 MHz using manual tap selection	0
		1	Reserved	
5	Iclk	0	Iclock source external	0
		1	Iclock source internal	
4	Csel	0	CS 0	0
		1	CS 1	

Table 4-14 summarizes the OSPI pin configuration done by ROM code for OSPI boot device.

**Table 4-14. OSPI Pin Usage**

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Tx En/Dis	Pinmux Sel
MCU_OSPI0_CLK0	MCU_OSPI0_CLK	Disable	Up	0	Disable	Enable	0
MCU_OSPI0_LBCLK0	MCU_OSPI0_LBCLK0	Disable	Up	0	Enable	Enable	0
MCU_OSPI0_DQS	MCU_OSPI0_DQS	Disable	Up	0	Enable	Disable	0
MCU_OSPI0_D0	MCU_OSPI0_D0	Enable	Up	0	Enable	Enable	0
MCU_OSPI0_D1	MCU_OSPI0_D1	Enable	Up	0	Enable	Enable	0
MCU_OSPI0_D2	MCU_OSPI0_D2	Enable	Up	0	Enable	Enable	0
MCU_OSPI0_D3	MCU_OSPI0_D3	Enable	Up	0	Enable	Enable	0
MCU_OSPI0_D4	MCU_OSPI0_D4	Enable	Up	0	Enable	Enable	0
MCU_OSPI0_D5	MCU_OSPI0_D5	Enable	Up	0	Enable	Enable	0
MCU_OSPI0_D6	MCU_OSPI0_D6	Enable	Up	0	Enable	Enable	0
MCU_OSPI0_D7	MCU_OSPI0_D7	Enable	Up	0	Enable	Enable	0
MCU_OSPI0_CSn0	MCU_OSPI0_CSn0	Enable	Up	0	Disable	Enable	0
MCU_OSPI0_CSn1	MCU_OSPI0_CSn1	Enable	Up	0	Disable	Enable	0
MCU_OSPI1_LBCLK0	MCU_OSPI0_CSn2	Enable	Up	0	Disable	Enable	1
MCU_OSPI1_DQS	MCU_OSPI0_CSn3	Enable	Up	0	Disable	Enable	1

#### 4.3.5 xSPI Boot Device Configuration

The xSPI protocol defines 1S-1S-1S mode for general backwards compatibility, and 8D-8D-8D for maximum throughput (where bit-width (1 or 8) and data rate (S or D for \*S\*ingle Data rate or \*D\*ouble Data rate)).

For 1S-1S-1S mode of operation (Bit-width =1, Single Data Rate): the Command and Address issued are 8 bits and 24 bits, respectively. The Read Command issued is 0x0b, followed by zero for address and 8 dummy cycles. The frequency of operation supported is 50 MHz.

For 8D-8D-8D mode of operation (Bit-width =8, Double Data Rate): the Command and Address issued are 8 bits and 32 bits, respectively. The Read Command issued is 0x0b or 0xee, followed by zero for address, 16 or 20 dummy cycles. The frequency of operation supported is 25 MHz. Additionally, the flash is expected to be configured in 8D mode out of POR through the nonvolatile configuration register.

For SFDP mode, ROM starts operation in 1S-1S-1S mode, then reads the SFDP header from flash memory to get the 8D-8D-8D switching sequence, Read Command, CMD Extension, and Byte Order. SFDP parsing of ROM is described below and on successful parsing ROM, issues the 8D-8D-8D command switching sequence. It then reads the boot image in 8D-8D-8D mode with the read command specified in the SFDP header.

The following boot mode pin configuration and corresponding pin usage and mux configuration are shown below. This is the xSPI boot mode.

**Table 4-15. MCU\_BOOTMODE Pin Map OSPI**

9	8	7	6	5	4	3	2	1	0
Rsvd (not for boot use)	Rsvd	MCU Only	Primary Boot Mode A				PLL Config		
X	X	X	X	1	1	0	X	X	X

**Table 4-16. BOOTMODE Pin Map OSPI**

7	6	5	4	3	2	1	0
Backup Boot Mode Config	Primary Boot Mode Config			Backup Boot Mode			Primary Boot B
X	0/1	0/1	0/1	X	X	X	1

**Table 4-17. xSPI Boot Configuration Fields**

BOOTMODE Pins	Field	Value	Description	MCU Only=1 Value
6	SFDP	0	SFDP (Serial Flash Discovery Parameter) Disabled	0
		1	SFDP Enabled	
5	Pin Cmd	0	0x0B Read Command	0
		1	0xEE Read Command	
4	Mode	0	SPI-STR (1S-1S-1S) at 50 MHz	0
		1	OCTAL-DTR (8D-8D-8D) at 25 MHz	

**Table 4-18. xSPI Pin Usage**

Package Name	Function Name	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Tx En/Dis	Pinmux Sel	Pad configuration Register
MCU_OSPI0_CLK0	MCU_OSPI0_CLK	Disable	Up	0	Disable	Enable	0	WKUP_PADCONFIG_0
MCU_OSPI0_LBCLK0	MCU_OSPI0_LBCLK0	Disable	Up	0	Enable	Enable	0	WKUP_PADCONFIG_1
MCU_OSPI0_DQS	MCU_OSPI0_DQS	Disable	Up	0	Enable	Disable	0	WKUP_PADCONFIG_2
MCU_OSPI0_D0	MCU_OSPI0_D0	Enable	Up	0	Enable	Enable	0	WKUP_PADCONFIG_3
MCU_OSPI0_D1	MCU_OSPI0_D1	Enable	Up	0	Enable	Enable	0	WKUP_PADCONFIG_4
MCU_OSPI0_D2	MCU_OSPI0_D2	Enable	Up	0	Enable	Enable	0	WKUP_PADCONFIG_5
MCU_OSPI0_D3	MCU_OSPI0_D3	Enable	Up	0	Enable	Enable	0	WKUP_PADCONFIG_6
MCU_OSPI0_D4	MCU_OSPI0_D4	Enable	Up	0	Enable	Enable	0	WKUP_PADCONFIG_7

**Table 4-18. xSPI Pin Usage (continued)**

Package Name	Function Name	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Tx En/Dis	Pinmux Sel	Pad configuration Register
MCU_OSPI0_D5	MCU_OSPI0_D5	Enable	Up	0	Enable	Enable	0	WKUP_PADCONFIG_8
MCU_OSPI0_D6	MCU_OSPI0_D6	Enable	Up	0	Enable	Enable	0	WKUP_PADCONFIG_9
MCU_OSPI0_D7	MCU_OSPI0_D7	Enable	Up	0	Enable	Enable	0	WKUP_PADCONFIG_10
MCU_OSPI0_CS0	MCU_OSPI0_CS0	Enable	Up	0	Disable	Enable	0	WKUP_PADCONFIG_11
MCU_OSPI0_CS1	MCU_OSPI0_CS1	Enable	Up	0	Disable	Enable	0	WKUP_PADCONFIG_12

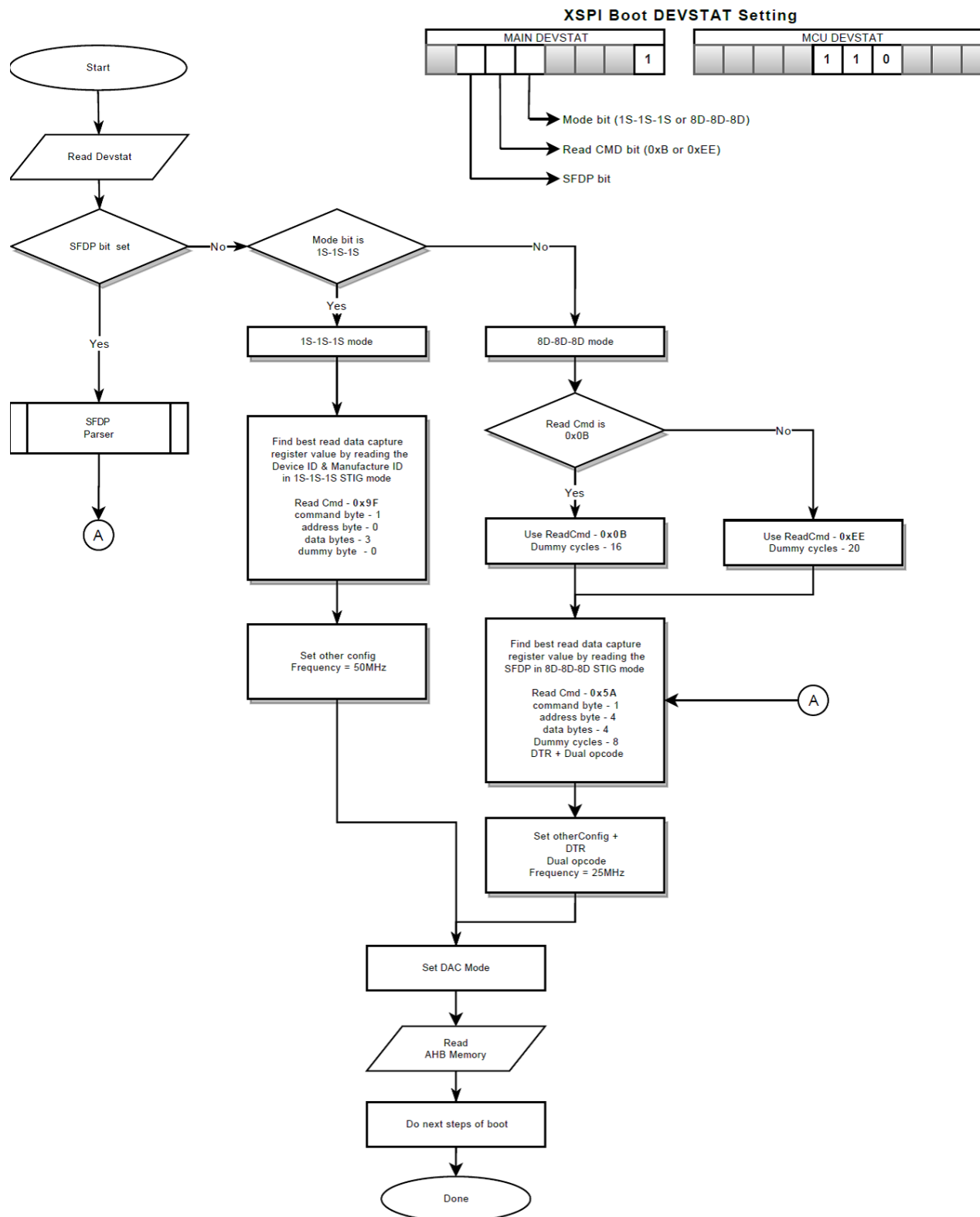
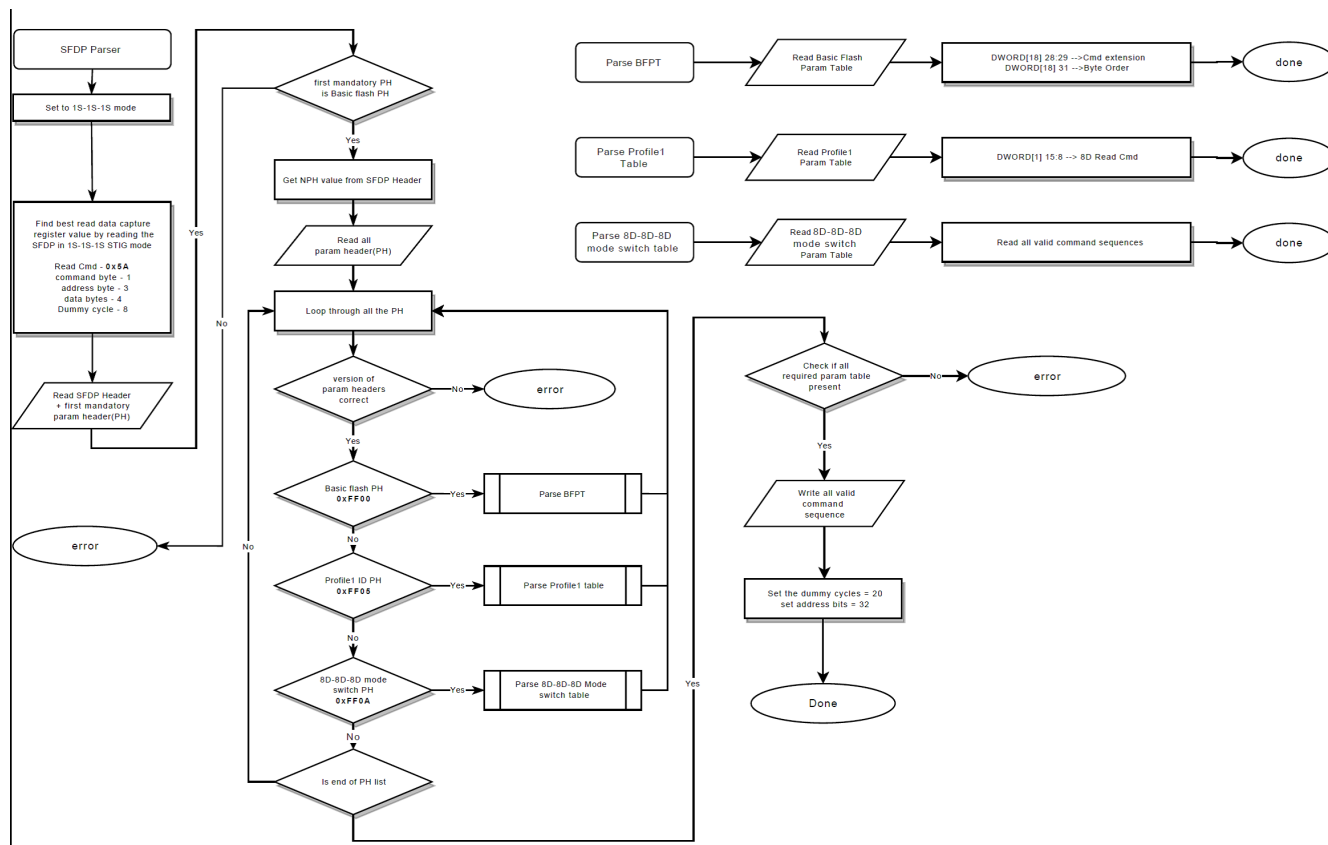


Figure 4-5. xSPI Flow Chart


**Figure 4-6. ROM SFDP Parser Flow**



### 4.3.6 QSPI Boot Device Configuration

Table 4-19 shows configuration pins assignment to functions when boot mode is the QSPI on OSPI mode.

**Table 4-19. QSPI Boot Configuration Fields**

BOOTMODE Pins	Field	Value	Description	MCU Only=1 Value
6	Port	0	Port 1	0
		1	Port 0	
5	Iclk	0	Iclock source external	0
		1	Iclock source internal	
4	Csel	0	CS 0	0
		1	CS 1	

Table 4-20 summarizes the OSPI pin configuration done by ROM code for QSPI boot device on port 0.

**Table 4-20. QSPI Port 0 Pin Usage**

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Tx En/Dis	Pinmux Sel
MCU_OSPI0_CLK0	MCU_OSPI0_CLK	Disable	Up	0	Disable	Enable	0
MCU_OSPI0_LBCLK0	MCU_OSPI0_LBCLK0	Disable	Up	0	Enable	Enable	0
MCU_OSPI0_DQS	MCU_OSPI0_DQS	Disable	Up	0	Enable	Disable	0
MCU_OSPI0_D0	MCU_OSPI0_D0	Enable	Up	0	Enable	Enable	0
MCU_OSPI0_D1	MCU_OSPI0_D1	Enable	Up	0	Enable	Enable	0
MCU_OSPI0_D2	MCU_OSPI0_D2	Enable	Up	0	Enable	Enable	0
MCU_OSPI0_D3	MCU_OSPI0_D3	Enable	Up	0	Enable	Enable	0
MCU_OSPI0_CSn0	MCU_OSPI0_CSn0	Enable	Up	0	Disable	Enable	0
MCU_OSPI0_CSn1	MCU_OSPI0_CSn1	Enable	Up	0	Disable	Enable	0
MCU_OSPI1_LBCLK0	MCU_OSPI0_CSn2	Enable	Up	0	Disable	Enable	1
MCU_OSPI1_DQS	MCU_OSPI0_CSn3	Enable	Up	0	Disable	Enable	1

Table 4-21 summarizes the OSPI pin configuration done by ROM code for QSPI boot device on port 1.

**Table 4-21. QSPI Port 1 Pin Usage**

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Tx En/Dis	Pinmux Sel
MCU_OSPI1_CLK0	MCU_OSPI1_CLK	Disable	Up	0	Disable	Enable	0
MCU_OSPI1_LBCLK0	MCU_OSPI1_LBCLK0	Disable	Up	0	Enable	Enable	0
MCU_OSPI1_D0	MCU_OSPI1_D0	Enable	Up	0	Enable	Enable	0
MCU_OSPI1_D1	MCU_OSPI1_D1	Enable	Up	0	Enable	Enable	0
MCU_OSPI1_D2	MCU_OSPI1_D2	Enable	Up	0	Enable	Enable	0
MCU_OSPI1_D3	MCU_OSPI1_D3	Enable	Up	0	Enable	Enable	0
MCU_OSPI1_CSn0	MCU_OSPI1_CSn0	Enable	Up	0	Disable	Enable	0
MCU_OSPI1_CSn1	MCU_OSPI1_CSn1	Enable	Up	0	Disable	Enable	0

### 4.3.7 SPI Boot Device Configuration

Table 4-22 shows configuration pins assignment to functions when boot mode is the SPI on OSPI port mode. The SPI bus will be run at 4.156 MHz.

**Table 4-22. SPI Boot Configuration Fields**

BOOTMODE Pins	Field	Value	Description	MCU Only=1 Value
6 (7) <sup>(1)</sup>	Port	0	Port 0	0
		1	Port 1	
5	Mode	0	SPI Mode 0	0
		1	SPI Mode 3	
4	Csel	0	CS 0	0
		1	CS 1	

(1) When SPI is the backup mode and MCU Only=0

Table 4-23 summarizes the OSPI pin configuration done by ROM code for SPI boot device on port 0.

**Table 4-23. SPI Port 0 Pin Usage**

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Tx En/Dis	Pinmux Sel
MCU_OSPI0_CLK0	MCU_OSPI0_CLK	Disable	Up	0	Disable	Enable	0
MCU_OSPI0_LBCLK0	MCU_OSPI0_LBCLK0	Disable	Up	0	Enable	Enable	0
MCU_OSPI0_DQS	MCU_OSPI0_DQS	Disable	Up	0	Enable	Disable	0
MCU_OSPI0_D0	MCU_OSPI0_D0	Enable	Up	0	Enable	Enable	0
MCU_OSPI0_D1	MCU_OSPI0_D1	Enable	Up	0	Enable	Enable	0
MCU_OSPI0_CSn0	MCU_OSPI0_CSn0	Enable	Up	0	Disable	Enable	0
MCU_OSPI0_CSn1	MCU_OSPI0_CSn1	Enable	Up	0	Disable	Enable	0
MCU_OSPI1_LBCLK0	MCU_OSPI0_CSn2	Enable	Up	0	Disable	Enable	1
MCU_OSPI1_DQS	MCU_OSPI0_CSn3	Enable	Up	0	Disable	Enable	1

Table 4-24 summarizes the OSPI pin configuration done by ROM code for SPI boot device on port 1.

**Table 4-24. SPI Port 1 Pin Usage**

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Tx En/Dis	Pinmux Sel
MCU_OSPI1_CLK0	MCU_OSPI1_CLK	Disable	Up	0	Disable	Enable	0
MCU_OSPI1_LBCLK0	MCU_OSPI1_LBCLK0	Disable	Up	0	Enable	Enable	0
MCU_OSPI1_D0	MCU_OSPI1_D0	Enable	Up	0	Enable	Enable	0
MCU_OSPI1_D1	MCU_OSPI1_D1	Enable	Up	0	Enable	Enable	0
MCU_OSPI1_CSn0	MCU_OSPI1_CSn0	Enable	Up	0	Disable	Enable	0
MCU_OSPI1_CSn1	MCU_OSPI1_CSn1	Enable	Up	0	Disable	Enable	0

### 4.3.8 I2C Boot Device Configuration

Table 4-25 shows configuration pins assignment to functions when boot mode is the I2C mode.

**Table 4-25. I2C Boot Configuration Fields**

BOOTMODE Pins	Field	Value	Description	MCU Only=1 Value
6	Bus reset	0	Hung bus reset attempt after 1 ms	0
		1	No hung bus reset attempted	
5 (7) <sup>(1)</sup>	Mode	0	SoC I2C module is master (boot from flash/EEPROM)	0
		1	SoC I2C module is slave	
4	Address	0	0x50 (master mode, EEPROM's address), 0x10 (slave mode, own address)	0
		1	0x51 (master mode, EEPROM's address), 0x11 (slave mode, own address)	

(1) When I2C is the backup mode and MCU Only=0

The I<sup>2</sup>C bus is considered inactive if the data line is low and clock remains high for the specified timeout time. Recovery consists of driving the clock a stop condition is detected. A stop condition is a transition on the data line from 0 to 1 while the clock line is high. If the clock line is stuck low there is no way to take control of the bus.

Table 4-26 summarizes the I2C pin configuration done by ROM code for I<sup>2</sup>C boot device.

**Table 4-26. I2C Pin Usage**

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Tx En/Dis	Pinmux Sel
MCU_I2C0_SCL	MCU_I2C0_SCL	Enable	Up	0	Enable	Enable	0
MCU_I2C0_SDA	MCU_I2C0_SDA	Enable	Up	0	Enable	Enable	0

### 4.3.9 MMC/SD Card Boot Device Configuration

Table 4-27 shows configuration pins assignment to functions when boot mode is the MMC/SD card mode.

**Table 4-27. MMC/SD Card Boot Configuration Fields**

BOOTMODE Pins	Field	Value	Description	MCU Only=1 Value
6 (7) <sup>(1)</sup>	Port	0	Port 0	N/A
		1	Port 1	
5	Bus Width	0	4 bit	N/A
		1	1 bit	
4	FS/Raw	0	Filesystem mode	N/A
		1	Raw Mode	

(1) When MMCSD is the backup mode

Table 4-28 summarizes the MMC pin configuration done by ROM code for MMC/SD Card boot device on port 1.

#### Note

Note that MMC Port 0 has no pin mux.

**Table 4-28. MMC/SD Card Port 1 Pin Usage**

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Tx En/Dis	Pinmux Sel
MMC1_DAT3 <sup>(1)</sup>	MMC1_DAT3	Enable	Up	0	Enable	Enable	0
MMC1_DAT2 <sup>(1)</sup>	MMC1_DAT2	Enable	Up	0	Enable	Enable	0
MMC1_DAT1 <sup>(1)</sup>	MMC1_DAT1	Enable	Up	0	Enable	Enable	0
MMC1_DAT0	MMC1_DAT0	Enable	Up	0	Enable	Enable	0
MMC1_CLK	MMC1_CLK	Disable	Up	0	Disable	Enable	0
MMC1_CMD	MMC1_CMD	Enable	Up	0	Enable	Enable	0
MMC1_SDCD	MMC1_SDCD	Enable	Up	0	Enable	Disable	0

(1) In 1-bit mode, only MMC1\_DAT0 is configured.

### 4.3.10 Ethernet Boot Device Configuration

Table 4-29 shows configuration pins assignment to functions when boot mode is the Ethernet RGMII mode.

**Table 4-29. Ethernet RGMII Boot Configuration Fields**

BOOTMODE Pins	Field	Value	Description	MCU Only=1 Value
6	Clkout	0	25 MHz clock not generated on MCU CLKOUT	0
		1	25 MHz clock generated on MCU CLKOUT	
5	Delay	0	RGMII with internal Tx delay	0
		1	RGMII with external Tx delay	
4	Link stat	0	PHY scan used for speed/duplex setup	0
		1	RGMII status register used for speed/duplex setup	

Table 4-30 shows configuration pins assignment to functions when boot mode is the Ethernet RMII mode.

**Table 4-30. Ethernet RMII Boot Configuration Fields**

BOOTMODE Pins	Field	Value	Description	MCU Only=1 Value
6	Clk out	0	50 MHz clock not generated on MCU CLKOUT	0
		1	50 MHz clock generated on MCU CLKOUT	
5	Clk src	0	External clock source	0
		1	Internal clock source	
4	Reserved	X	Not used	N/A

Table 4-31 shows configuration pins assignment to functions when boot mode is the Ethernet is the backup mode.

**Table 4-31. Ethernet Backup Boot Configuration Field**

BOOTMODE Pins	Field	Value	Description	MCU Only=1 Value
7 <sup>(1)</sup>	Interface	0	RGMII with internal Tx delay	N/A
		1	RMII with external clock source	

(1) When Ethernet is the backup mode

Table 4-32 summarizes the RGMII pin configuration done by ROM code for Ethernet boot device on RGMII port.

**Table 4-32. RGMII Pin Usage**

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Tx En/Dis	Pinmux Sel
MCU_RGMII1_TX_CTL	MCU_RGMII1_TX_CTL	Disable	Down	0	Disable	Enable	0
MCU_RGMII1_RX_CTL	MCU_RGMII1_RX_CTL	Disable	Down	0	Enable	Disable	0
MCU_RGMII1_TD3	MCU_RGMII1_TD3	Disable	Down	0	Disable	Enable	0
MCU_RGMII1_TD2	MCU_RGMII1_TD2	Disable	Down	0	Disable	Enable	0
MCU_RGMII1_TD1	MCU_RGMII1_TD1	Disable	Down	0	Disable	Enable	0
MCU_RGMII1_TD0	MCU_RGMII1_TD0	Disable	Down	0	Disable	Enable	0
MCU_RGMII1_TXC	MCU_RGMII1_TXC	Disable	Down	0	Enable	Enable	0
MCU_RGMII1_RXC	MCU_RGMII1_RXC	Disable	Down	0	Enable	Disable	0
MCU_RGMII1_RD3	MCU_RGMII1_RD3	Disable	Down	0	Enable	Disable	0
MCU_RGMII1_RD2	MCU_RGMII1_RD2	Disable	Down	0	Enable	Disable	0
MCU_RGMII1_RD1	MCU_RGMII1_RD1	Disable	Down	0	Enable	Disable	0
MCU_RGMII1_RD0	MCU_RGMII1_RD0	Disable	Down	0	Enable	Disable	0
MCU_MDIO0_MDIO	MCU_MDIO0_MDIO	Enable	Up	0	Enable	Enable	0

**Table 4-32. RGMII Pin Usage (continued)**

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Tx En/Dis	Pinmux Sel
MCU_MDIO0_MDC	MCU_MDIO0_MDC	Enable	Up	0	Disable	Enable	0
WKUP_GPIO_11	MCU_CLKOUT0	Disable	Down	0	Disable	Enable	6

Table 4-33 summarizes the RMII pin configuration done by ROM code for Ethernet boot device on RMII port.

**Table 4-33. RMII Pin Usage**

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Tx En/Dis	Pinmux Sel
MCU_RGMII1_TX_CTL	MCU_RMII1_CRSDV	Disable	Down	0	Enable	Disable	1
MCU_RGMII1_RX_CTL	MCU_RMII1_RX_ER	Disable	Down	0	Enable	Disable	1
MCU_RGMII1_TD1	MCU_RMII1_TXD1	Disable	Down	0	Disable	Enable	1
MCU_RGMII1_TD0	MCU_RMII1_TXD0	Disable	Down	0	Disable	Enable	1
MCU_RGMII1_TXC	MCU_RMII1_TX_EN	Disable	Down	0	Disable	Enable	1
MCU_RGMII1_RXC	MCU_RMII1_REF_CLK	Disable	Down	0	Enable	Enable	1
MCU_RGMII1_RD1	MCU_RMII1_RXD1	Disable	Down	0	Enable	Disable	1
MCU_RGMII1_RD0	MCU_RMII1_RXD0	Disable	Down	0	Enable	Disable	1
WKUP_GPIO_11	MCU_CLKOUT0	Disable	Down	0	Disable	Enable	6
MCU_MDIO0_MDIO	MCU_MDIO0_MDIO	Enable	Up	0	Enable	Enable	0
MCU_MDIO0_MDC	MCU_MDIO0_MDC	Enable	Up	0	Disable	Enable	0

### 4.3.11 USB Boot Device Configuration

Table 4-34 shows configuration pins assignment to functions when boot mode is the USB mode.

**Table 4-34. USB Boot Configuration Fields**

BOOTMODE Pins	Field	Value	Description	MCU Only=1 Value
6 (7) <sup>(1)</sup>	Port	0	Port 0	N/A
		1	Port 1	
5	Mode	0	DFU (USB device mode)	N/A
		1	Reserved	
4	Lane Swap	0	D+/D- lines are not swapped	N/A
		1	D+/D- lines are swapped	

(1) When USB is the backup mode

Table 4-35 summarizes the USB pin configuration done by ROM code for USB boot device on port 0.

**Table 4-35. USB Port 0 Pin Usage**

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Tx En/Dis	Pinmux Sel
USB0_DRVVBUS	USB0_DRVVBUS	Enable	Down	0	Disable	Enable	0

Table 4-36 summarizes the USB pin configuration done by ROM code for USB boot device on port 1.

**Table 4-36. USB Port 1 Pin Usage**

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Tx En/Dis	Pinmux Sel
USB0_DRVVBUS	USB1_DRVVBUS	Enable	Down	0	Disable	Enable	1

#### Note

Note that other USB pins do not have pin mux options.



### 4.3.12 PCIe Boot Device Configuration

Table 4-37 shows configuration pins assignment to functions when boot mode is the PCIe mode.

**Table 4-37. PCIe Boot Configuration Fields**

BOOTMODE Pins	Field	Value	Description	MCU Only=1 Value
6	Port	0	Port 0	N/A
		1	Port 1	
5	N Lanes	0	Max lanes	N/A
		1	Reserved	
4	Clocking	0	From external pins	N/A
		1	From internal source	

#### Note

Note that PCIe (SERDES) pins do not have pin mux options.

### 4.3.13 UART Boot Device Configuration

ROM Code always configures the UART port to 115200 kbaud, 8-n-1 mode, and the XMODEM protocol is used to transfer the boot data.

**Table 4-38. UART Boot Configuration Fields**

BOOTMODE Pins	Field	Value	Description	MCU Only=1 Value
6	Reserved	-	Not used	N/A
5	Reserved	-	Not used	N/A
4 (7) <sup>(1)</sup>	Port	0	MCU UART (port 0)	0
		1	WKUP UART (port 1)	

(1) When UART is the backup mode and MCU Only=0

Table 4-39 summarizes the UART pin configuration done by ROM code for UART host on port 0.

**Table 4-39. UART Port 0 Pin Usage**

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Tx En/Dis	Pinmux Sel
WKUP_GPIO0_12	MCU_UART_TXD	Enable	Up	0	Disable	Enable	0
WKUP_GPIO0_13	MCU_UART_RXD	Enable	Up	0	Enable	Disable	0

Table 4-40 summarizes the UART pin configuration done by ROM code for UART host on port 1.

**Table 4-40. UART Port 1 Pin Usage**

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Tx En/Dis	Pinmux Sel
WKUP_UART_TXD	WKUP_UART_TXD	Enable	Up	0	Disable	Enable	0
WKUP_UART_RXD	WKUP_UART_RXD	Enable	Up	0	Enable	Disable	0

#### 4.3.14 GPMC NOR Boot Device Configuration

Table 4-41 shows configuration pins assignment to functions when boot mode is the GPMC NOR mode.

**Table 4-41. GPMC NOR Boot Configuration Fields**

BOOTMODE Pins	Field	Value	Description	MCU Only=1 Value
6-5	A/D Mux	0	Non-mux	N/A
		1	Reserved	
		2	Reserved	
		3	Non-mux	
4	Csel	0	Chip select 0	N/A
		1	Chip select 1	

Table 4-42 summarizes the GPMC pin configuration done by ROM code for GPMC non-muxed memory.

**Table 4-42. GPMC non-muxed Pin Usage**

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Tx En/Dis	Pinmux Sel
PRG0_PRU0_GPO5	GPMC0_AD0	Disable	Down	0	Enable	Enable	8
PRG0_PRU0_GPO7	GPMC0_AD1	Disable	Down	0	Enable	Enable	8
PRG0_PRU0_GPO8	GPMC0_AD2	Disable	Down	0	Enable	Enable	8
PRG0_PRU0_GPO9	GPMC0_AD3	Disable	Down	0	Enable	Enable	8
PRG0_PRU0_GPO10	GPMC0_AD4	Disable	Down	0	Enable	Enable	8
PRG0_PRU0_GPO17	GPMC0_AD7	Disable	Down	0	Enable	Enable	8
PRG0_PRU0_GPO18	GPMC0_AD6	Disable	Down	0	Enable	Enable	8
PRG0_PRU1_GPO5	GPMC0_AD8	Disable	Down	0	Enable	Enable	8
PRG0_PRU1_GPO7	GPMC0_AD9	Disable	Down	0	Enable	Enable	8
PRG0_PRU1_GPO8	GPMC0_AD10	Disable	Down	0	Enable	Enable	8
PRG0_PRU1_GPO9	GPMC0_AD11	Disable	Down	0	Enable	Enable	8
PRG0_PRU1_GPO10	GPMC0_AD12	Disable	Down	0	Enable	Enable	8
PRG0_PRU1_GPO17	GPMC0_AD13	Disable	Down	0	Enable	Enable	8
PRG0_PRU1_GPO18	GPMC0_AD14	Disable	Down	0	Enable	Enable	8
PRG0_PRU1_GPO19	GPMC0_AD15	Disable	Down	0	Enable	Enable	8
PRG0_MDIO0_MDC	GPMC_A0	Disable	Down	0	Disable	Enable	8
RGMII5_TX_CTL	GPMC_A1	Disable	Down	0	Disable	Enable	8
RGMII5_RX_CTL	GPMC_A2	Disable	Down	0	Disable	Enable	8
RGMII5_TD3	GPMC_A3	Disable	Down	0	Disable	Enable	8
RGMII5_TD2	GPMC_A4	Disable	Down	0	Disable	Enable	8
RGMII5_TD1	GPMC_A5	Disable	Down	0	Disable	Enable	8
RGMII5_TD0	GPMC_A6	Disable	Down	0	Disable	Enable	8
RGMII5_TXC	GPMC_A7	Disable	Down	0	Disable	Enable	8
RGMII5_RXC	GPMC_A8	Disable	Down	0	Disable	Enable	8
RGMII5_RD3	GPMC_A9	Disable	Down	0	Disable	Enable	8
RGMII5_RD2	GPMC_A10	Disable	Down	0	Disable	Enable	8
RGMII5_RD1	GPMC_A11	Disable	Down	0	Disable	Enable	8
RGMII5_RD0	GPMC_A12	Disable	Down	0	Disable	Enable	8
RGMII6_TX_CTL	GPMC_A13	Disable	Down	0	Disable	Enable	8
RGMII6_RX_CTL	GPMC_A14	Disable	Down	0	Disable	Enable	8
RGMII6_TD3	GPMC_A15	Disable	Down	0	Disable	Enable	8

**Table 4-42. GPMC non-muxed Pin Usage (continued)**

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Tx En/Dis	Pinmux Sel
RGMII6_TD2	GPMC_A16	Disable	Down	0	Disable	Enable	8
RGMII6_TD1	GPMC_A17	Disable	Down	0	Disable	Enable	8
RGMII6_TD0	GPMC_A18	Disable	Down	0	Disable	Enable	8
RGMII6_TXC	GPMC_A19	Disable	Down	0	Disable	Enable	8
RGMII6_RXC	GPMC_A20	Disable	Down	0	Disable	Enable	8
RGMII6_RD3	GPMC_A21	Disable	Down	0	Disable	Enable	8
RGMII6_RD2	GPMC_A22	Disable	Down	0	Disable	Enable	8
PRG0_PRU1_GPO2	GPMC_A23	Disable	Down	0	Disable	Enable	8
PRG0_PRU1_GPO4	GPMC_A24	Disable	Down	0	Disable	Enable	8
PRG0_PRU1_GPO6	GPMC_A25	Disable	Down	0	Disable	Enable	8
PRG0_PRU1_GPO11	GPMC_A26	Disable	Down	0	Disable	Enable	8
PRG0_MDIO0_MDIO	GPMC_A27	Disable	Down	0	Disable	Enable	8
PRG1_PRU0_GPO9	GPMC0_ADVn_ALE	Disable	Up	0	Disable	Enable	8
PRG1_PRU0_GPO8	GPMC0_OEn_Ren	Disable	Up	0	Disable	Enable	8
PRG1_PRU0_GPO5	GPMC0_Wen	Disable	Up	0	Disable	Enable	8
PRG1_PRU0_GP10	GPMC0_BEOn_CLE	Disable	Up	0	Disable	Enable	8
RGMII6_RD1	GPMC0_BE1n	Disable	Up	0	Disable	Enable	8
PRG1_PRU0_GPO1	GPMC_WAIT0	Enable	Down	0	Enable	Disable	8
PRG1_PRU0_GPO2	GPMC_WAIT1	Enable	Down	0	Enable	Disable	8
PRG1_PRU1_GPO5	GPMC0_WPn	Disable	Down	0	Disable	Enable	8
RGMII6_RD0	GPMC0_DIR	Disable	Up	0	Disable	Enable	8
PRG1_PRU1_GPO9	GPMC0_CSn0	Enable	Up	0	Disable	Enable	8
PRG1_PRU1_GPO8	GPMC0_CSn1	Enable	Up	0	Disable	Enable	8
PRG1_PRU0_GPO4	GPMC0_CSn2	Enable	Up	0	Disable	Enable	8
PRG1_PRU0_GPO6	GPMC0_CSn3	Enable	Up	0	Disable	Enable	8

Table 4-43 summarizes the GPMC pin configuration done by ROM code for GPMC Address/Data muxed memory.

**Table 4-43. GPMC A/D muxed Pin Usage**

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Tx En/Dis	Pinmux Sel
PRG0_PRU0_GPO5	GPMC0_AD0	Disable	Down	0	Enable	Enable	8
PRG0_PRU0_GPO7	GPMC0_AD1	Disable	Down	0	Enable	Enable	8
PRG0_PRU0_GPO8	GPMC0_AD2	Disable	Down	0	Enable	Enable	8
PRG0_PRU0_GPO9	GPMC0_AD3	Disable	Down	0	Enable	Enable	8
PRG0_PRU0_GPO10	GPMC0_AD4	Disable	Down	0	Enable	Enable	8
PRG0_PRU0_GPO17	GPMC0_AD7	Disable	Down	0	Enable	Enable	8
PRG0_PRU0_GPO18	GPMC0_AD6	Disable	Down	0	Enable	Enable	8
PRG0_PRU1_GPO5	GPMC0_AD8	Disable	Down	0	Enable	Enable	8
PRG0_PRU1_GPO7	GPMC0_AD9	Disable	Down	0	Enable	Enable	8
PRG0_PRU1_GPO8	GPMC0_AD10	Disable	Down	0	Enable	Enable	8
PRG0_PRU1_GPO9	GPMC0_AD11	Disable	Down	0	Enable	Enable	8
PRG0_PRU1_GPO10	GPMC0_AD12	Disable	Down	0	Enable	Enable	8
PRG0_PRU1_GPO17	GPMC0_AD13	Disable	Down	0	Enable	Enable	8

**Table 4-43. GPMC A/D muxed Pin Usage (continued)**

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Tx En/Dis	Pinmux Sel
PRG0_PRU1_GPO18	GPMC0_AD14	Disable	Down	0	Enable	Enable	8
PRG0_PRU1_GPO19	GPMC0_AD15	Disable	Down	0	Enable	Enable	8
RGMII6_TD2	GPMC_A16	Disable	Down	0	Disable	Enable	8
RGMII6_TD1	GPMC_A17	Disable	Down	0	Disable	Enable	8
RGMII6_TD0	GPMC_A18	Disable	Down	0	Disable	Enable	8
RGMII6_TXC	GPMC_A19	Disable	Down	0	Disable	Enable	8
RGMII6_RXC	GPMC_A20	Disable	Down	0	Disable	Enable	8
RGMII6_RD3	GPMC_A21	Disable	Down	0	Disable	Enable	8
RGMII6_RD2	GPMC_A22	Disable	Down	0	Disable	Enable	8
PRG0_PRU1_GPO2	GPMC_A23	Disable	Down	0	Disable	Enable	8
PRG0_PRU1_GPO4	GPMC_A24	Disable	Down	0	Disable	Enable	8
PRG0_PRU1_GPO6	GPMC_A25	Disable	Down	0	Disable	Enable	8
PRG0_PRU1_GPO11	GPMC_A26	Disable	Down	0	Disable	Enable	8
PRG0_MDIO0_MDIO	GPMC_A27	Disable	Down	0	Disable	Enable	8
PRG1_PRU0_GPO9	GPMC0_ADVn_ALE	Disable	Up	0	Disable	Enable	8
PRG1_PRU0_GPO8	GPMC0_OEn_Ren	Disable	Up	0	Disable	Enable	8
PRG1_PRU0_GPO5	GPMC0_Wen	Disable	Up	0	Disable	Enable	8
PRG1_PRU0_GP10	GPMC0_BEOn_CLE	Disable	Up	0	Disable	Enable	8
RGMII6_RD1	GPMC0_BE1n	Disable	Up	0	Disable	Enable	8
PRG1_PRU0_GPO1	GPMC_WAIT0	Enable	Down	0	Enable	Disable	8
PRG1_PRU0_GPO2	GPMC_WAIT1	Enable	Down	0	Enable	Disable	8
PRG1_PRU1_GPO5	GPMC0_WPn	Disable	Down	0	Disable	Enable	8
RGMII6_RD0	GPMC0_DIR	Disable	Up	0	Disable	Enable	8
PRG1_PRU1_GPO9	GPMC0_CSn0	Enable	Up	0	Disable	Enable	8
PRG1_PRU1_GPO8	GPMC0_CSn1	Enable	Up	0	Disable	Enable	8
PRG1_PRU0_GPO4	GPMC0_CSn2	Enable	Up	0	Disable	Enable	8
PRG1_PRU0_GPO6	GPMC0_CSn3	Enable	Up	0	Disable	Enable	8

Table 4-44 summarizes the GPMC pin configuration done by ROM code for GPMC Address/Address/Data muxed memory.

**Table 4-44. GPMC A/A/D muxed Pin Usage**

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Tx En/Dis	Pinmux Sel
PRG0_PRU0_GPO5	GPMC0_AD0	Disable	Down	0	Enable	Enable	8
PRG0_PRU0_GPO7	GPMC0_AD1	Disable	Down	0	Enable	Enable	8
PRG0_PRU0_GPO8	GPMC0_AD2	Disable	Down	0	Enable	Enable	8
PRG0_PRU0_GPO9	GPMC0_AD3	Disable	Down	0	Enable	Enable	8
PRG0_PRU0_GPO10	GPMC0_AD4	Disable	Down	0	Enable	Enable	8
PRG0_PRU0_GPO17	GPMC0_AD7	Disable	Down	0	Enable	Enable	8
PRG0_PRU0_GPO18	GPMC0_AD6	Disable	Down	0	Enable	Enable	8
PRG0_PRU1_GPO5	GPMC0_AD8	Disable	Down	0	Enable	Enable	8
PRG0_PRU1_GPO7	GPMC0_AD9	Disable	Down	0	Enable	Enable	8
PRG0_PRU1_GPO8	GPMC0_AD10	Disable	Down	0	Enable	Enable	8
PRG0_PRU1_GPO9	GPMC0_AD11	Disable	Down	0	Enable	Enable	8

**Table 4-44. GPMC A/A/D muxed Pin Usage (continued)**

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Tx En/Dis	Pinmux Sel
PRG0_PRU1_GPO10	GPMC0_AD12	Disable	Down	0	Enable	Enable	8
PRG0_PRU1_GPO17	GPMC0_AD13	Disable	Down	0	Enable	Enable	8
PRG0_PRU1_GPO18	GPMC0_AD14	Disable	Down	0	Enable	Enable	8
PRG0_PRU1_GPO19	GPMC0_AD15	Disable	Down	0	Enable	Enable	8
PRG1_PRU0_GPO9	GPMC0_ADVn_ALE	Disable	Up	0	Disable	Enable	8
PRG1_PRU0_GPO8	GPMC0_OEn_Ren	Disable	Up	0	Disable	Enable	8
PRG1_PRU0_GPO5	GPMC0_Wen	Disable	Up	0	Disable	Enable	8
PRG1_PRU0_GP10	GPMC0_BEOn_CLE	Disable	Up	0	Disable	Enable	8
RGMII6_RD1	GPMC0_BE1n	Disable	Up	0	Disable	Enable	8
PRG1_PRU0_GPO1	GPMC_WAIT0	Enable	Down	0	Enable	Disable	8
PRG1_PRU0_GPO2	GPMC_WAIT1	Enable	Down	0	Enable	Disable	8
PRG1_PRU1_GPO5	GPMC0_WPn	Disable	Down	0	Disable	Enable	8
RGMII6_RD0	GPMC0_DIR	Disable	Up	0	Disable	Enable	8
PRG1_PRU1_GPO9	GPMC0_CSn0	Enable	Up	0	Disable	Enable	8
PRG1_PRU1_GPO8	GPMC0_CSn1	Enable	Up	0	Disable	Enable	8
PRG1_PRU0_GPO4	GPMC0_CSn2	Enable	Up	0	Disable	Enable	8
PRG1_PRU0_GPO6	GPMC0_CSn3	Enable	Up	0	Disable	Enable	8

### 4.3.15 eMMC Boot Device Configuration

Table 4-45 shows configuration pins assignment to functions when boot mode is the eMMC mode.

**Table 4-45. eMMC Boot Configuration Fields**

BOOTMODE Pins	Field	Value	Description	MCU Only=1 Value
6	Port	0	Port 0	N/A
		1	Port 1	
5	Bus Width	0	Max width by port (8-bit on port 0, 4-bit on port 1)	N/A
		1	1-bit only	
4	Voltage	0	1.8V	N/A
		1	3.3V	

Table 4-43 summarizes the MMC pin configuration done by ROM code for eMMC on port 1.

#### Note

Note that MMC Port 0 has no pin mux options.

**Table 4-46. eMMC Port 1 Pin Usage**

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Tx En/Dis	Pinmux Sel
MMC1_DAT3	MMC1_DAT3	Enable	Up	0	Enable	Enable	0
MMC1_DAT2	MMC1_DAT2	Enable	Up	0	Enable	Enable	0
MMC1_DAT1	MMC1_DAT1	Enable	Up	0	Enable	Enable	0
MMC1_DAT0	MMC1_DAT0	Enable	Up	0	Enable	Enable	0
MMC1_CLK	MMC1_CLK	Disable	Up	0	Disable	Enable	0
MMC1_CMD	MMC1_CMD	Enable	Up	0	Enable	Enable	0
MMC1_SDCD	MMC1_SDCD	Enable	Up	0	Enable	Disable	0

#### 4.3.15.1 eMMC Flash

To support eMMC boot mode across a device warm reset, eMMC flash has the following requirements:

1. The reset line must be connected to the eMMC flash input reset pin.
2. The eMMC flash ext\_csd[162] RST\_n\_ENABLE must be set to the expected configuration for the warm-reset signal to propagate to the flash device for the boot process to succeed. Possible configurations are shown in Table 4-47.

#### Note

For warm\_reset eMMC boot to function correctly, 0x1 must be written to RST\_n\_ENABLE in the ext\_csd[162] Register. This will enable the RST\_n signal.

**Table 4-47. ext\_csd[162] Register**

Bit	Field	Type	Description
7:2	Reserved		Reserved
1:0	RST_n_ENABLE	R/W	0x0: RST_n signal is temporarily disabled (default) 0x1: RST_n signal is permanently enabled 0x2 RST_n signal is permanently disabled 0x3: Reserved



### 4.3.16 PLL Configuration

ROM code must be aware of the reference clock provided to PLLs. That is, the speed of the quartz crystal, or the clock supplied by an external clock oscillator. On how to indicate the PLL reference clock, see , *PLL Reference Clock Selection*.

See [Section 5.4, Clock Management](#), for the PLL reference clock scheme.

ROM code configures only PLLs which are required during boot. Therefore, if a PLL is required for the backup boot mode but not the primary boot mode, and if the backup boot mode never executes, then the PLLs required for backup boot are not enabled. [Table 4-48](#) lists the enabled PLLs according to module or boot peripheral.

**Table 4-48. PLL Configuration by Boot Mode**

MCU_PLL0	MCU_PLL1	MCU_PLL2	Main PLL0	Main PLL1	Main PLL2	Main PLL3	Boot Mode
✓	✓						OSPI/QSPI/SPI
✓		✓					Ethernet
✓	✓						I2C
✓	✓						UART
✓			✓				MMCSD
✓			✓	✓			USB
✓			✓				GPCM NOR
✓			✓		✓		PCIe

#### Note

All clock frequencies are in megahertz.

#### 4.3.16.1 MCU\_PLL0, MCU\_PLL2, Main PLL0, and Main PLL3

[Table 4-49](#) summarizes the ROM code settings used to configure MCU\_PLL0, MCU\_PLL2, Main PLL0, and Main PLL3 for the supported input clocks.

**Table 4-49. PLL Configuration for MCU\_PLL0, MCU\_PLL2, Main PLL0, and Main PLL3**

2000 MHz VCO frequency								
Ref clk	Refdiv	Pfd freq	Fbdiv	Frac	Vco	Delta	Postdiv	
19.2	1	19.2	104	2796202	2000	$3.97 \times 10^{-8}$	0 or 1	
20	1	20	100	0	2000	0	0 or 1	
24	1	24	83	5592405	2000	$1.99 \times 10^{-8}$	0 or 1	
25	1	25	80	0	2000	0	0 or 1	
26	1	26	76	15486661	2000	$5.50 \times 10^{-8}$	0 or 1	
27	1	27	74	1242757	2000	$4.42 \times 10^{-8}$	0 or 1	

#### 4.3.16.2 MCU\_PLL1

The MCU\_PLL1 is a fractional PLL configured with a VCO frequency of 2400 MHz.

**Table 4-50. PLL Configuration for MCU\_PLL1**

2400 MHz VCO frequency								
Ref clk	Refdiv	Pfd freq	Fbdiv	Frac	Vco	Delta	Postdiv	
19.2	1	19.2	125	0	2400	0	0	
20	1	20	120	0	2400	0	0	
24	1	24	100	0	2400	0	0	
25	1	96	192	0	2400	0	0	
26	1	26	92	5162220	2400	$1.8 \times 10^{-8}$	0	

**Table 4-50. PLL Configuration for MCU\_PLL1 (continued)**

2400 MHz VCO frequency							
Ref clk	Refdiv	Pfd freq	Fbdiv	Frac	Vco	Delta	Postdiv
27	1	27	88	14913081	2400	$5.53 \times 10^{-8}$	0

#### 4.3.16.3 Main PLL1

The main PLL is a fractional PLL configured with a VCO frequency of 1920 MHz.

**Table 4-51. PLL Configuration for Main PLL1**

1920 MHz VCO frequency							
Ref clk	Refdiv	Pfd freq	Fbdiv	Frac	Vco	Delta	Postdiv
19.2	1	19.2	100	0	1920	0	0
20	1	20	96	0	1920	0	0
24	1	24	80	0	1920	0	0
25	1	25	76	13421773	1920	$4.77 \times 10^{-8}$	0
26	1	26	73	14196106	1920	$5.04 \times 10^{-8}$	0
27	1	27	71	1864135	1920	$6.62 \times 10^{-8}$	0

#### 4.3.16.4 Main PLL2

Main PLL2 is an fractional PLL configured with a VCO frequency of 1800 MHz.

**Table 4-52. PLL Configuration for Main PLL 2**

1800 MHz VCO frequency							
Ref clk	Refdiv	Pfd freq	Fbdiv	Frac	Vco	Delta	Postdiv
19.2	1	19.2	93	12582912	1800	0	0
20	1	20	90	0	1800	0	0
24	1	24	75	0	1800	0	0
25	1	25	72	0	1800	0	0
26	1	26	69	3871665	1800	$1.38 \times 10^{-8}$	0
27	1	27	66	11184811	1800	$3.97 \times 10^{-8}$	0

#### 4.3.16.5

##### Note

Note that the bringup and configuration of bootmode-specific PLLs by ROM code will result in a bootmode-specific device clocking setup which for example has an impact on which peripheral modules can readily be clocked and used by SBL/SPL prior to loading and bringing up System Firmware (SYSFW).

## 4.4 Boot Parameter Tables

The boot parameter tables direct the main module boot process. On cold boot the tables are created based on pin strapped values (see [Section 4.3, Boot Mode Pins](#)) and built-in data. The ROM Code supports two parameter tables stored as an array in a fixed memory address in the MSRAM, each of size 512 bytes. The ROM will attempt to boot using the primary table. On boot failure, ROM Code will retry using the second table.

Using two tables handles two cases.

- The first is in initial board manufacture where the primary boot table specifies boot from a flash device, and the flash is blank. ROM Code would then switch to the secondary boot mode which would receive the image externally (Ethernet, USB, PCIe, UART) and this image would flash the boot image.
- The second case is failure due to total flash corruption. In all flash parameter tables there exist backup addresses within the primary boot mode. This covers the problem of a flash update failure with a backup image present on the same flash device.

The boot tables reside at a fixed location in memory which is described in [Section 4.7.2, Global Memory Addresses Used by ROM Code](#).

### 4.4.1 Common Header

These boot parameter tables have certain parameters common across all the boot modes, while the rest of the parameters are unique to the boot modes. The common entries in the boot parameter table are shown in [Table 4-53](#).

**Table 4-53. Boot Parameter Table Common Header**

Byte Offset	Size (bytes)	Name	Description
0	2	Length	The length of the table
2	2	Checksum	Ones complement checksum over length bytes in the table. If 0 the checksum is not validated.
4	2	Peripheral	Identifies the boot peripheral and format of the table after the common header. See <a href="#">Table 4-54</a>
6	2	Reserved	Reserved
8	4	Timeout	Timeout for this boot mode, in milliseconds
12	4	Magic	Magic value 0x01AD0911
16	40	PLL Config 0	PLL Configuration 0. See <a href="#">Table 4-55</a>
56	40	PLL Config 1	PLL Configuration 1
96	40	PLL Config 2	PLL Configuration 2
136	40	PLL Config 3	PLL Configuration 3
176	40	PLL Config 4	PLL Configuration 4
216	40	PLL Config 5	PLL Configuration 5

[Table 4-54](#) lists the possible boot modes used in the boot parameter tables.

**Table 4-54. Boot Peripheral Selection**

Peripheral Field Value	Description
0	Sleep (No boot)
10	Ethernet Reserved
20	Ethernet BOOTP/TFTP (general)
21	Ethernet RGMII specific
22	Ethernet RMII specific
30	PCIe
31	PCIe 1-lane
32	PCIe 2-lane
40	I2C

**Table 4-54. Boot Peripheral Selection (continued)**

Peripheral Field Value	Description
50	SPI
60	UART
70	USB DFU
71	USB Reserved
72	USB Reserved
80	QSPI
85	OSPI
90	Hyperflash Reserved
100	MMC/SD Card (general)
101	eMMC (general)
102	MMC 1-bit
103	MMC 4-bit
104	MMC 8-bit
110	GPMC
120	UFS Reserved
130	xSPI

#### 4.4.2 PLL Setup

Table 4-55 through describe the PLL configuration fields.

**Table 4-55. Boot Parameter Table PLL Configuration**

Byte Offset	Size (bytes)	Name	Description
0	1	Domain/cfg	See Table 4-56
1	1	PLL number	PLL number indexed from 0. See , for PLL numbers.
2	1	Input source	See Table 4-58
3	1	PLL Type	This field must be 1 to indicate an SCPLL
4	4	Input Ref Clock	The PLL input clock, in Q16.16 format
8	4	Feed back divider, integer part	Integer value of feedback divider
12	4	Feed back divider, fractional part	Fractional portion of feedback divider. Total divider is the Integer part + (Fractional part / 2 <sup>24</sup> )
16	1	Ref divider	Input clock pre-divider
17	1	Post divider 1	Output post divider 1
18	1	Post divider 2	Output post divider 2
19	1	Reserved	Reserved
20	2	Hsdiv Enable	Bit map. A set bit indicates that the corresponding hsdiv is enabled.
22	2	Reserved	Reserved
24	16	Hsdiv[16]	Array of hs divider values.

**Table 4-56. PLL Domain and Enable Configuration**

7	6	5	4	3	2	1	0
Reserved		Enable		Reserved		Domain	

**Table 4-57. PLL Domain and Enable Field Description**

Field	Value	Description
Enable	0	PLL not configured
	1	PLL enabled only if currently disabled or in bypass
	2	PLL is unconditionally enabled. If currently enabled with a different configuration the PLL is first disabled

**Table 4-57. PLL Domain and Enable Field Description (continued)**

Field	Value	Description
Domain	3	PLL is unconditionally disabled
	0	PLL is in the MCU domain
	1	PLL is in the MAIN domain

**Table 4-58. PLL Reference Source Bit Fields**

7	6	5	4	3	2	1	0
Source Type				Source Index			

**Table 4-59. PLL Reference Source Field Description**

Field	Value	Description
Source Type	0	Source is HFOSC
	1	Source is external pin
	2	Reserved
	3-7	Reserved
Source Index	0-31	Source index (HFOSC[0-31] or pin[0-31], depending on Source Type) HFOSC[0] – WKUP_HFOSC0 HFOSC[1] – HFOSC1 (in MAIN domain) PIN[1] – EXT_REFCLK1 pin (not all PLLs, see <a href="#">Section 5.4, Clocking</a> )

#### 4.4.3 PCIe Boot Parameter Table

[Table 4-60](#) shows the boot parameter table for PCIe boot. Must be preceded with the common boot parameters described in [Table 4-53](#).

**Table 4-60. PCIe Boot Parameter Table**

Byte Offset	Size (bytes)	Name	Default Value	Description
256	4	portNum	From pins	Physical port number
260	4	AddrWid	64	PCIe address width
264	4	LinkRate	8000	Link rate in MHz
268	4	RefClkKHz	100000	Serdes reference clock in kHz
272	4	Nlanes	From pins	Number of PCIe lanes configured (link width)
276	4	Rsvd	4	Reserved
280	4	Rsvd	256	Reserved
284	4	Rsvd	256	Reserved
288	4	Rsvd	0	Reserved
292	4	Rsvd	0	Reserved
296	4	Vendor ID	0x104C	PCIe Vendor ID value (read from control registers)
300	4	Device ID	0xB00D	PCIe Device ID value (read from control registers)
304	4	Class code/revision ID	0x04800001	PCIe class code and revision ID value
308	4	Internal Clock	From Pins	If non-zero, internal (SoC) ref clock is used
312	4	sscEnable	1	Enable spread spectrum clock
316	4	RefSrc	From Pins	Selects serdes reference clock internal source. Refer to , <i>Reference Clock Distribution</i> for details.

#### 4.4.4 I2C Boot Parameter Table

[Table 4-61](#) shows the boot parameter table for I2C boot. Must be preceded with the common boot parameters described in [Table 4-53](#).

**Table 4-61. I2C Boot Parameter Table**

Byte Offset	Size (bytes)	Name	Default Value	Description
256	1	Port	0	Physical port number
257	1	Mode	From Pins	0x4E = I2C Master, 0x72 = I2C slave
258	1	Dev Addr	From Pins	I2C address when slave mode (0x10 or 0x11)
259	1	Reserved	0	Reserved
260	4	Mod Clock	0	I2C Module input clock. If 0, it is computed by ROM code.
264	2	Bus Freq	400	I2C Master mode bus frequency, in kHz
266	2	Bus Addr	From Pins	I2C Master mode storage device's address (0x50 or 0x51)
268	2	Read Index	0	Index to the active read offset (0 or 1)
270	2	Read Offset 0	0x0000	I2C Master mode read offset
272	2	Read Offset 1	0x8000	I2C Master mode backup read offset
274	2	Reserved	0	Reserved
276	2	Reserved	0	Reserved
280	2	Busy Timeout	From pins	Number of $\mu$ s before a bus recovery is attempted. In units of microseconds in Q3 number format. Value of 0 disables bus recovery attempts.

#### 4.4.5 OSPI/QSPI/SPI/xSPI Boot Parameter Table

Table 4-62 shows the boot parameter table for OSPI, QSPI, or SPI boot. Table 4-63 shows the boot parameter table for xSPI. Must be preceded with the common boot parameters described in Table 4-53.

**Table 4-62. OSPI/QSPI/SPI Boot Parameter Table**

Byte Offset	Size (bytes)	Name	Default Value	Description
256	1	Port	0	Physical port number
257	1	Mode on	From Pins	If non-zero, the mode byte will be sent
258	1	Instruct Width	From Pins	Number of pins used to send instructions (1, 2, 4, 8)
259	1	Address Width	From Pins	Number of pins used to send address (1, 2, 4, 8)
260	1	Data Width	From Pins	Number of pins used to received data (1, 2, 4, 8)
261	1	Address Size	24	16 (SPI only), 24, and 32 bits are the valid address sizes
262	1	Mode	0	OSPI clock polarity and phase mode
263	1	CSEL	From Pins	Chip select number (0–3)
264	1	Read Cmd	From Pins	Command used to read read data
265	1	Mode byte	0	Value used for the mode byte (when active)
266	1	Dummy Cycles	From pins	Number of dummy cycles sent after the read command
267	1	clkRecovery	From pins	Clock recovery
268	1	dqsEnable	0	Enable DQS
269	1	Reserved	0	Reserved
270	2	Module Freq	0	The OSPI module frequency after PLL enable, in kHz. If 0, ROM code uses the value from the module clock tables.
272	4	Bus Frequency	From pins	The OSPI bus frequency, in kHz
276	4	Delay	0x08080808 or 0x01010101	The chip select read delays. Default value is based on the read command
280	4	Tap Delay	0xFFFFFFFF	The read tap selection. If 0xFFFFFFFF, the ROM code will scan the taps to find the best delay. The result will then overwrite the value in this table.
284	4	Internal Clk	From pins	0 = external (dqs) 1 = internal
288	4	notDAC	From pins	When 0, DAC mode is used

**Table 4-62. OSPI/QSPI/SPI Boot Parameter Table (continued)**

Byte Offset	Size (bytes)	Name	Default Value	Description
292	4	Read Index	0	Index to the active read address (0-1)
296	4	Read Addr 0	0x000000	The initial flash read address
300	4	Read Addr 1	0x400000 (0x4000 SPI)	Backup read address
304	4	Reserved	0	Reserved
308	4	Reserved	0	Reserved

**Table 4-63. xSPI Parameter Table**

Byte Offset	Size (bytes)	Name	Default Value	Description
256	1	Port	0	Physical port number
257	1	Mode on	From Pins	If non-zero the mode byte will be sent
258	1	Instruct Width	From Pins	Number of pins used to send instructions (1, 8)
259	1	Address Width	From Pins	Number of pins used to send address (1,8)
260	1	Data Width	From Pins	Number of pins used to received data (1,8)
261	1	Address Size	24	24 or 32 bits are the valid address sizes
262	1	Mode	0	QSPI clock polarity and phase mode
263	1	CSEL	From Pins	Chip select number (0-n) not the bitfield
264	1	Read Cmd	From Pins	Command used to read read data
265	1	Mode byte	0	Value used for the mode byte (when active)
266	1	Dummy Cycles	From pins	Number of dummy cycles sent after the read command
267	1	clkRecovery	From pins	Clock recovery
268	1	dqsEnable	0	Enable dqs
269	1	ddrEnable	From pins	OSPI DDR mode operation
270	2	Module Freq	0	The QSPI module frequency after PLL enable, in kHz. If 0 the ROM uses the value from the module clock tables.
272	4	Bus Frequency	From pins	The QSPI bus frequency, in kHz
276	4	Delay	0x08080808 or 0x01010101	The chip select read delays. Default value is based on the read command
280	1	SFDP	From Pins	Enables SFDP parser for 1S-1S-1S to 8D-8D-8D switching



**Table 4-63. xSPI Parameter Table (continued)**

Byte Offset	Size (bytes)	Name	Default Value	Description
282	4	Tap Delay	0xffffffff	The read tap selection. If 0xffffffff the ROM will scan the taps to find the best delay. The result will then overwrite the value in this table.
286	4	Internal Clk	From pins	0 = external (dqs) 1 = internal
290	4	indac	From pins	When 0 XIP mode is used
294	4	Read Index	0	Index to the active read address
298	4	Read Addr 0	0	The initial flash read address
302	4	Read Addr 1	0x400000	Backup read address

#### 4.4.6 GPMC NOR Boot Parameter Table

Table 4-64 shows the boot parameter table for GPMC NOR boot. Must be preceded with the common boot parameters described in Table 4-53.

**Table 4-64. GPMC NOR Boot Parameter Table**

Byte Offset	Size (bytes)	Name	Default Value	Description
256	4	refClkKHz	0	The module functional clock frequency, in kHz. 0 = ROM code computes the value.
260	4	csSizeMb	64	The size of each chip-select, in MB
264	1	Csel	From pins	The chip-select to use (0-3)
265	1	adMux	From pins	The address/data multiplexing used. 0 = A/D parallel, 1 = A/A/D mux, 2 = A/D mux
266	1	Width	16	Data bus width
267	1	Reserved	0	Reserved
268	4	Read index	0	The currently active read offset (0-1)
272	4	Read offset 0	0x000000	Read address offset 0
276	4	Read offset 1	0x400000	Backup read address offset 1
280	4	Reserved	0	Reserved
284	4	Reserved	0	Reserved

#### 4.4.7 Ethernet Boot Parameter Table

Table 4-65 is shown segmented into four sections:

1. The first section contains information required to configure the device hardware
2. The second section contains information used by the top level Ethernet module to execute the boot
3. The third section contains information for the Ethernet stack code.
4. The fourth section contains information for creating the BOOTP packet (vendor string, ID string and debug string), and holds information returned in the BOOTP response (Default route and file name)

Table 4-65 shows the boot parameter table for Ethernet boot. Must be preceded with the common boot parameters described in Table 4-53.

**Table 4-65. Ethernet Boot Parameter Table**

Byte Offset	Size (bytes)	Name	Default Value	Description
Hardware Configuration Options				

**Table 4-65. Ethernet Boot Parameter Table (continued)**

Byte Offset	Size (bytes)	Name	Default Value	Description
256	2	Mod Freq	0	Module clock frequency, kHz. If 0, ROM code computes the value.
258	1	Port Num	0	Physical port number
259	1	Interface	From Pins	0 = RGMII with internal delay 1 = RGMII with external delay 2 = RMII
260	1	Init Level	0	0 = Initialize only not enabled modules 1 = Full ethernet sub-system initialization
261	1	Clock out enable	From pins	0x10 = MCU_CLKOUT enable 0x11 = MCU_CLKOUT disable
262	1	Clk out freq	From pins	0x20 = MCU_CLKOUT 25 MHz (RGMII) 0x21 = MCU_CLKOUT 50 MHz (RMII)
263	1	RMII Clk In	From pins	0x60 = RMII internal (SoC) clock 0x61 = RMII external clock
264	1	Port Enable	0x01	Bit map. A set bit indicates that the corresponding physical port will be enabled
265	1	Phy Query	From pins	0x30 = speed/duplex determined from RGMII status register 0x31 = MDIO used to query PHY 0x32 = Use fixed speed/duplex values from offset 266/267.
266	1	Speed		0x40 = full speed (1Gbit for RGMII, 100Mbit for RMII) 0x41 = slow speed (100Mbit for RGMII, 10Mbit for RMII)
267	1	Duplex		0x50 = full duplex 0x51 = half duplex
Main Level Boot Control				
268	1	Bootp enable	1	0 = Image information already in this structure 1 = Use BOOTP to get boot image information
269	1	Reserved	0	Reserved
270	2	Bootp Timeout	4000	BOOTP timeout in milli-seconds
272	2	TFTP timeout	1000	TFTP timeout in milli-seconds
274	2	Bootp retries	10	Number of BOOTP retries before fail
276	2	TFTP retries	10	Number of TFTP retries before fail
278	2	Reserved	0	Reserved
Network Stack Configuration (plus TFTP server ID)				
280	6	MAC Address	From E-fuse	MAC address of the device
286	2	Reserved	0	Reserved
288	4	Device IP	0	IP address of the device. Valid only if BOOTP not enabled
292	4	Net Mask	0	Net mask. Valid only if BOOTP not enabled
296	4	Tftp server IP	0	TFTP server IP. Valid only if BOOTP not enabled
BOOTP send and receive Information				
300	20	Vendor String	"TI K3 Bootp Boot"	BOOTP request vendor string. Valid only if BOOTP not enabled
320	9	Client ID	1-mac-address-0	Client ID. See <a href="#">RFC1700</a>
329	1	ID len	7	Client ID length
330	45	Debug array	SOC ID up to size available	Debug array output
375	1	Debug len	Varies	The number of valid bytes in debug array
376	4	Next hop	0	Next hop IP address. Valid only if bootp not enabled
380	4	Default Route	0	IP default route IP address. Valid only if bootp not enabled
384	128	Boot filename	0	Boot filename. Valid only if bootp not enabled

#### 4.4.8 USB Boot Parameter Table

Table 4-66 shows the boot parameter table for USB boot. Must be preceded with the common boot parameters described in Table 4-53.

**Table 4-66. USB Boot Parameter Table**

Byte Offset	Size (bytes)	Name	Default Value	Description
256	4	Port	From pins	Physical port number. Always set to 0.
260	4	Reseved	0	Reseved
264	4	Base address 0	From pins (port)	Base address of USB subsystem (CMN)
268	4	Base address 1	From pins (port)	Base address of controller
272	4	phyBaseAddress	From pins (port)	Base address of USB PHY module
276	4	modRefClkKHz	varies	Module reference clock, in kHz
280	4	Vendor ID	0x0451	USB vendor ID. Read from control registers.
284	4	Product ID	0x6163	USB product ID. Read from control registers.
288	4	String Table addr	0x4182BCA8	Pointer to the string table in RAM
292	2	Vendor String offset	0	Offset to vendor string in string table
298	2	Prod string offset	32	Offset to product string in string table
300	2	Serial num string offset	64	Offset to serial number string in string table
302	2	Timeout	5000	USB timeout in milliseconds
304	2	Mode	1	1 = DFU ≠1 = Reserved
305	1	Lane reverse	0	Host mode only. Boot file name in Unicode characters
306	2	Reseved	0	Reseved
308	4	Block size	4096	DFU block size

#### 4.4.9 MMCSD Boot Parameter Table

Table 4-67 shows the boot parameter table for eMMC, MMC or SD card boot. Must be preceded with the common boot parameters described in Table 4-53.

**Table 4-67. MMCSD Boot Parameter Table**

Byte Offset	Size (bytes)	Name	Default Value	Description
256	1	Port	From Pins	Physical port number
257	1	eMMC	From Pins	0 = SD/MMC, else eMMC
258	1	bootAck	1 for eMMC 0 for SD/MMC	If 1 and in eMMC mode the controller expects a boot ack from the eMMC
259	1	Media	1 for eMMC 0 for SD/MMC	0 = auto detect card type 1 = MMC 2 = SD
260	1	busWidth	8 for eMMC From Pins for SD/MMC	Number of data pins (1, 4, or 8). If 0, max supported pins of the port are used.
261	1	bootAlt	1 for eMMC 0 for SD/MMC	If 1 and in eMMC mode, the controller uses the alt boot method (CMD1 with arg 0xFFFF_FFFA) to initiate data transfer
262	1	sigVolt	0x18 for port0 0x33 for port1	Sets the signal voltage for the controller. 0x18 = 1.8V 0x33 = 3.3V
263	1	Reserved	0	Reserved
264	4	Max Bus Freq	0	Max bus frequency in kHz. If 0, the value is determined by reading the CSD register from the card (still maxes out at 25 MHz)

**Table 4-67. MMCSD Boot Parameter Table (continued)**

Byte Offset	Size (bytes)	Name	Default Value	Description
268	4	refClkKHz	0	Module reference clock frequency, in kHz. If 0, the ROM code computes the value.
272	4	respTimeout	40000	The timeout period on initial card read, in milli-seconds, Q3 number format
276	128	Filename	"\tiboot3.bin"	For SD/MMC, boot filename in 16-bit Unicode characters (max 64)
404	4	Mode	0x1144D091	0x1144D091 = file system boot 0x1144C180 = raw image boot
408	4	CardIsInit	0	0 = card not yet initialized 1 = card has been initialized
412	4	Rsvd	0	Reserved
416	4	RawIndex	0	Current active read offset (0 or 1)
420	4	Rsvd	0	Reserved
424	4	Raw Offset 0	0x000000	Raw read offset
428	4	Raw Offset 1	0x400000	Backup raw read offset
432	4	Rsvd	0	Reserved
436	4	Rsvd	0	Reserved

#### 4.4.10 UART Boot Parameter Table

Table 4-68 shows the boot parameter table for UART boot. Must be preceded with the common boot parameters described in Table 4-53.

**Table 4-68. UART Boot Parameter Table**

Byte Offset	Size (bytes)	Name	Default Value	Description
256	1	Magic	0x49	Required Magic value
257	1	Protocol	63 (0x3F)	Specifies the transfer protocol = XMODEM
258	2	Reserved	0	Reserved
260	1	Max error Count	10	Error count resulting in boot abort
261	1	Ack timeout	3	Timeout in seconds on ack
262	1	Char timeout	20	Inter-character timeout in milliseconds
263	1	Reserved	0	Reserved
264	4	Port	From Pins	Physical port number
268	4	Mod Ref Clk	48000	Module reference clock, in kHz
272	4	Data Rate	115200	Baud rate (bps)
276	1	Parity	0	0=none, 1=odd, 2=even
277	1	Data bits	8	Only 8 data bit width is supported
278	1	Stop bits	2	Stop bits in Q7.1 format (2 = 1 stop bit)
279	1	Flow Control	0	0=none, 1= RTS/CTS
280	1	Over sample	16	Only 16× and 13× oversample are supported
281	1	Magic 2	0xB7	Required magic value

## 4.5 Boot Image Format

### 4.5.1 Overall Structure

The boot image consists of an X.509 Certificate, followed immediately by a boot image blob.

X.509 Certificate (Variable Size) (Optional)
Boot Image Blob (Variable Size)

### 4.5.2 X.509 Certificate

The X.509 certificate is described in [RFC5280](#). Section 4.1 of the specification describes the format.

The X.509 fields relevant to the public boot (taken from RFC5280) are shown below.

```

Certificate ::= SEQUENCE {
    tbsCertificate      TBSCertificate,
    signatureAlgorithm  AlgorithmIdentifier,
    signatureValue      BIT STRING }
TBSCertificate ::= SEQUENCE {
    version             [0] EXPLICIT Version DEFAULT v1,
    serialNumber        CertificateSerialNumber,
    signature           AlgorithmIdentifier,
    issuer              Name,
    validity            Validity,
    subject             Name,
    subjectPublicKeyInfo SubjectPublicKeyInfo,
    issuerUniqueID      [1] IMPLICIT UniqueIdentifier OPTIONAL,
                        -- If present, version MUST be v2 or v3
    subjectUniqueID     [2] IMPLICIT UniqueIdentifier OPTIONAL,
                        -- If present, version MUST be v2 or v3
    extensions          [3] EXPLICIT Extensions OPTIONAL
                        -- If present, version MUST be v3
}
Extensions ::= SEQUENCE SIZE (1..MAX) OF Extension
Extension ::= SEQUENCE {
    extnID              OBJECT IDENTIFIER,
    critical            BOOLEAN DEFAULT FALSE,
    extnValue           OCTET STRING
                        -- contains the DER encoding of an ASN.1 value
                        -- corresponding to the extension type identified
                        -- by extnID
}

```

In general, an X.509 certificate contains a public key which has been signed by a private key. The public ROM code does not directly use the keys. In non-secure devices, the public key value is in general a don't care condition. The exception is certificates containing a degenerate RSA public key. GP devices with a degenerate RSA key allow for integrity checking of most (but not all) of the certificate.

The public MCU ROM only needs to extract some of information from the X.509 formatted structure:

- The total size of the X.509 Certificate
- The total size of the boot image

The total size of the X.509 Certificate is determined by reading the length of the sequence containing the certificate. The length of the image is determined by parsing the certificate to find the extension field which holds the image length.

The ROM defines several extensions that are used only by TI for boot. These are placed in the extensions field of the TBS certificate.

### 4.5.3 Organizational Identifier (OID)

OID (organizational identifier) values are represented as a tree structure. TI has the following node registered:

1.3.6.1.4.1.294: iso(1), identified-organization(3), dod(6), internet(1), private(4), enterprise(1), Texas Instruments(294)

ROM code adds the following branch after *Texas Instruments: device-boot(1)*. The OID values shown in this section are leaves off the device boot branch.

#### 4.5.4 X.509 Extensions Specific to Boot

These values are not defined in any standard, but created by TI for boot.

##### 4.5.4.1 Boot Info (OID 1.3.6.1.4.1.294.1.1)

This extension must be present on all boot images. It is from this extension that the image length is extracted.

```
bootInfo ::= SEQUENCE {
    cert_type:  INTEGER,      -- identifies the certificate type
    boot_core:  INTEGER,      -- identifies the boot core
    core_opts:  INTEGER,      -- 32 or 64 bit boot core target
    load_addr:  OCTET STRING, -- Global address image destination
    image_size: INTEGER,      -- Image size in bytes
}
```

**Table 4-69. Certificate Type Values**

Value	Description
0x0000_0001	Primary boot image
0x0000_0002	Firmware image

**Table 4-70. Boot Core Values**

Value	Description
0x00	Firmware (DMSC) image
0x08	DMSC certificate
0x10	MCU image
0x20	Reserved

**Table 4-71. Core Options Bit Fields**

31	2	1	0
Reserved		Split	Mode

**Table 4-72. Core Options Field Description**

Bits	Field	Value	Description
1	Split	0	Dual MCU set to lockstep (two cores in lockstep)
		1	Dual MCU set to split mode (two independent cores)
0	Mode	0	MCU starts execution in Arm® mode
		1	MCU starts execution in Thumb® mode

##### 4.5.4.2 Image Integrity (OID 1.3.6.1.4.1.294.1.2)

```
imageIntegrity ::= SEQUENCE {
    sha_type:  OID,      -- Identifies the SHA type
    hash:      OCTET STRING -- The SHA of the boot image
}
```

#### 4.5.5 Generating X.509 Certificates

X.509 Certificates are generated using OpenSSL and a configuration script to supply values in the extension fields.

#### 4.5.5.1 Key Generation

The SBL must always be signed with a given OpenSSL key - Secure and GP devices. Secure must have encryption and authentication, GP device must only have authentication. The key used for authentication can be random or specific. If a random key is generated and SBL is signed with this, the ROM will copy the SBL image for authentication using memcpy. With this key, ROM code will be directed to use DMA to load the SBL for authentication which saves boot time.

##### 4.5.5.1.1 Degenerate RSA Keys

Degenerate RSA keys are valid RSA keys with the private exponent set to 1. This results in the signature field being equal to the digest, since in RSA:

$$\text{Signature} = \text{digest}^{\text{privExp}} \bmod n^{\text{privExp}} \bmod n^{\text{privExp}} \quad (1)$$

Where  $n$  is the key size. Since the hash used is SHA-512 and the signature is an ASN.1 sequence containing the OID defining which has been used as well as the hash value, the degenerate RSA must have a value of  $n$  greater than the maximum digest size. Typically 1024-bit is chosen.

The following sequence is used to generate degenerate RSA keys:

1. Create a random RSA key:

```
openssl genrsa -out key.pem 1024
```

2. Convert to text:

```
openssl rsa -in key.pem -text -noout > key.txt
```

3. Create an asn1 template for the degenerate key called degenerateKey.txt. Simply copy the values for modulus, prime (listed as  $p$  in key.txt), prime 2 (listed as  $q$ ), and coefficient (listed as  $\text{coeff}$ ). Set the public and private key exponents to 1, as well as the values for  $e1$  and  $e2$ . See the example below.
4. Convert the template to DER:

```
openssl asn1parse -genconf degenerateKey.txt -out degenerateKey.der
```

5. Sanity check the key:

```
openssl rsa -in degenerateKey.der -inform der -text -check
```

6. If there are no errors create the degenerate key pem file:

```
openssl rsa -in degenerateKey.der -inform der -outform pem -out degenerateKey.pem
```

An example degenerateKey.txt file is shown.

```
asn1=SEQUENCE:rsa_key
[rsa_key]
version=INTEGER:0
modulus=INTEGER:<copied from key.txt>
pubExp=INTEGER:1
privExp=INTEGER:1
p=INTEGER:<copied from key.txt>
q=INTEGER:<copied from key.txt>
e1=INTEGER:1
e2=INTEGER:1
coeff=INTEGER:<copied from key.txt>
```

Note that when copying the multi-byte fields from key.txt it is necessary to remove the colons, concatenate the lines and add a preceding 0x.



### 4.5.5.2 Configuration Script

An example openssl configuration script is shown below. Not all extensions are required, but all possible are shown.

```
[ req ]
distinguished_name = req_distinguished_name
x509_extensions = v3_ca
prompt = no
dirstring_type = nobmp
[ req_distinguished_name ]
C = GB
ST = HI
L = Boston
O = Texas Instruments., Inc.
OU = DSP
CN = Bob
emailAddress = Bob@hou.ti.com
[ v3_ca ]
basicConstraints = CA:true
1.3.6.1.4.1.294.1.1 = ASN1:SEQUENCE:boot_seq
1.3.6.1.4.1.294.1.2 = ASN1:SEQUENCE:image_integrity
1.3.6.1.4.1.294.1.3 = ASN1:SEQUENCE:swrv
1.3.6.1.4.1.294.1.4 = ASN1:SEQUENCE:encryption
1.3.6.1.4.1.294.1.5 = ASN1:SEQUENCE:key_derivation
1.3.6.1.4.1.294.1.7 = ANSI:SEQUENCE:pllControl
1.3.6.1.4.1.294.1.8 = ANSI:SEQUENCE:debug
[ boot_seq ]
certType = INTEGER:1
bootCore = INTEGER:16
bootArchWidth = INTEGER:32
destAddr = FORMAT:HEX,OCT:bc934b00
imageSize = INTEGER:0x00004860
[ image_integrity ]
shaType = OID:1.3.14.3.2.26
shaValue = FORMAT:HEX,OCT:4cf4d59ef77b5d9ab28d2ceb3c9fe83cb52ae6d2
[ swrv ]
rollback = INTEGER:0x00010001
[ encryption ]
Iv = FORMAT:HEX,OCT:00112233445566778899aabbccddeeff
Rstring = FORMAT:HEX,OCT:00112233445566778899aabbccddeeff101112131415161718191a1b1c1d1e1f
Icount = INTEGER:1
Salt = FORMAT:HEX,OCT:00112233445566778899aabbccddeeff
[ pllControl ]
pll0_num = INTEGER:0
pll0_cfg = FORMAT:HEX,OCT:00345678900
pll1_num = INTEGER:1
pll0_cfg = FORMAT:HEX,OCT:00345678900
[ debug ]
uid = FORMAT:HEX,OCT:00345678900
type = INTEGER:1
dbgE = INTEGER:0
secDbgEn = INTEGER:0
```

The certificate is then generated using the following openssl command:

```
openssl req -new -x509 -key <private_key_pem_file> -nodes -out <output_x.509_pem_file> -config
<config_file> -sha512
```

If a delegate key is being signed, then add the option -signkey <sign\_key\_pem\_file> to the command above.

### 4.5.6 Image Data

The image data (blob) is considered simply as a byte stream. On devices that are multiple bytes wide (for example, PCIe) the image must be formatted so that all multi-byte fields match the endianness of the device. The MCU will always run in little endian mode.

## 4.6 Boot Modes

### 4.6.1 I2C Bootloader Operation

#### 4.6.1.1 I2C Initialization Process

In the I2C boot mode, the ROM Code configures the MCU\_I2C0 peripheral.

MCU\_I2C0 can operate either in master mode or in slave mode. In the master mode, the boot master drives the I2C slave device where the image is stored. In the slave mode, the data transfer is driven by an external I2C master device. The master device is usually another SoC device or a FPGA.

A detailed summary of the I2C boot parameter table and the BOOTMODE pins definitions are listed in [Section 4.4.4, I2C Boot Parameter Table](#) and [Section 4.3.8, I2C Boot Device Configuration](#).

##### 4.6.1.1.1 Block Size

ROM code will read 0x800 bytes before processing the data. The last block must be padded to the next 0x800 byte boundary and it must be padded with zeros.

##### 4.6.1.1.2

The boot code does not support byte address to bus address wrapping. So the maximum image size that can be access is 64 kbytes. What this means is that if the read address is 0xFF00, the read size is 0x200 and the I2C bus address is 0x50, then 0x100 bytes will be read from 0xFF00 at bus address 0x50, followed by 0x100 bytes from 0x0000 at bus address 0x50. The bus address does not increment when the address rolls over.

#### 4.6.1.2 I2C Loading Process

##### 4.6.1.2.1 Loading a Boot Image From EEPROM

In this mode, the MCU\_I2C0 peripheral is configured as I2C master.

ROM Code will start reading from the I2C EEPROM at the specified I2C bus address. This read will be done beginning at the specified base address offset. Data will be read in 2-KB chunks. The data will be stored at the address specified in the boot header. It will continue reading image data until a complete image has been read. When the complete image has been read, the ROM Code will branch to the start address of the image.

##### 4.6.1.2.2 Loading Image In Slave Mode

The ROM Code also provides the option to set the I2C to slave mode and connect to external master device, which can then send the image to boot from. The default slave address is set to the value specified in the boot configuration in [Section 4.4.4, I2C Boot Parameter Table](#).

ROM Code will initialize the MCU\_I2C0 hardware and then start polling the specified I2C port for data. As data is read from the I2C port, it will be stored at the address specified in the boot header. The ROM Code will continue polling for and reading data until a complete image has been read. Data will be read and processed in 2-Kbyte chunks. The I2C master should pad the image with zeroes to be a multiple of this size. When the complete image has been read, the ROM Code will branch to the base address of the image.

### 4.6.2 SPI Bootloader Operation

#### 4.6.2.1 SPI Initialization Process

In the SPI boot mode, the ROM Code initializes the OSPI peripheral to SPI bus at 4.156 MHz. The image is read from the EEPROM connected to the corresponding OSPI port and chip-select. The the starting offset to be read can be provided by the user through the BOOTMODE pins ([Section 4.3.7, SPI Boot Device Configuration](#)). A detailed summary of the SPI boot parameter table and the boot configuration definitions are listed in [Section 4.4.5, OSPI/QSPI/SPI Boot Parameter Table](#).

#### 4.6.2.2 SPI Loading Process

The SPI boot loading process is similar to the loading process explained in [Section 4.6.1.2.1, Loading a Boot Image From EEPROM](#), except that the image is read from a NOR flash. The data formats supported are also the same as in [Section 4.6.1.2, I2C Loading Process](#). All the other loading processes that are detailed for I2C master mode work for the SPI boot mode, too.

### 4.6.3 QSPI Bootloader Operation

#### 4.6.3.1 QSPI Initialization Process

In the QSPI boot mode, the ROM Code initializes the OSPI peripheral in DAC mode. The image is read from the QSPI flash connected to the selected chip-select. See [Section 4.3.6, QSPI Boot Device Configuration](#) for OSPI port settings.

A detailed summary of the QSPI boot parameter table and the boot configuration definitions are listed in [Section 4.4.5, QSPI Boot Parameter Table](#).

#### 4.6.3.2 QSPI Loading Process

QSPI boot mode is not executable-in-place (XIP). ROM code first copies boot image into on-chip RAM and then executes it.

### 4.6.4 OSPI Bootloader Operation

#### 4.6.4.1 OSPI Initialization Process

In the OSPI boot mode, the ROM Code initializes the OSPI peripheral in DAC mode. The image is read from the OSPI flash connected to the selected chip-select. See [Section 4.3.4, OSPI Boot Device Configuration](#) for OSPI port settings.

A detailed summary of the OSPI boot parameter table and the boot configuration definitions are listed in [Section 4.4.5, OSPI Boot Parameter Table](#).

#### 4.6.4.2 OSPI Loading Process

OSPI boot mode is not executable-in-place (XIP). ROM code first copies boot image into on-chip RAM and then executes it.

### 4.6.5 PCIe Bootloader Operation

#### 4.6.5.1 PCIe Initialization Process

In the PCIe boot mode, the ROM Code configures the PCIe and SerDes from information it obtains from the boot parameter table, see [Section 4.4.3, PCIe Boot Parameter Table](#)

#### 4.6.5.2 PCIe Loading Process

PCIe boot mode is not executable-in-place (XIP). ROM code first copies boot image into on-chip RAM and then executes it.

### 4.6.6 GPMC NOR Bootloader Operation

#### 4.6.6.1 GPMC NOR Initialization Process

In this mode, the ROM Code configures the GPMC interface based on the configuration parameters specified in the boot parameter table for the GPMC NOR boot mode, see [Section 4.4.6, GPMC NOR Boot Parameter Table](#). The boot parameter structure definition for the GPMC NOR boot mode and the parts of this table that can be configured by the BOOTMODE pins are detailed in [Section 4.3.14, GPMC NOR Boot Device Configuration](#).

The default timings cannot be used since they are based on a 19.2 MHz module clock. The GPMC module is clocked by Main PLL0 (MAIN\_SYSCLK0) divided by 2, which is 250 MHz. This is the FCLK input to the module, yielding a 4-ns period. The following configurations are setup by boot:

**Table 4-73. GPMC NOR Timing Configuration**

Field	Default		ROM Code	
	Value	Time (ns)	Value	Time (ns)
CSONTime	1	4	1	4
CSRdOffTime	16	64	28	112
ADVOnTime	4	16	8	32
ADVRdOffTime	5	20	11	44
OEOOnTime	6	24	13	52

**Table 4-73. GPMC NOR Timing Configuration (continued)**

Field	Default		ROM Code	
	Value	Time (ns)	Value	Time (ns)
OEOffTime	16	64	28	112
RdAccessTime	15	60	27	108
RdCycleTime	17	68	30	120
AdvAadMuxOnTime	1	4	1	4
AdvAadMuxOffTime	2	8	4	16
OeAadMuxOnTime	1	4	1	4
OeAadMuxOffTime	3	12	6	24

#### 4.6.6.2 GPMC NOR Loading Process

GPMC NOR boot mode is not executable-in-place (XIP). ROM code first copies boot image into on-chip RAM and then executes it.

#### 4.6.7 Ethernet Bootloader Operation

##### 4.6.7.1 Ethernet Initialization Process

When the device is set to boot through the Ethernet mode, the ROM Code configures PHY, NAVSS, and the CPSW. These initial configurations, as well as the interface mode (RMII, RGMII), are determined by querying the BOOTMODE pins, see [Section 4.3.10, Ethernet Boot Device Configuration](#), and the boot parameter table for the Ethernet boot, see [Section 4.4.7, Ethernet Boot Parameter Table](#).

The queue manager is setup to route the packets to queues polled by the ROM Code.

##### 4.6.7.2 Ethernet Loading Process

After device configuration, the bootloader performs a standard BOOTP/TFTP boot. The device sends a BOOTP request with its MAC address to a host TFTP server to be assigned an IP from a pool of addresses. After this connection is established, the device is able to receive image data encapsulated in Ethernet packets, see [Section 4.6.7.2.1](#). Data received is stripped of its network headers and the boot data is stored in internal RAM. When the transfer completes and the image is found in good integrity, the ROM Code will branch to the address defined in the Boot Info field of the boot header.

##### 4.6.7.2.1 Ethernet Boot Data Formats

Ethernet boot uses the BOOTP/TFTP protocol for downloads. Only IPv4 is supported.

###### 4.6.7.2.1.1 Limitations

- Received packets cannot be IP fragmented (should not be a problem in most systems since the BOOTP/TFTP packets have fixed lengths of small size)
- Only DIX Ethernet headers are supported.
- 802.3 with SNAP/LLC not supported
- DIX Ethernet with VLAN not supported
- 802.3 with VLAN and SNAP/LLC not supported

###### 4.6.7.2.1.2 BOOTP Request

###### 4.6.7.2.1.2.1 MAC Header (DIX)

- Destination MAC = value from parameter table (default is broadcast)
- Source MAC = value from parameter table (default is e-fuse value)
- Type = IPv4 (0x0800)

###### 4.6.7.2.1.2.2 IPv4 Header

- Version = 4
- Header length = 0
- TOS = 0

- Len = computed during operation
- ID = 0x0001
- Flags + Fragment offset = 0
- TTL = 0x10
- Protocol = UDP (17)
- Header checksum = Computed during operation
- SRC IP = 0.0.0.0
- Dest IP = 255.255.255.255

#### 4.6.7.2.1.2.3 UDP Header

- Source port = BOOTP client (68 decimal)
- Destination Port = BOOTP server (67 decimal)
- Length = computed during operation
- Checksum = computed during operation

#### 4.6.7.2.1.2.4 BOOTP Payload

- Opcode = Request (1)
- HW Type = Ethernet (1)
- HW Addr Len = 6
- Hop Count = 0
- Transaction ID = 1
- Number of seconds = 0
- Client IP = 0.0.0.0
- Your IP = 0.0.0.0
- Server IP = 0.0.0.0
- Gateway IP = 0.0.0.0
- Client HW Address = Device MAC address (from parameter table)
- Server hostname = NULL
- Filename = NULL
- Option 60, Vendor ID string, from parameter table
- Option 61, Client ID string, from parameter table

#### 4.6.7.2.1.2.5 TFTP

There are no ROM code specific TFTP configurations.

#### 4.6.7.3 Ethernet Hand Over Process

Once the ROM Code receives the valid packet, it decodes it to get the image sections and loads them in the appropriate memory location. After the image is loaded and validated, the ROM Code starts the boot image execution.

#### 4.6.8 USB Bootloader Operation

The USB boot mode is used to read the boot image from external USB host .

See [Section 4.3.11, USB Boot Device Configuration](#) and [Section 4.4.8, USB Boot Parameter Table](#) for the available configuration options.

More information about USB DFU protocol can be found at [http://www.usb.org/developers/docs/devclass\\_docs/DFU\\_1.1.pdf](http://www.usb.org/developers/docs/devclass_docs/DFU_1.1.pdf).

After successful enumeration, the ROM Code will start reading from the external host as specified by the BOOTMODE pins. It will continue reading data from the memory and storing it in internal RAM until a complete image has been read. When the complete image has been read and found in good integrity, the ROM Code will branch to the address defined in the Boot Info field of the boot header.

#### 4.6.8.1 USB-Specific Attributes

##### 4.6.8.1.1 DFU Device Mode

- Vendor ID = 0x451
- Product ID = 0x6162
- Device ID = 0

#### 4.6.9 MMCSD Bootloader Operation

Only single data rate with backward compatible interface timing is supported.

See [Section 4.3.9, MMCSD Boot Device Configuration](#) and [Section 4.4.9, MMCSD Boot Parameter Table](#) for the available configuration options.

If the card type is set to autodetect, then an initial discovery phase is performed:

1. Send CMD 0. (GO IDLE, both for MMC and SD)
2. Send CMD 55. If the card responds then the card type is assumed to be SD and the boot attempt fails (the device do not support SD boot). On timeout the code sends CMD 1. If there is a response then the card type is MMC.

The ROM Code will start reading from the MMCSD memory boot sector, or filesystem, as specified by the BOOTMODE pins. It will continue reading data from the memory and storing it in internal RAM until a complete image has been read. When the complete image has been read and found in good integrity, the ROM Code will branch to the address defined in the Boot Info field of the boot header.

#### 4.6.10 UART Bootloader Operation

##### 4.6.10.1 Initialization Process

In the UART boot mode, the selected UART module (port) is the only peripheral configured. The baud rate, data, parity, and stop bits are configured based on the information in the UART boot parameter table. The boot parameter table definitions and the boot configuration values that can be set are in [Section 4.3.13, UART Boot Device Configuration](#) and [Section 4.4.10, UART Boot Parameter Table](#).

Once the ROM Code configures the UART, it sends the UART pings for few seconds, which can be seen in the host. The pings consist of an ASCII capital C character. The UART boot mode supports only the CRC mode of XMODEM and does not support CHECKSUM mode.

##### 4.6.10.2 UART Loading Process

Before the ping from the device stops, load the boot image from the host using the XMODEM protocol.

##### 4.6.10.2.1 UART XMODEM

The XMODEM protocol is used to transfer boot data. Only CRC mode is supported (not checksum), with both 128- and 1024-byte block sizes. The general, format of received frames is shown in [Table 4-74](#) and [Table 4-75](#).

**Table 4-74. XMODEM 1024- and 128-byte Data Frames**

STX	Block Num	Inv Block Num	1024 data bytes			CRC	CRC
SOH	Block Num	Inv Block Num	128 data bytes	CRC	CRC		

**Table 4-75. XMODEM Data Frame Fields**

Field	Value	Description
STX	0x02	The start character for 1024-byte CRC data blocks
SOH	0x01	The start character for 128-byte CRC data block
Block Num	0x01-0xFF – 0x00	The block number. The first block has value 1, and the block number wraps around 0xFF to 0
Inv Block Num	0xFE-0x00	The inverse block number (bit inverse of the block number)
CRC	Calculated	The 16-bit CRC generated from the polynomial 0x1021

The XMODEM protocol is implemented as a half-duplex protocol as shown in [Table 4-76](#).

**Table 4-76. Example of XMODEM Transfer protocol**

Transmitter Sends		Receiver Sends
	←	Ping ('C')
Frame 1	→	
	←	ACK (or NACK)
Frame 2	→	
	←	ACK (or NACK)
EOT	→	
	←	ACK (or NACK)

#### 4.6.10.3 UART Hand-Over Process

Once the complete image has been read and found in good integrity, the ROM Code will branch to the address defined in the Boot Info field of the boot header.



## 4.7 Boot Memory Maps

### 4.7.1 Memory Layout/MPU

Table 4-77 shows an overview of the MPU configuration. In the R5 MPU, higher numbered regions have priority, therefore in Table 4-77, where two regions overlap, the region on the right defines the memory attributes.

**Table 4-77. Memory Layout/MPU**

Memory Address	Regions			
0x0000_0000	Region 0 non-executable full access	Region 10		
0x0000_003F				
...				
0x4101_0000		Region 3 TCM User access	Region 4 TCM Priv access	
0x4101_0FFF				
0x4101_1000				
0x4101_7FFF				
...				
0x4180_0000		Region 1 ROM User access	Region 2 ROM Priv access	
0x4180_7FFF				
0x4180_8000				
0x4183_FFFF				
...				
0x41C0_0000		Region 5 RAM All access non-cacheable		
0x41CB_FFFF				
0x41CC_0000			Region 6 cacheable	
0x41CD_FFFF				
0x41CE_0000			Region 7 cacheable	
0x41CE_FFFF				
0x41CF_0000			Region 8 non-cacheable non-executable	
0x41CF_DFFF				
0x41CF_E000				Region 9 cacheable
0x41CF_FFFF				
...				
0xFFFF_FFFF				

### 4.7.2 Global Memory Addresses Used by ROM Code

The ROM code uses several global memory addresses that are useful for debugging. They are shown in Table 4-78.

**Table 4-78. Global Memory Addresses**

Group	Address	Size (bytes)	Content
Version	0x4182_FF80	64	ROM code version (stored in ROM)
Free MSRAM	0x41C0_0000	768k	Loadable space for SBL
Warning/Error Logs	0x41CF_E000	512	Warning entries in cold boot
Warning/Error Logs	0x41CF_E200	512	Severe entries in cold boot
Warning/Error Logs	0x41CF_E400	256	Critical entries in cold boot
Message Logs	0x41CF_E580	512	Circular message buffer
Message Logs	0x41CF_E780	20	Circular message context log
Warning/Error Logs	0x41CF_E800	512	Warning entries in warm boot
Warning/Error Logs	0x41CF_EA00	512	Severe entries in warm boot

**Table 4-78. Global Memory Addresses (continued)**

Group	Address	Size (bytes)	Content
Warning/Error Logs	0x41CF_EC00	256	Critical entries in warm boot
Warning/Error Logs	0x41CF_ED00	40	Cold boot log context
Warning/Error Logs	0x41CF_ED28	40	Warm boot log context
Warning/Error Logs	0x41CF_ED50	4	Active context pointer (not index) to one of the cold or warm contexts
Trace	0x41CF_ED80	24	Boot trace context
Trace	0x41CF_ED98	1024	Boot trace entry buffers
Parameter Tables	0x41CF_FBFC	4	Parameter table index (currently active table, 0 or 1)
Parameter Tables	0x41CF_FC00	512	Parameter table 0
Parameter Tables	0x41CF_FE00	512	Parameter table 1

The ROM code version information is a structure shown in [Table 4-79](#)

**Table 4-79. ROM Code Version**

Field	Address	Size (bytes)	Value for PG1
Version Number	0x4182_FF80	4	0x00010000 (1.0.0)
Version Date	0x4182_FF84	8	"02/24/19"
Device Name	0x4182_FF8C	12	"j7es"
Commit ID	0x4182_FF98	40	"ff0ea75aac317ea6042724fe775412bca006e697"

#### 4.7.3 Memory Reserved by ROM Code

ROM code reserves all of TCM B (BTCM) and the upper memory address of on-chip RAM (MCU\_MSRAM0). The initial section of MCU\_MSRAM0 is available for the booted image. This is the only memory available for the boot image since neither MSMC nor TCM A (ATCM) is powered up or available for the boot image.

[Table 4-80](#) shows ROM code memory usage.

**Table 4-80. Memory Reserved By ROM Code**

Memory	Address	ROM Code Usage
TCM B (BTCM)	0x0000	Reserved by ROM Code
	0x7FFF	
MCU_MSRAM0	0x41C0_0000	Free for boot image
	0x41CB_FFFF	
	0x41CC_0000	Reserved by ROM Code
	0x41CF_FFFF	



5.1 Control Module (CTRL_MMR).....	328
5.2 Power.....	360
5.3 Reset.....	462
5.4 Clocking.....	472

## 5.1 Control Module (CTRL\_MMR)

The CTRL\_MMR is a module on the device that controls top-level device behavior in various states. It contains registers for configuration, bootstrap signals, I/O pad multiplexing, clock selection, and many others. There are three CTRL\_MMR modules in this device - CTRL\_MMR0 which resides in VD\_CORE, MCU\_CTRL\_MMR0 in VD\_MCU and WKUP\_CTRL\_MMR0 in the VD\_WKUP voltage domain.

There are also two other modules (SEC\_MMR0 and MCU\_SEC\_MMR0) that have registers to control the configuration of the MAIN domain R5 and C66 processors and the MCU domain processors.

The following sections describe all these modules.

### 5.1.1 WKUP\_CTRL\_MMR0

The following sections describe the WKUP\_CTRL\_MMR0 module.

#### 5.1.1.1 WKUP\_CTRL\_MMR0 Overview

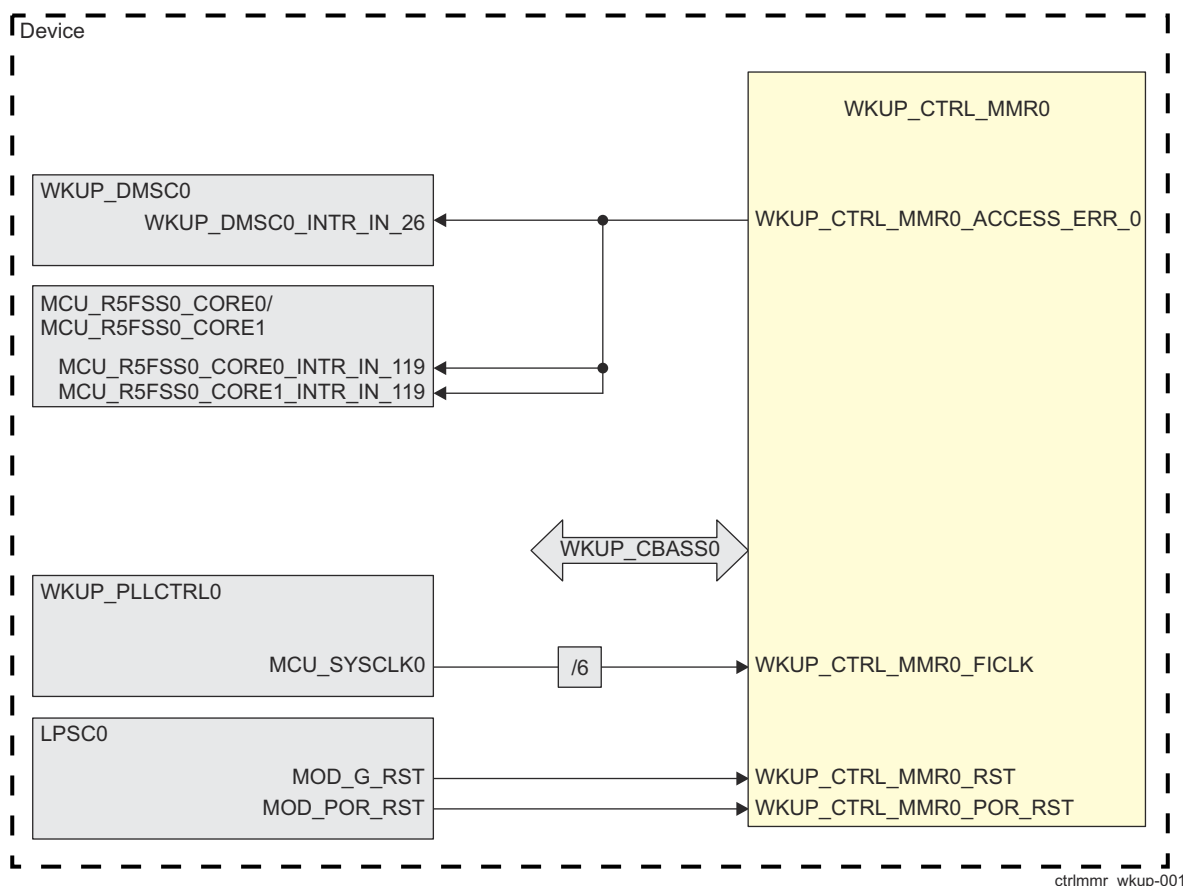
The WKUP\_CTRL\_MMR0 module has registers associated mainly with:

- [Device pad configuration](#)
- [Kick protection \(register write protection\)](#)
- [WKUP\\_CTRL\\_MMR0 module interrupts](#)
- [Clock selection](#)
- [Device features](#)
- [POK Modules](#)
- [Reset related registers](#)
- [PRG Modules](#)
- [Voltage glitch detectors](#)
- [I/O debounce control](#)
- and a few other registers and register groups.

### 5.1.1.2 WKUP\_CTRL\_MMR0 Integration

This section describes WKUP\_CTRL\_MMR0 integration in the device, including information about clocks, resets, and hardware requests.

A single WKUP\_CTRL\_MMR0 module is integrated in the device WKUP domain. [Figure 5-1](#) shows the integration of WKUP\_CTRL\_MMR0.



**Figure 5-1. WKUP\_CTRL\_MMR0 Integration**

[Table 5-1](#) through [Table 5-3](#) summarize the integration of the WKUP\_CTRL\_MMR0 module in the device WKUP domain.

**Table 5-1. WKUP\_CTRL\_MMR0 Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
WKUP_CTRL_MMR0	WKUP_PSC0	PD0	LPSC0	WKUP_CBASS0

**Table 5-2. WKUP\_CTRL\_MMR0 Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
WKUP_CTRL_MMR0	WKUP_CTRL_MMR0_FICLK	MCU_SYSCCLK0/6	WKUP_PLLECTRLO	Functional and interface clock for the WKUP_CTRL_MMR0 module with frequency equal to MCU_SYSCCLK0 divided by 6.
Resets				

**Table 5-2. WKUP\_CTRL\_MMR0 Clocks and Resets (continued)**

Module Instance	Module Reset Input	Source Reset Signal	Source	Description
WKUP_CTRL_MMR0	WKUP_CTRL_MMR0_RST	MOD_G_RST	LPSC0	Module level main reset
	WKUP_CTRL_MMR0_POR_RST	MOD_POR_RST	LPSC0	Module power-on reset

**Table 5-3. WKUP\_CTRL\_MMR0 Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
WKUP_CTRL_MM R0	WKUP_CTRL_MMR0_ACCESS_ ERR_0	WKUP_DMSC0_INTR_IN_26	WKUP_DMSC0	Interrupt indicating protection, addressing, lock violation.	Level
		MCU_R5FSS0_CORE0_INTR _IN_119	MCU_R5FSS0_COR E0		
		MCU_R5FSS0_CORE1_INTR _IN_119	MCU_R5FSS0_COR E1		
DMA Events					
Module Instance	Module DMA Event	Destination DMA Event Input	Destination	Description	Type
WKUP_CTRL_MM - R0	-	-	-	-	-

### Note

For more information about WKUP\_CTRL\_MMR0\_ACCESS\_ERR\_0, see [Section 5.1.1.3.1.3](#).

For more information on the interconnects, see [Chapter 3, System Interconnect](#).

For more information on the power, reset and clock management, see the corresponding sections within [Chapter 5, Device Configuration](#).

For more information on the device interrupt controllers, see [Section 9.2, Interrupt Controllers](#).



### 5.1.1.3 WKUP\_CTRL\_MMR0 Functional Description

#### 5.1.1.3.1 Description for WKUP\_CTRL\_MMR0 Register Types

The following sub-sections provide summary and description for most of the registers which are part of the WKUP\_CTRL\_MMR0 module memory space. The rest of the registers are described in the corresponding chapters where the functionality associated with particular WKUP\_CTRL\_MMR0 register is covered in more detail.

##### 5.1.1.3.1.1 Pad Configuration Registers

The pad multiplexing scheme is same as described in [Section 5.1.3.3.1.1](#). For more details, refer to that section. The only difference is that the CTRLMMR\_WKUP\_PADCONFIG0 to CTRLMMR\_WKUP\_PADCONFIG93 registers control the signal multiplexing of modules in the device MCU and WKUP domains.

##### 5.1.1.3.1.2 Kick Protection Registers

The functionality of the WKUP\_CTRL\_MMR0 kick protection registers is same as described in [Section 5.1.3.3.1.2](#). For more details, refer to that section.

[Table 5-4](#) summarizes the WKUP\_CTRL\_MMR0 kick protection registers.

**Table 5-4. WKUP\_CTRL\_MMR0 Partition Unlock Values**

Register	Partition <sup>(1)</sup>	Unlock Value	Offset Range
CTRLMMR_WKUP_LOCK0_KICK0	Partition 0	68EF 3490h	0000h to 1FFFh
CTRLMMR_WKUP_LOCK0_KICK1	Partition 0	D172 BC5Ah	
CTRLMMR_WKUP_LOCK1_KICK0	Partition 1	68EF 3490h	4000h to 5FFFh
CTRLMMR_WKUP_LOCK1_KICK1	Partition 1	D172 BC5Ah	
CTRLMMR_WKUP_LOCK2_KICK0	Partition 2	68EF 3490h	8000h to 9FFFh
CTRLMMR_WKUP_LOCK2_KICK1	Partition 2	D172 BC5Ah	
CTRLMMR_WKUP_LOCK3_KICK0	Partition 3	68EF 3490h	C000h to DFFFh
CTRLMMR_WKUP_LOCK3_KICK1	Partition 3	D172 BC5Ah	
CTRLMMR_WKUP_LOCK6_KICK0	Partition 6	68EF 3490h	1 8000h to 1 9FFFh
CTRLMMR_WKUP_LOCK6_KICK1	Partition 6	D172 BC5Ah	
CTRLMMR_WKUP_LOCK7_KICK0	Partition 7	68EF 3490h	1 C000h to 1 DFFFh
CTRLMMR_WKUP_LOCK7_KICK1	Partition 7	D172 BC5Ah	

(1) Not listed partitions are reserved and not used.

##### 5.1.1.3.1.3 WKUP\_CTRL\_MMR0 Module Interrupts

The WKUP\_CTRL\_MMR0 module has one interrupt request - the WKUP\_CTRL\_MMR0\_ACCESS\_ERR\_0. The interrupt behavior is same as described in [Section 5.1.3.3.1.3](#). For more details, refer to that section. When reading replace the corresponding registers with the following:

- CTRLMMR\_WKUP\_INTR\_RAW\_STAT
- CTRLMMR\_WKUP\_INTR\_STAT\_CLR
- CTRLMMR\_WKUP\_INTR\_EN\_SET
- CTRLMMR\_WKUP\_INTR\_EN\_CLR
- CTRLMMR\_WKUP\_FAULT\_ADDR
- CTRLMMR\_WKUP\_FAULT\_TYPE
- CTRLMMR\_WKUP\_FAULT\_ATTR
- CTRLMMR\_WKUP\_FAULT\_CLR

#### 5.1.1.3.1.4 Clock Selection Registers

There are registers within the WKUP\_CTRL\_MMR0 module address space that are used to select clocks for the MAIN PLLs and several other modules. [Table 5-5](#) summarizes these registers. Related information can also be found in *Clocking*.

**Table 5-5. Summary of the WKUP\_CTRL\_MMR0 Clock Selection Registers**

Register Name	Register Name
CTRLMMR_WKUP_MCU_PLL_CLKSEL	CTRLMMR_WKUP_MAIN_PLL13_CLKSEL
CTRLMMR_WKUP_PER_CLKSEL	CTRLMMR_WKUP_MAIN_PLL14_CLKSEL
CTRLMMR_WKUP_USART_CLKSEL	CTRLMMR_WKUP_MAIN_PLL15_CLKSEL
CTRLMMR_WKUP_GPIO_CLKSEL	CTRLMMR_WKUP_MAIN_PLL16_CLKSEL
CTRLMMR_WKUP_MAIN_PLL0_CLKSEL	CTRLMMR_WKUP_MAIN_PLL17_CLKSEL
CTRLMMR_WKUP_MAIN_PLL1_CLKSEL	CTRLMMR_WKUP_MAIN_PLL18_CLKSEL
CTRLMMR_WKUP_MAIN_PLL2_CLKSEL	CTRLMMR_WKUP_MAIN_PLL19_CLKSEL
CTRLMMR_WKUP_MAIN_PLL3_CLKSEL	CTRLMMR_WKUP_MAIN_PLL23_CLKSEL
CTRLMMR_WKUP_MAIN_PLL4_CLKSEL	CTRLMMR_WKUP_MAIN_PLL24_CLKSEL
CTRLMMR_WKUP_MAIN_PLL5_CLKSEL	CTRLMMR_WKUP_MAIN_PLL25_CLKSEL
CTRLMMR_WKUP_MAIN_PLL6_CLKSEL	CTRLMMR_WKUP_MAIN_SYCLK_CTRL
CTRLMMR_WKUP_MAIN_PLL7_CLKSEL	CTRLMMR_WKUP_MCU_SPI0_CLKSEL
CTRLMMR_WKUP_MAIN_PLL8_CLKSEL	CTRLMMR_WKUP_MCU_SPI1_CLKSEL
CTRLMMR_WKUP_MAIN_PLL12_CLKSEL	

#### 5.1.1.3.1.5 Device Feature Registers

The following registers within the WKUP\_CTRL\_MMR0 module address space provide status information regarding specific device features:

- CTRLMMR\_WKUP\_DEVICE\_FEATURE0
- CTRLMMR\_WKUP\_DEVICE\_FEATURE1
- CTRLMMR\_WKUP\_DEVICE\_FEATURE2
- CTRLMMR\_WKUP\_DEVICE\_FEATURE3
- CTRLMMR\_WKUP\_DEVICE\_FEATURE4
- CTRLMMR\_WKUP\_DEVICE\_FEATURE5
- CTRLMMR\_WKUP\_DEVICE\_FEATURE6

#### 5.1.1.3.1.6 POK Module Registers

[Table 5-6](#) summarizes the registers associated with the device POK modules.

**Table 5-6. Summary of the POK Registers**

Register Name	Register Name
CTRLMMR_WKUP_POK_VDDA_PMIC_IN_CTRL	CTRLMMR_WKUP_POK_VDD_CORE_UV_CTRL
CTRLMMR_WKUP_POK_VDDSHV_WKUP_GEN_UV_CTRL	CTRLMMR_WKUP_POK_VDD_CPU_UV_CTRL
CTRLMMR_WKUP_POK_VDDR_MCU_UV_CTRL	CTRLMMR_WKUP_POK_VMON_EXT_UV_CTRL
CTRLMMR_WKUP_POK_VDD_MCU_OV_CTRL	CTRLMMR_WKUP_POK_VDDR_CORE_UV_CTRL
CTRLMMR_WKUP_POK_VDDSHV_WKUP_GEN_OV_CTRL	CTRLMMR_WKUP_POK_VDD_CORE_OV_CTRL
CTRLMMR_WKUP_POK_VDDR_MCU_OV_CTRL	CTRLMMR_WKUP_POK_VDD_CPU_OV_CTRL
CTRLMMR_WKUP_POR_POKHV_UV_CTRL	CTRLMMR_WKUP_POK_VMON_EXT_OV_CTRL
CTRLMMR_WKUP_POR_POKLV_UV_CTRL	CTRLMMR_WKUP_POK_VDDR_CORE_OV_CTRL
CTRLMMR_WKUP_POR_POKHV_OV_CTRL	

#### 5.1.1.3.1.7 Power and Reset Related Registers

The following are the reset related registers within the WKUP\_CTRL\_MMR0 module address space:

- CTRLMMR\_WKUP\_POR\_CTRL
- CTRLMMR\_WKUP\_POR\_STAT
- CTRLMMR\_WKUP\_RESET\_SRC\_STAT
- CTRLMMR\_WKUP\_MAIN\_POR\_TO\_CTRL
- CTRLMMR\_WKUP\_POR\_RST\_CTRL
- CTRLMMR\_WKUP\_MAIN\_WARM\_RST\_CTRL
- CTRLMMR\_WKUP\_RST\_STAT
- CTRLMMR\_WKUP\_MCU\_WARM\_RST\_CTRL

#### 5.1.1.3.1.8 PRG Related Registers

The following are the PRG related registers within the WKUP\_CTRL\_MMR0 module address space:

- CTRLMMR\_WKUP\_PRG0\_CTRL
- CTRLMMR\_WKUP\_PRG0\_STAT
- CTRLMMR\_WKUP\_PRG1\_CTRL
- CTRLMMR\_WKUP\_PRG1\_STAT
- CTRLMMR\_WKUP\_MAIN\_PRG\_CTRL
- CTRLMMR\_WKUP\_MAIN\_PRG\_STAT

#### 5.1.1.3.1.9 Voltage Glitch Detect Control and Status Registers

There are glitch detector circuits monitoring the voltage of the device voltage domains. WKUP\_CTRL\_MMR0 has registers that provide control and status information associated with these glitch detectors. [Table 5-7](#) summarizes the voltage glitch detect control and status registers.

**Table 5-7. Summary of the Voltage Glitch Detect Registers**

Register Name	Register Name
CTRLMMR_WKUP_VDD_CPU_GLDTC_CTRL	CTRLMMR_WKUP_VDDR_CPU_GLDTC_STAT
CTRLMMR_WKUP_VDD_CORE_GLDTC_CTRL	CTRLMMR_WKUP_VDDR_CORE_GLDTC_STAT
CTRLMMR_WKUP_VDDR_CPU_GLDTC_CTRL	CTRLMMR_WKUP_VDD_MCU_GLDTC_CTRL
CTRLMMR_WKUP_VDDR_CORE_GLDTC_CTRL	CTRLMMR_WKUP_VDDR_MCU_GLDTC_CTRL
CTRLMMR_WKUP_VDD_CPU_GLDTC_STAT	CTRLMMR_WKUP_VDD_MCU_GLDTC_STAT
CTRLMMR_WKUP_VDD_CORE_GLDTC_STAT	CTRLMMR_WKUP_VDDR_MCU_GLDTC_STAT

#### 5.1.1.3.1.10 I/O Debounce Control Registers

Some device pads have debounce logic. The following WKUP\_CTRL\_MMR0 registers are used to configure the debounce period:

- CTRLMMR\_WKUP\_DBOUNCE\_CFG1
- CTRLMMR\_WKUP\_DBOUNCE\_CFG2
- CTRLMMR\_WKUP\_DBOUNCE\_CFG3
- CTRLMMR\_WKUP\_DBOUNCE\_CFG4
- CTRLMMR\_WKUP\_DBOUNCE\_CFG5
- CTRLMMR\_WKUP\_DBOUNCE\_CFG6

The CTRLMMR\_WKUP\_DBOUNCE\_CFG1 register contains the debounce period for pads with PADCONFIGx[13-11] DEBOUNCE\_SEL fields set to 1h. The CTRLMMR\_WKUP\_DBOUNCE\_CFG2 register contains the debounce period for pads with PADCONFIGx[13-11] DEBOUNCE\_SEL fields set to 2h and so on. The CTRLMMR\_WKUP\_DBOUNCE\_CFG6 register contains the debounce period for pads with PADCONFIGx[13-11] DEBOUNCE\_SEL fields set to 6h.

---

**Note**

The debounce logic is not associated with all signals that can be multiplexed on a pad. Only certain signals can use it.

---

For information about the PADCONFIGx registers, see *Pad Configuration Registers*.

[Table 5-8](#) shows the debounce period values to load in the CTRLMMR\_WKUP\_DBOUNCE\_CFGx[5-0] DB\_CFG field.

---

**Note**

The debounce clock selection should happen before the consumer of the debounced signals is enabled as the clock multiplexers for the debounce logic are NOT glitch-free.

---

**Table 5-8. Debounce Period Values**

CTRLMMR_WKUP_DBOUNCE_CF Gx[5-0] DB_CFG Decimal Value	GPIO Case (32.768kHz (1))	EQEP Case (20MHz <sup>(1)</sup> )	CTRLMMR_WKUP_DBOUNCE_CF Gx[5-0] DB_CFG Decimal Value	GPIO Case (20MHz <sup>(1)</sup> )	EQEP Case (250MHz <sup>(1)</sup> )
	Delay[ms]	Delay[μs]		Delay[μs]	Delay[ns]
0	bypassed	bypassed	32	0.05	4
1	1.95	3.2	33	0.1	8
2	2.93	4.8	34	0.15	12
3	3.91	6.4	35	0.2	16
4	4.88	8	36	0.25	20
5	5.86	9.6	37	0.3	24
6	6.84	11.2	38	0.35	28
7	7.81	12.8	39	0.4	32
8	8.79	14.4	40	0.45	36
9	9.77	16	41	0.5	40
10	10.74	17.6	42	0.55	44
11	11.72	19.2	43	0.6	48
12	12.7	20.8	44	0.65	52
13	13.67	22.4	45	0.7	56
14	14.65	24	46	0.75	60
15	15.63	25.6	47	0.8	64
16	16.6	27.2	48	0.85	68
17	17.58	28.8	49	0.9	72
18	18.55	30.4	50	0.95	76
19	19.53	32	51	1	80
20	20.51	33.6	52	1.05	84
21	21.48	35.2	53	1.1	88
22	15.63	25.6	54	25.6	2.048
23	31.25	51.2	55	51.2	4.096
24	46.88	76.8	56	76.8	6.144
25	62.5	102.4	57	102.4	8.192
26	78.13	128	58	128	10.24
27	93.75	153.6	59	153.6	12.288
28	109.38	179.2	60	179.2	14.336
29	125	204.8	61	204.8	16.384
30	140.63	230.4	62	230.4	18.432
31	156.25	256	63	256	20.48

(1) These are the debounce logic clock frequencies.

### 5.1.2 MCU\_CTRL\_MMR0

The following sections describe the MCU\_CTRL\_MMR0 module.

#### 5.1.2.1 MCU\_CTRL\_MMR0 Overview

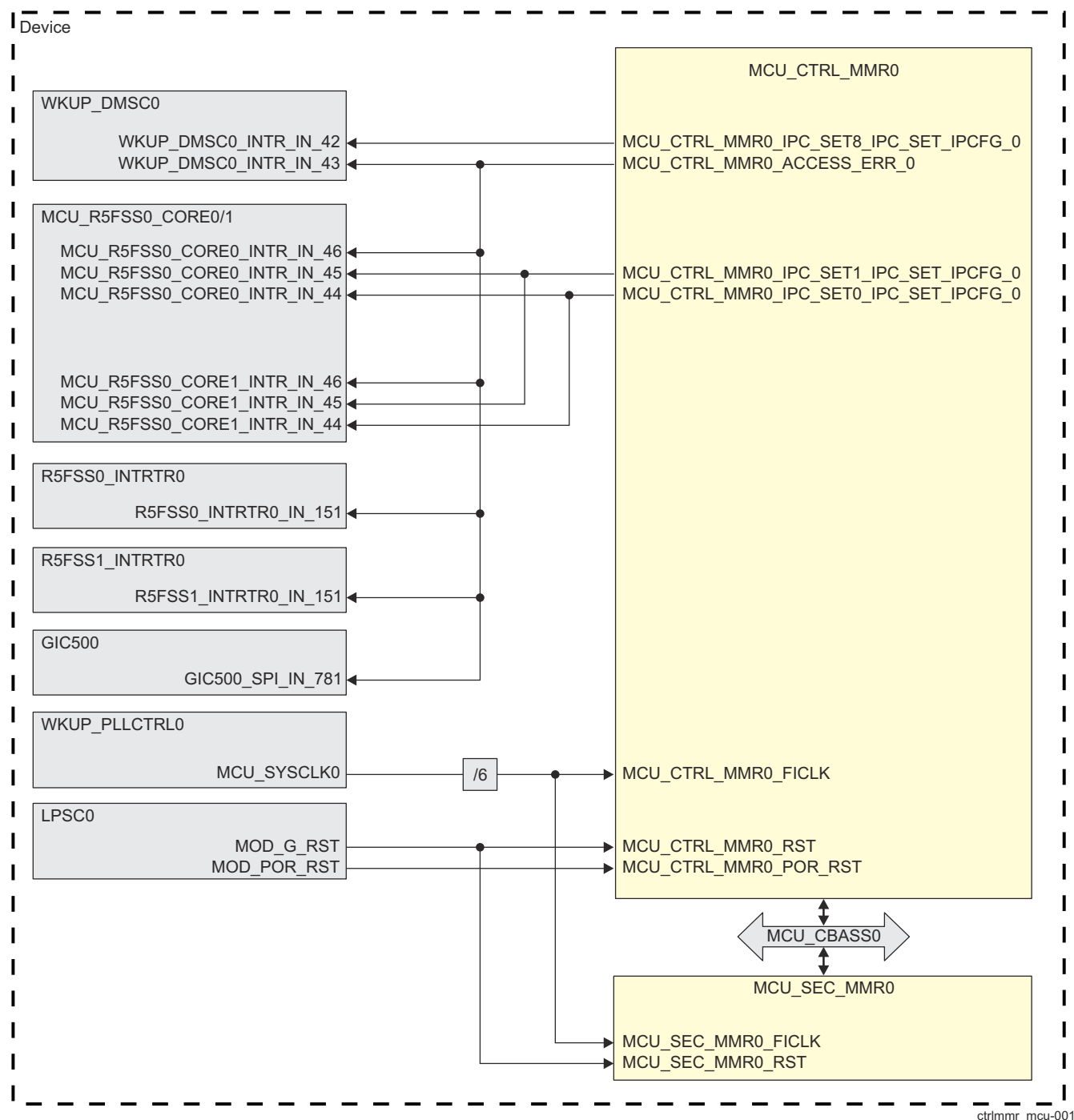
The MCU\_CTRL\_MMR0 module has registers associated mainly with:

- [Kick protection \(register write protection\)](#)
- [MCU\\_CTRL\\_MMR0 module interrupts](#)
- [Inter-processor communication](#)
- [Timer I/O multiplexing](#)
- [Clock muxing and division](#)
- [MCU\\_CPSW0 MAC address](#)
- and a few other registers and register groups.

### 5.1.2.2 MCU\_CTRL\_MMR0 Integration

This section describes MCU\_CTRL\_MMR0 and MCU\_SEC\_MMR0 integration in the device, including information about clocks, resets, and hardware requests.

One MCU\_CTRL\_MMR0 and one MCU\_SEC\_MMR0 modules are integrated in the device MCU domain. [Figure 5-2](#) shows their integration.



**Figure 5-2. MCU\_CTRL\_MMR0 and MCU\_SEC\_MMR0 Integration**



Table 5-9 through Table 5-11 summarize the integration of the MCU\_CTRL\_MMR0 and MCU\_SEC\_MMR0 modules in the device MCU domain.

**Table 5-9. MCU\_CTRL\_MMR0 and MCU\_SEC\_MMR0 Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
MCU_CTRL_MMR0 MCU_SEC_MMR0	WKUP_PSC0	PD0	LPSC0	MCU_CBASS0

**Table 5-10. MCU\_CTRL\_MMR0 and MCU\_SEC\_MMR0 Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
MCU_CTRL_MMR0	MCU_CTRL_MMR0_FICLK	MCU_SYSCCLK0/6	WKUP_PLLCTRL0	Functional and interface clock for the MCU_CTRL_MMR0 module with frequency equal to MCU_SYSCCLK0 divided by 6.
MCU_SEC_MMR0	MCU_SEC_MMR0_FICLK	MCU_SYSCCLK0/6	WKUP_PLLCTRL0	Functional and interface clock for the MCU_SEC_MMR0 module with frequency equal to MCU_SYSCCLK0 divided by 6.

Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
MCU_CTRL_MMR0	MCU_CTRL_MMR0_RST	MOD_G_RST	LPSC0	Module level main reset
	MCU_CTRL_MMR0_POR_RST	MOD_POR_RST	LPSC0	Module power-on reset
MCU_SEC_MMR0	MCU_SEC_MMR0_RST	MOD_G_RST	LPSC0	MCU_SEC_MMR0 module reset

**Table 5-11. MCU\_CTRL\_MMR0 Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCU_CTRL_MM R0	MCU_CTRL_MMR0_ACCES S_ERR_0	GIC500_SPI_IN_781	GIC500	Interrupt indicating protection, addressing, lock violation.	Level
		WKUP_DMSC0_INTR_IN_43	WKUP_DMSC0		
		R5FSS0_INTRTR0_IN_151	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_151	R5FSS1_INTRTR0		
		MCU_R5FSS0_CORE0_INTR_IN_46	MCU_R5FSS0_CORE0		
	MCU_CTRL_MMR0_IPC_SE T0_IPC_SET_IPCFG_0	MCU_R5FSS0_CORE1_INTR_IN_46	MCU_R5FSS0_CORE1	Interrupt generated by writing 1h to CTRLMMR_MCU_IPC_SET0[0].	Level
		MCU_R5FSS0_CORE0_INTR_IN_44	MCU_R5FSS0_CORE0		
	MCU_CTRL_MMR0_IPC_SE T1_IPC_SET_IPCFG_0	MCU_R5FSS0_CORE1_INTR_IN_44	MCU_R5FSS0_CORE1	Interrupt generated by writing 1h to CTRLMMR_MCU_IPC_SET1[0].	Level
		MCU_R5FSS0_CORE0_INTR_IN_45	MCU_R5FSS0_CORE0		
	MCU_CTRL_MMR0_IPC_SE T8_IPC_SET_IPCFG_0	MCU_R5FSS0_CORE1_INTR_IN_45	MCU_R5FSS0_CORE1	Interrupt generated by writing 1h to CTRLMMR_MCU_IPC_SET8[0].	Level

**Table 5-11. MCU\_CTRL\_MMR0 Hardware Requests (continued)**

DMA Events					
Module Instance	Module DMA Event	Destination DMA Event Input	Destination	Description	Type
MCU_CTRL_MM R0	-	-	-	-	-

### Note

For more information about MCU\_CTRL\_MMR0\_ACCESS\_ERR\_0, see [Section 5.1.2.3.1.2](#).

For more information about MCU\_IPC\_INTx, see [Section 5.1.2.3.1.3](#).

### 5.1.2.3 MCU\_CTRL\_MMR0 Functional Description

#### 5.1.2.3.1 Description for MCU\_CTRL\_MMR0 Register Types

The following sub-sections provide summary and description for most of the registers which are part of the MCU\_CTRL\_MMR0 module memory space. The rest of the registers are described in the corresponding chapters where the functionality associated with particular MCU\_CTRL\_MMR0 register is covered in more detail.

##### 5.1.2.3.1.1 Kick Protection Registers

The functionality of the MCU\_CTRL\_MMR0 kick protection registers is same as described in [Section 5.1.3.3.1.2](#). For more details, refer to that section.

[Table 5-12](#) summarizes the MCU\_CTRL\_MMR0 kick protection registers.

**Table 5-12. MCU\_CTRL\_MMR0 Partition Unlock Values**

Register	Partition <sup>(1)</sup>	Unlock Value	Offset Range
CTRLMMR_MCU_LOCK0_KICK0	Partition 0	68EF 3490h	0000h to 1FFFh
CTRLMMR_MCU_LOCK0_KICK1	Partition 0	D172 BC5Ah	
CTRLMMR_MCU_LOCK1_KICK0	Partition 1	68EF 3490h	4000h to 5FFFh
CTRLMMR_MCU_LOCK1_KICK1	Partition 1	D172 BC5Ah	
CTRLMMR_MCU_LOCK2_KICK0	Partition 2	68EF 3490h	8000h to 9FFFh
CTRLMMR_MCU_LOCK2_KICK1	Partition 2	D172 BC5Ah	
CTRLMMR_MCU_LOCK3_KICK0	Partition 3	68EF 3490h	C000h to DFFFh
CTRLMMR_MCU_LOCK3_KICK1	Partition 3	D172 BC5Ah	

(1) Not listed partitions are reserved and not used.

##### 5.1.2.3.1.2 MCU\_CTRL\_MMR0 Module Interrupts

The MCU\_CTRL\_MMR0 module has one interrupt request - the MCU\_CTRL\_MMR0\_ACCESS\_ERR\_0. The interrupt behavior is same as described in [Section 5.1.3.3.1.3](#). For more details, refer to that section. When reading replace the corresponding registers with the following:

- CTRLMMR\_MCU\_INTR\_RAW\_STAT
- CTRLMMR\_MCU\_INTR\_STAT\_CLR
- CTRLMMR\_MCU\_INTR\_EN\_SET
- CTRLMMR\_MCU\_INTR\_EN\_CLR
- CTRLMMR\_MCU\_FAULT\_ADDR
- CTRLMMR\_MCU\_FAULT\_TYPE
- CTRLMMR\_MCU\_FAULT\_ATTR
- CTRLMMR\_MCU\_FAULT\_CLR

##### 5.1.2.3.1.3 Inter-processor Communication Registers

The inter-processor communication (IPC) registers are used for generating interrupts to the device cores. The IPC scheme is same as described in [Section 5.1.3.3.1.4](#). For more details, refer to that section.

[Table 5-13](#) summarizes the IPC registers.

**Table 5-13. Summary of the MCU\_CTRL\_MMR0 IPC Registers**

IPC Register	Writing 1h to bit 0 results in:	Writing 1h to one of the bits [31-4] results in:
CTRLMMR_MCU_IPC_SET0	<ul style="list-style-type: none"> <li>• interrupt generation to MCU R5 Core 0</li> <li>• setting the IPC_SET bit to 1h</li> <li>• setting the corresponding IPC_CLR bit to 1h</li> </ul>	setting that bit and the corresponding IPC_SRC_CLR bit to 1h
CTRLMMR_MCU_IPC_SET1	<ul style="list-style-type: none"> <li>• interrupt generation to MCU R5 Core 1</li> <li>• setting the IPC_SET bit to 1h</li> <li>• setting the corresponding IPC_CLR bit to 1h</li> </ul>	setting that bit and the corresponding IPC_SRC_CLR bit to 1h

**Table 5-13. Summary of the MCU\_CTRL\_MMR0 IPC Registers (continued)**

IPC Register	Writing 1h to bit 0 results in:	Writing 1h to one of the bits [31-4] results in:
CTRLMMR_MCU_IPC_SET8	<ul style="list-style-type: none"> <li>interrupt generation to WKUP_DMSC0</li> <li>setting the IPC_SET bit to 1h</li> <li>setting the corresponding IPC_CLR bit to 1h</li> </ul>	setting that bit and the corresponding IPC_SRC_CLR bit to 1h
CTRLMMR_MCU_IPC_CLR0	<ul style="list-style-type: none"> <li>clearing the interrupt to MCU R5 Core 0</li> <li>clearing the IPC_CLR bit</li> <li>clearing the corresponding IPC_SET bit</li> </ul>	clearing (set to 0h) that bit and the corresponding IPC_SRC_SET bit
CTRLMMR_MCU_IPC_CLR1	<ul style="list-style-type: none"> <li>clearing the interrupt to MCU R5 Core 1</li> <li>clearing the IPC_CLR bit</li> <li>clearing the corresponding IPC_SET bit</li> </ul>	clearing (set to 0h) that bit and the corresponding IPC_SRC_SET bit
CTRLMMR_MCU_IPC_CLR8	<ul style="list-style-type: none"> <li>clearing the interrupt to WKUP_DMSC0</li> <li>clearing the IPC_CLR bit</li> <li>clearing the corresponding IPC_SET bit</li> </ul>	clearing (set to 0h) that bit and the corresponding IPC_SRC_SET bit

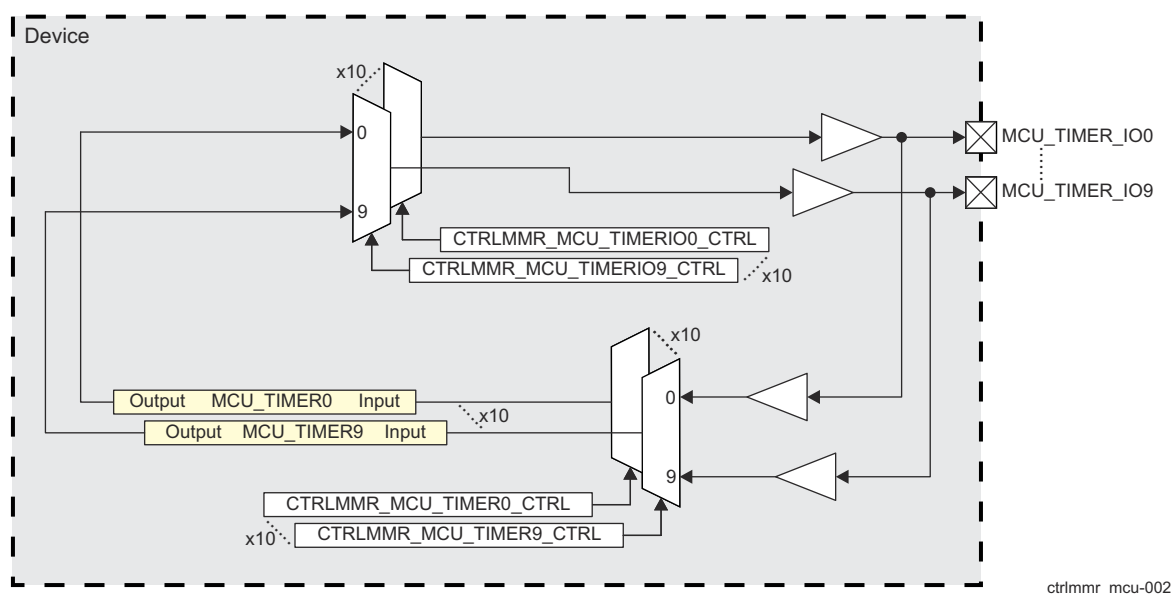
### Note

For latency reasons the IPC registers are not write protected by KICK registers which means that they can be written to without a need for performing unlocking procedure as described in *Kick Protection Registers*.

#### 5.1.2.3.1.4 Timer I/O Muxing Control Registers

Not all timer inputs and outputs are pinned out of the device. Through the MCU timer I/O muxing registers the inputs and outputs of the MCU domain timers can be made available on the MCU\_TIMER\_IO[9:0] device pads.

Each timer input can be configured to be driven by one of the MCU\_TIMER\_IO[9:0] pads through the corresponding bits in the CTRLMMR\_MCU\_TIMER0\_CTRL to CTRLMMR\_MCU\_TIMER9\_CTRL registers. Each of the MCU\_TIMER\_IO[9:0] pads can be configured to be driven by any of the timer outputs through the corresponding bits in the CTRLMMR\_MCU\_TIMERIO0\_CTRL to CTRLMMR\_MCU\_TIMERIO9\_CTRL registers. Figure 5-3 shows the TIMER I/O multiplexing scheme for the MCU domain.


**Figure 5-3. MCU Domain TIMER I/O Multiplexing Scheme**

For more information about each timer, see *Timers*.

#### 5.1.2.3.1.5 Clock Muxing and Division Registers

There are registers within the MCU\_CTRL\_MMR0 module address space dedicated to clock muxing and division for some device modules. [Table 5-14](#) summarizes these registers. Related information can also be found in *Clocking*.

**Table 5-14. Summary of the MCU\_CTRL\_MMR0 Clock Muxing and Division Registers**

Register Name	Register Name	Register Name
CTRLMMR_MCU_CLKOUT0_CTRL	CTRLMMR_MCU_ENET_CLKSEL	CTRLMMR_MCU_TIMER6_CLKSEL
CTRLMMR_MCU_EFUSE_CLKSEL	CTRLMMR_MCU_R5_CORE0_CLKSEL	CTRLMMR_MCU_TIMER7_CLKSEL
CTRLMMR_MCU_MCAN0_CLKSEL	CTRLMMR_MCU_TIMER0_CLKSEL	CTRLMMR_MCU_TIMER8_CLKSEL
CTRLMMR_MCU_MCAN1_CLKSEL	CTRLMMR_MCU_TIMER1_CLKSEL	CTRLMMR_MCU_TIMER9_CLKSEL
CTRLMMR_MCU_OSPI0_CLKSEL	CTRLMMR_MCU_TIMER2_CLKSEL	CTRLMMR_MCU_RTIO_CLKSEL
CTRLMMR_MCU_OSPI1_CLKSEL	CTRLMMR_MCU_TIMER3_CLKSEL	CTRLMMR_MCU_RT11_CLKSEL
CTRLMMR_MCU_ADC0_CLKSEL	CTRLMMR_MCU_TIMER4_CLKSEL	CTRLMMR_MCU_USART_CLKSEL
CTRLMMR_MCU_ADC1_CLKSEL	CTRLMMR_MCU_TIMER5_CLKSEL	

#### 5.1.2.3.1.6 MCU\_CPSW0 MAC Address Registers

Each device has a unique 48-bit MAC address which is stored in the MCU\_CTRL\_MMR0 registers shown in [Table 5-15](#).

**Table 5-15. MCU\_CPSW0 MAC Address Registers**

Bit Field	Description
CTRLMMR_MCU_MAC_ID0[31-0] MACID_LO	Contains the first, second, third and fourth octets of the MCU_CPSW0 MAC address
CTRLMMR_MCU_MAC_ID1[15-0] MACID_HI	Contains the fifth and sixth octets of the MCU_CPSW0 MAC address

### 5.1.3 CTRL\_MMR0

The following sections describe the CTRL\_MMR0 module.

#### 5.1.3.1 CTRL\_MMR0 Overview

The CTRL\_MMR0 module has registers associated mainly with:

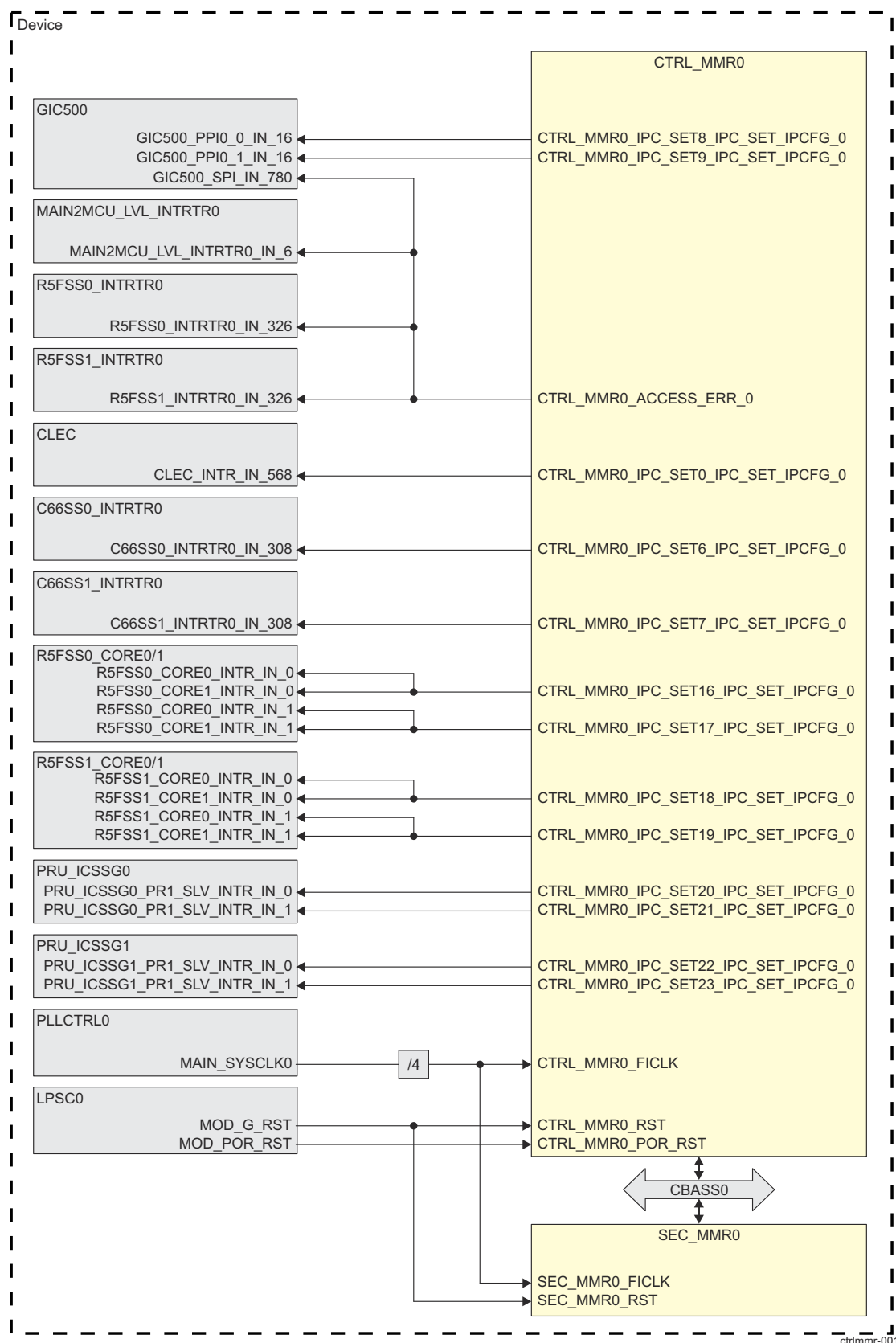
- [Device pad configuration](#)
- [Kick protection \(register write protection\)](#)
- [CTRL\\_MMR0 module interrupts](#)
- [Inter-processor communication](#)
- [Timer I/O multiplexing](#)
- [EHRPWM/EQEP control and status](#)
- [PRU\\_ICSSG control](#)
- [Clock muxing and division](#)
- [Ethernet port operation control](#)
- [PCIe operation control](#)
- [SERDES lane function control](#)
- [DDRSS dynamic frequency change](#)
- and a few other registers and register groups.

### 5.1.3.2 CTRL\_MMR0 Integration

This section describes CTRL\_MMR0 and SEC\_MMR0 integration in the device, including information about clocks, resets, and hardware requests.

One CTRL\_MMR0 and one SEC\_MMR0 modules are integrated in the device MAIN domain. [Figure 5-4](#) shows their integration.





**Figure 5-4. CTRL\_MMR0 and SEC\_MMR0 Integration**

Table 5-16 through Table 5-18 summarize the integration of the CTRL\_MMR0 and SEC\_MMR0 modules in the device MAIN domain.

**Table 5-16. CTRL\_MMR0 and SEC\_MMR0 Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
CTRL_MMR0 SEC_MMR0	PSC0	PD0	LPSC0	CBASS0

**Table 5-17. CTRL\_MMR0 and SEC\_MMR0 Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
CTRL_MMR0	CTRL_MMR0_FICLK	MAIN_SYSCCLK0/4	PLLCTRL0	Functional and interface clock for the CTRL_MMR0 module with frequency equal to MAIN_SYSCCLK0 divided by 4.
SEC_MMR0	SEC_MMR0_FICLK	MAIN_SYSCCLK0/4	PLLCTRL0	Functional and interface clock for the SEC_MMR0 module with frequency equal to MAIN_SYSCCLK0 divided by 4.

Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
CTRL_MMR0	CTRL_MMR0_RST	MOD_G_RST	LPSC0	Module level main reset
	CTRL_MMR0_POR_RST	MOD_POR_RST	LPSC0	Module power-on reset
SEC_MMR0	SEC_MMR0_RST	MOD_G_RST	LPSC0	SEC_MMR0 module reset

**Table 5-18. CTRL\_MMR0 Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
CTRL_MMR0	CTRL_MMR0_ACCESS_ERR_0	GIC500_SPI_IN_780	GIC500	Interrupt indicating protection, addressing, lock violation.	Level
		MAIN2MCU_LVL_INTRTR0_IN_6	MAIN2MCU_LVL_INTRTR0		
		R5FSS0_INTRTR0_IN_326	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_326	R5FSS1_INTRTR0		
	CTRL_MMR0_IPC_SET0_IPC_SE T_IPCFG_0	CLEC_INTR_IN_568	CLEC	Interrupt generated by writing 1h to CTRLMMR_IPC_SET0[0].	Level
	CTRL_MMR0_IPC_SET6_IPC_SE T_IPCFG_0	C66SS0_INTRTR0_IN_308	C66SS0_INTRTR0	Interrupt generated by writing 1h to CTRLMMR_IPC_SET6[0].	Level
	CTRL_MMR0_IPC_SET7_IPC_SE T_IPCFG_0	C66SS1_INTRTR0_IN_308	C66SS1_INTRTR0	Interrupt generated by writing 1h to CTRLMMR_IPC_SET7[0].	Level
	CTRL_MMR0_IPC_SET8_IPC_SE T_IPCFG_0	GIC500_PPIO_0_IN_16	GIC500	Interrupt generated by writing 1h to CTRLMMR_IPC_SET8[0].	Level

**Table 5-18. CTRL\_MMR0 Hardware Requests (continued)**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
	CTRL_MMR0_IPC_SET9_IPC_SE T_IPCFG_0	GIC500_PPIO_1_IN_16	GIC500	Interrupt generated by writing 1h to CTRLMMR_IPC_SET9[0].	Level
	CTRL_MMR0_IPC_SET16_IPC_SE T_IPCFG_0	R5FSS0_CORE0_INTR_IN_0	R5FSS0_CORE0	Interrupt generated by writing 1h to CTRLMMR_IPC_SET16[0].	Level
		R5FSS0_CORE1_INTR_IN_0	R5FSS0_CORE1		
	CTRL_MMR0_IPC_SET17_IPC_SE T_IPCFG_0	R5FSS0_CORE0_INTR_IN_1	R5FSS0_CORE0	Interrupt generated by writing 1h to CTRLMMR_IPC_SET17[0].	Level
		R5FSS0_CORE1_INTR_IN_1	R5FSS0_CORE1		
	CTRL_MMR0_IPC_SET18_IPC_SE T_IPCFG_0	R5FSS1_CORE0_INTR_IN_0	R5FSS1_CORE0	Interrupt generated by writing 1h to CTRLMMR_IPC_SET18[0].	Level
		R5FSS1_CORE1_INTR_IN_0	R5FSS1_CORE1		
	CTRL_MMR0_IPC_SET19_IPC_SE T_IPCFG_0	R5FSS1_CORE0_INTR_IN_1	R5FSS1_CORE0	Interrupt generated by writing 1h to CTRLMMR_IPC_SET19[0].	Level
		R5FSS1_CORE1_INTR_IN_1	R5FSS1_CORE1		
	CTRL_MMR0_IPC_SET20_IPC_SE T_IPCFG_0	PRU_ICSSG0_PR1_SLV_INTR_IN_0	PRU_ICSSG0	Interrupt generated by writing 1h to CTRLMMR_IPC_SET20[0].	Level
	CTRL_MMR0_IPC_SET21_IPC_SE T_IPCFG_0	PRU_ICSSG0_PR1_SLV_INTR_IN_1	PRU_ICSSG0	Interrupt generated by writing 1h to CTRLMMR_IPC_SET21[0].	Level
	CTRL_MMR0_IPC_SET22_IPC_SE T_IPCFG_0	PRU_ICSSG1_PR1_SLV_INTR_IN_0	PRU_ICSSG1	Interrupt generated by writing 1h to CTRLMMR_IPC_SET22[0].	Level
	CTRL_MMR0_IPC_SET23_IPC_SE T_IPCFG_0	PRU_ICSSG1_PR1_SLV_INTR_IN_1	PRU_ICSSG1	Interrupt generated by writing 1h to CTRLMMR_IPC_SET23[0].	Level
DMA Events					
Module Instance	Module DMA Event	Destination DMA Event Input	Destination	Description	Type
CTRL_MMR - 0	-	-	-	-	-

**Note**

For more information about CTRL\_MMR0\_ACCESS\_ERR\_0, see [Section 5.1.3.3.1.3](#).

### 5.1.3.3 CTRL\_MMR0 Functional Description

#### 5.1.3.3.1 Description for CTRL\_MMR0 Register Types

The following sub-sections provide summary and description for most of the registers which are part of the CTRL\_MMR0 module memory space. The rest of the registers are described in the corresponding chapters where the functionality associated with particular CTRL\_MMR0 register is covered in more detail.

##### 5.1.3.3.1.1 Pad Configuration Registers

The pad configuration registers are used to configure most of the device pads. Each pad configuration register (CTRLMMR\_PADCONFIG0 to CTRLMMR\_PADCONFIG172) is associated only with one pad and has bits as described in [Table 5-19](#).

**Table 5-19. Description Of The Pad Configuration Register Bits**

Bit	Field <sup>(1)</sup>	Type	Description
31	LOCK	R/W	Pad configuration register lock bit. 0h - The corresponding pad configuration register is unlocked. Its value can be modified. Further writes are allowed. 1h - The corresponding pad configuration register is locked. Its value can not be modified. Further writes are not allowed.
30	WKUP_EVT	R	Wakeup event status 0h - No wake event on pin 1h - Wake event occurred on pin
29	WKUP_EN	R/W	Wakeup enable 0h - Wakeup operation disabled 1h - Wakeup operation enabled
28-27	RESERVED	R	Reserved
26	DSOUT_VAL	R/W	Specifies the output logic value for a pad when device is in Deep Sleep mode. 0h - Output value is 0 1h - Output value is 1
25	DSOUT_DIS	R/W	Deep Sleep output disable 0h - Output enabled 1h - Output disabled
24	DS_EN	R/W	Deep Sleep override control 0h - I/O keeps its previous state when Deep Sleep mode is active 1h - I/O state is forced to OFF mode value when Deep Sleep mode is active
23	ISO_BYP	R/W	Isolation Bypass 0h - I/O isolation is preserved 1h - I/O isolation is bypassed
22	ISO_OVR	R/W	Isolation Override 0h - I/O isolation is preserved 1h - I/O isolation is overridden
21	TX_DIS	R/W	Disables the driver for a pad 0h - Driver is enabled 1h - Driver is disabled

**Table 5-19. Description Of The Pad Configuration Register Bits (continued)**

Bit	Field <sup>(1)</sup>	Type	Description
20-19	DRV_STR	R/W	<p>Drive Strength Control.</p> <p>Selects the drive strength value for LVCMOS pins. (Does not apply to other pin types). Nominal drive strength is recommended and is the value used for data manual AC timing . Fast or Slow modes can be used where necessary, and should be validated using IBIS simulations to determine the appropriate timing derate from the data manual.</p> <p>0h - Nominal (recommended)  1h - Fast  2h - Slow  3h - Reserved</p>
18	RXACTIVE	R/W	<p>Input enable for a pad</p> <p>0h - Receiver disabled  1h - Receiver enabled</p>
17-16	RESERVED	R	Reserved
15	FORCE_DS_EN	R/W	<p>Enable pad Deep Sleep controls by overriding DMSC gating</p> <p>0h - Deep Sleep pad controls are gated by the DMSC  1h - Deep Sleep pad controls are activated. DMSC gating logic is overridden.</p>
14	ST_EN	R/W	<p>Receiver Schmitt Trigger enable</p> <p>0h - Schmitt trigger input disabled  1h - Schmitt trigger input enabled</p>
13-11	DEBOUNCE_SEL	R/W	<p>Selects the debounce period for a pad. For more information, see <a href="#">Section 5.1.1.3.1.10</a>.</p> <p>0h: Debounce period of 0  1h: Debounce period as specified via the CTRLMMR_WKUP_DBOUNCE_CFG1[5-0] DB_CFG field  2h: Debounce period as specified via the CTRLMMR_WKUP_DBOUNCE_CFG2[5-0] DB_CFG field  3h: Debounce period as specified via the CTRLMMR_WKUP_DBOUNCE_CFG3[5-0] DB_CFG field  4h: Debounce period as specified via the CTRLMMR_WKUP_DBOUNCE_CFG4[5-0] DB_CFG field  5h: Debounce period as specified via the CTRLMMR_WKUP_DBOUNCE_CFG5[5-0] DB_CFG field  6h: Debounce period as specified via the CTRLMMR_WKUP_DBOUNCE_CFG6[5-0] DB_CFG field  7h: Reserved</p>
10-6	RESERVED	R	Reserved
5-4	VGPI0_SEL	R/W	<p>Virtual MAIN_GPIO instance select.</p> <p>These bits select which instance of GPIO is used for this I/O pad. This allows protection between two different processor virtual worlds. These bits have no effect if GPIO mode (MUXMODE = 7h) is not selected for the pad.</p> <p>0h - Implement GPIO in GPIO_0/1 instance  1h - Implement GPIO in GPIO_2/3 instance  2h - Implement GPIO in GPIO_4/5 instance  3h - Implement GPIO in GPIO_6/7 instance</p>

**Table 5-19. Description Of The Pad Configuration Register Bits (continued)**

Bit	Field <sup>(1)</sup>	Type	Description
3-0	MUXMODE	R/W	Selects the desired multiplexing mode for a pad 0h - Mux Mode 0 1h - Mux Mode 1 2h - Mux Mode 2 3h - Mux Mode 3 4h - Mux Mode 4 5h - Mux Mode 5 6h - Mux Mode 6 7h - Mux Mode 7 8h - Mux Mode 8 9h - Mux Mode 9 Ah - Mux Mode 10 Bh - Mux Mode 11 Ch - Mux Mode 12 Dh - Mux Mode 13 Eh - Mux Mode 14 Fh - Mux Mode 15

(1) Some bits may not be present in all PADCONFIGx registers.

Many of the device pads support pad multiplexing. This means that their function can be independently chosen from two or more options. The selection of functions available on each pad is enumerated in table “*Pin Multiplexing*” of the device-specific Datasheet. The desired function is selected via the MUXMODE field of the associated pad configuration register.

The CTRLMMR\_PADCONFIG0 to CTRLMMR\_PADCONFIG172 registers control the signal multiplexing of modules in the device MAIN domain.

#### 5.1.3.3.1.2 Kick Protection Registers

The CTRL\_MMR0 module has a protection mechanism which prevents spurious writes from changing the values of its registers. The LOCKi\_KICK0 and LOCKi\_KICK1 registers are used for this purpose. A write is required first to the LOCKi\_KICK0[31-1] KEY field and then to the LOCKi\_KICK1[31-0] KEY field with exact data values to unlock the protection mechanism. Once released then all registers within Partition “i” having write permissions can be written to. The read only registers are still read only. An indication for unlocked Partition “i” is when the LOCKi\_KICK0[0] UNLOCKED bit is set to 1h. When the protection mechanism is locked (indicated by LOCKi\_KICK0[0] UNLOCKED = 0h) none of the registers within Partition “i” can be written to. They can only be read.

Table 5-20 shows the values that must be written to the LOCKi\_KICK0 and LOCKi\_KICK1 registers to unlock each partition. Writing any other data value to either of these registers locks the protection mechanism and blocks any writes to the registers that reside in Partition “i”.

**Table 5-20. Partition Unlock Values**

Register	Partition <sup>(1)</sup>	Unlock Value	Offset Range
CTRLMMR_LOCK0_KICK0	Partition 0	68EF 3490h	0000h to 1FFFh
CTRLMMR_LOCK0_KICK1	Partition 0	D172 BC5Ah	
CTRLMMR_LOCK1_KICK0	Partition 1	68EF 3490h	4000h to 5FFFh
CTRLMMR_LOCK1_KICK1	Partition 1	D172 BC5Ah	
CTRLMMR_LOCK2_KICK0	Partition 2	68EF 3490h	8000h to 9FFFh
CTRLMMR_LOCK2_KICK1	Partition 2	D172 BC5Ah	
CTRLMMR_LOCK3_KICK0	Partition 3	68EF 3490h	C000h to DFFFh
CTRLMMR_LOCK3_KICK1	Partition 3	D172 BC5Ah	

**Table 5-20. Partition Unlock Values (continued)**

Register	Partition <sup>(1)</sup>	Unlock Value	Offset Range
CTRLMMR_LOCK5_KICK0	Partition 5	68EF 3490h	1 4000h to 1 5FFFh
CTRLMMR_LOCK5_KICK1	Partition 5	D172 BC5Ah	
CTRLMMR_LOCK6_KICK0	Partition 6	68EF 3490h	1 8000h to 1 9FFFh
CTRLMMR_LOCK6_KICK1	Partition 6	D172 BC5Ah	
CTRLMMR_LOCK7_KICK0	Partition 7	68EF 3490h	1 C000h to 1 DFFFh
CTRLMMR_LOCK7_KICK1	Partition 7	D172 BC5Ah	

(1) Not listed partitions are reserved and not used.

### Note

In order to ensure that all registers from all partitions are write protected, software must always re-lock the protection mechanism after completing the register writes.

The following registers are exceptions and are not write protected by the LOCK<sub>i</sub>\_KICK0 and LOCK<sub>i</sub>\_KICK1 registers:

- The *Inter-processor Communication Registers*
- The *Kick Protection Registers* described in this section.

The *Inter-processor Communication Registers* registers are not write protected in order to reduce the access latency.

#### 5.1.3.3.1.3 CTRL\_MMR0 Module Interrupts

The CTRL\_MMR0 module has one interrupt request, the CTRL\_MMR0\_ACCESS\_ERR\_0, which is associated with the following registers:

- CTRLMMR\_INTR\_RAW\_STAT — interrupt raw status register
- CTRLMMR\_INTR\_STAT\_CLR — interrupt status register
- CTRLMMR\_INTR\_EN\_SET — interrupt enable register
- CTRLMMR\_INTR\_EN\_CLR — interrupt disable register

The following applies for the interrupt behavior of the CTRL\_MMR0 module:

- The CTRL\_MMR0 module asserts its interrupt line only if the interrupts are enabled by setting to 1h the corresponding bits in the CTRLMMR\_INTR\_EN\_SET register. These interrupts can be disabled by setting to 1h the corresponding bits in the CTRLMMR\_INTR\_EN\_CLR register.
- After an interrupt has been serviced, software must clear the corresponding status flag. This is done by setting to 1h the corresponding bit in the CTRLMMR\_INTR\_STAT\_CLR register which also clears the corresponding bit in the CTRLMMR\_INTR\_RAW\_STAT register. The status flags in the CTRLMMR\_INTR\_RAW\_STAT register are set even if the corresponding interrupt is disabled unlike those in the CTRLMMR\_INTR\_STAT\_CLR register, which are set only if the corresponding interrupt is enabled.
- An interrupt is also generated by the CTRL\_MMR0, if certain bit in the CTRLMMR\_INTR\_RAW\_STAT register is set to 1h and the corresponding interrupt is enabled through the CTRLMMR\_INTR\_EN\_SET register. This feature is useful during user software debugging. In addition, even if interrupts are not enabled the corresponding raw flag in the CTRLMMR\_INTR\_RAW\_STAT register is set to 1h when an IRQ condition occurs.
- Even if interrupts are not enabled, a status bit in the CTRLMMR\_INTR\_RAW\_STAT register can also be cleared by setting to 1h the corresponding bit in the CTRLMMR\_INTR\_STAT\_CLR register.

Table 5-21 lists the interrupt events which can assert the (CTRL\_MMR0\_ACCESS\_ERR\_0) interrupt line.



**Table 5-21. CTRL\_MMR0 Events**

Event Flag	Event Mask	Description
CTRLMMR_INTR_RAW_STAT[2] LOCK_ERR CTRLMMR_INTR_STAT_CLR[2] EN_LOCK_ERR	CTRLMMR_INTR_EN_SET[2] LOCK_ERR_EN_SET CTRLMMR_INTR_EN_CLR[2] LOCK_ERR_EN_CLR	Lock violation interrupt. Occurs when writing to a register in a locked CTRL_MMR0 partition.
CTRLMMR_INTR_RAW_STAT[1] ADDR_ERR CTRLMMR_INTR_STAT_CLR[1] EN_ADDR_ERR	CTRLMMR_INTR_EN_SET[1] ADDR_ERR_EN_SET CTRLMMR_INTR_EN_CLR[1] ADDR_ERR_EN_CLR	Addressing violation interrupt. Occurs when accessing an illegal address inside the CTRL_MMR0 module.
CTRLMMR_INTR_RAW_STAT[0] PROT_ERR CTRLMMR_INTR_STAT_CLR[0] EN_PROT_ERR	CTRLMMR_INTR_EN_SET[0] PROT_ERR_EN_SET CTRLMMR_INTR_EN_CLR[0] PROT_ERR_EN_CLR	Protection violation interrupt. Occurs when a register is accessed without the required secure/privilege level permissions.

When an error event as described in [Table 5-21](#) occurs, the error associated details are captured in the CTRLMMR\_FAULT\_ADDR, CTRLMMR\_FAULT\_TYPE and CTRLMMR\_FAULT\_ATTR registers. CTRLMMR\_FAULT\_ADDR contains the address of the first fault access. CTRLMMR\_FAULT\_TYPE and CTRLMMR\_FAULT\_ATTR contain status attributes associated with the first fault access. To clear the contents of these three registers and allow them to latch the attributes of the next fault the CTRLMMR\_FAULT\_CLR[0] CLEAR bit must be set to 1h.

#### 5.1.3.3.1.4 Inter-processor Communication Registers

The inter-processor communication registers are used for generating interrupts to the device cores. Any master having access to the CTRL\_MMR0 registers can generate an interrupt by writing 1h to the CTRLMMR\_IPC\_SETx[0] bits. The interrupt is cleared when 1h is written to the CTRLMMR\_IPC\_CLRx[0] bits. For example, writing 1h to CTRLMMR\_IPC\_SET8[0] IPC\_SET generates an interrupt to A72 core 0, and writing 1h to CTRLMMR\_IPC\_CLR8[0] IPC\_CLR clears this event. These registers provide also a *Source ID* facility through the CTRLMMR\_IPC\_SETx[31-4] IPC\_SRC\_SET and CTRLMMR\_IPC\_CLRx[31-4] IPC\_SRC\_CLR bit fields by which up to 28 different sources of interrupts can be identified. Allocation of the source bits to source processor and meaning is entirely based on software convention. Virtually, anything can be a source for these fields as this is completely controlled by software.

Writing 1h to an IPC\_SRC\_SET bit sets to 1h both the IPC\_SRC\_SET and corresponding IPC\_SRC\_CLR bit. Writing 1h to an IPC\_SRC\_CLR bit sets to 0h (clears) both the IPC\_SRC\_CLR and corresponding IPC\_SRC\_SET bit. The same logic applies also to the IPC\_SETx[0] IPC\_SET and IPC\_CLRx[0] IPC\_CLR bits. [Table 5-22](#) summarizes the inter-processor communication (IPC) registers.

**Table 5-22. Summary of the IPC Registers**

IPC Register	Writing 1h to bit 0 results in:	Writing 1h to one of the bits [31-4] results in:
CTRLMMR_IPC_SET0	<ul style="list-style-type: none"> <li>interrupt generation to C71x core 0</li> <li>setting the IPC_SET bit to 1h</li> <li>setting the corresponding IPC_CLR bit to 1h</li> </ul>	setting that bit and the corresponding IPC_SRC_CLR bit to 1h
CTRLMMR_IPC_SET6	<ul style="list-style-type: none"> <li>interrupt generation to C66 core 0</li> <li>setting the IPC_SET bit to 1h</li> <li>setting the corresponding IPC_CLR bit to 1h</li> </ul>	setting that bit and the corresponding IPC_SRC_CLR bit to 1h
CTRLMMR_IPC_SET7	<ul style="list-style-type: none"> <li>interrupt generation to C66 core 1</li> <li>setting the IPC_SET bit to 1h</li> <li>setting the corresponding IPC_CLR bit to 1h</li> </ul>	setting that bit and the corresponding IPC_SRC_CLR bit to 1h
CTRLMMR_IPC_SET8	<ul style="list-style-type: none"> <li>interrupt generation to A72 core 0</li> <li>setting the IPC_SET bit to 1h</li> <li>setting the corresponding IPC_CLR bit to 1h</li> </ul>	setting that bit and the corresponding IPC_SRC_CLR bit to 1h

**Table 5-22. Summary of the IPC Registers (continued)**

IPC Register	Writing 1h to bit 0 results in:	Writing 1h to one of the bits [31-4] results in:
CTRLMMR_IPC_SET9	<ul style="list-style-type: none"> <li>interrupt generation to A72 core 1</li> <li>setting the IPC_SET bit to 1h</li> <li>setting the corresponding IPC_CLR bit to 1h</li> </ul>	setting that bit and the corresponding IPC_SRC_CLR bit to 1h
CTRLMMR_IPC_SET16	<ul style="list-style-type: none"> <li>interrupt generation to MAIN R5 core 0</li> <li>setting the IPC_SET bit to 1h</li> <li>setting the corresponding IPC_CLR bit to 1h</li> </ul>	setting that bit and the corresponding IPC_SRC_CLR bit to 1h
CTRLMMR_IPC_SET17	<ul style="list-style-type: none"> <li>interrupt generation to MAIN R5 core 1</li> <li>setting the IPC_SET bit to 1h</li> <li>setting the corresponding IPC_CLR bit to 1h</li> </ul>	setting that bit and the corresponding IPC_SRC_CLR bit to 1h
CTRLMMR_IPC_SET18	<ul style="list-style-type: none"> <li>interrupt generation to MAIN R5 core 2</li> <li>setting the IPC_SET bit to 1h</li> <li>setting the corresponding IPC_CLR bit to 1h</li> </ul>	setting that bit and the corresponding IPC_SRC_CLR bit to 1h
CTRLMMR_IPC_SET19	<ul style="list-style-type: none"> <li>interrupt generation to MAIN R5 core 3</li> <li>setting the IPC_SET bit to 1h</li> <li>setting the corresponding IPC_CLR bit to 1h</li> </ul>	setting that bit and the corresponding IPC_SRC_CLR bit to 1h
CTRLMMR_IPC_SET20	<ul style="list-style-type: none"> <li>interrupt generation to PRU_ICSSG0 INTC</li> <li>setting the IPC_SET bit to 1h</li> <li>setting the corresponding IPC_CLR bit to 1h</li> </ul>	setting that bit and the corresponding IPC_SRC_CLR bit to 1h
CTRLMMR_IPC_SET21	<ul style="list-style-type: none"> <li>interrupt generation to PRU_ICSSG0 INTC</li> <li>setting the IPC_SET bit to 1h</li> <li>setting the corresponding IPC_CLR bit to 1h</li> </ul>	setting that bit and the corresponding IPC_SRC_CLR bit to 1h
CTRLMMR_IPC_SET22	<ul style="list-style-type: none"> <li>interrupt generation to PRU_ICSSG1 INTC</li> <li>setting the IPC_SET bit to 1h</li> <li>setting the corresponding IPC_CLR bit to 1h</li> </ul>	setting that bit and the corresponding IPC_SRC_CLR bit to 1h
CTRLMMR_IPC_SET23	<ul style="list-style-type: none"> <li>interrupt generation to PRU_ICSSG1 INTC</li> <li>setting the IPC_SET bit to 1h</li> <li>setting the corresponding IPC_CLR bit to 1h</li> </ul>	setting that bit and the corresponding IPC_SRC_CLR bit to 1h
CTRLMMR_IPC_CLR0	<ul style="list-style-type: none"> <li>clearing the interrupt to C71x core 0</li> <li>clearing the IPC_CLR bit</li> <li>clearing the corresponding IPC_SET bit</li> </ul>	clearing (set to 0h) that bit and the corresponding IPC_SRC_SET bit
CTRLMMR_IPC_CLR6	<ul style="list-style-type: none"> <li>clearing the interrupt to C66 core 0</li> <li>clearing the IPC_CLR bit</li> <li>clearing the corresponding IPC_SET bit</li> </ul>	clearing (set to 0h) that bit and the corresponding IPC_SRC_SET bit
CTRLMMR_IPC_CLR7	<ul style="list-style-type: none"> <li>clearing the interrupt to C66 core 1</li> <li>clearing the IPC_CLR bit</li> <li>clearing the corresponding IPC_SET bit</li> </ul>	clearing (set to 0h) that bit and the corresponding IPC_SRC_SET bit
CTRLMMR_IPC_CLR8	<ul style="list-style-type: none"> <li>clearing the interrupt to A72 core 0</li> <li>clearing the IPC_CLR bit</li> <li>clearing the corresponding IPC_SET bit</li> </ul>	clearing (set to 0h) that bit and the corresponding IPC_SRC_SET bit
CTRLMMR_IPC_CLR9	<ul style="list-style-type: none"> <li>clearing the interrupt to A72 core 1</li> <li>clearing the IPC_CLR bit</li> <li>clearing the corresponding IPC_SET bit</li> </ul>	clearing (set to 0h) that bit and the corresponding IPC_SRC_SET bit

**Table 5-22. Summary of the IPC Registers (continued)**

IPC Register	Writing 1h to bit 0 results in:	Writing 1h to one of the bits [31-4] results in:
CTRLMMR_IPC_CLR16	<ul style="list-style-type: none"> <li>clearing the interrupt to MAIN R5 core 0</li> <li>clearing the IPC_CLR bit</li> <li>clearing the corresponding IPC_SET bit</li> </ul>	clearing (set to 0h) that bit and the corresponding IPC_SRC_SET bit
CTRLMMR_IPC_CLR17	<ul style="list-style-type: none"> <li>clearing the interrupt to MAIN R5 core 1</li> <li>clearing the IPC_CLR bit</li> <li>clearing the corresponding IPC_SET bit</li> </ul>	clearing (set to 0h) that bit and the corresponding IPC_SRC_SET bit
CTRLMMR_IPC_CLR18	<ul style="list-style-type: none"> <li>clearing the interrupt to MAIN R5 core 2</li> <li>clearing the IPC_CLR bit</li> <li>clearing the corresponding IPC_SET bit</li> </ul>	clearing (set to 0h) that bit and the corresponding IPC_SRC_SET bit
CTRLMMR_IPC_CLR19	<ul style="list-style-type: none"> <li>clearing the interrupt to MAIN R5 core 3</li> <li>clearing the IPC_CLR bit</li> <li>clearing the corresponding IPC_SET bit</li> </ul>	clearing (set to 0h) that bit and the corresponding IPC_SRC_SET bit
CTRLMMR_IPC_CLR20	<ul style="list-style-type: none"> <li>clearing the interrupt to PRU_ICSSG0 INTC</li> <li>clearing the IPC_CLR bit</li> <li>clearing the corresponding IPC_SET bit</li> </ul>	clearing (set to 0h) that bit and the corresponding IPC_SRC_SET bit
CTRLMMR_IPC_CLR21	<ul style="list-style-type: none"> <li>clearing the interrupt to PRU_ICSSG0 INTC</li> <li>clearing the IPC_CLR bit</li> <li>clearing the corresponding IPC_SET bit</li> </ul>	clearing (set to 0h) that bit and the corresponding IPC_SRC_SET bit
CTRLMMR_IPC_CLR22	<ul style="list-style-type: none"> <li>clearing the interrupt to PRU_ICSSG1 INTC</li> <li>clearing the IPC_CLR bit</li> <li>clearing the corresponding IPC_SET bit</li> </ul>	clearing (set to 0h) that bit and the corresponding IPC_SRC_SET bit
CTRLMMR_IPC_CLR23	<ul style="list-style-type: none"> <li>clearing the interrupt to PRU_ICSSG1 INTC</li> <li>clearing the IPC_CLR bit</li> <li>clearing the corresponding IPC_SET bit</li> </ul>	clearing (set to 0h) that bit and the corresponding IPC_SRC_SET bit

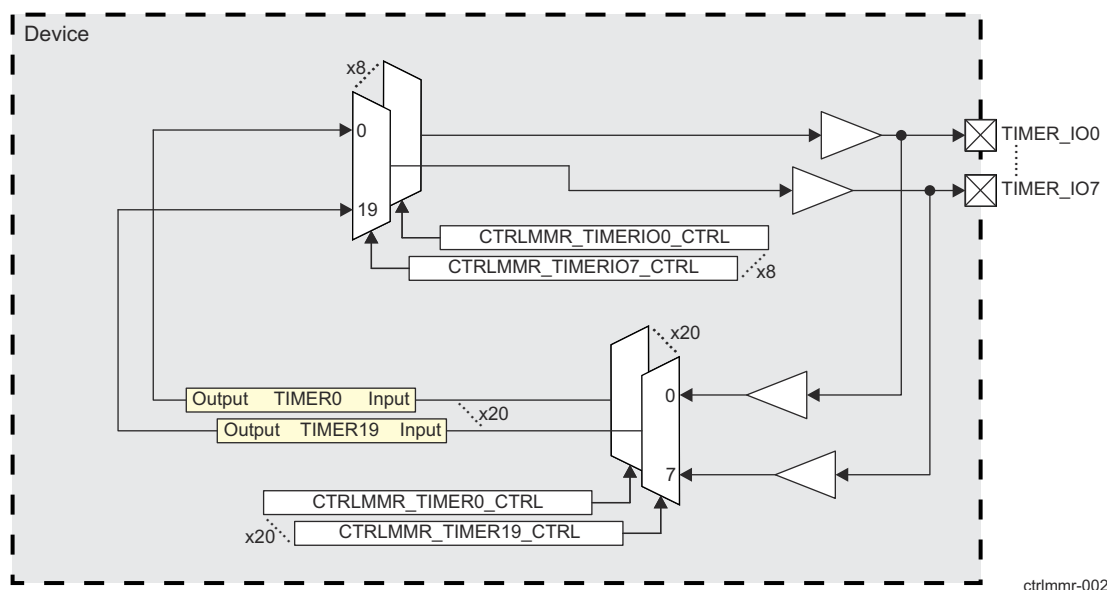
#### Note

For latency reasons the IPC registers are not write protected by KICK registers which means that they can be written to without a need for performing unlocking procedure as described in *Kick Protection Registers*.

#### 5.1.3.3.1.5 Timer I/O Muxing Control Registers

Not all timer inputs and outputs are pinned out of the device. Through the timer I/O muxing registers the inputs and outputs of the MAIN domain timers can be made available on the TIMER\_IO[7:0] device pads.

Each timer input can be configured to be driven by one of the TIMER\_IO[7:0] pads through the corresponding bits in the CTRLMMR\_TIMER0\_CTRL to CTRLMMR\_TIMER19\_CTRL registers. Each of the TIMER\_IO[7:0] pads can be configured to be driven by any of the timer outputs through the corresponding bits in the CTRLMMR\_TIMERIO0\_CTRL to CTRLMMR\_TIMERIO7\_CTRL registers. [Figure 5-5](#) shows the TIMER I/O multiplexing scheme for the MAIN domain.



**Figure 5-5. MAIN Domain TIMER I/O Multiplexing Scheme**

For more information about each timer, see *Timers*.

#### 5.1.3.3.1.6 EHRPWM/EQEP Control and Status Registers

The CTRL\_MMR0 registers which provide control and status information for the EHRPWM/EQEP modules are shown in [Table 5-23](#).

**Table 5-23. Summary of the EHRPWM/EQEP Control and Status Registers**

Register Name	Description	Associated Functionality Described in:
CTRLMMR_EPWM0_CTRL CTRLMMR_EPWM1_CTRL CTRLMMR_EPWM2_CTRL CTRLMMR_EPWM3_CTRL CTRLMMR_EPWM4_CTRL CTRLMMR_EPWM5_CTRL	Time base clock, source of PWM synchronization input and other controls for the EHRPWM <sup>(1)</sup> module	<i>EPWM<sup>(1)</sup> Modules Time Base Clock Gating and Daisy-Chain Connectivity between EPWM Modules in Enhanced Pulse Width Modulation (EPWM) Module</i>
CTRLMMR_SOCA_SEL CTRLMMR_SOCB_SEL	Start of Conversion output source	<i>ADC start of conversion signals (PWM_SOCA and PWM_SOCB) in Enhanced Pulse Width Modulation (EPWM) Module</i>
CTRLMMR_EQEP_STAT	Provides EQEP <sup>x</sup> phase error status information	<i>EPWM Integration Device Specific EQEP Features in Enhanced Quadrature Encoder Pulse (EQEP) Module</i>

(1) The terms "EPWM" and "EHRPWM" are interchangeable in this section.

#### 5.1.3.3.1.7 PRU\_ICSSG Control Registers

There are registers within the CTRL\_MMR0 module address space related to several PRU\_ICSSG features. These registers are shown in [Table 5-24](#).

**Table 5-24. PRU\_ICSSG Control Registers**

Register Name	Register Name
CTRLMMR_ICSSG0_CTRL0	CTRLMMR_ICSSG1_CTRL0
CTRLMMR_ICSSG0_CTRL1	CTRLMMR_ICSSG1_CTRL1

### 5.1.3.3.1.8 Clock Muxing and Division Registers

There are registers within the CTRL\_MMR0 module address space dedicated to clock muxing and division for some device modules. [Table 5-25](#) summarizes these registers. Related information can also be found in [Section 5.4 Clocking](#).

**Table 5-25. Summary of the Clock Muxing and Division Registers**

Register Name	Register Name	Register Name
CTRLMMR_OBSCLK0_CTRL	CTRLMMR_SPI6_CLKSEL	CTRLMMR_ATL_AWS1_SEL
CTRLMMR_OBSCLK1_CTRL	CTRLMMR_SPI7_CLKSEL	CTRLMMR_ATL_AWS2_SEL
CTRLMMR_CLKOUT_CTRL	CTRLMMR_USART0_CLK_CTRL	CTRLMMR_ATL_AWS3_SEL
CTRLMMR_GTC_CLKSEL	CTRLMMR_USART1_CLK_CTRL	CTRLMMR_ATL_CLKSEL
CTRLMMR_EFUSE_CLKSEL	CTRLMMR_USART2_CLK_CTRL	CTRLMMR_AUDIO_REFCLK0_CTRL
CTRLMMR_ICSSG0_CLKSEL	CTRLMMR_USART3_CLK_CTRL	CTRLMMR_AUDIO_REFCLK1_CTRL
CTRLMMR_ICSSG1_CLKSEL	CTRLMMR_USART4_CLK_CTRL	CTRLMMR_AUDIO_REFCLK2_CTRL
CTRLMMR_PCIE_REFCLK0_CLKSEL	CTRLMMR_USART5_CLK_CTRL	CTRLMMR_AUDIO_REFCLK3_CTRL
CTRLMMR_PCIE_REFCLK1_CLKSEL	CTRLMMR_USART6_CLK_CTRL	CTRLMMR_DPIO_CLK_CTRL
CTRLMMR_PCIE_REFCLK2_CLKSEL	CTRLMMR_USART7_CLK_CTRL	CTRLMMR_DPI1_CLK_CTRL
CTRLMMR_PCIE_REFCLK3_CLKSEL	CTRLMMR_USART8_CLK_CTRL	CTRLMMR_DPHY0_CLKSEL
CTRLMMR_PCIE0_CLKSEL	CTRLMMR_USART9_CLK_CTRL	CTRLMMR_DSS_DISPC0_CLKSEL1
CTRLMMR_PCIE1_CLKSEL	CTRLMMR_MCASP0_CLKSEL	CTRLMMR_DSS_DISPC0_CLKSEL2
CTRLMMR_PCIE2_CLKSEL	CTRLMMR_MCASP1_CLKSEL	CTRLMMR_DSS_DISPC0_CLKSEL3
CTRLMMR_PCIE3_CLKSEL	CTRLMMR_MCASP2_CLKSEL	CTRLMMR_EDP_PHY0_CLKSEL
CTRLMMR_CPSW_CLKSEL	CTRLMMR_MCASP3_CLKSEL	CTRLMMR_EDP0_CLK_CTRL
CTRLMMR_NAVSS_CLKSEL	CTRLMMR_MCASP4_CLKSEL	CTRLMMR_WWD0_CLKSEL
CTRLMMR_EMMC0_CLKSEL	CTRLMMR_MCASP5_CLKSEL	CTRLMMR_WWD1_CLKSEL
CTRLMMR_EMMC1_CLKSEL	CTRLMMR_MCASP6_CLKSEL	CTRLMMR_WWD15_CLKSEL
CTRLMMR_EMMC2_CLKSEL	CTRLMMR_MCASP7_CLKSEL	CTRLMMR_WWD16_CLKSEL
CTRLMMR_UFS0_CLKSEL	CTRLMMR_MCASP8_CLKSEL	CTRLMMR_WWD24_CLKSEL
CTRLMMR_GPMC_CLKSEL	CTRLMMR_MCASP9_CLKSEL	CTRLMMR_WWD25_CLKSEL
CTRLMMR_USB0_CLKSEL	CTRLMMR_MCASP10_CLKSEL	CTRLMMR_WWD28_CLKSEL
CTRLMMR_USB1_CLKSEL	CTRLMMR_MCASP11_CLKSEL	CTRLMMR_WWD29_CLKSEL
CTRLMMR_TIMER0_CLKSEL	CTRLMMR_MCASP0_AHCLKSEL	CTRLMMR_WWD30_CLKSEL
CTRLMMR_TIMER1_CLKSEL	CTRLMMR_MCASP1_AHCLKSEL	CTRLMMR_WWD31_CLKSEL
CTRLMMR_TIMER2_CLKSEL	CTRLMMR_MCASP2_AHCLKSEL	CTRLMMR_SERDES0_CLKSEL
CTRLMMR_TIMER3_CLKSEL	CTRLMMR_MCASP3_AHCLKSEL	CTRLMMR_SERDES0_CLK1SEL
CTRLMMR_TIMER4_CLKSEL	CTRLMMR_MCASP4_AHCLKSEL	CTRLMMR_SERDES1_CLKSEL
CTRLMMR_TIMER5_CLKSEL	CTRLMMR_MCASP5_AHCLKSEL	CTRLMMR_SERDES1_CLK1SEL
CTRLMMR_TIMER6_CLKSEL	CTRLMMR_MCASP6_AHCLKSEL	CTRLMMR_SERDES2_CLKSEL
CTRLMMR_TIMER7_CLKSEL	CTRLMMR_MCASP7_AHCLKSEL	CTRLMMR_SERDES2_CLK1SEL
CTRLMMR_TIMER8_CLKSEL	CTRLMMR_MCASP8_AHCLKSEL	CTRLMMR_SERDES3_CLKSEL
CTRLMMR_TIMER9_CLKSEL	CTRLMMR_MCASP9_AHCLKSEL	CTRLMMR_SERDES3_CLK1SEL
CTRLMMR_TIMER10_CLKSEL	CTRLMMR_MCASP10_AHCLKSEL	CTRLMMR_MCAN0_CLKSEL
CTRLMMR_TIMER11_CLKSEL	CTRLMMR_MCASP11_AHCLKSEL	CTRLMMR_MCAN1_CLKSEL
CTRLMMR_TIMER12_CLKSEL	CTRLMMR_ASRC_RXSYNC0_SEL	CTRLMMR_MCAN2_CLKSEL
CTRLMMR_TIMER13_CLKSEL	CTRLMMR_ASRC_RXSYNC1_SEL	CTRLMMR_MCAN3_CLKSEL
CTRLMMR_TIMER14_CLKSEL	CTRLMMR_ASRC_RXSYNC2_SEL	CTRLMMR_MCAN4_CLKSEL
CTRLMMR_TIMER15_CLKSEL	CTRLMMR_ASRC_RXSYNC3_SEL	CTRLMMR_MCAN5_CLKSEL
CTRLMMR_TIMER16_CLKSEL	CTRLMMR_ASRC_TXSYNC0_SEL	CTRLMMR_MCAN6_CLKSEL

**Table 5-25. Summary of the Clock Muxing and Division Registers (continued)**

Register Name	Register Name	Register Name
CTRLMMR_TIMER17_CLKSEL	CTRLMMR_ASRC_TXSYNC1_SEL	CTRLMMR_MCAN7_CLKSEL
CTRLMMR_TIMER18_CLKSEL	CTRLMMR_ASRC_TXSYNC2_SEL	CTRLMMR_MCAN8_CLKSEL
CTRLMMR_TIMER19_CLKSEL	CTRLMMR_ASRC_TXSYNC3_SEL	CTRLMMR_MCAN9_CLKSEL
CTRLMMR_SPI0_CLKSEL	CTRLMMR_ATL_BWS0_SEL	CTRLMMR_MCAN10_CLKSEL
CTRLMMR_SPI1_CLKSEL	CTRLMMR_ATL_BWS1_SEL	CTRLMMR_MCAN11_CLKSEL
CTRLMMR_SPI2_CLKSEL	CTRLMMR_ATL_BWS2_SEL	CTRLMMR_MCAN12_CLKSEL
CTRLMMR_SPI3_CLKSEL	CTRLMMR_ATL_BWS3_SEL	CTRLMMR_MCAN13_CLKSEL
CTRLMMR_SPI5_CLKSEL	CTRLMMR_ATL_AWS0_SEL	

**5.1.3.3.1.9 Ethernet Port Operation Control Registers**

There are registers within the CTRL\_MMR0 module address space that control the operation of each ethernet port in the device main domain. These registers are summarized in [Table 5-26](#).

**Table 5-26. Ethernet Port Operation Control Registers**

Register Name	Register Name
CTRLMMR_ENET1_CTRL	CTRLMMR_ENET5_CTRL
CTRLMMR_ENET2_CTRL	CTRLMMR_ENET6_CTRL
CTRLMMR_ENET3_CTRL	CTRLMMR_ENET7_CTRL
CTRLMMR_ENET4_CTRL	CTRLMMR_ENET8_CTRL

**5.1.3.3.1.10 PCIe Operation Control Registers**

There are registers within the CTRL\_MMR0 module address space that configure the lane operation and generation support for each PCIe module in the device main domain. These registers are summarized in [Table 5-27](#).

**Table 5-27. PCIe Operation Control Registers**

Register Name	Register Name
CTRLMMR_PCIE0_CTRL	CTRLMMR_PCIE2_CTRL
CTRLMMR_PCIE1_CTRL	CTRLMMR_PCIE3_CTRL

**5.1.3.3.1.11 SERDES Lane Function Control Registers**

There are registers within the CTRL\_MMR0 module address space that select the function of each lane for each SERDES module in the device main domain. These registers are shown in [Table 5-28](#).

**Table 5-28. SERDES Lane Function Control Registers**

Register Name	Register Name
CTRLMMR_SERDES0_LN0_CTRL	CTRLMMR_SERDES3_LN0_CTRL
CTRLMMR_SERDES0_LN1_CTRL	CTRLMMR_SERDES3_LN1_CTRL
CTRLMMR_SERDES1_LN0_CTRL	CTRLMMR_SERDES4_LN0_CTRL
CTRLMMR_SERDES1_LN1_CTRL	CTRLMMR_SERDES4_LN1_CTRL
CTRLMMR_SERDES2_LN0_CTRL	CTRLMMR_SERDES4_LN2_CTRL
CTRLMMR_SERDES2_LN1_CTRL	CTRLMMR_SERDES4_LN3_CTRL

**5.1.3.3.1.12 DDRSS Dynamic Frequency Change Registers**

There are registers within the CTRL\_MMR0 module address space that are used to dynamically change the DDR clock frequency to support LPDDR4 Frequency Set Point (FSP). These registers are shown in [Table 5-29](#). For more information, see [Section 8.2.4.5 DDRSS Dynamic Frequency Change Interface](#) in [Chapter 8 Memory Controllers](#).

**Table 5-29. DDRSS Dynamic Frequency Change Registers**

Register Name	Register Name
CTRLMMR_CHNG_DDR4_FSP_REQ	CTRLMMR_DDR4_FSP_CLKCHNG_REQ
CTRLMMR_CHNG_DDR4_FSP_ACK	CTRLMMR_DDR4_FSP_CLKCHNG_ACK



## 5.2 Power

This chapter describes the power-management architecture implemented in the device.

The Power Management is presented in several general sections - Power Management Subsystems, including both System and Control Modules; Device Power States; Dynamic and Thermal Management.

### 5.2.1 Power Management Overview

Power management architecture of the device is built over three levels of power controls: clock-, power-, and voltage supplies to the subsystem modules, namely domains. A domain is a group of modules or subsections of the device that share a common entity (for example, common clock source, common voltage source, or common power switch).

A set of Power Domains (PD) are defined for the device, where clock and power supplies to the domain may be controlled by Power Sleep Controllers (PSC), see [Section 5.2.2.2.1, Power Sleep Controller and Local Power Sleep Controllers](#). This section describes functionality of PSC, as well as list of power domains, control registers and hierarchical dependencies of PDs.

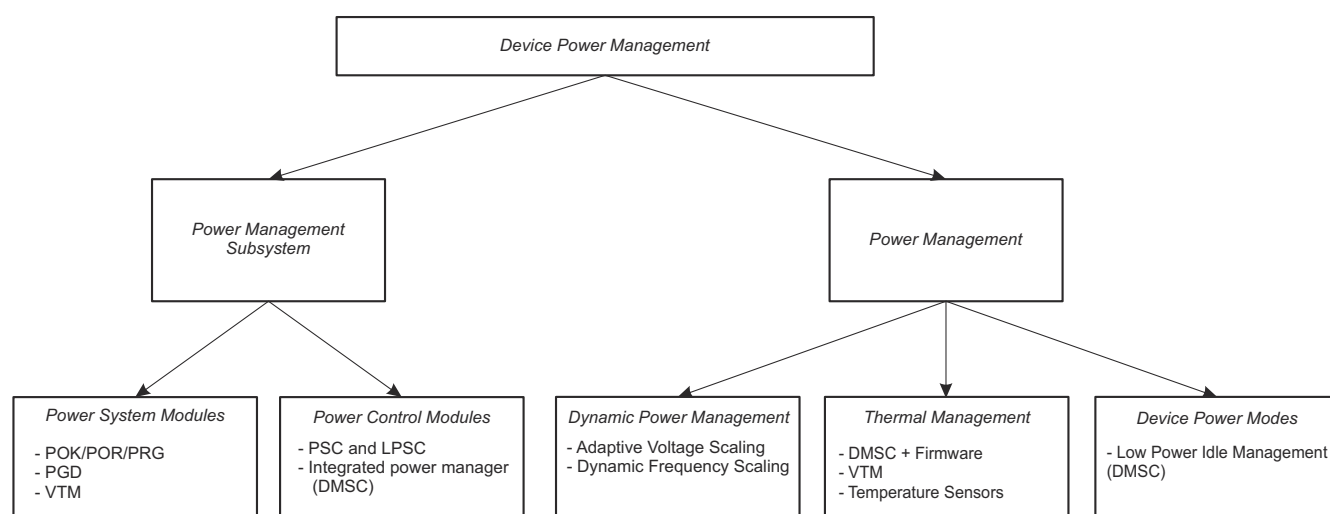
One or multiple power domains may share a single power supply, this level of grouping are referred as Voltage Domains (VD). Some voltage domains may be turned off by externally switching off the power supplies to these domains. [Section 5.2.2.2.1.3.1, WKUP\\_PSC0 Device-Specific Information](#), and [Section 5.2.2.2.1.3.1, PSC0 Device-Specific Information](#), give list of VDs and corresponding PDs on them.

Device power states are defined based on the combination of VD and PD ON/OFF states, as described in [Section 5.2.3, Device Power States](#)

Initial configuration of the device clocking tree and power domains, as well as power state transitions, may be complex due to hierarchical dependencies. Thus a Device Management and Security Controller (DMSC) is integrated to off-load some of these tasks from main controllers. DMSC also serves as a wake-up controller and generator, maintaining minimum hardware state machines when all programmable cores are turned off. Power management functions of the DMSC are described in [Section 5.2.2.2.2, Integrated Power Management \(DMSC\)](#), for functional reference. Standard applications shall use the DMSC Firmware APIs to perform high-level power management functions.

Additional hardware modules for temperature sensing, thermal management, power monitoring, and reset functions related to power ramping, are described in [Section 5.2.2.1, Power System Modules](#).

[Figure 5-6](#) shows the overview of the common power management of the device.



power-001

**Figure 5-6. Overview of Common Power Management**

## 5.2.2 Power Management Subsystems

### 5.2.2.1 Power Management Unit

The Power Management Unit in consists of a Reference system and a Safety system.

- **Reference System:**

The Reference system generates internally used power supply and reference rails. It ensures reliable power on sequencing with the PMU and generates a reset signal based on coarse voltage level checks given to the external power supplies. The Reference system also contains the 1.8V LDO which generates a 1.8V output on the VDDA18\_LDO pin. The 1.8V on VDDA18\_LDO can be connected to VDDA18 and VDDA18\_OSC\_PLL on the board to provide the 1.8V supply to the Analog Circuit and PLL. This connection has to be done external to the device as there is no internal connection path for VDDA18\_LDO to source either VDDA18 or VDDA\_OSC\_PLL.

Using the 1.8V LDO as the VDDA18 source allows the 1.8V LDO to source multiple internal blocks. These blocks include the internal ADC voltage reference buffers as well as other analog loads. Because the ADC reference buffers are sourced internally through the VDDA18 input, the VDDA18\_LDO output should not be connected as source for the ADC VREFHI inputs.

- **Safety System:**

The Safety System contains the safety comparators and temperature sensor to monitor the system supplies and temperature. The glitch filtered output of the voltage monitors are connected to the Error Signaling Module (ESM) and provide an error signal if the supply is not within the voltage thresholds set for the individual monitored supplies.

#### 5.2.2.1.1 Power OK (POK) Modules

---

#### Note

All POK modules are inWKUP domain regardless of the domain of the voltage they monitor.

---



---

#### Note

In this family of devices, the Power-on-reset module is used only for voltage monitoring, as three additional POK modules. For more information about POK functionality in POR module - voltage being monitored and type (under voltage or over voltage), see [Power on Reset \(POR\) Module](#).

---

**Table 5-30. POK Allocation Within Device Domains**

Module Instance	Domain		
	WKUP	MCU	MAIN
WKUP_POK0	✓	-	-
WKUP_POK1	✓	-	-
WKUP_POK2	✓	-	-
WKUP_POK3	✓	-	-
WKUP_POK4	✓	-	-
WKUP_POK5	✓	-	-
WKUP_POK6	✓	-	-
WKUP_POK7	✓	-	-
WKUP_POK8	✓	-	-
WKUP_POK9	✓	-	-
WKUP_POK10	✓	-	-
WKUP_POK11	✓	-	-
WKUP_POK12	✓	-	-
WKUP_POK13	✓	-	-

POK modules are responsible for accurately detecting the voltage levels. Each module is trimmed to account for process and temperature variations. The trim values are provided by eFuse chains enabled by a POR/Reset Generator (PRG) module.

Two types of POK modules are implemented in this family of devices - POK and POK\_SA. See [Table 5-31](#) about the type of a POK and the voltage it is monitoring. More information about the two types of POKs is given further in this section.

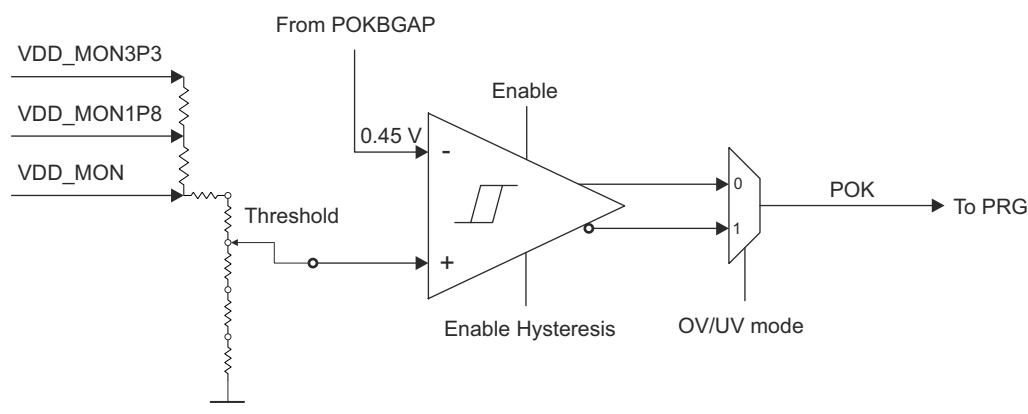
**Table 5-31. POK Modules Type and Monitored Voltage**

Module Instance	Connected to PRG	Type	Voltage Monitored	OV/UV <sup>(1)</sup>
WKUP_POK0	WKUP_PRG0	POK_SA	VMON_ER_VSYS pin <sup>(2)</sup>	UV
WKUP_POK1	WKUP_PRG0	POK	VDDSHV0_MCU	UV
WKUP_POK2	WKUP_PRG0	POK	VDDAR_MCU	UV
WKUP_POK3	WKUP_PRG0	POK	VDD_MCU	OV
WKUP_POK4	WKUP_PRG0	POK	VDDSHV0_MCU	OV
WKUP_POK5	WKUP_PRG0	POK	VDDAR_MCU	OV
WKUP_POK6	WKUP_PRG2	POK	VDD_CORE	UV
WKUP_POK7	WKUP_PRG2	POK	VDD_CPU	UV
WKUP_POK8	WKUP_PRG2	POK	VMON_IR_VEXT pin	UV
WKUP_POK9	WKUP_PRG2	POK	VDDAR_CORE	UV
WKUP_POK10	WKUP_PRG2	POK	VDD_CORE	OV
WKUP_POK11	WKUP_PRG2	POK	VDD_CPU	OV
WKUP_POK12	WKUP_PRG2	POK	VMON_IR_VEXT pin	OV
WKUP_POK13	WKUP_PRG2	POK	VDDAR_CORE	OV

(1) UV: POK is set for under voltage detection, OV: POK is set for over voltage detection.

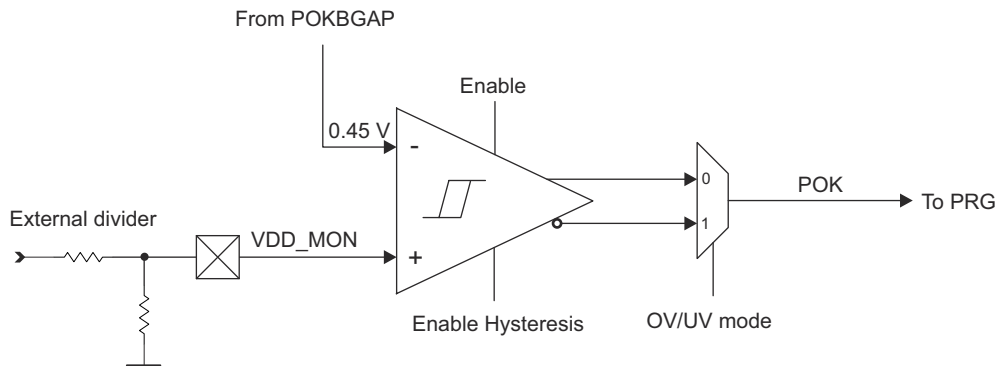
(2) Typically used to monitor PMIC's VSYS voltage.

[Figure 5-7](#) and [Figure 5-8](#) represent the block diagrams of the two POK types.



pok-001

**Figure 5-7. POK Block Diagram**



pok-002

**Figure 5-8. POK\_SA Block Diagram**

Vref for both POK and POK\_SA types in Figure 5-7 and Figure 5-8 is supplied from POR module, see [Power on Reset \(POR\) Module](#).

#### 5.2.2.1.1.1 POK Programming Model

Table 5-32 and Table 5-33 show the programmable features for the POK modules based on the block diagrams shown in Figure 5-7 and Figure 5-8. Furthermore, each POK module has its own dedicated register to set its programmable features. Table 5-34 lists those registers.

**Table 5-32. POK Programmable Features**

Programmable Feature	Register Bitfield of the POK Dedicated Register
POK hysteresis	[31] HYST_EN
POK over or under voltage detection mode	[7] OVER_VOLT_DET
POK programmable resistors for voltage comparator	[6-0] POK_TRIM

**Table 5-33. POK\_SA Programmable Features**

Programmable Feature	Register Bitfield of the POK Dedicated Register
POK hysteresis	[31] HYST_EN
POK over or under voltage detection mode	[0] OVER_VOLT_DET

**Table 5-34. POK Control Registers**

POK	Voltage Monitored <sup>(1)</sup>	Register <sup>(2)</sup>
WKUP_PRG0		
WKUP_POK0	VMON_ER_VSYS pin	CTRLMMR_WKUP_POK_VDDA_PMIC_IN_CTRL
WKUP_POK1	VDDSHV0_MCU (UV)	CTRLMMR_WKUP_POK_VDDSHV_WKUP_GEN_UV_CTRL
WKUP_POK2	VDDAR_MCU (UV)	CTRLMMR_WKUP_POK_VDDR_MCU_UV_CTRL
WKUP_POK3	VDD_MCU (OV)	CTRLMMR_WKUP_POK_VDD_MCU_OV_CTRL
WKUP_POK4	VDDSHV0_MCU (OV)	CTRLMMR_WKUP_POK_VDDSHV_WKUP_GEN_OV_CTRL
WKUP_POK5	VDDAR_MCU (OV)	CTRLMMR_WKUP_POK_VDDR_MCU_OV_CTRL
WKUP_PRG2		
WKUP_POK6	VDD_CORE (UV)	CTRLMMR_WKUP_POK_VDD_CORE_UV_CTRL
WKUP_POK7	VDD_CPU (UV)	CTRLMMR_WKUP_POK_VDD_CPU_UV_CTRL
WKUP_POK8	VMON_IR_VEXT pin (UV)	CTRLMMR_WKUP_POK_VMON_EXT_UV_CTRL
WKUP_POK9	VDDAR_CORE (UV)	CTRLMMR_WKUP_POK_VDDR_CORE_UV_CTRL
WKUP_POK10	VDD_CORE (OV)	CTRLMMR_WKUP_POK_VDD_CORE_OV_CTRL
WKUP_POK11	VDD_CPU (OV)	CTRLMMR_WKUP_POK_VDD_CPU_OV_CTRL
WKUP_POK12	VMON_IR_VEXT pin (OV)	CTRLMMR_WKUP_POK_VMON_EXT_OV_CTRL

**Table 5-34. POK Control Registers (continued)**

POK	Voltage Monitored <sup>(1)</sup>	Register <sup>(2)</sup>
WKUP_POK13	VDDAR_CORE (OV)	CTRLMMR_WKUP_POK_VDDR_CORE_OV_CTRL

- (1) UV means POK is set for under voltage detection, OV means POK is set for over voltage detection.  
(2) For more information about control and status registers, see *Control Module (CTRL\_MMR)*.

#### Note

All POKs given in [Table 5-34](#) are enabled via corresponding bit in CTRLMMR\_WKUP\_PRG0\_CTRL and CTRLMMR\_WKUP\_MAIN\_PRG\_CTRL. Status of the POKs can be monitored via CTRLMMR\_WKUP\_PRG0\_STAT and CTRLMMR\_WKUP\_MAIN\_PRG\_STAT. For more information about control registers, see *Control Module (CTRL\_MMR)*.

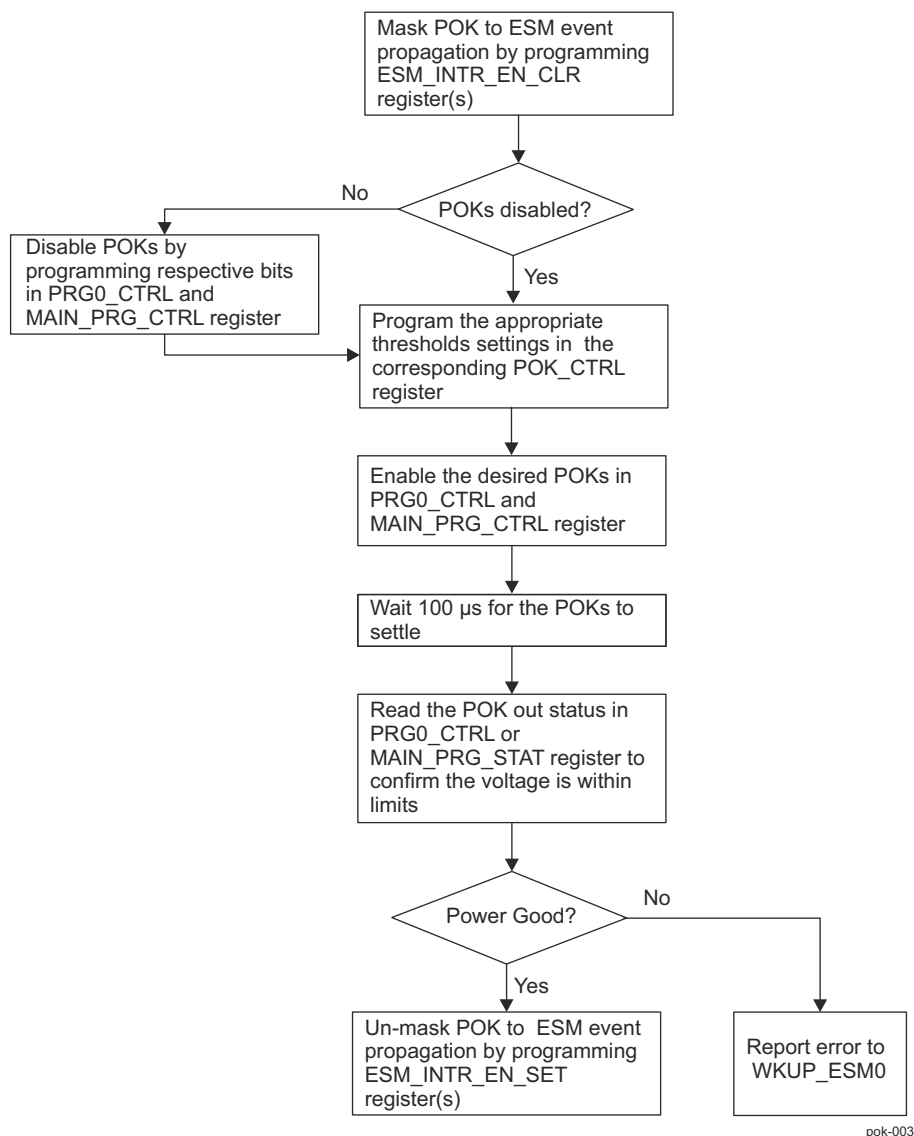
The possible values of the monitored voltage differ among the POK types, see [Table 5-35](#). If the voltage values are programmable, they are set through [6-0] POK\_TRIM bitfield in the corresponding POK control register, see [Table 5-34](#).

**Table 5-35. Values of the Possible Monitored Voltages**

POK Type	Monitored Voltage Values
POK	3.3 V, 1.8 V, core voltages from 0.6 V to 1.5 V
POK_SA	0.45 V (allows for external resistor divider for a wide range of scaled voltages)

### POK Threshold Setting Programming Sequence

After initial boot, it is the responsibility of software to program the appropriate POK threshold settings and then enable the POK monitoring for desired supply rails. The sequence shown in [Figure 5-9](#) has to be followed.

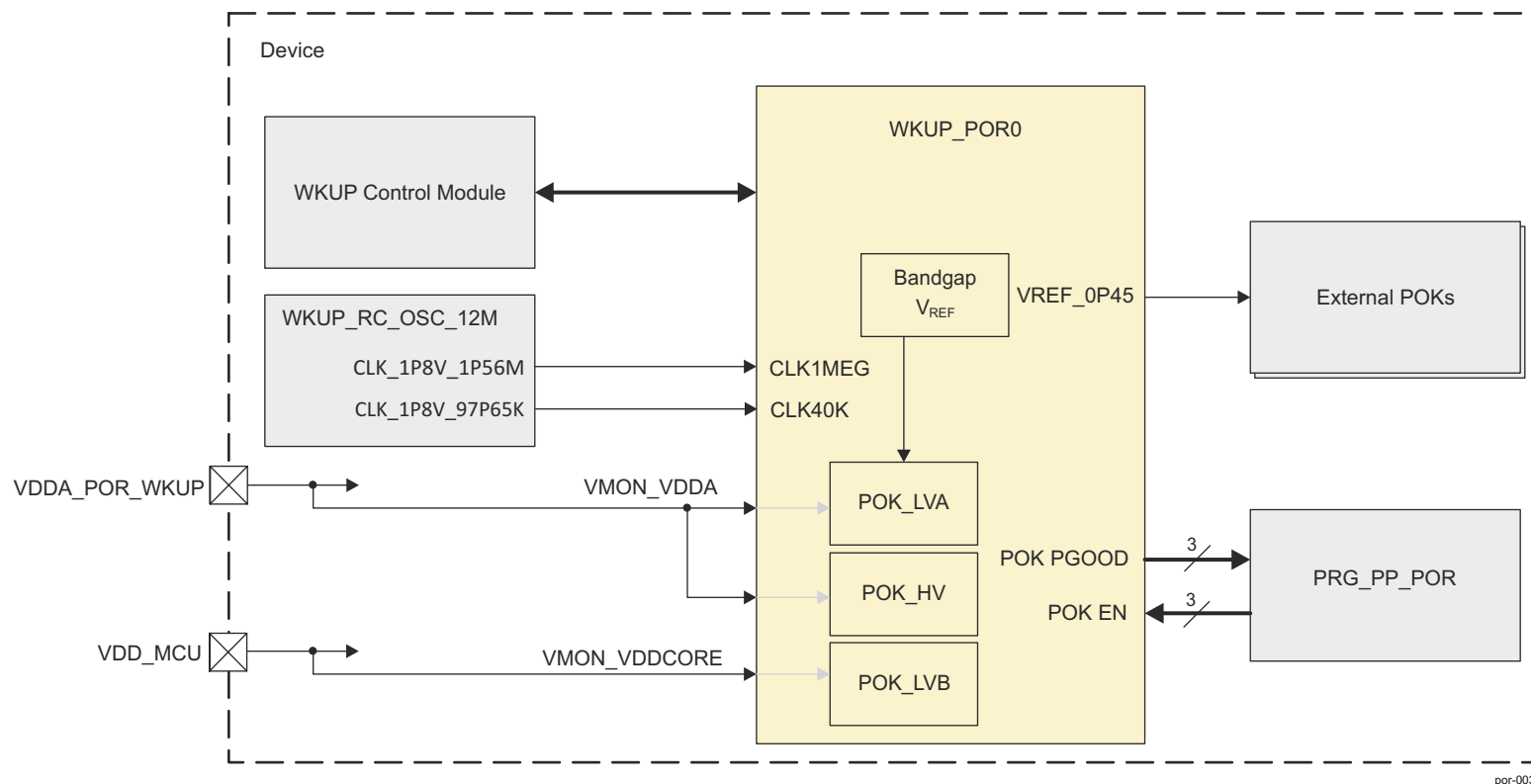

**Figure 5-9. POK Programming Model**





### 5.2.2.1.2.2 POR Integration

The device supports one instance of a Power on Reset (POR) module - WKUP\_POR0. WKUP\_POR0 is located in WKUP domain. [Figure 5-11](#) shows the integration of WKUP\_POR0.



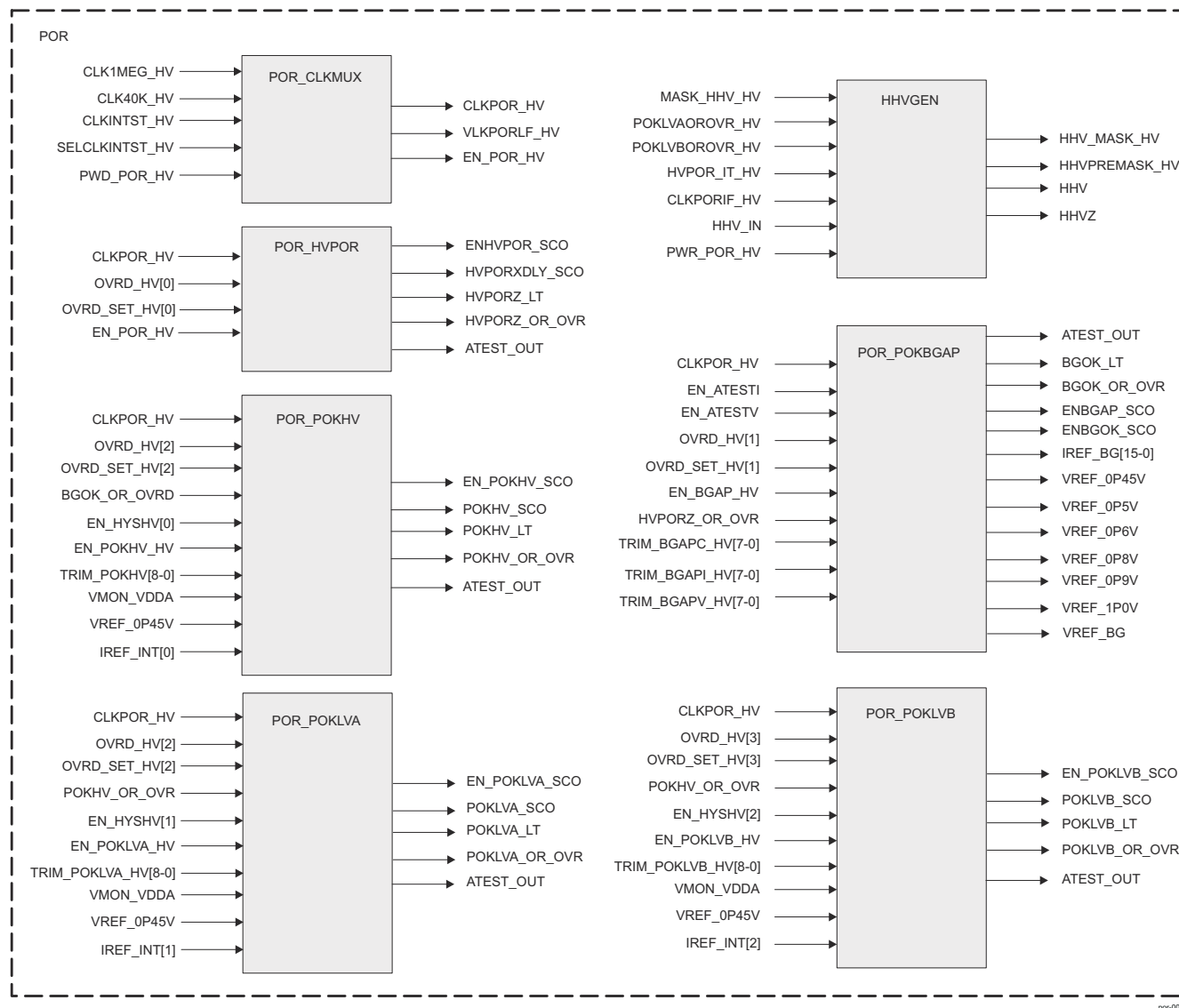
**Figure 5-11. POR Integration**

**Table 5-37. POR Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
WKUP_POR0	CLK1MEG	CLK_1P8V_1P56M	WKUP_RC_OSC_12M	Clock from RCOSC
	CLK40K	CLK_1P8V_97P65K	WKUP_RC_OSC_12M	Clock from RCOSC

### 5.2.2.1.2.3 POR Functional Description

Figure 5-12 shows the block diagram of a POR module.



**Figure 5-12. POR Block Diagram**

The modules in the block diagrams are:

- POR\_HVPOR, system supply POR is used to reset the custom logic at power-up and keep it in reset until system supply is sufficiently high for logic cells to operate reliably.
- Bandgap (POR\_POKBAP, shortly named as BGAP) provides reference for the comparator within POR module and also provides reference to external supply monitor circuits. This circuit also includes a Bandgap OK (BGOK) check to ensure bandgap is within expected range. This is a very crude check to ensure BG is not stuck at ground or supply levels.
- System Supply Power OK comparators, POR\_POKHV and POR\_POKLVA, shortly named as POHVH and POKLVA, check the system supply for under voltage and over voltage detection, respectively. POR\_POKHV is enabled after a wait time following BGOK assertion in order to allow BG to settle to expected accuracy. POR\_POKLVA is enabled once POR\_POKHV is set.
- POR\_POKLVB, shortly named POKLVB, checks VDD supply for under voltage detection.

System power ramp-up and reset deassertion sequence is:

1. VDD (and all other supplies) is ramped up ahead of the system supply, VDDA.
2. As VDDA ramps up internal signals follows this ramp, but as soon as supply is sufficiently high for operation of HVPOR circuit HVPORZ signal will be driven to low (reset) state. At the same time HHV signal that is used for level shifter isolation is pulled high.
3. Once VDDA ramps up to the HVPOR threshold, HVPORZ toggles high. HVPOR threshold is set by the minimum supply level required for the proper operation of the custom digital circuits. Width HVPORZ should be wide enough to properly reset the custom digital circuits
4. HVPORZ goes through an analog delay to provide additional time for the reset. Once HVPORZ\_DLY toggles high, HVPORZ synchronized with CLKPOR\_HV to produce HVPORZ\_OR\_OVR signal that is used to generate EN\_BGAP.
5. BGAP circuitry includes a BGOK circuitry that is set once BGAP is settled. BGOK gets synchronized with clock to produce BGOK\_OR\_OVR signal that is used to generate EN\_POKHV.
6. POKHV checks VDDA supply against the BGAP reference and if it is above the threshold level POKHV is set that gets synchronized to clock to produce POKHV\_OR\_OVR signal. POKHV\_OR\_OVR is used to generate EN\_POKLVA and EN\_POKLVB that enables the comparator that checks VMON\_VDDA (vdda) for over voltage and VMON\_VDDCORE (vdd) for under voltage detection.
7. POKLVA and POKLVB check VMON\_VDDA (VDDA) and VMON\_VDDCORE, against bandgap reference and produce synchronized output signals POKLVA\_OR\_OVR and POKLVB\_OR\_OVR. This is the last step in voltage checks within the POR\_IP. POKLVA\_OR\_OVR and POKLVB\_OR\_OVR is fed to the HHVGEN block that combines these two to generate pre HHV signal and further processes the combined signal to implement HHV masking functionality and a minimum of 250- $\mu$ s delay before de-asserting the HHV.
8. HHV signal is then level shifted to VDD domain to generate the reset signal, SOC\_PORZ for SoC.

#### Note

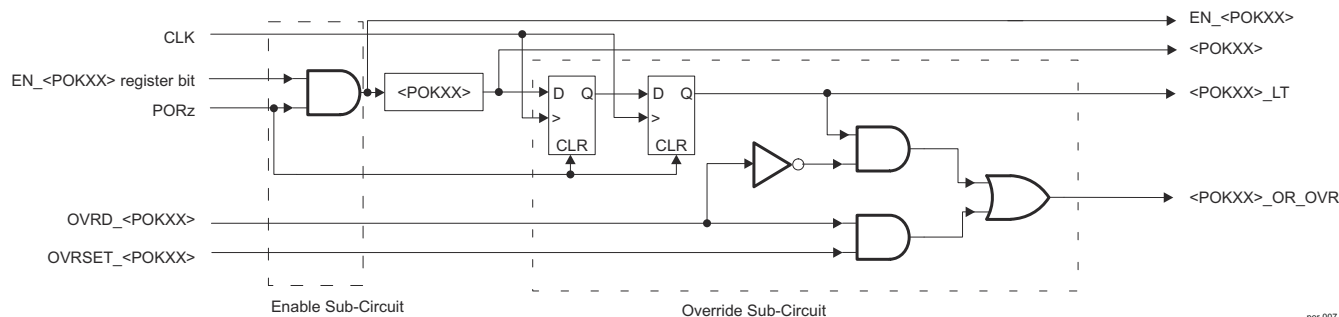
In this family of devices, only the POK functionality of the POR module is used. That is, in step 8, HHV signal is not resetting the SoC.

POR systems response to system power supply ramp down is:

1. Once VDDA cross below the threshold of the POK\_VDDA, POK flags failure and synchronously issues reset to the core domain and asserts HHV. At the same time, in anticipation of failure in VBG state machine resets system back to the step 5 of the above-described system power ramp-up and reset deassertion sequence.
2. As the supply ramps down further, it crosses system POR threshold and a full system reset is issued. As system reboots, the POR starts its operation from the step 1 of the above-described system power ramp-up and reset deassertion sequence.

#### POR Sub-Block Enable and Override

The core of the POR sub-blocks, PORHV, BGAP (BGOK is the comparator output), POKHV and POKLVA and POKLVB have a wrapper around them that controls the enable for these blocks and also process the comparator outputs according to the override. The wrapper architecture is common for all subblocks and is depicted in [Figure 5-13](#) BGAP (BGOK is the comparator output), POKHV, POKLVA and POKLVB.



<POKXX> - is either of BGAP, POKHV, POKLVA and POKLVB

**Figure 5-13. POKHV Wrapper**

Controls, enable, override and override set signal, are not effective until HHV is released.

## Debug and Power-Down Modes

### Note

In this family of devices only POK functionality of POR module is used.

Debug and powerdown mode of the POR module are enabled through PORDEBUGEN\_HV, and PWD\_POR\_HV device pins (both VDDA/1.8V signal), respectively. Truth table for operation modes is given in [Table 5-38](#).

Debug mode enables the override controls for the comparators by forcing the level shifters for these control signals out of HHV state.

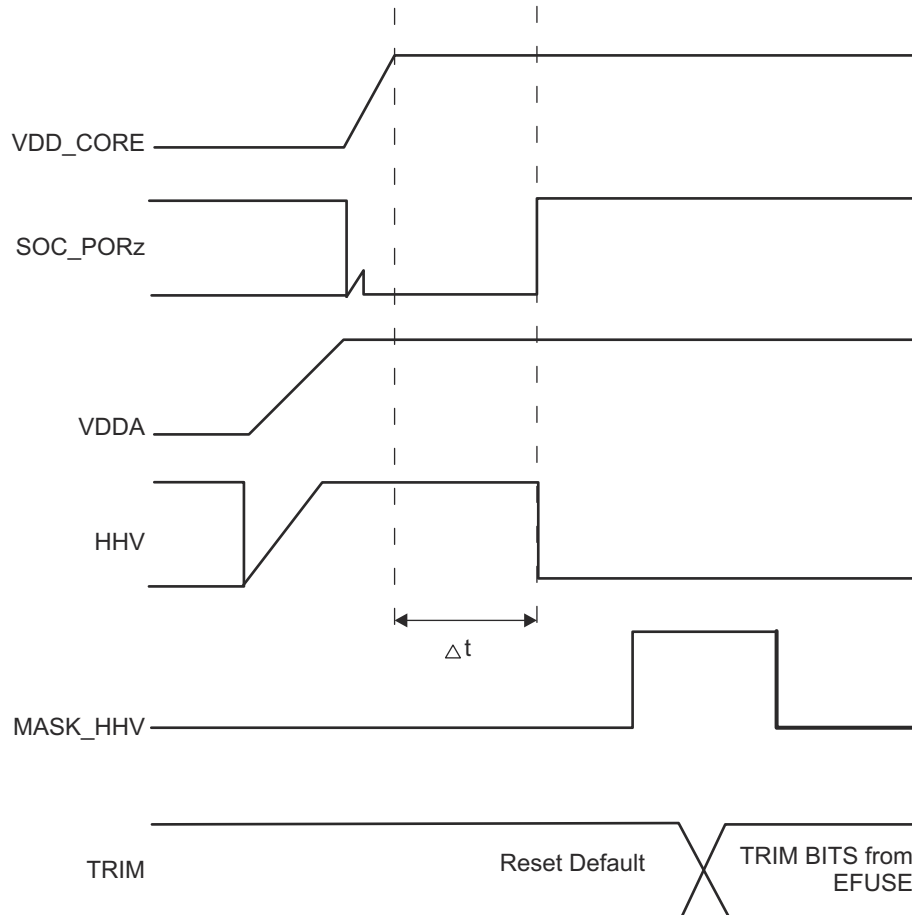
During powerdown PORHV block is disabled at its output is forced to logic low forcing all following blocks into disable state. Also, the HHV signal is forced to logic low and SOC\_PORZ is forced to logic high. During powerdown mode if debug mode is also enabled, subblocks of the POR module may be enabled through override control signals, but SOC\_PORZ and HHV state cannot be modified.

**Table 5-38. POR Operational Mode**

PORDEBUGEN_HV	PWD_POR_HV	VDDA	VDD	HHV	SOC_PORZ	POR Module State
Z	Z	OFF	OFF	Z	Z	Not Powered
Z	Z	OFF	ON	Z	0	Reverse ISO Not Powered
0	0	ON	OFF	1	X	Functional
0	0	ON	ON	0	1	Functional
0	1	ON	ON	0	1	Powerdown
1	0	ON	ON	0	1	DebugMode Operational
1	1	ON	ON	0	1	Mixed Mode (POR_HV powerdown Debug Enable)

## HHV/SOC\_PORZ Masking and Trimming

POR module allows HHV/SOC\_PORZ outputs to be masked after initial reset de-assertion as shown in [Figure 5-14](#). Masking feature of the HHV/SOC\_PORZ should be used before applying the EFUSE trim values in order to avoid re-assertion of the reset due to the transient associated with the new trim values. This feature should be used during functional mode testing of the POR on ATE or bench. HHV Mask may be selected to be applied during normal operation in order to access internal comparator outputs



**Figure 5-14. HHV masking and trimming**

### External (HHVIN\_HV) HHV/SOC\_PORZ Pin Functionality

The POR module gates the internal generated reset signal with the external pin input (HHVIN\_HV) just before outputting HHV.SOC\_PORZ signals. If external reset feature is not available then HHVIN\_HV port of the POR module should be tied low.

### POK operational modes in POR

Modes of operations for the POK blocks in the POR module can be controlled through control signals, HHV, EN\_POK, POK\_SET\_THRESHOLD[8]. [Table 5-39](#) defines the modes of operation.

**Table 5-39. POK Operational Mode**

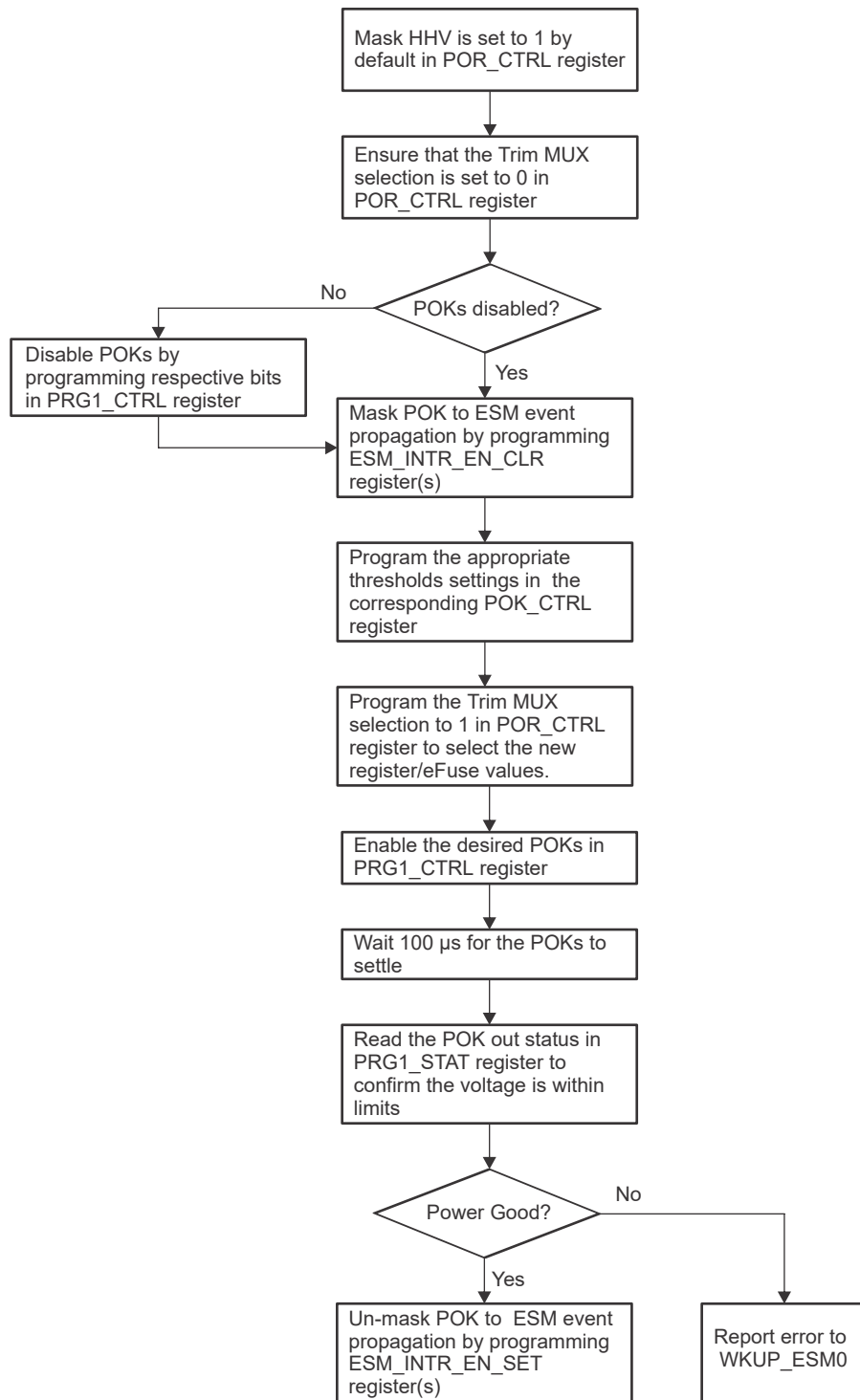
HHV	EN_POK	ENPOK18 (internal)	POK_SET_THRES HOLD<8>	POK_OUT	Comments
1	x	0	x	0	HHV State
1	X	1	0	UVD	Power Up supply Checks
1	X	1	1	OVD	Power Up supply Checks
0	0	0	0	1	Disable State for UVD
0	0	0	1	1	Disable State for OVD
0	1		0	UVD	UVD Mode POK_OUT=1 if power is good.
0	1		1	OVD	OVD Mode POK_OUT=1 if power is good.

#### 5.2.2.1.2.4 POR Programming Model

The three internal POKs for the POR module and their hystereses can be independently enabled from CTRLMMR\_WKUP\_POR\_POKHV\_UV\_CTRL, CTRLMMR\_WKUP\_POR\_POKLV\_UV\_CTRL, CTRLMMR\_WKUP\_POR\_POKHV\_OV\_CTRL, and CTRLMMR\_WKUP\_POR\_BANDGAP\_CTRL. For more information about control registers, see *Control Module (CTRL\_MMR)*.

The three POKs in the POR module are enabled via corresponding bit in CTRLMMR\_WKUP\_WKUP\_PRG1\_CTRL. Their current state can be monitored via corresponding bit in CTRLMMR\_WKUP\_WKUP\_PRG1\_CTRL.

[Figure 5-15](#) shows the programming sequence for the three POKs in the POR module.



**Figure 5-15. POKs in POR Module Programming**



### 5.2.2.1.3 PoR/Reset Generator (PRG) Modules

#### 5.2.2.1.3.1 PRG Overview

A PRG module is responsible for implementing enable/disable controls and event/interrupt masking logic on all POK outputs. [Table 5-40](#) shows PRG modules allocation within device domains.

**Table 5-40. PRG Modules Allocation within Device Domains**

Module Instance	Domain		
	WKUP	MCU	MAIN
WKUP_PRG0	✓	-	-
WKUP_PRG1	✓	-	-
WKUP_PRG2	✓	-	-

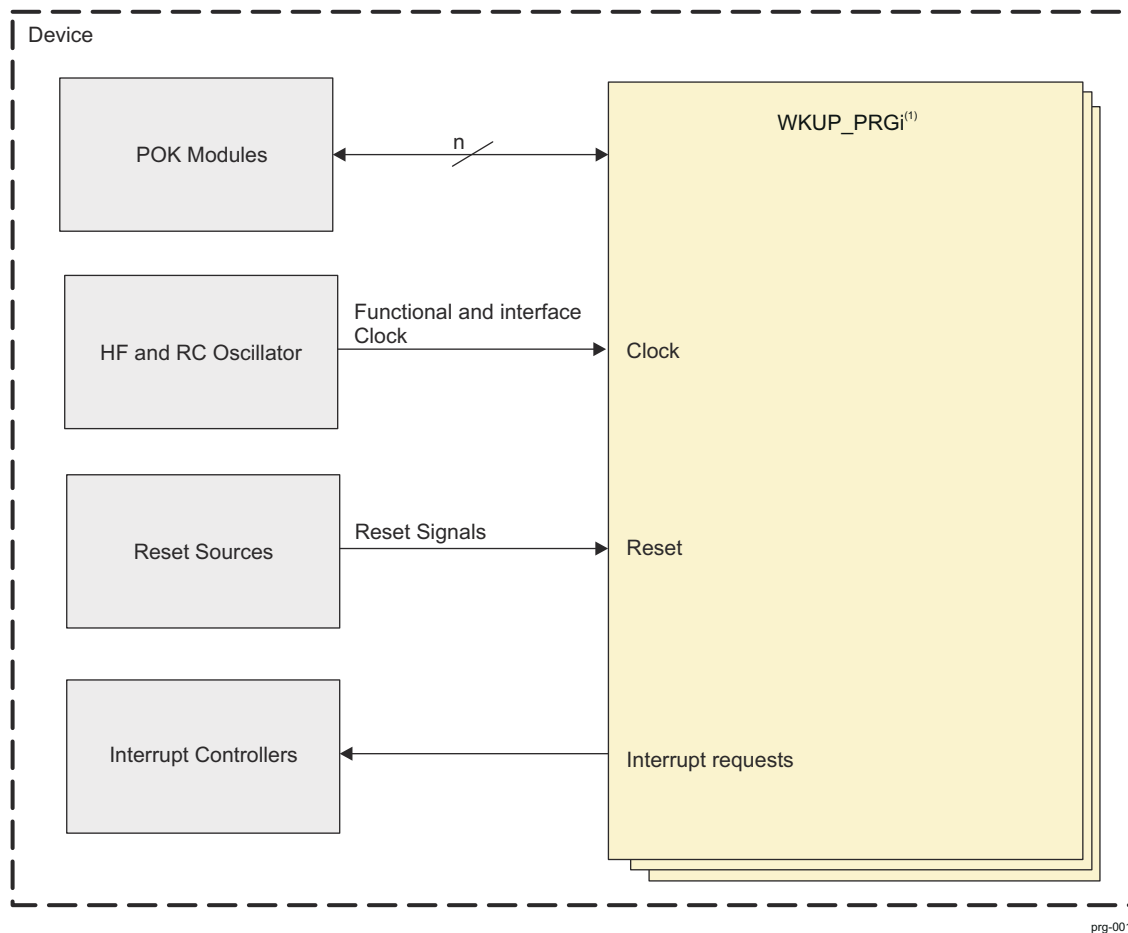
[Table 5-41](#) shows the device domains to which are PRG dedicated for.

**Table 5-41. Dedicated Domains**

Domain/Module	PRG Module
WKUP/MCU	WKUP_PRG0
MCU	WKUP_PRG1 <sup>(1)</sup>
MAIN	WKUP_PRG2

(1) The internal three POKs of POR module are connected to WKUP\_PRG1.

Figure 5-16 shows the PRG module overview.

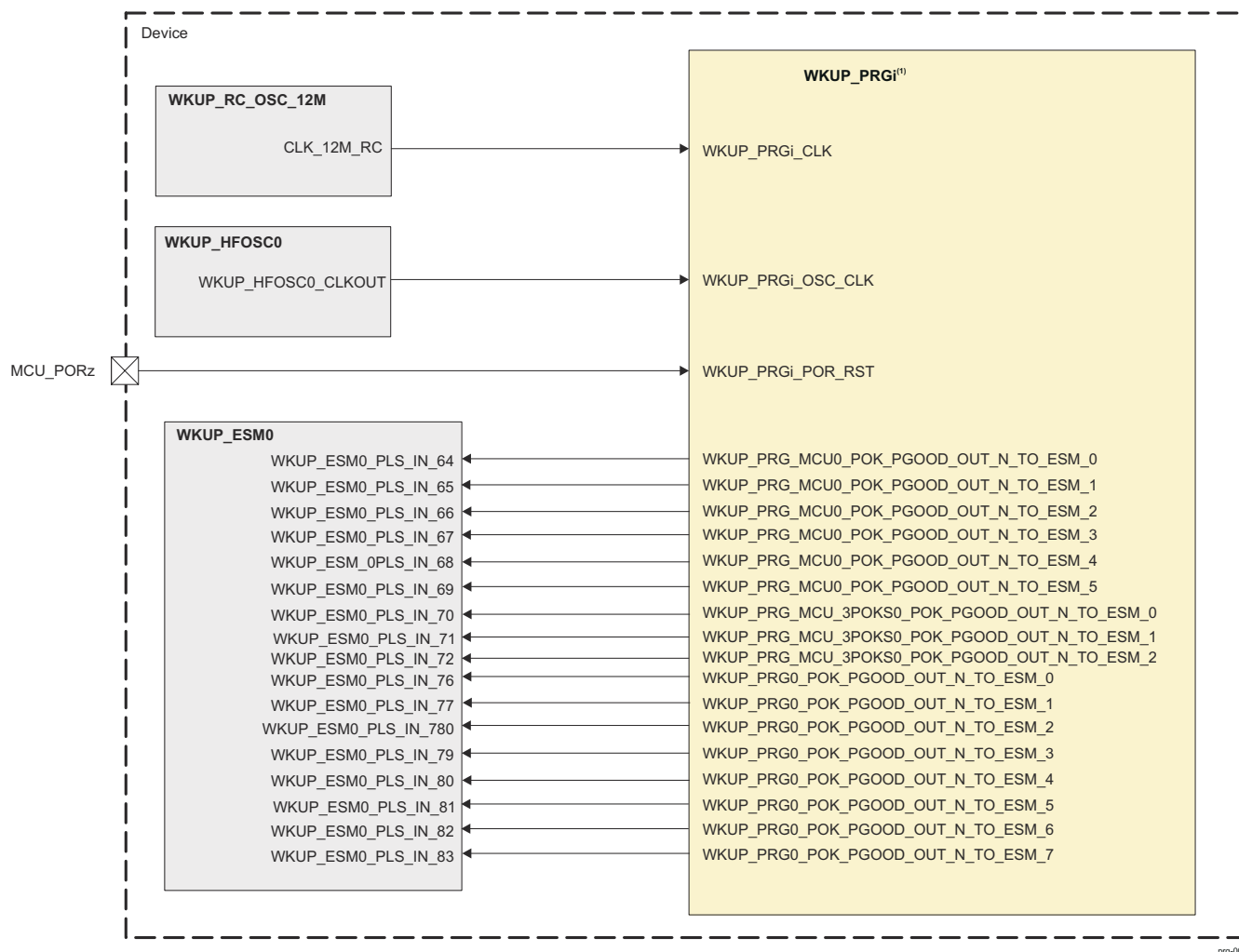


(<sup>1</sup>) i = 0, 1, or 2

**Figure 5-16. PRG Overview**

#### 5.2.2.1.3.2 PRG Integration

Figure 5-17 shows the integration of PRG modules.



(1) i = 0, 1, 2

**Figure 5-17. PRG Integration**

Table 5-42 through Table 5-44 summarize the integration of the PRG modules in the device.

**Table 5-42. PRG Integration Attributes**

Module Instance	Attributes	
	Power Sleep Controller	Power Domain
WKUP_PRG0	No PSC interface	PD0 (WKUP_PSC0)
WKUP_PRG1	No PSC interface	PD0 (WKUP_PSC0)
WKUP_PRG1	No PSC interface	PD0 (WKUP_PSC0)

**Table 5-43. PRG Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
WKUP_PRG0	WKUP_PRG0_CLK	CLK_12M_RC	WKUP_RC_OSC_12M	Clock from internal RC oscillator
	WKUP_PRG0_OSC_CLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	Clock from HFOSC0
WKUP_PRG1	WKUP_PRG1_CLK	CLK_12M_RC	WKUP_RC_OSC_12M	Clock from internal RC oscillator
	WKUP_PRG1_OSC_CLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	Clock from HFOSC0
WKUP_PRG2	WKUP_PRG2_CLK	CLK_12M_RC	WKUP_RC_OSC_12M	Clock from internal RC oscillator
	WKUP_PRG2_OSC_CLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	Clock from HFOSC0

**Table 5-43. PRG Clocks and Resets (continued)**

Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
WKUP_PRG0	WKUP_PRG0_POR_RST	MCU_PORz	MCU_PORz pin	WKUP_PRG0 Power-On Reset
WKUP_PRG1	WKUP_PRG1_POR_RST	MCU_PORz	MCU_PORz pin	WKUP_PRG1 Power-On Reset
WKUP_PRG2	WKUP_PRG2_POR_RST	MCU_PORz	MCU_PORz pin	WKUP_PRG1 Power-On Reset

**Table 5-44. PRG Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
WKUP_PRG0	WKUP_PRG_MCU0_POK_PGO OD_OUT_N_TO_ESM_0	WKUP_ESM0_PLS_IN_6 4	WKUP_ESM0	POK Power-not-good to ESM	Pulse
	WKUP_PRG_MCU0_POK_PGO OD_OUT_N_TO_ESM_1	WKUP_ESM0_PLS_IN_6 5	WKUP_ESM0	POK Power-not-good to ESM	Pulse
	WKUP_PRG_MCU0_POK_PGO OD_OUT_N_TO_ESM_2	WKUP_ESM0_PLS_IN_6 6	WKUP_ESM0	POK Power-not-good to ESM	Pulse
	WKUP_PRG_MCU0_POK_PGO OD_OUT_N_TO_ESM_3	WKUP_ESM0_PLS_IN_6 7	WKUP_ESM0	POK Power-not-good to ESM	Pulse
	WKUP_PRG_MCU0_POK_PGO OD_OUT_N_TO_ESM_4	WKUP_ESM0_PLS_IN_6 8	WKUP_ESM0	POK Power-not-good to ESM	Pulse
	WKUP_PRG_MCU0_POK_PGO OD_OUT_N_TO_ESM_5	WKUP_ESM0_PLS_IN_6 9	WKUP_ESM0	POK Power-not-good to ESM	Pulse
WKUP_PRG1	WKUP_PRG_MCU_3POKS0_P OK_PGOOD_OUT_N_TO_ESM _0	WKUP_ESM0_PLS_IN_7 0	WKUP_ESM0	POK Power-not-good to ESM	Pulse
	WKUP_PRG_MCU_3POKS0_P OK_PGOOD_OUT_N_TO_ESM _1	WKUP_ESM0_PLS_IN_7 1	WKUP_ESM0	POK Power-not-good to ESM	Pulse
	WKUP_PRG_MCU_3POKS0_P OK_PGOOD_OUT_N_TO_ESM _2	WKUP_ESM0_PLS_IN_7 2	WKUP_ESM0	POK Power-not-good to ESM	Pulse
WKUP_PRG2	WKUP_PRG0_POK_PGOOD_O UT_N_TO_ESM_0	WKUP_ESM0_PLS_IN_7 6	WKUP_ESM0	POK Power-not-good to ESM	Pulse
	WKUP_PRG0_POK_PGOOD_O UT_N_TO_ESM_1	WKUP_ESM0_PLS_IN_7 7	WKUP_ESM0	POK Power-not-good to ESM	Pulse
	WKUP_PRG0_POK_PGOOD_O UT_N_TO_ESM_2	WKUP_ESM0_PLS_IN_7 8	WKUP_ESM0	POK Power-not-good to ESM	Pulse
	WKUP_PRG0_POK_PGOOD_O UT_N_TO_ESM_3	WKUP_ESM0_PLS_IN_7 9	WKUP_ESM0	POK Power-not-good to ESM	Pulse
	WKUP_PRG0_POK_PGOOD_O UT_N_TO_ESM_4	WKUP_ESM0_PLS_IN_8 0	WKUP_ESM0	POK Power-not-good to ESM	Pulse
	WKUP_PRG0_POK_PGOOD_O UT_N_TO_ESM_5	WKUP_ESM0_PLS_IN_8 1	WKUP_ESM0	POK Power-not-good to ESM	Pulse
	WKUP_PRG0_POK_PGOOD_O UT_N_TO_ESM_6	WKUP_ESM0_PLS_IN_8 2	WKUP_ESM0	POK Power-not-good to ESM	Pulse
	WKUP_PRG0_POK_PGOOD_O UT_N_TO_ESM_7	WKUP_ESM0_PLS_IN_8 3	WKUP_ESM0	POK Power-not-good to ESM	Pulse

#### 5.2.2.1.3.3 PRG Programming Model

The PRG modules registers used to control and monitor the status of POKs modules are:

- CTRLMMR\_WKUP\_PRG0\_CTRL
- CTRLMMR\_WKUP\_PRG0\_STAT
- CTRLMMR\_WKUP\_PRG1\_CTRL
- CTRLMMR\_WKUP\_PRG1\_STAT
- CTRLMMR\_WKUP\_MAIN\_PRG\_CTRL
- CTRLMMR\_WKUP\_MAIN\_PRG\_STAT

For more information about control registers, see *Control Module (CTRL\_MMR)*.

#### 5.2.2.1.4 Power Glitch Detect (PGD) Modules

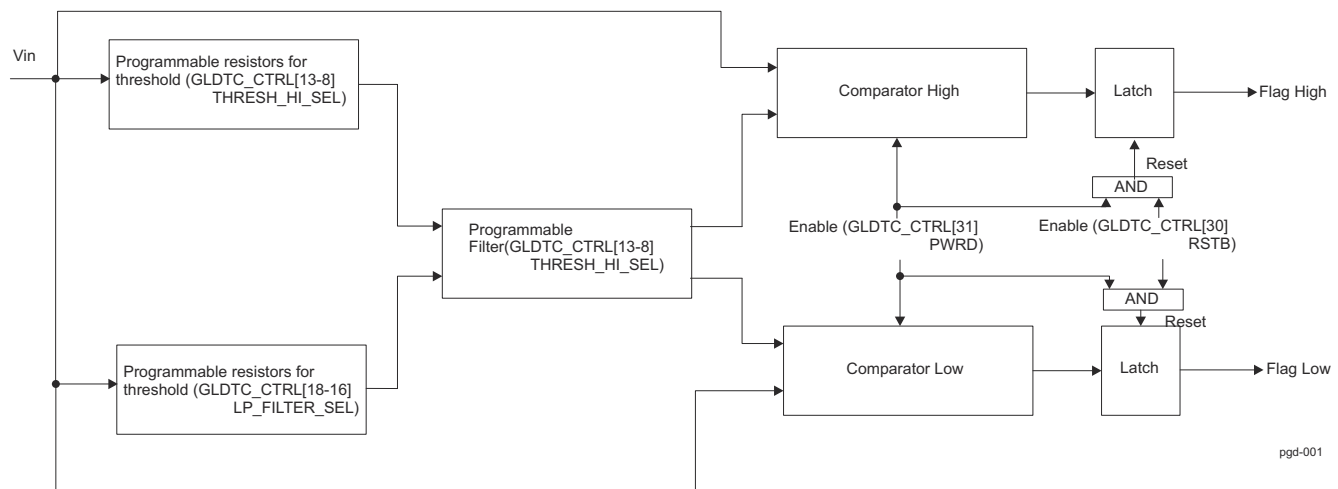
A Power Glitch Detect (PGD) circuit is used to detect short duration “glitches” on the core and MPU power supplies. The PGD provides a low output when no glitch is detected and a high output when a glitch is detected.

Table 5-45 shows PGD modules allocation within device domains.

**Table 5-45. PGD Modules Allocation within Device Domains**

Module Instance	Domain		
	WKUP	MCU	MAIN
MCU_PGD0	-	✓	-
MCU_PGD1	-	✓	-
PGD0	-	-	✓
PGD1	-	-	✓
PGD2	-	-	✓
PGD3	-	-	✓

Figure 5-18 shows the PGD block diagram.



**Figure 5-18. PGD Block Diagram**

Table 5-46 summarize the PGD integration.

**Table 5-46. PGD Integration Summary**

Module Instance	Monitored Voltage	PGD Control Register	PGD Status Register
MCU_PGD0	VDD_MCU	CTRLMMR_WKUP_VDD_MCU_GLDTC_CTRL <sup>(1)</sup>	CTRLMMR_WKUP_VDD_MCU_GLDTC_STAT <sup>(1)</sup>
MCU_PGD1	VDDR_MCU (SRAM)	CTRLMMR_WKUP_VDDR_MCU_GLDTC_CTRL <sup>(1)</sup>	CTRLMMR_WKUP_VDDR_MCU_GLDTC_STAT <sup>(1)</sup>
PGD0	VDD_CPU	CTRLMMR_WKUP_VDD_CPU_GLDTC_CTRL <sup>(1)</sup>	CTRLMMR_WKUP_VDD_CPU_GLDTC_STAT <sup>(1)</sup>
PGD1	VDD_CORE	CTRLMMR_WKUP_VDD_CORE_GLDTC_CTRL <sup>(1)</sup>	CTRLMMR_WKUP_VDD_CORE_GLDTC_STAT <sup>(1)</sup>
PGD2	VDDR_CPU (SRAM)	CTRLMMR_WKUP_VDDR_CPU_GLDTC_CTRL <sup>(1)</sup>	CTRLMMR_WKUP_VDDR_CPU_GLDTC_STAT <sup>(1)</sup>
PGD3	VDDR_CORE (SRAM)	CTRLMMR_WKUP_VDDR_CORE_GLDTC_CTRL <sup>(1)</sup>	CTRLMMR_WKUP_VDDR_CORE_GLDTC_STAT <sup>(1)</sup>

(1) For more information about control registers, see *Control Module (CTRL\_MMR)*.

**Table 5-47. PGD Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCU_PGD0	VDD_MCU_GLDTC_STAT_THRESH_HI_FLAG <sup>(1)</sup>	WKUP_ESM_LVL_EVT_IN_21	WKUP_ESM0	MCU_PGD0 Threshold High Flag level interrupt	Level
	VDD_MCU_GLDTC_STAT_THRESH_LOW_FLAG <sup>(1)</sup>	WKUP_ESM_LVL_EVT_IN_20	WKUP_ESM0	MCU_PGD0 Threshold Low Flag level interrupt	Level
MCU_PGD1	VDDR_MCU_GLDTC_STAT_THRESH_HI_FLAG <sup>(1)</sup>	WKUP_ESM_LVL_EVT_IN_23	WKUP_ESM0	MCU_PGD1 Threshold High Flag level interrupt	Level
	VDDR_MCU_GLDTC_STAT_THRESH_LOW_FLAG <sup>(1)</sup>	WKUP_ESM_LVL_EVT_IN_22	WKUP_ESM0	MCU_PGD1 Threshold Low Flag level interrupt	Level
PGD0	VDD_CORE_GLDTC_STAT_THRESH_HI_FLAG <sup>(1)</sup>	WKUP_ESM_LVL_EVT_IN_25	WKUP_ESM0	PGD0 Threshold High Flag level interrupt	Level
	VDD_CORE_GLDTC_STAT_THRESH_LOW_FLAG <sup>(1)</sup>	WKUP_ESM_LVL_EVT_IN_24	WKUP_ESM0	PGD0 Threshold Low Flag level interrupt	Level
PGD1	VDD_CPU_GLDTC_STAT_THRESH_HI_FLAG <sup>(1)</sup>	WKUP_ESM_LVL_EVT_IN_29	WKUP_ESM0	PGD1 Threshold High Flag level interrupt	Level
	VDD_CPU_GLDTC_STAT_THRESH_LOW_FLAG <sup>(1)</sup>	WKUP_ESM_LVL_EVT_IN_28	WKUP_ESM0	PGD1 Threshold Low Flag level interrupt	Level
PGD2	VDDR_CORE_GLDTC_STAT_THRESH_HI_FLAG <sup>(1)</sup>	WKUP_ESM_LVL_EVT_IN_27	WKUP_ESM0	PGD2 Threshold High Flag level interrupt	Level
	VDDR_CORE_GLDTC_STAT_THRESH_LOW_FLAG <sup>(1)</sup>	WKUP_ESM_LVL_EVT_IN_26	WKUP_ESM0	PGD2 Threshold Low Flag level interrupt	Level
PGD3	VDDR_CPU_GLDTC_STAT_THRESH_HI_FLAG <sup>(1)</sup>	WKUP_ESM_LVL_EVT_IN_31	WKUP_ESM0	PGD3 Threshold High Flag level interrupt	Level
	VDDR_CPU_GLDTC_STAT_THRESH_LOW_FLAG <sup>(1)</sup>	WKUP_ESM_LVL_EVT_IN_30	WKUP_ESM0	PGD3 Threshold Low Flag level interrupt	Level

(1) The interrupt flag is mapped also to the corresponding status register, shown in [Table 5-47](#).

### Note

A flag status bit in corresponding STAT register is cleared by clearing the [30] RSTB bit in the corresponding CTRL register, see for STAT and CTRL registers for a certain PGD module.



### 5.2.2.1.5 Voltage and Thermal Manager (VTM)

#### 5.2.2.1.5.1 VTM Overview

This section describes the Voltage and Thermal Manager (VTM) module in the device.

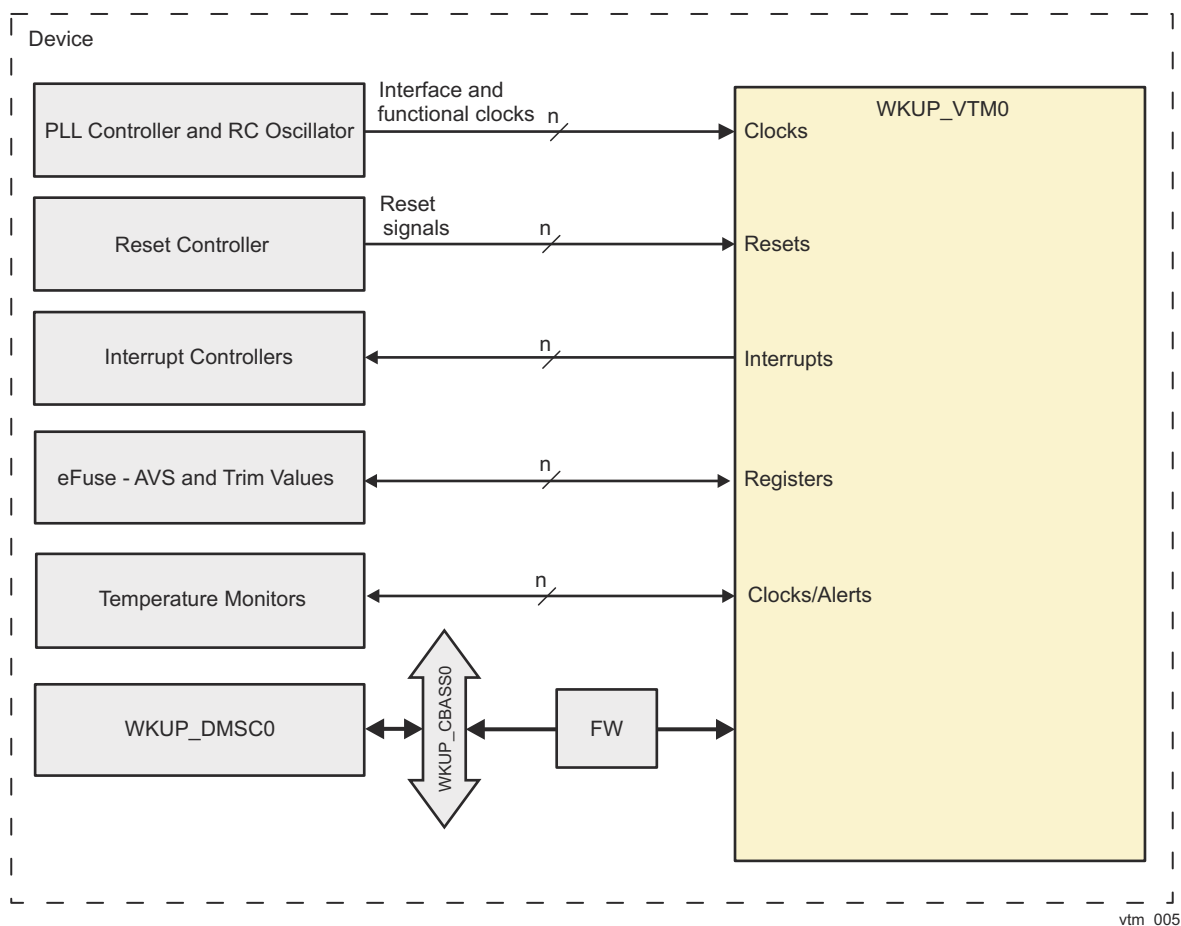
VTM provides the control, status as well as interrupt and event generation related to integrated temperature sensors and thermal events programmed by the user. VTM also contains a set of memory mapped registers that store device-specific operating voltages (AVSVNOM) set during device testing. These values may be used for the system AVS software to adjust operating voltages for the device to achieve optimal operating power.

The device supports one VTM module - WKUP\_VTM0. Table 5-48 shows VTM module allocation within device domains.

**Table 5-48. VTM Module Allocation within Device Domains**

Module Instance	Domain		
	WKUP	MCU	MAIN
WKUP_VTM0	✓	-	-

Figure 5-19 shows VTM module allocation within device domains.



**Figure 5-19. VTM Overview**

#### 5.2.2.1.5.1.1 VTM Features

VTM module supports the following features:

- AVS-Class0 only
- Supports up to 8 temperature monitors

- Programming of temperature-crossing thresholds
- Signals when programmed thresholds are exceeded - up to 3 alerts:
  - Three full reference 10-bit temperature threshold points, THPT1, THPT2 and THPT0.
- Supports up to 8 temperature monitors
- Allows resolution of 0.5°C for temperature reading and threshold point temperature alert/interrupt generation.
- Supports PMIC/LDO set-up with Class-0 VDD-VID settings
- Support of tentative customization of OPP voltage per device (in support of multiple OPPs)
- AVS-voltage or thermal management for up to 8 voltage domains
- Maximum temperature alert
- Supports one shot sampling mode and continuous monitoring mode for the sensors

---

**Note**

The VTM of this family of devices has only 5 temperature sensors.

---

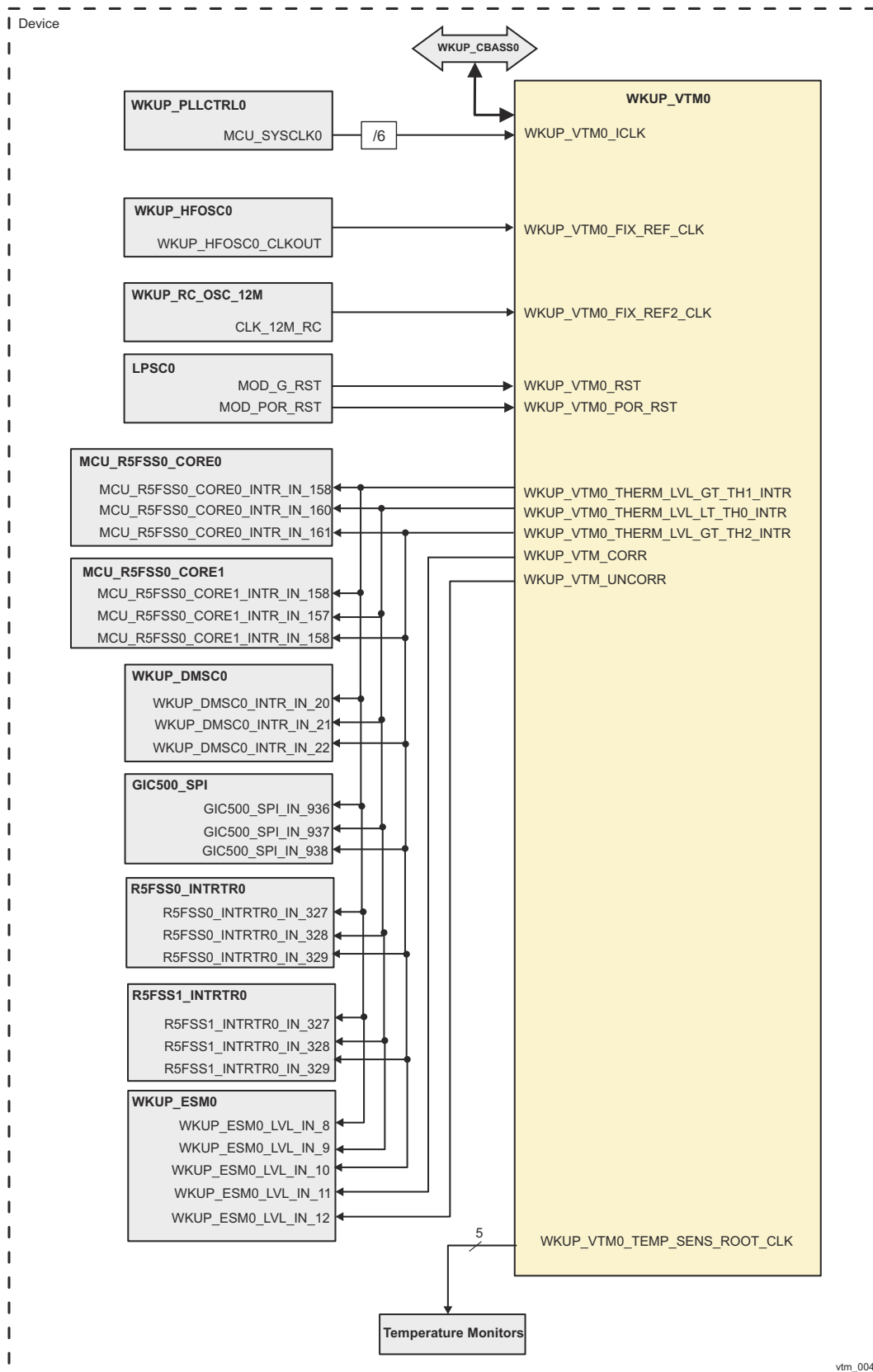
**5.2.2.1.5.1.2 VTM Not Supported Features**

- No support for AVS-Class0 with temperature compensation (AVS0 + TC)
- No integrated I2C inside VTM
- No direct hardware triggering of I2C transactions by VTM
- No support for voltage and thermal management of IO voltage domains
- No oversampling of the temperature reading
- No support for eFuse defaults of all register values

**5.2.2.1.5.2 VTM Integration**

A single VTM module is integrated in the device WKUP domain. [Figure 5-20](#) shows the integration of WKUP\_VTM0.

In this family of devices, five temperature monitors are used.



**Figure 5-20. WKUP\_VTM0 Integration**

Table 5-49 through Table 5-51 summarize the integration of WKUP\_VTM0 in device WKUP domain.

**Table 5-49. WKUP\_VTM0 Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
WKUP_VTM0	WKUP_PSC0	PD0	LPSC0	WKUP_CBASS0

**Table 5-50. WKUP\_VTM0 Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
WKUP_VTM0	WKUP_VTM0_ICLK	MCU_SYSCCLK0/6	WKUP_PLLCTRL0	WKUP_VTM interface clock
	WKUP_VTM0_FIX_R EF_CLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	Reference clock, used for the sensors
	WKUP_VTM0_FIX_R EF2_CLK	CLK_12M_RC	WKUP_RC_OSC_12M	Reference clock, used for the sensors during LPM
WKUP_TEMPSEN SOR0	WKUP_TEMPSENSO R0_CLK	WKUP_VTM0_TEMP_SENS_ROOT_CLK	WKUP_VTM0	Temperature monitor modules clock (1 Mhz to 2 MHz)
TEMPSENSOR0	TEMPSENSOR0_CL K	WKUP_VTM0_TEMP_SENS_ROOT_CLK	WKUP_VTM0	Temperature monitor modules clock (1 Mhz to 2 MHz)
TEMPSENSOR1	TEMPSENSOR1_CL K	WKUP_VTM0_TEMP_SENS_ROOT_CLK	WKUP_VTM0	Temperature monitor modules clock (1 Mhz to 2 MHz)
TEMPSENSOR2	TEMPSENSOR2_CL K	WKUP_VTM0_TEMP_SENS_ROOT_CLK	WKUP_VTM0	Temperature monitor modules clock (1 Mhz to 2 MHz)
TEMPSENSOR3	TEMPSENSOR3_CL K	WKUP_VTM0_TEMP_SENS_ROOT_CLK	WKUP_VTM0	Temperature monitor modules clock (1 Mhz to 2 MHz)
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
WKUP_VTM0	WKUP_VTM0_RST	MOD_G_RST	LPSC0	WKUP_VTM0 Reset
	WKUP_VTM0_POR_ RST	MOD_POR_RST	LPSC0	WKUP_VTM0 Power-On Reset

### Note

Software must disable all sensors before switching the reference clock.

**Table 5-51. WKUP\_VTM0 Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
WKUP_VTM0	WKUP_VTM0_THERM_L VL_GT_TH1_INTR_0	MCU_R5FSS0_CORE0_I NTR_IN_158	MCU_R5FSS0_CORE 0	WKUP_VTM0 over temperature interrupt	Level
		MCU_R5FSS0_CORE1_I NTR_IN_158	MCU_R5FSS0_CORE 1		
		WKUP_DMSC0_INTR_I N_20	WKUP_DMSC0		
		GIC500_SPI_IN_936	GIC500 SPI		
		WKUP_ESM0_LVL_IN_8	WKUP_ESM0		

**Table 5-51. WKUP\_VTM0 Hardware Requests (continued)**

	R5FSS0_INTRTR0_IN_3	R5FSS0_INTRTR0		
	27			
	R5FSS1_INTRTR0_IN_3	R5FSS1_INTRTR0		
	27			
WKUP_VTM0_THERM_L VL_LT_TH0_INTR_0	MCU_R5FSS0_CORE0_I NTR_IN_160	MCU_R5FSS0_CORE 0	WKUP_VTM0 under temperature interrupt	Level
	MCU_R5FSS0_CORE1_I NTR_IN_160	MCU_R5FSS0_CORE 1		
	WKUP_DMSC0_INTR_I N_21	WKUP_DMSC0		
	GIC500_SPI_IN_937	GIC500 SPI		
	WKUP_ESM0_LVL_IN_9	WKUP_ESM0		
	R5FSS0_INTRTR0_IN_3	R5FSS0_INTRTR0		
	28			
	R5FSS1_INTRTR0_IN_3	R5FSS1_INTRTR0		
	27			
WKUP_VTM0_THERM_L VL_GT_TH2_INTR_0	MCU_R5FSS0_CORE0_I NTR_IN_161	MCU_R5FSS0_CORE 0	WKUP_VTM0 maximum temperature interrupt	Level
	MCU_R5FSS0_CORE1_I NTR_IN_161	MCU_R5FSS0_CORE 1		
	WKUP_DMSC0_INTR_I N_22	WKUP_DMSC0		
	GIC500_SPI_IN_938	GIC500 SPI		
	WKUP_ESM0_LVL_IN_1	WKUP_ESM0		
	0			
	R5FSS0_INTRTR0_IN_3	R5FSS0_INTRTR0		
	29			
	R5FSS1_INTRTR0_IN_3	R5FSS1_INTRTR0		
	27			
WKUP_VTM_CORR_LE VEL_0	WKUP_ESM0_LVL_IN_1 1	WKUP_ESM0	WKUP_VTM0 correctable error	Level
WKUP_VTM_UNCORR_ LEVEL_0	WKUP_ESM0_LVL_IN_1 2	WKUP_ESM0	WKUP_VTM0 uncorrectable error	Level

For more information about logical processing of the internal interrupts of VTM, see [Section 5.2.2.1.5.3.2, VTM Temperature Driven Alerts and Interrupts](#).

### 5.2.2.1.5.3 VTM Functional Description

#### 5.2.2.1.5.3.1 VTM Temperature Status and Thermal Management

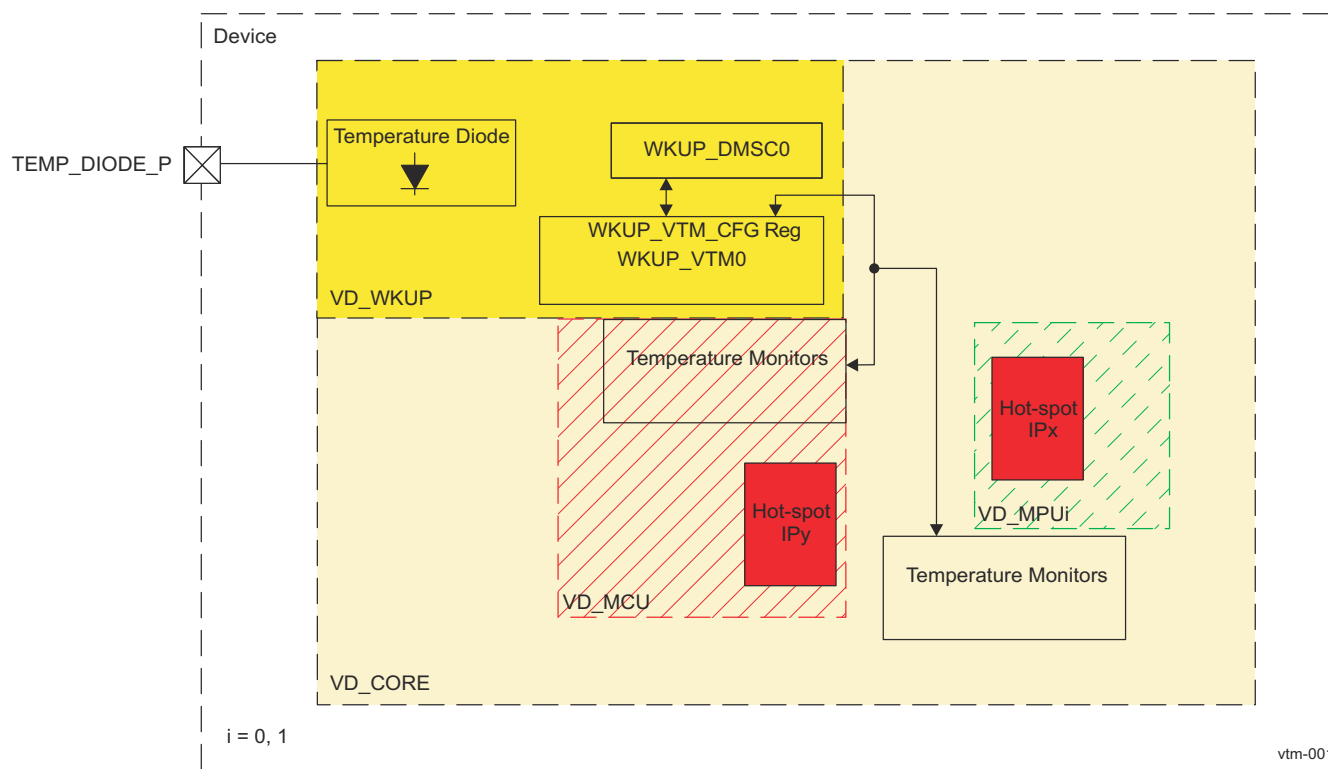
VTM module provides temperature reading and interrupt/alert information for thermal management.

Figure 5-21 shows a high level view of modules required to implement AVS-Class0 and thermal management in the device, VTM included.

VTM controls the temperature monitors in the die. A single VTM can control up to 8 monitors via its registers. The VTM enables the sensors periodically to keep the reported data continually updated because the device sensors are not periodical. The temperature value coming back from the monitor is captured and reported by the VTM registers. The VTM keeps the sensors in reset state when not enabled to save power and reduce sensor usage to maximize the sensor life.

#### Note

This device has 5 integrated temperature monitors.



**Figure 5-21. Top-level IPs for AVS-Class0 and Thermal Management**

#### Note

A temperature diode sensor is used to measure silicon junction temperature, for more information about temperature diode, see the device-specific Datasheet.

#### 5.2.2.1.5.3.1.1 10-bit Temperature Values Versus Temperature

Table 5-52 shows a table lookup method for translating code to temperature from temperature monitors which correspond to the temperature measured read from the VTM0\_VTM\_TMPSENS\_STAT\_J [9-0] DATA\_OUT bitfields. Table 5-52 also provides the values for the temperatures coded in VTM0\_VTM\_MISC\_CTRL2 [25-16] MAXT\_OUTRG\_ALERT\_THR0 and VTM0\_VTM\_MISC\_CTRL2 [9-0] MAXT\_OUTRG\_ALERT\_THR thresholds.

**Table 5-52. Temperature Translation Table**

Parameter Type	Value								
Temperature [°C] (rounded)	-40	-25	0	25	50	75	100	125	150
Temperature [°C] (unrounded)	-40.03	-24.95	0.01	25.02	49.94	75.03	100.02	124.97	150.00
DATA_OUT Code	28	77	164	260	366	485	620	773	949

**Table 5-53** shows a table that gives coefficients of polynomial to calculate code or temperature. The equation to calculate code or temperature is:

$$y = a4 \times x^4 + a3 \times x^3 + a2 \times x^2 + a1 \times x + a0$$

**Table 5-53. Equation Method to Calculate Code or Temperature**

Function	Unit	Coefficient				
		a4	a3	a2	a1	a0
Code = f(Temperature)	Decimal value of code	4.9847E-08	1.6972E-05	6.8759E-03	3.6509E+00	1.6407E+02
Temperature = f(Code)	°C	-9.2627E-12	6.0373E-08	-1.7058E-04	3.2512E-01	-4.9002E+01

#### 5.2.2.1.5.3.2 VTM Temperature Driven Alerts and Interrupts

Each temperature-monitor register group of VTM can be configured to sample the temperature of its corresponding temperature monitor and trigger up to 3 alert (level) signals:

- GT\_TH1\_ALERT (overtemperature alert comparator result) - generated by the 10-bit temperature threshold-point1 (THPT1). Set in VTM\_TMPSENSx\_TH[25-16] TH1\_VAL.
- GT\_TH2\_ALERT (overtemperature alert comparator result) - 10-bit incremental poing generated by threshold-point2 (THPT2). Set in VTM\_TMPSENSx\_TH2[9-0] TH2\_VAL.
- LT\_TH0\_ALERT (undertemperature alert comparator result) - 10-bit incremental poing generated by threshold-point0 (THPT0). Set in VTM\_TMPSENSx\_TH[9-0] TH0\_VAL.

#### Note

The following inequality shall always be valid: THPT2 > THPT1 > THPT0.

#### Note

The device also supports absolute maximum alert (when the die temperature exceeds the programmed temperature in WKUP\_VTM\_MISC\_CTRL2[9-0] MAXT\_OUTRG\_ALERT\_THR) - THERM\_MAXTEMP\_OUTRANGE\_ALERT. For more information about how to handle it, see [Section 5.2.2.1.5.3.7.1, VTM Maximum Temperature Outrange Alert](#).

The level interrupts can be used as follows:

- GT\_TH1\_INT: An early alert for firmware/software to start doing high-temperature thermal management.
- GT\_TH2\_INT: A late interrupt whose level high can be used to trigger a hardware-voltage domain reset.
- LT\_TH0\_INT: A follow-up interrupt to GT\_TH1\_INT which will report to the firmware/software that the temperature has drop to a safe level to eliminate the thermal-management actions triggered in GT\_TH1\_INT.

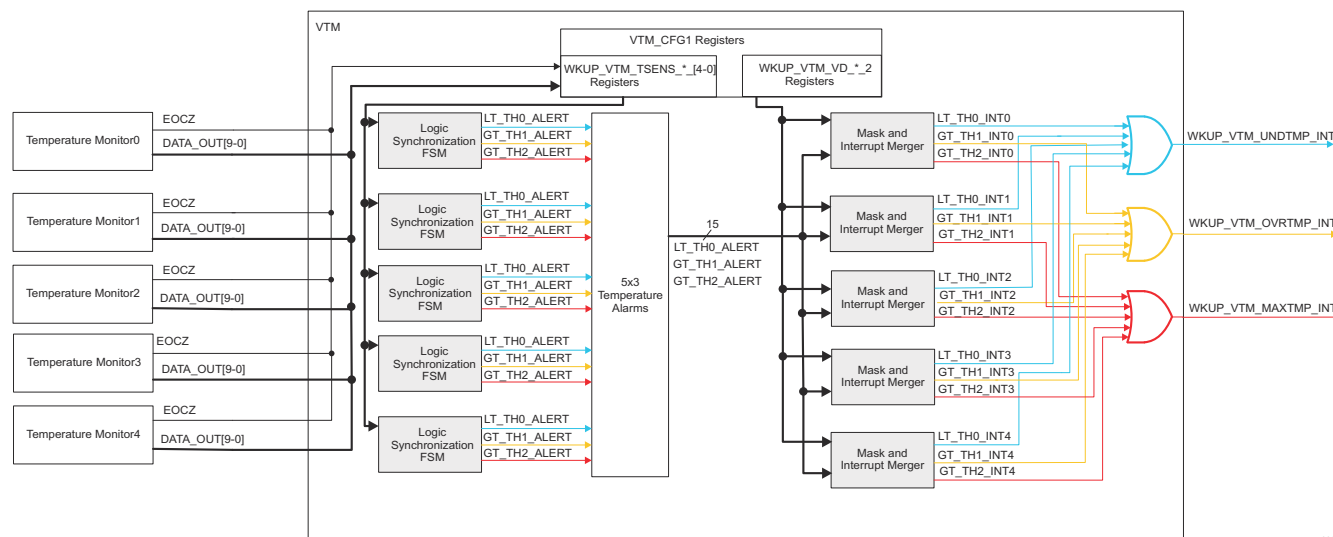
LT\_TH0\_INT, if enabled, will get triggered always when the temperature being read is less than THPT0, regardless of whether GT\_TH1\_INT and GT\_TH2\_INT are enabled, or have ever been triggered. And therefore if LT\_TH0\_INT has to be generated, then firmware/software is responsible to enable the LT\_TH0\_INT only as part of the interrupt service routine of GT\_TH1\_INT and GT\_TH2\_INT. Otherwise it will keep triggering when is not needed.



Each voltage-domain register group can be set to enable and generate the 3 interrupts related to that particular voltage domain. For enabling and disabling the three types of interrupts see the corresponding group of registers:

- for GT\_TH1\_INT - WKUP\_VTM\_GT\_TH1\_INT\_RAW\_STAT\_SET, WKUP\_VTM\_GT\_TH1\_INT\_EN\_STAT\_CLR, WKUP\_VTM\_GT\_TH1\_INT\_EN\_SET, WKUP\_VTM\_GT\_TH1\_INT\_EN\_CLR
- for GT\_TH2\_INT - WKUP\_VTM\_GT\_TH2\_INT\_RAW\_STAT\_SET, WKUP\_VTM\_GT\_TH2\_INT\_EN\_STAT\_CLR, WKUP\_VTM\_GT\_TH2\_INT\_EN\_SET, WKUP\_VTM\_GT\_TH2\_INT\_EN\_CLR
- for LT\_TH0\_INT - WKUP\_VTM\_LT\_TH0\_INT\_RAW\_STAT\_SET, WKUP\_VTM\_LT\_TH0\_INT\_EN\_STAT\_CLR, WKUP\_VTM\_LT\_TH0\_INT\_EN\_SET, WKUP\_VTM\_LT\_TH0\_INT\_EN\_CLR.

All 3 alerts from the 5 temperature monitors are available as inputs of the mask and alert merging logic block of each voltage domain, such that the temperature monitors that are relevant to each voltage domain can be selected as the contributors to the generation of the 3 combined interrupts in each voltage domain (5 shown in sketch; 3 ). This logic is presented in Figure 5-22. The THERM\_MAXTEMP\_OUTRANGE\_ALERT is not shown in this figure. Notice that the same temperature sensor can contribute to more than one voltage domain and each voltage domain can have multiple sensors contributing to the interrupt generation in that VD, see WKUP\_VTM\_VD\_EVT\_SET\_j and WKUP\_VTM\_VD\_EVT\_CLR\_j registers.



**Figure 5-22. VTM Alert and Interrupt Generation**

The interrupts need to be active even when the functional clock is off. In this mode they will be driven by the sensor clock.

#### Note

The interrupts are only active when the sensor is in continuous mode. A one-shot sampling of the sensor will not trigger any interrupts.

Table 5-54 presents the connection of VTM TEMPSENSOR registers groups to voltage domains.

**Table 5-54. VTM TEMPSENSOR Register Groups Mapping to Voltage Domains**

Register Group	Voltage Domain
WKUP_VTM_TMPSENS_*_0	Sensor in VD_WKUP.
WKUP_VTM_TMPSENS_*_1	Sensor in VD_CORE near near MPU region.
WKUP_VTM_TMPSENS2_*_2	Sensor in VD_CC near C7x region.

**Table 5-54. VTM TEMPSensor Register Groups Mapping to Voltage Domains  
(continued)**

WKUP_VTM_TMPSENS_*_3	Sensor in VD_CORE near GPU region.
WKUP_VTM_TMPSENS_*_4	Sensor in VD_CORE near CPSW0 and R5FSS region.
WKUP_VTM_TMPSENS_*[7-6]	Not-used.

There are only 3 interrupts that come out of the VTM module: THERM\_LVL\_LT\_TH0\_INTR, THERM\_LVL\_GT\_TH1\_INTR, THERM\_LVL\_GT\_TH2\_INTR. The interrupt contributions of the 5 voltage domain groups are ORed together to only produce the 3 interrupts that go out. Once an interrupt is detected via DMSC/Compute\_cluster, the firmware/software will check which of the 5 voltage domains is the source of the interrupt.

#### Note

Software shall read the corresponding flags in each of the connected voltage domains to identify which voltage domain is active.

#### 5.2.2.1.5.3.3 VTM VID Voltage Domains

Table 5-55 presents the connection of VTM VD registers groups to voltage domains.

**Table 5-55. VTM VD Register Groups Mapping to Voltage Domains**

Register Group	Voltage Domain
WKUP_VTM_*_VD_*[1-0]	Not-used.
WKUP_VTM_*_VD_*_2	Maps to VD_CC
WKUP_VTM_*_VD_*[7-3]	Not-used.

Table 5-56 shows the relation between the VID bit-field values and voltage.

**Table 5-56. VID Bit-field Values and Their Corresponding Voltage**

VID Bitfield Value, HEX	Voltage, V	VID Bitfield Value, HEX	Voltage, V	VID Bitfield Value, HEX	Voltage, V	VID Bitfield Value, HEX	Voltage, V
0x0-0x1D	Invalid Values	0x3B	0.745	0x59	0.895	0x77	1.045
0x1E	0.6	0x3C	0.75	0x5A	0.9	0x78	1.05
0x1F	0.605	0x3D	0.755	0x5B	0.905	0x79	1.055
0x20	0.61	0x3E	0.76	0x5C	0.91	0x7A	1.06
0x21	0.615	0x3F	0.765	0x5D	0.915	0x7B	1.065
0x22	0.62	0x40	0.77	0x5E	0.92	0x7C	1.07
0x23	0.625	0x41	0.775	0x5F	0.925	0x7D	1.075
0x24	0.63	0x42	0.78	0x60	0.93	0x7E	1.08
0x25	0.635	0x43	0.785	0x61	0.935	0x7F	1.085
0x26	0.64	0x44	0.79	0x62	0.94	0x80	1.09
0x27	0.645	0x45	0.795	0x63	0.945	0x81	1.095
0x28	0.65	0x46	0.8	0x64	0.95	0x82	1.1
0x29	0.655	0x47	0.805	0x65	0.955	0x83	1.11
0x2A	0.66	0x48	0.81	0x66	0.96	0x84	1.12
0x2B	0.665	0x49	0.815	0x67	0.965	0x85	1.13
0x2C	0.67	0x4A	0.82	0x68	0.97	0x86	1.14
0x2D	0.675	0x4B	0.825	0x69	0.975	0x87	1.15
0x2E	0.68	0x4C	0.83	0x6A	0.98	0x88	1.16
0x2F	0.685	0x4D	0.835	0x6B	0.985	0x89	1.17

**Table 5-56. VID Bit-field Values and Their Corresponding Voltage (continued)**

VID Bitfield Value, HEX	Voltage, V	VID Bitfield Value, HEX	Voltage, V	VID Bitfield Value, HEX	Voltage, V	VID Bitfield Value, HEX	Voltage, V
0x30	0.69	0x4E	0.84	0x6C	0.99	0x8A	1.18
0x31	0.695	0x4F	0.845	0x6D	0.995	0x8B	1.19
0x32	0.7	0x50	0.85	0x6E	1	0x8C	1.2
0x33	0.705	0x51	0.855	0x6F	1.005	0x8D	1.21
0x34	0.71	0x52	0.86	0x70	1.01	0x8E	1.22
0x35	0.715	0x53	0.865	0x71	1.015	0x8F	1.23
0x36	0.72	0x54	0.87	0x72	1.02	0x90	1.24
0x37	0.725	0x55	0.875	0x73	1.025	0x91	1.25
0x38	0.73	0x56	0.88	0x74	1.03	0x92-0xFF	Invalid Values
0x39	0.735	0x57	0.885	0x75	1.035		
0x3A	0.74	0x58	0.89	0x76	1.04		

**5.2.2.1.5.3.4 VTM Clocking**

The VTM has 2 reference clock inputs, WKUP\_VTM0\_FIX\_REF\_CLK and WKUP\_VTM0\_FIX\_REF2\_CLK. The WKUP\_VTM0\_FIX\_REF\_CLK is the high precision clock that is used for the sensors. The WKUP\_VTM0\_FIX\_REF2\_CLK is a lower precision clock that can be used during low power scenarios to keep the sensors running when the WKUP\_VTM0\_FIX\_REF\_CLK is shut off. WKUP\_VTM\_CLK\_CTRL[31] TSENS\_CLK\_SEL and WKUP\_VTM\_CLK\_CTRL[4-0] TSENS\_CLK\_DIV are used to switch between the clocks and to choose the divider value. All the sensors in all VDs controlled by this VTM will run off of the same reference clock.

**Note**

Software must disable all sensors before switching the reference clock.

For more information about propagation sequence of the sensor changes, see [Section 5.2.2.1.5.3.7.3, Sensors Programming Sequences](#).

**5.2.2.1.5.3.5 VTM Retention Interface**

VTM has a general purpose retention interface, consistent of 16 sets of 3 signals that can operate totally independently in order to provide retention for 16 signals in the SoC:

- LPMODE\_SLEEP\_RETIN[15:0]
- LPMODE\_SLEEP\_RETEN[15:0]
- LPMODE\_SLEEP\_RETOUT[15:0]

The input LPMODE\_SLEEP\_RETIN[n] is sampled and synchronized continuously while LPMODE\_SLEEP\_RETEN[n] = 0. The synchronized output is send out via LPMODE\_SLEEP\_RETOUT[n]. Once LPMODE\_SLEEP\_RETEN[n] = 1, then the synchronizer will latch the value and continue providing that latched value via LPMODE\_SLEEP\_RETOUT[n].

**5.2.2.1.5.3.6 VTM ECC Aggregator****Note**

For more information about ECC Aggregator functionality, see *ECC Aggregator*.

### 5.2.2.1.5.3.7 VTM Programming Model

#### 5.2.2.1.5.3.7.1 VTM Maximum Temperature Outrange Alert

THERM\_MAXTEMP\_OUTRANGE\_ALERT is used at the device level in order to generate device warm reset. The alert is asserted when the device overheats and exceeds the maximum temperature limit specified for the device, namely 125°C.

The following is the sequence to assert the THERM\_MAXTEMP\_OUTRANGE\_ALERT:

1. For the output to go active ('1'), the VTM has to be configured to operate with at least one of its temperature sensors enabled, and WKUP\_VTM\_TMPSENS\_CTRL\_j[11] MAXT\_OUTRG\_EN = 1 for the corresponding enabled sensor.
2. In addition, WKUP\_VTM\_MISC\_CTRL[0] ANYMAXT\_OUTRG\_ALERT\_EN has to be set to '1'.
3. With the previous configuration set and at least one of the temperature sensors properly configured, and enabled, if the temperature of the die ever exceeds the programmed value in WKUP\_VTM\_MISC\_CTRL2, the VTM output will be driven '1' once the sensor detects the programmed maximum temperature.
4. At the device level the VTM output, THERM\_MAXTEMP\_OUTRANGE\_ALERT, will drive the PLL controllers into warm reset state with the PLL controllers entering CHIP\_0\_RST condition, at the same time that all the PLLs go into clock bypass mode.
5. By the actions described in No.4, the device will considerably reduce its power consumption, which over the passing of a few seconds will cause the device to reduce its temperature.
6. The VTM sensor continues to read the temperature and once the code corresponding to the programmed value in WKUP\_VTM\_MISC\_CTRL2[25-16] MAXT\_OUTRG\_ALERT\_THR0 is detected the VTM drives THERM\_MAXTEMP\_OUTRANGE\_ALERT = 0.
7. After No.6 is fulfilled, the PLL controllers no longer have the THERM\_MAXTEMP\_OUTRANGE\_ALERT driving the CHIP\_0\_RST request and thus the PLL controllers bring the SoC out of reset and the boot sequence re-starts. At the same time the PLLs get out of bypass mode and start with their target programmed frequencies for the boot process.

#### Note

THERM\_MAXTEMP\_OUTRANGE\_ALERT output of VTM is mapped to WKUP\_VTM\_MISC\_CTRL[0] ANY\_MAXT\_OUTRG\_ALERT\_EN.

Notice that exceeding of the programmed value in WKUP\_VTM\_MISC\_CTRL2[9-0] MAXT\_OUTRG\_ALERT\_THR boundary doesn't result in a PORz (cold reset) event, but instead a CHIP\_0\_RST event. Otherwise if cold reset is applied to the MCU/WKUP domains of the device, the VTM will be fully reset and the output THERM\_MAXTEMP\_OUTRANGE\_ALERT will be cleared to '0' immediately.

#### 5.2.2.1.5.3.7.2 Temperature Monitor during Low Power Modes

For more information about device Low Power Modes (LPM), see *Device Power States*.

The following sequence has to be follow in LPM:

1. During MCU\_ONLY LPM VD\_CORE is OFF and VD\_MCU and VD\_WKUP are ON, so:
  - WKUP\_VTM\_TMPSENS\_CTRL\_1[1] TMPSOFF needs to be programmed for the sensor to be disabled and WKUP\_VTM\_TMPSENS\_CTRL\_1, and VTM\_TMPSENS\_TRIM\_1 registers settings for this sensor to go to its reset state.
2. If the exit from low power modes in the device results in the VD\_CORE also being ON, then at that point the remaining TMPSENS modules can be enabled (WKUP\_VTM\_TMPSENS\_CTRL\_1, WKUP\_VTM\_TMPSENS\_TRIM\_1 registers).

#### 5.2.2.1.5.3.7.3 Sensors Programming Sequences

##### Sensors Control Modifications Sequence:

1. Write changes to registers.
2. Wait for 900ns to 1us (long enough for sensor period of slowest 1.15 MHz sensor clock plus some extra).
3. New changes can be written to registers.

### Sensors TRIM Pins Modifications Sequence:

1. Clear WKUP\_VTM\_TMPSENS\_CTRL\_j[6] CLRZ to disable the sensor, so that it has the ENA input low.
2. Wait for 900ns to 1us for the sensor clock to toggle and register the updated ENA pin .
3. Update the WKUP\_VTM\_TMPSENS\_TRIM\_j registers.
4. Wait for 900ns to 1us for the sensor clock to toggle and register to updated TRIM pins while ENA is low.
5. Now the sensor can be enabled normally.

### Reset Sensors Sequence:

1. Set the WKUP\_VTM\_TMPSENS\_CTRL\_j[6] CLRZ bit to enable the sensor (so that it has the ENA input high).
2. Wait for 900ns to 1us for the sensor clock to toggle and register the updated ENA pin.
3. Clear WKUP\_VTM\_TMPSENS\_CTRL\_j[6] CLRZ to disable and reset the sensor (as it requires an ENA transition from 1 to 0 to reset initially) .
4. Wait for 900ns to 1us for the sensor clock to toggle and register the updated ENA pin.
5. Now the sensor can be configured and enabled normally.

#### 5.2.2.1.5.3.8 AVS-Class0

Adaptive Voltage Scaling (AVS) Class 0 is a procedure for lowering the voltage on certain device power rails. AVS Class 0 attempts to normalize the power consumption across all devices. The optimal voltage for each AVS supported rail of each device is determined after analysis in the factory, based on the strength of the device during manufacturing. This value is written in the device eFuse where it can be read through dedicated registers.

AVS can be enabled by system software after primary boot. Refer to *AVS Support* on how to enable AVS based on these registers.

#### 5.2.2.1.6 Distributed Power Clock and Reset Controller (DPCR)

DPCR module is instantiated at the chip level for every power managed sub-chip to support power management features, local clock alignment, DFT power management overrides, isolation controls for self-test and clocking for LBIST. It is also help standardizing interfaces between moduels/sub-chips and central power management, clocking and reset infrastructure at the chip-level.

DPCR contains following sub-modules:

- Distributed Clock Aligner - aligns clock dividers inside power managed block/sub-chip whenever the block goes through power-down/power-up sequence.
- Distributed Clock Aligner for other PLLs - similar functionality as the Distributed Clock Aligner and is used for sub-chips clocked by other PLLs in addition to main/core PLL.
- Distributed RAM Power Controller - is used for power management sequencing for embedded memories. It supports up to eight separate memory groups.
- Distributed Power Chain Controller - controls sequencing of power switch chains inside power managed sub-chips.

All of the above modules are entirely hardware modules, the exception is Distributed RAM Power Controller. It has RAM\_SLEEPMODE pins that are conntrolled via PSC (see corresponding WKUP\_PSC0\_PDCTL\_y[14-12] PDMODE or PSC0\_PDCTL\_y[14-12] PDMODE). [Table 5-57](#) shows the memory power state dependend on the value of the bitfields in PSC registers.

**Table 5-57. RAM\_SLEEPMODE Settings for Memory Power States**

RAM_SLEEPMODE Value	Memory Power State
0x0	Core Off, RAM Array Off, RAM Periphery Off
0x1	Core Off, RAM Array Retention, RAM Periphery off (DeepSleep)
0x2	Reserved <sup>(1)</sup>
0x3	Reserved
0x4	Core Retention, RAM Array Off, RAM Periphery Off

**Table 5-57. RAM\_SLEEPMODE Settings for Memory Power States (continued)**

RAM_SLEEPMODE Value	Memory Power State
0x5	Core Retention, RAM Array Retention, RAM Periphery Off (DeepSleep)
0x6	Reserved
0x7	Reserved
0x8	Core On, RAM Array Off, RAM Periphery Off
0x9	Core On, RAM Array Retention, RAM Periphery Off (DeepSleep)
0xA	Core On, RAM Array Retention, RAM Periphery Off (LightSleep)
0xB	Core On, RAM Array Retention, RAM Periphery On
0xC	Reserved
0xD	Reserved
0xE	Reserved
0xF	Core On, RAM Array On, RAM Periphery On

(1) Reserved values default to 0xF (Core On, RAM Array and Periphery On)

### 5.2.2.2 Power Control Modules

The Power Control Modules are divided into two section, dependent on their functionality - Power Sleep Controller and Local Power Sleep section and Integrated Power Management (DMSC) section.

#### 5.2.2.2.1 Power Sleep Controller and Local Power Sleep Controllers

The Power Sleep Controller (PSC) includes a Global Power Sleep Controller (GPSC) and a number of Local Power Sleep Controllers (LPSC) that control overall device power by turning off unused power domains and gating off clocks to individual peripherals and modules. The PSC provides the user with an interface to control several important power and clock operations.

The device has two PSC - WKUP\_PSC0 and PSC0 in WKUPSS and MAIN SoC, respectively.

##### 5.2.2.2.1.1 PSC Terminology

Term	Definition
<b>LPSC</b>	Local Power Sleep Controller; one per module (domain)
<b>GPSC</b>	Global Power/Sleep Controller; manages the LPSCs
<b>PSC</b>	Power Sleep Controller, including one GPSC and multiple LPSCs

##### 5.2.2.2.1.2 PSC Features

The PSC includes the following features:

- Provides software interface to:
  - Control module power on or off
  - Control module clock on or off
- Supports emulation features: power, clock, and reset.

For more information on how to control the PSC see [Section 5.2.2.2.1.5, PSC: Executing State Transitions and PSC Registers](#).

#### Note

PSC functions are controlled via DMSC. For more information how to use DMSC, see TISCI API available at ti.com.

##### 5.2.2.2.1.3 PSC: Device Power-Management Layout

The device is divided into a number of separate voltage domains each containing specific processing cores and peripherals. This division enables the device to achieve lower power dissipation profiles by allowing the power supplies to unused domains to be completely turned off.

Each PSC in the device has several power domains that can be turned on for operation or off to minimize power dissipation. The Global Power Sleep Controller (GPSC) is used to control the power gating of various power domains.

Clock gating to each logic block is managed by the Local Power Sleep Controllers (LPSCs). For modules with a dedicated clock or multiple clocks, the LPSC communicates with the PLL controller to enable and disable that module's clock(s) at the source. For modules that share a clock with other modules, the LPSC controls the clock gating logic for each module.

##### 5.2.2.2.1.3.1 WKUP\_PSC0 Device-Specific Information

[Table 5-58](#) provides the device-level view with module asSoCiations to the clock, power, and voltage domains.



**Table 5-58. WKUP\_PSC0 Power Management Device-Level Layout**

VD Name	PD Name	PD Index	LPSC Name	LPSC Index	Modules
VD_WKUP/MCU	GP_CORE_CTL_WKUP	0	LPSC_WKUP_ALWAYS ON	0	MCU_PDMA_G2_0, MCU_PDMA_ADC0, MCU_CLK8_ECC_AGGR0_CFG, MCU_PDMA_G1_0, MCU_PDMA_G0_0, MCU_CPT2_PROBEs, MCU_CBASS_FW0, MCU_ARM_ATB_FUNNEL0, MCU_CPT2_AGGREGATOR0, MCU_CBASS0, MCU_NAVSS0, MCU_CPSW_2GUSS0, MCU_FSS0, MCU_SPI0, MCU_SPI1, MCU_SPI2, MCU_UART0, MCU_I2C0, MCU_I2C1, MCU_TIMER0, MCU_TIMER1, MCU_TIMER2, MCU_TIMER3, MCU_TIMER4, MCU_TIMER5, MCU_TIMER6, MCU_TIMER7, MCU_TIMER8, MCU_TIMER9, MCU_DCC0, MCU_DCC1, MCU_DCC2, MCU_ESM0, MCU_PSR0M0, MCU_SRAM0, MCU_SEC_MMR0, MCU_PLL_CFG0, MCU_CTRL_MMR0, WKUP_DPPA0, WKUP_GPIOMUX_INTRTR0, WKUP_CLK4_ECC_AGGR0_CFG, WKUP_VTM0, WKUP_CBASS_FW0, WKUP_CBASS0, WKUP_CTRL_MMR0, WKUP_PSC0, WKUP_CTRL_MMR0, WKUP_ESM0
			LPSC_DMSC	1	WKUP_DMSC0
			LPSC_DEBUG2DMSC	2	-
			LPSC_WKUP_GPIO	3	WKUP_GPIO0, WKUP_UART0, WKUP_I2C0
			LPSC_WKUPMCU2MAIN	4	-
			LPSC_MAIN2WKUPMCU	5	-
			LPSC_MCU_TEST	6	MCU_EFUSE0, MCU_PBI0ST0
			LPSC_MCU_DEBUG	7	-
			LPSC_MCU_MCAN_0	8	MCU_MCANSS0
			LPSC_MCU_MCAN_1	9	MCU_MCANSS1
			LPSC_MCU_OSPI_0	10	MCU_FSS_OSPI0
			LPSC_MCU_OSPI_1	11	MCU_FSS_OSPI1
			LPSC_MCU_HYPERBUS	12	MCU_FSS0_HPB0
			LPSC_MCU_I3C_0	13	MCU_I3C0
			LPSC_MCU_I3C_1	14	MCU_I3C1
			LPSC_MCU_ADC_0	15	MCU_ADC0
			LPSC_MCU_ADC_1	16	MCU_ADC1
			LPSC_WKUP_SPARE0	17	-
			LPSC_WKUP_SPARE1	18	-
	PD_MCU_R5FF0FSS	1	LPSC_MCU_R5FF0_0	19	MCU_RT0I0, MCU_R5FF0_CORE0
			LPSC_MCU_R5FF0_1	20	MCU_RT0I1, MCU_R5FF0_CORE1
			LPSC_MCU_R5FF0FSS_PBI0ST_0	21	MCU_R5FF0_PBI0ST

### Note

For details on power domain state transitions, please refer to [Section 5.2.2.2.1.5, Executing State Transitions](#).

[Table 5-59](#) presents presents Power Domain features for WKUP\_PSC0.

**Table 5-59. WKUP\_PSC0 Power Domain Features**

PD index	PD name	GP(always on)/PD/ PDM(memory power domain only)	Default Power Domain state	PD State Software Controlled
0	GP_CORE_CTL_WKUP	GP <sup>(1)</sup>	AO <sup>(3)</sup>	NO
1	PD_MCU_R5FF0FSS	PD <sup>(2)</sup>	ON	YES

(1) Group with common power - means that there are no power switches. The group is considered always-on.

(2) Power switched domain - means this is a full power switched domain, there are logic power-switches and memory power-switches if there are memories.

(3) AO = AlwaysOn

[Table 5-60](#) presents LPSCs features.

**Table 5-60. WKUP\_PSC0 LPSC Features**

LPSC Index	LPSC Name	Default LPSC State	Efuse Disable Availability	LPSC State Software Controlled	Reset Isolation
0	LPSC_WKUP_ALWAYSON	ON	N	NO	N
1	LPSC_DMSC	ON	N	YES	N
2	LPSC_DEBUG2DMSC	ON	N	YES	N
3	LPSC_WKUP_GPIO	ON	N	YES	N
4	LPSC_WKUPMCU2MAIN	OFF	N	YES	N
5	LPSC_MAIN2WKUPMCU	ON	N	YES	N
6	LPSC_MCU_TEST	ON	N	YES	N
7	LPSC_MCU_DEBUG	ON	N	YES	N
8	LPSC_MCU_MCAN_0	ON	Y	YES	N
9	LPSC_MCU_MCAN_1	ON	Y	YES	N
10	LPSC_MCU_OSPI_0	ON	Y	YES	N
11	LPSC_MCU_OSPI_1	ON	Y	YES	N
12	LPSC_MCU_HYPERBUS	ON	Y	YES	N
13	LPSC_MCU_I3C_0	OFF	N	YES	N
14	LPSC_MCU_I3C_1	OFF	N	YES	N
15	LPSC_MCU_ADC_0	OFF	N	YES	N
16	LPSC_MCU_ADC_1	OFF	N	YES	N
17	LPSC_WKUP_SPARE0	OFF	Y	YES	N
18	LPSC_WKUP_SPARE1	OFF	Y	YES	N
19	LPSC_MCU_R5FF0_0	OFF	N	YES	N
20	LPSC_MCU_R5FF0_1	OFF	N	YES	N
21	LPSC_MCU_R5FF0FSS_PBIST_0	OFF	N	YES	N

### Note

For details on module state transitions, see [Section 5.2.2.2.1.5, PSC: Executing State Transitions](#).

### 5.2.2.2.1.3.2 PSC0 Device-Specific Information

Table 5-61 provides the device-level view with module associations to the clock, power, and voltage domains.

**Table 5-61. PSC0 Power Management Device-Level Layout**

VD Name	PD Name	PD Index	LPSC Name	LPSC Index	Modules
VD_CORE	GP_CORE_CTL	0	LPSC_MAIN_ALWAYSON	0	R5FSS0_INTRTR0, R5FSS1_INTRTR0, CSI_PSILSS0, CPT2_HC_AGGRO, CPT2_AC_AGGRO, C66SS1_INTRTR0, GPIO2, GPIO3, GPIO4, GPIO5, GPIO6, GPIO7, PDMA_UART_PSILSS0, MSRAM0, PDMA_MISC_PSILSS0, PDMA_AASRC_PSILSS0, PDMA_DEBUG_PSILSS0, PDMA_UART_G2, PDMA_UART_G1, PDMA_UART_G0, PDMA_G3, PDMA_G2, PDMA_G1, PDMA_G0, PDMA_MCASP_G1, PDMA_MCASP_G0, PDMA_AASRC0, C66SS0_INTRTR0, CSITXRX_CLK4_ECC_AGGRO, MAINCLK4_ECC_AGGR1, MAINCLK2_ECC_AGGRO, INFRACLK2_ECC_AGGRO_CFG, NAVSS0_ECC_AGGRO, DEBUG_SUSPENDRTR0, MAIN2MCU_PLS_INTRTR0, TIMESYNC_INTRTR0, CMP_EVT_ROUTER0, GPIOMUX_INTRTR0, MAIN2MCU_LVL_INTRTR0, CPT2_PROBES, CBASS_FW0, ESM0, DCC0, DCC1, DCC2, DCC3, DCC4, DCC5, DCC6, DCC7, DCC8, DCC9, DCC10, DCC11, DCC12, PLL_CTRL0, PSC0, SEC_MMR0, PLL0_CFG, CTRL_MMR0, eFUSE0, GTC0, BIST, CPT2_AGGRO, NAVSS0, PSRAM_ARM_C71_BOOTVECTOR, PSRAM_ARM_C66_BOOTVECTOR, TIMER4, TIMER5, TIMER6, TIMER7, TIMER8, TIMER9, TIMER10, TIMER11, TIMER12, TIMER13, TIMER14, TIMER15, TIMER16, TIMER17, TIMER18, TIMER19, GPIO0, GPIO1, MSMC0, GIC0
			LPSC_MAIN_TEST	1	DFTSS0
			LPSC_MAIN_PBIT	2	PBIST_RC_NB, PBIST_RC_NAVSS, PBIST_HC, PBIST_INFRA0
			LPSC_PER_AUDIO	3	ASRC0, MCASP0, MCASP1, MCASP2, MCASP3, MCASP4, MCASP5, MCASP6, MCASP7, MCASP8, MCASP9, MCASP10, MCASP11
			LPSC_PER_ATL	4	ATL0
			LPSC_PER_MLB	5	MLBSS0
			LPSC_PER_MOTOR	6	ECAP0, ECAP1, ECAP2, EQEP0, EQEP1, EQEP2, EPWM0, EPWM1, EPWM2, EPWM3, EPWM4, EPWM5
			LPSC_PER_MISCIO	7	UART0, UART1, UART2, UART3, UART4, UART5, UART6, UART7, UART8, UART9, MCSPI0, MCSPI1, MCSPI2, MCSPI3, MCSPI4, MCSPI5, MCSPI6, MCSPI7, I2C0, I2C1, I2C2, I2C3, I2C4, I2C5, I2C6
			LPSC_PER_GPMC	8	ELM0, GPMC0
			LPSC_PER_VPFE	9	VPFE0
			LPSC_PER_VPE	10	-
			LPSC_PER_SPARE0	11	-
			LPSC_PER_SPARE1	12	-

**Table 5-61. PSC0 Power Management Device-Level Layout (continued)**

VD Name	PD Name	PD Index	LPSC Name	LPSC Index	Modules		
			LPSC_MAIN_DEBUG	13	C66_DEBUG_CELL0, PDMA_DEBUG_C66, PDMA_DEBUG_CCMCU, DBG_CBASS0, C66_DEBUG_CELL1, DEBUG_CELL1, DEBUG_CELL0,CC_DEBUG_CELL0,CXSTM500SS0, DEBUGSS_CV0		
			LPSC_EMIF_DATA_0	14	-		
			LPSC_EMIF_CFG_0	15	DDRSS0		
			LPSC_EMIF_DATA_1	16	-		
			LPSC_EMIF_CFG_1	17	-		
VD_CC			LPSC_PER_SPARE2	18	-		
			LPSC_CC_TOP_PBIST	19	PBIST_CC_Top		
VD_CORE			LPSC_USB_0	20	USBSS0		
			LPSC_USB_1	21	USBSS1		
			LPSC_USB_2	22	-		
			LPSC_MMC4B_0	23	MMCSD1		
			LPSC_MMC4B_1	24	MMCSD2		
			LPSC_MMC8B_0	25	MMCSD0		
			LPSC_UFS_0	26	UFS0		
			LPSC_UFS_1	27	-		
			LPSC_PCIE_0	28	PCIE0		
			LPSC_PCIE_1	29	PCIE1		
			LPSC_PCIE_2	30	PCIE2		
			LPSC_PCIE_3	31	PCIE3		
			LPSC_SAUL	32	-		
			LPSC_PER_I3C	33	I3C0		
				PD_MCANSS	1	LPSC_MAIN_MCANSS_0	34
					LPSC_MAIN_MCANSS_1	35	MCANSS1
					LPSC_MAIN_MCANSS_2	36	MCANSS2
					LPSC_MAIN_MCANSS_3	37	MCANSS3
					LPSC_MAIN_MCANSS_4	38	MCANSS4
					LPSC_MAIN_MCANSS_5	39	MCANSS5
					LPSC_MAIN_MCANSS_6	40	MCANSS6
					LPSC_MAIN_MCANSS_7	41	MCANSS7
					LPSC_MAIN_MCANSS_8	42	MCANSS8
					LPSC_MAIN_MCANSS_9	43	MCANSS9

**Table 5-61. PSC0 Power Management Device-Level Layout (continued)**

VD Name	PD Name	PD Index	LPSC Name	LPSC Index	Modules
			LPSC_MAIN_MCA_NSS_10	44	MCANSS10
			LPSC_MAIN_MCA_NSS_11	45	MCANSS11
			LPSC_MAIN_MCA_NSS_12	46	MCANSS12
			LPSC_MAIN_MCA_NSS_13	47	MCANSS13
	PD_DSS	2	LPSC_DSS	48	DSS0
			LPSC_DSS_PBI	49	PBI
			LPSC_DSI	50	DSS_DSI0
			LPSC_EDP_0	51	EDP_PHY0, DSS_EDP0
			LPSC_EDP_1	52	-
			LPSC_CSIRX_0	53	CSI_RX_IF0
			LPSC_CSIRX_1	54	CSI_RX_IF1
			LPSC_CSIRX_2	55	-
			LPSC_CSITX_0	56	CSI_TX_IF0
			LPSC_TX_DPHY_0	57	DPHY_TX0
			LPSC_CSIRX_PHY_0	58	DPHY_RX0
			LPSC_CSIRX_PHY_1	59	DPHY_RX1
			LPSC_CSIRX_PHY_2	60	-
	PD_ICSS	3	LPSC_ICSSG_0	61	PRU_ICSSG0
			LPSC_ICSSG_1	62	PRU_ICSSG1
	PD_9GSS	4	LPSC_9GSS	63	CPSW0
	PD_SERDES_0	5	LPSC_SERDES_0	64	SERDES0
	PD_SERDES_1	6	LPSC_SERDES_1	65	SERDES1
	PD_SERDES_2	7	LPSC_SERDES_2	66	SERDES2
	PD_SERDES_3	8	LPSC_SERDES_3	67	SERDES3
	PD_SERDES_4	9	LPSC_SERDES_4	68	SERDES4
	PD_SERDES_5	10	LPSC_SERDES_5	69	-
	PD_TIMER	11	LPSC_DMTIMER_0	70	TIMER0
			LPSC_DMTIMER_1	71	TIMER1
			LPSC_DMTIMER_2	72	TIMER2
			LPSC_DMTIMER_3	73	TIMER3
VD_CC	PD_C71x_0	12	LPSC_C71x_0	74	MMA0, C71SS0, RTI_16
			LPSC_C71x_0_PBI	75	PBI for C71SS0
	PD_C71x_1	13	LPSC_C71x_1	76	-
			LPSC_C71x_1_PBI	77	-
	PD_A72_CLUSTER_0	14	LPSC_A72_CLUSTER_0	78	A72 cluster
			LPSC_A72_CLUSTER_0_PBI	79	PBI for A72 cluster
	PD_A72_0	15	LPSC_A72_0	80	RTI_0

**Table 5-61. PSC0 Power Management Device-Level Layout (continued)**

VD Name	PD Name	PD Index	LPSC Name	LPSC Index	Modules
	PD_A72_1	16	LPSC_A72_1	81	RTI_1
	PD_A72_CLUSTER_1	17	LPSC_A72_CLUSTER_1	82	-
			LPSC_A72_CLUSTER_1_PBI	83	-
	PD_A72_2	18	LPSC_A72_2	84	-
	PD_A72_3	19	LPSC_A72_3	85	-
VD_CORE	PD_GPUCOM	20	LPSC_GPUCOM	86	GPU_COMMON, RTI_15
			LPSC_GPUPBI	87	PBI_GPU0
	PD_GPUCORE	21	LPSC_GPUCORE	88	GPU_CORE
	PD_C66x_0	22	LPSC_C66X_0	89	RTI_24, C66_DSPSS0
			LPSC_C66X_PBI_T_0	90	PBI_C66_DSPSS0
	PD_C66x_1	23	LPSC_C66X_1	91	RTI_25, C66_DSPSS1
			LPSC_C66X_PBI_T_1	92	PBI_C66_DSPSS1
	PD_R5FSS_0	24	LPSC_R5FSS0_CORE0	93	RTI_28, R5FSS0_CORE0
			LPSC_R5FSS0_CORE1	94	RTI_29, R5FSS0_CORE1
			LPSC_R5FSS_PBI_ST_0	95	PBI_R5FSS0
	PD_R5FSS_1	25	LPSC_R5FSS1_CORE0	96	RTI_30, R5FSS1_CORE0
			LPSC_R5FSS1_CORE1	97	RTI_31, R5FSS1_CORE1
			LPSC_R5FSS_PBI_ST_1	98	PBI_R5FSS1
	PD_DECODE	26	LPSC_DECODE_0	99	D5520MPx
			LPSC_DECODE_PBI	100	PBI_DECODER
	PD_ENCODE	27	LPSC_ENCODE_0	101	VXE384MP2
			LPSC_ENCODE_PBI	102	PBI_ENCODER
	PD_DMPAC	28	LPSC_DMPAC	103	DMPAC_DOF0, DMPAC0
			LPSC_SDE	104	DMPAC_SDE0
			LPSC_DMPAC_PBI_ST	105	PBI_DMPAC
	PD_VPAC	29	LPSC_VPAC	106	VPAC_NF0, VPAC0_MSC, VPAC_LDC0, VPAC_VISS0, VPAC0
			LPSC_VPAC_PBI_T	107	PBI_VPAC0

**Note**

For details on power domain state transitions, please refer to [Section 5.2.2.2.1.5, Executing State Transitions](#).

[Table 5-62](#) presents Power Domain features for PSC0.

**Table 5-62. PSC0 Power Domain Features**

PD index	PD name	GP(always on)/PD/ PDM(memory power domain only)	Default Power Domain state	PD State Software Controlled
0	GP_CORE_CTL	GP <sup>(1)</sup>	AO <sup>(4)</sup>	YES
1	PD_MCANSS	PDRIO <sup>(2)</sup>	AO <sup>(4)</sup>	YES
2	PD_DSS	PDRIO <sup>(2)</sup>	AO <sup>(4)</sup>	YES
3	PD_ICSS	PDRIO <sup>(2)</sup>	AO <sup>(4)</sup>	YES
4	PD_9GSS	PDRIO <sup>(2)</sup>	AO <sup>(4)</sup>	YES
5	PD_SERDES_0	PDRIO <sup>(2)</sup>	AO <sup>(4)</sup>	YES
6	PD_SERDES_1	PDRIO <sup>(2)</sup>	AO <sup>(4)</sup>	YES
7	PD_SERDES_2	PDRIO <sup>(2)</sup>	AO <sup>(4)</sup>	YES
8	PD_SERDES_3	PDRIO <sup>(2)</sup>	AO <sup>(4)</sup>	YES
9	PD_SERDES_4	PDRIO <sup>(2)</sup>	AO <sup>(4)</sup>	YES
10	PD_SERDES_5	PDRIO <sup>(2)</sup>	AO <sup>(4)</sup>	YES
11	PD_TIMER	PDRIO <sup>(2)</sup>	AO <sup>(4)</sup>	YES
12	PD_C71x_0	PD <sup>(3)</sup>	OFF	NO
13	PD_C71x_1	PD <sup>(3)</sup>	OFF	NO
14	PD_A72_CLUSTER_0	PD <sup>(3)</sup>	OFF	NO
15	PD_A72_0	PD <sup>(3)</sup>	OFF	NO
16	PD_A72_1	PD <sup>(3)</sup>	OFF	NO
17	PD_A72_CLUSTER_1	PD <sup>(3)</sup>	OFF	NO
18	PD_A72_2	PD <sup>(3)</sup>	OFF	NO
19	PD_A72_3	PD <sup>(3)</sup>	OFF	NO
20	PD_GPUCOM	PD <sup>(3)</sup>	OFF	NO
21	PD_GPUCORE	PD <sup>(3)</sup>	OFF	NO
22	PD_C66x_0	PD <sup>(3)</sup>	OFF	NO
23	PD_C66x_1	PD <sup>(3)</sup>	OFF	NO
24	PD_R5FSS_0	PD <sup>(3)</sup>	OFF	NO
25	PD_R5FSS_1	PD <sup>(3)</sup>	OFF	NO
26	PD_DECODE	PD <sup>(3)</sup>	OFF	NO
27	PD_ENCODE	PD <sup>(3)</sup>	OFF	NO
28	PD_DMPAC	PD <sup>(3)</sup>	OFF	NO
29	PD_VPAC	PD <sup>(3)</sup>	OFF	NO

(1) Group with common power - means that there are no power switches. The group is considered always-on.

(2) Power switched domain for reset isolation - means this is a virtual power domain partition required to properly implement reset-isolation.

(3) Power switched domain - means this is a full power switched domain, there are logic power-switches and memory power-switches if there are memories.

(4) AO = AlwaysOn

Table 5-63 presents PSC0 LPSC features.

**Table 5-63. PSC0 LPSC Features**

LPSC Index	LPSC Name	Default LPSC State	Efuse Disable Availability	LPSC State Software Controlled	Reset Isolation
0	LPSC_MAIN_ALWAYS_ON	ON	N	NO	N
1	LPSC_MAIN_TEST	ON	N	YES	N
2	LPSC_MAIN_PBI	ON	N	YES	N
3	LPSC_PER_AUDIO	ON	N	YES	N



**Table 5-63. PSC0 LPSC Features (continued)**

LPSC Index	LPSC Name	Default LPSC State	Efuse Disable Availability	LPSC State Software Controlled	Reset Isolation
4	LPSC_PER_ATL	OFF	Y	YES	N
5	LPSC_PER_MLB	OFF	Y	YES	N
6	LPSC_PER_MOTOR	OFF	Y	YES	N
7	LPSC_PER_MISCIO	OFF	N	YES	N
8	LPSC_PER_GPMC	ON	N	YES	N
9	LPSC_PER_VPFE	ON	Y	YES	N
10	LPSC_PER_VPE	OFF	Y	YES	N
11	LPSC_PER_SPARE0	OFF	Y	YES	N
12	LPSC_PER_SPARE1	OFF	Y	YES	N
13	LPSC_MAIN_DEBUG	ON	N	YES	N
14	LPSC_EMIF_DATA_0	OFF	Y, Controlled by the same efuse bit	YES	N
15	LPSC_EMIF_CFG_0	OFF	Y, Controlled by the same efuse bit	YES	N
16	LPSC_EMIF_DATA_1	OFF	Y, Controlled by the same efuse bit	YES	N
17	LPSC_EMIF_CFG_1	OFF	Y, Controlled by the same efuse bit	YES	N
18	LPSC_PER_SPARE2	OFF	Y	YES	N
19	LPSC_CC_TOP_PBIIST	OFF	N	YES	N
20	LPSC_USB_0	OFF	Y	YES	n
21	LPSC_USB_1	OFF	Y	YES	n
22	LPSC_USB_2	OFF	Y	YES	N
23	LPSC_MM4B_0	OFF	Y	YES	n
24	LPSC_MM4B_1	OFF	Y	YES	n
25	LPSC_MM8B_0	OFF	Y	YES	n
26	LPSC_UFS_0	OFF	Y	YES	N
27	LPSC_UFS_1	OFF	Y	YES	N
28	LPSC_PCIE_0	OFF	Y	YES	n
29	LPSC_PCIE_1	OFF	Y	YES	n
30	LPSC_PCIE_2	OFF	Y	YES	n
31	LPSC_PCIE_3	OFF	Y	YES	n
32	LPSC_SAUL	OFF	Y	YES	N
33	LPSC_PER_I3C	OFF	Y	YES	N
34	LPSC_MAIN_MCANSS_0	OFF	Y	YES	N
35	LPSC_MAIN_MCANSS_1	OFF	Y	YES	N
36	LPSC_MAIN_MCANSS_2	OFF	Y	YES	N
37	LPSC_MAIN_MCANSS_3	OFF	Y	YES	N
38	LPSC_MAIN_MCANSS_4	OFF	Y	YES	N
39	LPSC_MAIN_MCANSS_5	OFF	Y	YES	N
40	LPSC_MAIN_MCANSS_6	OFF	Y	YES	N
41	LPSC_MAIN_MCANSS_7	OFF	Y	YES	N
42	LPSC_MAIN_MCANSS_8	OFF	Y	YES	N
43	LPSC_MAIN_MCANSS_9	OFF	Y	YES	N
44	LPSC_MAIN_MCANSS_10	OFF	Y	YES	N

**Table 5-63. PSC0 LPSC Features (continued)**

LPSC Index	LPSC Name	Default LPSC State	Efuse Disable Availability	LPSC State Software Controlled	Reset Isolation
45	LPSC_MAIN_MCANSS_11	OFF	Y	YES	N
46	LPSC_MAIN_MCANSS_12	OFF	Y	YES	N
47	LPSC_MAIN_MCANSS_13	OFF	Y	YES	N
48	LPSC_DSS	OFF	Y	YES	Y
49	LPSC_DSS_PBIIST	OFF	N	YES	N
50	LPSC_DSI	OFF	Y	YES	Y
51	LPSC_EDP_0	OFF	Y	YES	Y
52	LPSC_EDP_1	OFF	Y	YES	Y
53	LPSC_CSIRX_0	OFF	Y	YES	N
54	LPSC_CSIRX_1	OFF	Y	YES	N
55	LPSC_CSIRX_2	OFF	Y	YES	N
56	LPSC_CSITX_0	OFF	Y	YES	N
57	LPSC_TX_DPHY_0	OFF	Y	YES	Y
58	LPSC_CSIRX_PHY_0	OFF	Y	YES	N
59	LPSC_CSIRX_PHY_1	OFF	Y	YES	N
60	LPSC_CSIRX_PHY_2	OFF	Y	YES	N
61	LPSC_ICSSG_0	OFF	Y	YES	Y
62	LPSC_ICSSG_1	OFF	Y	YES	Y
63	LPSC_9GSS	OFF	Y	YES	Y
64	LPSC_SERDES_0	OFF	Y	YES	Y
65	LPSC_SERDES_1	OFF	Y	YES	Y
66	LPSC_SERDES_2	OFF	Y	YES	Y
67	LPSC_SERDES_3	OFF	Y	YES	Y
68	LPSC_SERDES_4	OFF	Y	YES	Y
69	LPSC_SERDES_5	OFF	Y	YES	Y
70	LPSC_DMTIMER_0	ON	N	YES	Y
71	LPSC_DMTIMER_1	ON	N	YES	Y
72	LPSC_DMTIMER_2	ON	N	YES	Y
73	LPSC_DMTIMER_3	ON	N	YES	Y
74	LPSC_C71x_0	OFF	Y	YES	N
75	LPSC_C71x_0_PBIIST	OFF	N	YES	N
76	LPSC_C71x_1	OFF	Y	YES	N
77	LPSC_C71x_1_PBIIST	OFF	N	YES	N
78	LPSC_A72_CLUSTER_0	OFF	Y	YES	N
79	LPSC_A72_CLUSTER_0_PBIIST	OFF	N	YES	N
80	LPSC_A72_0	OFF	Y	YES	N
81	LPSC_A72_1	OFF	Y	YES	N
82	LPSC_A72_CLUSTER_1	OFF	Y	YES	N
83	LPSC_A72_CLUSTER_1_PBIIST	OFF	N	YES	N
84	LPSC_A72_2	OFF	Y	YES	N
85	LPSC_A72_3	OFF	Y	YES	N
86	LPSC_GPUCOM	OFF	Y	YES	N
87	LPSC_GPUPBIIST	OFF	N	YES	N
88	LPSC_GPUCORE	OFF	Y	YES	N

**Table 5-63. PSC0 LPSC Features (continued)**

LPSC Index	LPSC Name	Default LPSC State	Efuse Disable Availability	LPSC State Software Controlled	Reset Isolation
89	LPSC_C66X_0	OFF	Y	YES	N
90	LPSC_C66X_PBIST_0	OFF	N	YES	N
91	LPSC_C66X_1	OFF	Y	YES	N
92	LPSC_C66X_PBIST_1	OFF	N	YES	N
93	LPSC_R5FSS0_CORE0	OFF	Y	YES	Y
94	LPSC_R5FSS0_CORE1	OFF	Y	YES	Y
95	LPSC_R5FSS_PBIST_0	OFF	N	YES	N
96	LPSC_R5FSS1_CORE0	OFF	Y	YES	Y
97	LPSC_R5FSS1_CORE1	OFF	Y	YES	Y
98	LPSC_R5FSS_PBIST_1	OFF	N	YES	N
99	LPSC_DECODE_0	OFF	Y	YES	N
100	LPSC_DECODE_PBIST	OFF	N	YES	N
101	LPSC_ENCODE_0	OFF	Y	YES	N
102	LPSC_ENCODE_PBIST	OFF	N	YES	N
103	LPSC_DMPAC	OFF	Y	YES	N
104	LPSC_SDE	OFF	Y	YES	N
105	LPSC_DMPAC_PBIST	OFF	N	YES	N
106	LPSC_VPAC	OFF	Y	YES	N
107	LPSC_VPAC_PBIST	OFF	N	YES	N

**Note**

For details on module state transitions, please refer to [Section 5.2.2.1.5, Executing State Transitions](#).

**Note**

For information about LPSC Dependencies, see [Section 5.2.2.1.3.3, LPSC Dependencies Overview](#).

### 5.2.2.2.1.3.3 LPSC Dependences Overview

Table 5-64, Table 5-65, Table 5-66, Table 5-67, and Table 5-68 depict the dependences among the LPSC in both PSCs - WKUP\_PSC0 and PSC0. The matrices are used in the following way:

- Enabling a LPSC - to check if another LPSC has to be enabled before a specific LPSC, take the specific LPSC column (find its name in one of the LPSC Dependences tables) and check if there is Y mark in this parts of the LPSC Dependences tables. All LPSC makred as Y should be enabled before enabling the specific LPSC.
- Disabling a LPSC - to disable a LPSC, take the specific LPSC row and check if there is Y mark in all five parts of the LPSC Dependences tables. All LPSC makred as Y should be disabled before enabling the specific LPSC.

In red are the LPSC that are always on. In green are those that are default on. For more information about default state and reset isolation of the LPSC see Table 5-60 and Table 5-63. For more information about reset isolation see *Reset*.

**Table 5-64. LPSC Dependences (Part 1)**

	LPS C_W KUP AL WAY SON	LPS C_D MSC	LPS C_D EBU G2D MSC	LPS C_W KUP GPIO	LPS C_W KUP MCU 2MA IN	LPS C_M AIN2 WK UPM CUM CU	LPS C_M CU_ TES T	LPS C_M CU_ DEB UG	LPS C_M CU_ MCA N_0	LPS C_M CU_ MCA N_1	LPS C_M CU_ OSP I_0	LPS C_M CU_ OSP I_1	LPS C_M CU_ HYP ERB US	LPS C_M CU_ 3C_ 0	LPS C_M CU_ 3C_ 1	LPS C_M CU_ ADC _0	LPS C_M CU_ ADC _1	LPS C_W KUP _SP ARE 0	LPS C_W KUP _SP ARE 1	LPS C_M CU_ R5F F0_0	LPS C_M CU_ R5F F0_1	LPS C_M CU_ R5F F0F SS_ PBI ST_ 0
LPSC_WKUP_ALWAYSON	N	Y	N	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N
LPSC_DMSC	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DEBUG2DMSC	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_WKUP_GPIO	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_WKUPMCU2MAIN	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN2WKUPMCUMCU	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_TEST	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_DEBUG	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_MCAN_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_MCAN_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_OSPI_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_OSPI_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_HYPERBUS	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_I3C_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_I3C_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_ADC_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Table 5-64. LPSC Dependences (Part 1) (continued)

	LPS C_W KUP _AL WAY SON	LPS C_D MSC	LPS C_D EBU G2D MSC	LPS C_W KUP _GPI O	LPS C_W KUP MCU 2MA IN	LPS C_M AIN2 WK UPM CUM CU	LPS C_M CU TES T	LPS C_M CU DEB UG	LPS C_M CU MCA N_0	LPS C_M CU MCA N_1	LPS C_M CU OSP I_0	LPS C_M CU OSP I_1	LPS C_M CU HYP ERB US	LPS C_M CU I 3C_0	LPS C_M CU I 3C_1	LPS C_M CU ADC _0	LPS C_M CU ADC _1	LPS C_W KUP _SP ARE 0	LPS C_W KUP _SP ARE 1	LPS C_M CU R5F F0_0	LPS C_M CU R5F F0_1	LPS C_M CU R5F F0F SS_ PBI ST_0
LPSC_MCU_ADC_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_WKUP_SPARE0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_WKUP_SPARE1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_R5FF0_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	P	Y
LPSC_MCU_R5FF0_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y
LPSC_MCU_R5FF0FSS_PBI0	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_ALWAYS0N	N	N	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_TEST	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_PBI0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_TX_DPHY_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_CSIRX_PHY_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_CSIRX_PHY_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_SERDES_4	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_SERDES_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_SERDES_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_SERDES_2	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_SERDES_3	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_USB_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_USB_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MMC4B_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MMC4B_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MMC8B_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_UFS_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DSS	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DSS_PBI0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DSI	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_EDP_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Table 5-64. LPSC Dependences (Part 1) (continued)

	LPS C_W KUP _AL WAY SON	LPS C_D MSC	LPS C_D EBU G2D MSC	LPS C_W KUP _GPI O	LPS C_W KUP MCU 2MA IN	LPS C_M AIN2 WK UPM CUM CU	LPS C_M CU TES T	LPS C_M CU DEB UG	LPS C_M CU MCA N_0	LPS C_M CU MCA N_1	LPS C_M CU OSP I_0	LPS C_M CU OSP I_1	LPS C_M CU HYP ERB US	LPS C_M CU I 3C_0	LPS C_M CU I 3C_1	LPS C_M CU ADC _0	LPS C_M CU ADC _1	LPS C_W KUP _SP ARE 0	LPS C_W KUP _SP ARE 1	LPS C_M CU R5F F0_0	LPS C_M CU R5F F0_1	LPS C_M CU R5F F0F SS_ PBI ST_0
LPSC_CSIRX_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_CSIRX_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_CSITX_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_ICSSG_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_ICSSG_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_PCIE_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_PCIE_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_PCIE_2	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_PCIE_3	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_9GSS	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_SAUL	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_PER_I3C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_MAIN_MCANSS_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_MAIN_MCANSS_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_MAIN_MCANSS_2	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_MAIN_MCANSS_3	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_MAIN_MCANSS_4	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_MAIN_MCANSS_5	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_MAIN_MCANSS_6	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_MAIN_MCANSS_7	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_MAIN_MCANSS_8	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_MAIN_MCANSS_9	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_MAIN_MCANSS_10	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_MAIN_MCANSS_11	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_MAIN_MCANSS_12	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_MAIN_MCANSS_13	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_DMTIMER_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	

Table 5-64. LPSC Dependences (Part 1) (continued)

	LPS C_W KUP _AL WAY SON	LPS C_D MSC	LPS C_D EBU G2D MSC	LPS C_W KUP _GPI O	LPS C_W KUP MCU 2MA IN	LPS C_M AIN2 WK UPM CUM CU	LPS C_M CU TES T	LPS C_M CU DEB UG	LPS C_M CU MCA N_0	LPS C_M CU MCA N_1	LPS C_M CU OSP I_0	LPS C_M CU OSP I_1	LPS C_M CU HYP ERB US	LPS C_M CU I 3C_0	LPS C_M CU I 3C_1	LPS C_M CU ADC _0	LPS C_M CU ADC _1	LPS C_W KUP _SP ARE 0	LPS C_W KUP _SP ARE 1	LPS C_M CU R5F F0_0	LPS C_M CU R5F F0_1	LPS C_M CU R5F F0F SS_ PBI ST_0
LPSC_DMTIMER_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DMTIMER_2	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DMTIMER_3	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_AUDIO	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_ATL	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_MLB	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_MOTOR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_MISCIO	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_GPMC	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_VPFE	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_SPARE0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_SPARE1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_DEBUG	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_EMIF_DATA_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_EMIF_CFG_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_CC_TOP_PBI	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_C71X_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_C71X_0_PBI	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_A72_CLUSTER_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_A72_CLUSTER_0_PBI	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_A72_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_A72_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_GPUCOM	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_GPU_PBI	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_GPUCORE	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_C66X_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_C66X_PBI	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N



Table 5-64. LPSC Dependences (Part 1) (continued)

	LPS C_W KUP _AL WAY SON	LPS C_D MSC	LPS C_D EBU G2D MSC	LPS C_W KUP _GPI O	LPS C_W KUP MCU 2MA IN	LPS C_M AIN2 WK UPM CUM CU	LPS C_M CU_ TES T	LPS C_M CU_ DEB UG	LPS C_M CU_ MCA N_0	LPS C_M CU_ MCA N_1	LPS C_M CU_ OSP I_0	LPS C_M CU_ OSP I_1	LPS C_M CU_ HYP ERB US	LPS C_M CU_ 3C_ 0	LPS C_M CU_ 3C_ 1	LPS C_M CU_ ADC _0	LPS C_M CU_ ADC _1	LPS C_W KUP _SP ARE 0	LPS C_W KUP _SP ARE 1	LPS C_M CU_ R5F F0_0	LPS C_M CU_ R5F F0_1	LPS C_M CU_ R5F F0F SS_ PBI ST_ 0
LPSC_C66X_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_C66X_PBI0_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_R5FSS_N_R5_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_R5FSS_N_R5_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_R5FSS_PBI0_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_R5FSS_Y_R5_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_R5FSS_Y_R5_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_R5FSS_PBI0_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_DECODE_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_DECODE_PBI0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_ENCODE_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_ENCODE_PBI0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_DMPAC	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_SDE	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_DMPAC_PBI0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_VPAC	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_VPAC_PBI0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	

Table 5-65. LPSC Dependences (Part 2)

	LPS C_M AIN _AL WAY SON	LPS C_M AIN _TE ST	LPS C_M AIN _PB IST	LPS C_T X_D PHY _0	LPS C_C SIR X_P HY_ 0	LPS C_C SIR X_P HY_ 1	LPS C_S ERD ES_ 4	LPS C_S ERD ES_ 0	LPS C_S ERD ES_ 1	LPS C_S ERD ES_ 2	LPS C_S ERD ES_ 3	LPS C_U SB_ 0	LPS C_U SB_ 1	LPS C_M MC4 B_0	LPS C_M MC4 B_1	LPS C_M MC8 B_0	LPS C_U FS_ 0	LPS C_D SS	LPS C_D SS_ PBI ST	LPS C_D SI	LPS C_E DP_ 0	LPS C_C SIR X_0	LPS C_C SIR X_1
LPSC_WKUP_ALWAYSON	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DM0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DEBUG2DM0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_WKUP_GPIO	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

**Table 5-65. LPSC Dependencies (Part 2) (continued)**

	LPS C_M AIN _AL WAY SON	LPS C_M AIN _TE ST	LPS C_M AIN _PB IST	LPS C_T X_D PHY _0	LPS C_C SIR X_P HY_0	LPS C_C SIR X_P HY_1	LPS C_S ERD ES_4	LPS C_S ERD ES_0	LPS C_S ERD ES_1	LPS C_S ERD ES_2	LPS C_S ERD ES_3	LPS C_U SB_0	LPS C_U SB_1	LPS C_M MC4 B_0	LPS C_M MC4 B_1	LPS C_M MC8 B_0	LPS C_U FS_0	LPS C_D SS	LPS C_D SS_PBI ST	LPS C_D SI	LPS C_E DP_0	LPS C_C SIR X_0	LPS C_C SIR X_1
LPSC_WKUPMCU2MAIN	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN2WKUPMCUMCU	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_TEST	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_DEBUG	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_MCAN_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_MCAN_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_OSPI_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_OSPI_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_HYPERBUS	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_I3C_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_I3C_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_ADC_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_ADC_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_WKUP_SPARE0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_WKUP_SPARE0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_R5FF0_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_R5FF0_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_R5FF0FSS_PBI0_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_ALWAYS0N	N	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N
LPSC_MAIN_TEST	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_PBI0ST	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_TX_DPHY_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N
LPSC_CSIRX_PHY_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N
LPSC_CSIRX_PHY_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y
LPSC_SERDES_4	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N
LPSC_SERDES_0	N	N	N	N	N	N	N	N	N	N	N	P	N	N	N	N	N	N	N	N	N	N	N
LPSC_SERDES_1	N	N	N	N	N	N	N	N	N	N	N	P	N	N	N	N	N	N	N	N	N	N	N
LPSC_SERDES_2	N	N	N	N	N	N	N	N	N	N	N	P	N	N	N	N	N	N	N	N	N	N	N
LPSC_SERDES_3	N	N	N	N	N	N	N	N	N	N	N	P	N	N	N	N	N	N	N	N	N	N	N

Table 5-65. LPSC Dependences (Part 2) (continued)

	LPS C_M AIN _AL WAY SON	LPS C_M AIN _TE ST	LPS C_M AIN _PB IST	LPS C_T X_D PHY _0	LPS C_C SIR X_P HY_0	LPS C_C SIR X_P HY_1	LPS C_S ERD ES_4	LPS C_S ERD ES_0	LPS C_S ERD ES_1	LPS C_S ERD ES_2	LPS C_S ERD ES_3	LPS C_U SB_0	LPS C_U SB_1	LPS C_M MC4 B_0	LPS C_M MC4 B_1	LPS C_M MC8 B_0	LPS C_U FS_0	LPS C_D SS	LPS C_D SS_PBI ST	LPS C_D SI	LPS C_E DP_0	LPS C_C SIR X_0	LPS C_C SIR X_1
LPSC_USB_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_USB_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MMC4B_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MMC4B_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MMC8B_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_UFS_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DSS	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N
LPSC_DSS_PBI	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DSI	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N
LPSC_EDP_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N
LPSC_CSIRX_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_CSIRX_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_CSITX_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_ICSSG_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_ICSSG_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PCIE_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PCIE_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PCIE_2	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PCIE_3	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_9GSS	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_SAUL	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_I3C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_2	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_3	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_4	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_5	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_6	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

**Table 5-65. LPSC Dependences (Part 2) (continued)**

	LPS C_M AIN _AL WAY SON	LPS C_M AIN _TE ST	LPS C_M AIN _PB IST	LPS C_T X_D PHY _0	LPS C_C SIR X_P HY_0	LPS C_C SIR X_P HY_1	LPS C_S ERD ES_4	LPS C_S ERD ES_0	LPS C_S ERD ES_1	LPS C_S ERD ES_2	LPS C_S ERD ES_3	LPS C_U SB_0	LPS C_U SB_1	LPS C_M MC4 B_0	LPS C_M MC4 B_1	LPS C_M MC8 B_0	LPS C_U FS_0	LPS C_D SS	LPS C_D SS_PBI ST	LPS C_D SI	LPS C_E DP_0	LPS C_C SIR X_0	LPS C_C SIR X_1
LPSC_MAIN_MCANSS_7	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_8	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_9	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_10	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_11	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_12	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_13	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DMTIMER_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DMTIMER_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DMTIMER_2	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DMTIMER_3	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_AUDIO	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_ATL	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_MLB	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_MOTOR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_MISCIO	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_GPMC	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_VPFE	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_SPARE0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_SPARE1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_DEBUG	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_EMIF_DATA_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_EMIF_CFG_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_CC_TOP_PBI	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_C71X_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_C71X_0_PBI	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_A72_CLUSTER_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_A72_CLUSTER_0_PBI	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_A72_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

**Table 5-65. LPSC Dependences (Part 2) (continued)**

	LPS C_M AIN _AL WAY SON	LPS C_M AIN _TE ST	LPS C_M AIN _PB IST	LPS C_T X_D PHY _0	LPS C_C SIR X_P HY_0	LPS C_C SIR X_P HY_1	LPS C_S ERD ES_4	LPS C_S ERD ES_0	LPS C_S ERD ES_1	LPS C_S ERD ES_2	LPS C_S ERD ES_3	LPS C_U SB_0	LPS C_U SB_1	LPS C_M MC4 B_0	LPS C_M MC4 B_1	LPS C_M MC8 B_0	LPS C_U FS_0	LPS C_D SS	LPS C_D SS_PBI ST	LPS C_D SI	LPS C_E DP_0	LPS C_C SIR X_0	LPS C_C SIR X_1
LPSC_A72_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_GPUCOM	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_GPU_PBIST	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_GPUCORE	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_C66X_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_C66X_PBIST_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_C66X_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_C66X_PBIST_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_R5FSS0_CORE0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_R5FSS0_CORE1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_R5FSS_PBIST_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_R5FSS1_CORE0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_R5FSS1_CORE1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_R5FSS_PBIST_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DECODE_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DECODE_PBIST	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_ENCODE_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_ENCODE_PBIST	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DMPAC	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_SDE	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DMPAC_PBIST	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_VPAC	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_VPAC_PBIST	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Table 5-66. LPSC Dependences (Part 3)

	LPS C_C SIT X_0	LPS C_I CSS G_0	LPS C_I CSS G_1	LPS C_P CIE _0	LPS C_P CIE _1	LPS C_P CIE _2	LPS C_P CIE _3	LPS C_9 GSS	LPS C_S AUL	LPS C_P ER I3C	LPS C_M AIN _MC ANS S_0	LPS C_M AIN _MC ANS S_1	LPS C_M AIN _MC ANS S_2	LPS C_M AIN _MC ANS S_3	LPS C_M AIN _MC ANS S_4	LPS C_M AIN _MC ANS S_5	LPS C_M AIN _MC ANS S_6	LPS C_M AIN _MC ANS S_7	LPS C_M AIN _MC ANS S_8	LPS C_M AIN _MC ANS S_9	LPS C_M AIN _MC ANS S_10	LPS C_M AIN _MC ANS S_11	LPS C_M AIN _MC ANS S_12
LPSC_WKUP_ALWAYS0N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DM5C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DEBUG2DM5C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_WKUP_GPIO	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_WKUPMCU2MAIN	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN2WKUPMCUMCU	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_TEST	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_DEBUG	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_MCAN_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_MCAN_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_OSPI_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_OSPI_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_HYPERBUS	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_I3C_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_I3C_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_ADC_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_ADC_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_WKUP_SPARE0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_WKUP_SPARE0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_R5FF0_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_R5FF0_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_R5FF0FSS_PBIST_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_ALWAYS0N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
LPSC_MAIN_TEST	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_PBIST	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_TX_DPHY_0	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_CSIRX_PHY_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_CSIRX_PHY_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Table 5-66. LPSC Dependencies (Part 3) (continued)

	LPS C_C SIT X_0	LPS C_I CSS G_0	LPS C_I CSS G_1	LPS C_P CIE _0	LPS C_P CIE _1	LPS C_P CIE _2	LPS C_P CIE _3	LPS C_9 GSS	LPS C_S AUL	LPS C_P ER I3C	LPS C_M AIN _MC ANS S_0	LPS C_M AIN _MC ANS S_1	LPS C_M AIN _MC ANS S_2	LPS C_M AIN _MC ANS S_3	LPS C_M AIN _MC ANS S_4	LPS C_M AIN _MC ANS S_5	LPS C_M AIN _MC ANS S_6	LPS C_M AIN _MC ANS S_7	LPS C_M AIN _MC ANS S_8	LPS C_M AIN _MC ANS S_9	LPS C_M AIN _MC ANS S_10	LPS C_M AIN _MC ANS S_11	LPS C_M AIN _MC ANS S_12
LPSC_SERDES_4	N	N	N	N	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_SERDES_0	N	N	N	Y	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_SERDES_1	N	N	P	N	Y	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_SERDES_2	N	N	P	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_SERDES_3	N	N	N	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_USB_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_USB_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MMC4B_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MMC4B_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MMC8B_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_UFS_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DSS	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DSS_PBI5T	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DSI	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_EDP_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_CSIRX_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_CSIRX_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_CSITX_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_ICSSG_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_ICSSG_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PCIE_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PCIE_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PCIE_2	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PCIE_3	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_9GSS	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_SAUL	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_I3C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N



**Table 5-66. LPSC Dependences (Part 3) (continued)**

	LPS C_C SIT X_0	LPS C_I CSS G_0	LPS C_I CSS G_1	LPS C_P CIE _0	LPS C_P CIE _1	LPS C_P CIE _2	LPS C_P CIE _3	LPS C_9 GSS	LPS C_S AUL	LPS C_P ER I3C	LPS C_M AIN _MC ANS S_0	LPS C_M AIN _MC ANS S_1	LPS C_M AIN _MC ANS S_2	LPS C_M AIN _MC ANS S_3	LPS C_M AIN _MC ANS S_4	LPS C_M AIN _MC ANS S_5	LPS C_M AIN _MC ANS S_6	LPS C_M AIN _MC ANS S_7	LPS C_M AIN _MC ANS S_8	LPS C_M AIN _MC ANS S_9	LPS C_M AIN _MC ANS S_10	LPS C_M AIN _MC ANS S_11	LPS C_M AIN _MC ANS S_12
LPSC_MAIN_MCANSS_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_2	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_3	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_4	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_5	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_6	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_7	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_8	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_9	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_10	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_11	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_12	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_13	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DMTIMER_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DMTIMER_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DMTIMER_2	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DMTIMER_3	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_AUDIO	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_ATL	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_MLB	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_MOTOR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_MISCIO	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_GPMC	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_VPFE	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_SPARE0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_SPARE1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_DEBUG	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_EMIF_DATA_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Table 5-66. LPSC Dependencies (Part 3) (continued)

	LPS C_C SIT X_0	LPS C_I CSS G_0	LPS C_I CSS G_1	LPS C_P CIE _0	LPS C_P CIE _1	LPS C_P CIE _2	LPS C_P CIE _3	LPS C_9 GSS	LPS C_S AUL	LPS C_P ER I3C	LPS C_M AIN _MC ANS S_0	LPS C_M AIN _MC ANS S_1	LPS C_M AIN _MC ANS S_2	LPS C_M AIN _MC ANS S_3	LPS C_M AIN _MC ANS S_4	LPS C_M AIN _MC ANS S_5	LPS C_M AIN _MC ANS S_6	LPS C_M AIN _MC ANS S_7	LPS C_M AIN _MC ANS S_8	LPS C_M AIN _MC ANS S_9	LPS C_M AIN _MC ANS S_10	LPS C_M AIN _MC ANS S_11	LPS C_M AIN _MC ANS S_12
LPSC_EMIF_CFG_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_CC_TOP_PBIST	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_C71X_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_C71X_0_PBIST	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_A72_CLUSTER_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_A72_CLUSTER_0_PBIST	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_A72_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_A72_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_GPUCOM	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_GPU_PBIST	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_GPUCORE	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_C66X_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_C66X_PBIST_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_C66X_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_C66X_PBIST_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_R5FSS0_CORE0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_R5FSS0_CORE1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_R5FSS_PBIST_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_R5FSS1_CORE0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_R5FSS1_CORE1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_R5FSS_PBIST_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DECODE_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DECODE_PBIST	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_ENCODE_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_ENCODE_PBIST	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DMPAC	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_SDE	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DMPAC_PBIST	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Table 5-66. LPSC Dependences (Part 3) (continued)

	LPS C_C SIT X_0	LPS C_I CSS G_0	LPS C_I CSS G_1	LPS C_P CIE _0	LPS C_P CIE _1	LPS C_P CIE _2	LPS C_P CIE _3	LPS C_9 GSS	LPS C_S AUL	LPS C_P ER_ I3C	LPS C_M AIN _MC S_0	LPS C_M AIN _MC S_1	LPS C_M AIN _MC S_2	LPS C_M AIN _MC S_3	LPS C_M AIN _MC S_4	LPS C_M AIN _MC S_5	LPS C_M AIN _MC S_6	LPS C_M AIN _MC S_7	LPS C_M AIN _MC S_8	LPS C_M AIN _MC S_9	LPS C_M AIN _MC S_1 0	LPS C_M AIN _MC S_1 1	LPS C_M AIN _MC S_1 2
LPSC_VPAC	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_VPAC_PBIST	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Table 5-67. LPSC Dependences (Part 4)

	LPS C_M AIN _MC ANS S_1 3	LPS C_D MTI ME R_0	LPS C_D MTI ME R_1	LPS C_D MTI ME R_2	LPS C_D MTI ME R_3	LPS C_P ER_ AUD IO	LPS C_P ER_ ATL	LPS C_P ER_ MLB	LPS C_P ER_ MO TOR	LPS C_P ER_ MIS CIO	LPS C_P ER_ GP MC	LPS C_P ER_ VPF E	LPS C_P ER_ SPA RE0	LPS C_P ER_ SPA RE1	LPS C_M AIN _DE BU G	LPS C_E MIF _DA TA_ 0	LPS C_E MIF _CF G_0	LPS C_C C_T OP_ PBI ST	LPS C_C 71X _0	LPS C_C 71X _0 PBI ST	LPS C_A 72_ CLU STE R_0	LPS C_A 72_ CLU STE R_0 _PB IST	LPS C_A 72_ 0
LPSC_WKUP_ALWAYS0N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DM5C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DEBUG2DM5C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_WKUP_GPIO	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_WKUPMCU2MAIN	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN2WKUPMCUMCU	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_TEST	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_DEBUG	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N	N	N	N	N
LPSC_MCU_MCAN_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_MCAN_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_OSPI_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_OSPI_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_HYPERBUS	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_I3C_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_I3C_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_ADC_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_ADC_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_WKUP_SPARE0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_WKUP_SPARE0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Table 5-67. LPSC Dependencies (Part 4) (continued)

	LPS C_M AIN _MC ANS S_1 3	LPS C_D MTI ME R_0	LPS C_D MTI ME R_1	LPS C_D MTI ME R_2	LPS C_D MTI ME R_3	LPS C_P ER_ AUD IO	LPS C_P ER_ ATL	LPS C_P ER_ MLB	LPS C_P ER_ MO TOR	LPS C_P ER_ MIS CIO	LPS C_P ER_ GP MC	LPS C_P ER_ VPF E	LPS C_P ER_ SPA RE0	LPS C_P ER_ SPA RE1	LPS C_M AIN _DE BUG	LPS C_E MIF _DA TA_ 0	LPS C_E MIF _CF G_0	LPS C_C C_T OP_ PBI ST	LPS C_C 71X _0	LPS C_C 71X _0 PBI ST	LPS C_A 72_ CLU STE R_0	LPS C_A 72_ CLU STE R_0 _PB IST	LPS C_A 72_ 0
LPSC_MCU_R5FF0_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_MCU_R5FF0_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_MCU_R5FF0FSS_PBI0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_MAIN_ALWAYS0N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y		Y	Y	Y	N	Y	N	
LPSC_MAIN_TEST	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_MAIN_PBI0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_TX_DPHY_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_CSIRX_PHY_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_CSIRX_PHY_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_SERDES_4	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_SERDES_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_SERDES_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_SERDES_2	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_SERDES_3	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_USB_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_USB_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_MMC4B_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_MMC4B_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_MMC8B_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_UFS_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_DSS	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_DSS_PBI0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_DSI	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_EDP_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_CSIRX_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_CSIRX_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_CSITX_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	

**Table 5-67. LPSC Dependencies (Part 4) (continued)**

	LPS C_M AIN _MC ANS S_1 3	LPS C_D MTI ME R_0	LPS C_D MTI ME R_1	LPS C_D MTI ME R_2	LPS C_D MTI ME R_3	LPS C_P ER_ AUD IO	LPS C_P ER_ ATL	LPS C_P ER_ MLB	LPS C_P ER_ MO TOR	LPS C_P ER_ MIS CIO	LPS C_P ER_ GP MC	LPS C_P ER_ VPF E	LPS C_P ER_ SPA RE0	LPS C_P ER_ SPA RE1	LPS C_M AIN _DE BU G	LPS C_E MIF _DA TA_ 0	LPS C_E MIF _CF G_ 0	LPS C_C C_T OP_ PBI ST	LPS C_C 71X _0	LPS C_C 71X _0 PBI ST	LPS C_A 72_ CLU STE R_0	LPS C_A 72_ CLU STE R_0 _PB IST	LPS C_A 72_ 0
LPSC_ICSSG_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_ICSSG_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PCIE_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PCIE_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PCIE_2	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PCIE_3	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_9GSS	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_SAUL	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_I3C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_2	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_3	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_4	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_5	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_6	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_7	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_8	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_9	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_10	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_11	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_12	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_13	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DMTIMER_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DMTIMER_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DMTIMER_2	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DMTIMER_3	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Table 5-67. LPSC Dependences (Part 4) (continued)

	LPS C_M AIN _MC ANS S_1 3	LPS C_D MTI ME R_0	LPS C_D MTI ME R_1	LPS C_D MTI ME R_2	LPS C_D MTI ME R_3	LPS C_P ER_ AUD IO	LPS C_P ER_ ATL	LPS C_P ER_ MLB	LPS C_P ER_ MO TOR	LPS C_P ER_ MIS CIO	LPS C_P ER_ GP MC	LPS C_P ER_ VPF E	LPS C_P ER_ SPA RE0	LPS C_P ER_ SPA RE1	LPS C_M AIN _DE BUG	LPS C_E MIF _DA TA_ 0	LPS C_E MIF _CF G_0	LPS C_C C_T OP_ PBI ST	LPS C_C C_71X _0	LPS C_C C_71X _0 PBI ST	LPS C_A 72_ CLU STE R_0	LPS C_A 72_ CLU STE R_0 _PB IST	LPS C_A 72_ 0
LPSC_PER_AUDIO	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_PER_ATL	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_PER_MLB	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_PER_MOTOR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_PER_MISCIO	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_PER_GPMC	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_PER_VPFE	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_PER_SPARE0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_PER_SPARE1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_MAIN_DEBUG	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_EMIF_DATA_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_EMIF_CFG_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N	N	N	
LPSC_CC_TOP_PBI	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_C71X_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	
LPSC_C71X_0_PBI	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_A72_CLUSTER_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	Y	
LPSC_A72_CLUSTER_0_PBI	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_A72_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_A72_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_GPUCOM	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_GPU_PBI	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_GPUCORE	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_C66X_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_C66X_PBI_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_C66X_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_C66X_PBI_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_R5FSS0_CORE0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	

**Table 5-67. LPSC Dependencies (Part 4) (continued)**

	LPS_C_M AIN_MC ANS_13	LPS_C_D MTI_ME R_0	LPS_C_D MTI_ME R_1	LPS_C_D MTI_ME R_2	LPS_C_D MTI_ME R_3	LPS_C_P ER_AUD IO	LPS_C_P ER_ATL	LPS_C_P ER_MLB	LPS_C_P ER_MO TOR	LPS_C_P ER_MIS CIO	LPS_C_P ER_GP MC	LPS_C_P ER_VPF E	LPS_C_P ER_SPA RE0	LPS_C_P ER_SPA RE1	LPS_C_M AIN_DE BUG	LPS_C_E MIF_DA TA_0	LPS_C_E MIF_CF G_0	LPS_C_C C_T_OP PBI_ST	LPS_C_C 71X_0	LPS_C_C 71X_0 PBI_ST	LPS_C_A 72_CLU STE_R_0	LPS_C_A 72_CLU STE_R_0 _PB IST	LPS_C_A 72_0
LPSC_R5FSS0_CORE1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_R5FSS_PBI0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_R5FSS1_CORE0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_R5FSS1_CORE1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_R5FSS_PBI1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_DECODE_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_DECODE_PBI0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_ENCODE_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_ENCODE_PBI0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_DMPAC	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_SDE	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_DMPAC_PBI0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_VPAC	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	
LPSC_VPAC_PBI0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	

**Table 5-68. LPSC DEPENDENCIES (PART 5)**

	LPS C_A 72_ 1	LPS C_G PUC OM	LPS C_G PU_ PBI ST	LPS C_G PUC ORE	LPS C_C 66X _0	LPS C_C 66X _PB IST_ 0	LPS C_C 66X _1	LPS C_C 66X _PB IST_ 1	LPS C_R 5FS S0_ CO RE0	LPS C_R 5FS S0_ CO RE1	LPS C_R 5FS S_P BIS T_0	LPS C_R 5FS S1_ CO RE0	LPS C_R 5FS S1_ CO RE1	LPS C_R 5FS S_P BIS T_1	LPS C_D ECO DE_ 0	LPS C_D ECO DE_ PBI ST	LPS C_E NC ODE _0	LPS C_E NC ODE _PB IST	LPS C_D MPA C	LPS C_S DE	LPS C_D MPA C_P BIS T	LPS C_V PAC	LPS C_V PAC _PB IST
LPSC_WKUP_ALWAYS0N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DM0C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DEBUG2DM0C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_WKUP_GPIO	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_WKUPMCU2MAIN	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN2WKUPMCUMCU	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_TEST	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N



Table 5-68. LPSC DEPENDENCES (PART 5) (continued)

	LPS C_A 72_ 1	LPS C_G PUC OM	LPS C_G PU PBI ST	LPS C_G PUC ORE	LPS C_C 66X _0	LPS C_C 66X _PB IST_ 0	LPS C_C 66X _1	LPS C_C 66X _PB IST_ 1	LPS C_R 5FS S0_ CO RE0	LPS C_R 5FS S0_ CO RE1	LPS C_R 5FS S_P BIS T_0	LPS C_R 5FS S1_ CO RE0	LPS C_R 5FS S1_ CO RE1	LPS C_R 5FS S_P BIS T_1	LPS C_D ECO DE_ 0	LPS C_D ECO DE_ PBI ST	LPS C_E NC ODE _0	LPS C_E NC ODE _PB IST	LPS C_D MPA C	LPS C_S DE	LPS C_D MPA C_P BIS T	LPS C_V PAC	LPS C_V PAC _PB IST
LPSC_MCU_DEBUG	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_MCAN_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_MCAN_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_OSPI_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_OSPI_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_HYPERBUS	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_I3C_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_I3C_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_ADC_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_ADC_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_WKUP_SPARE0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_WKUP_SPARE0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_R5FF0_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_R5FF0_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MCU_R5FF0FSS_PBIST_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_ALWAYS0N		Y	N		Y	N	Y	N	Y	Y	N	Y	Y	N	Y	N	Y	N	Y	N	N	Y	N
LPSC_MAIN_TEST	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_PBIST	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_TX_DPHY_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_CSIRX_PHY_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_CSIRX_PHY_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_SERDES_4	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_SERDES_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_SERDES_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_SERDES_2	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_SERDES_3	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_USB_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_USB_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MMC4B_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Table 5-68. LPSC DEPENDENCES (PART 5) (continued)

	LPS C_A 72_ 1	LPS C_G PUC OM	LPS C_G PU_ PBI ST	LPS C_G PUC ORE	LPS C_C 66X _0	LPS C_C 66X _PB IST_ 0	LPS C_C 66X _1	LPS C_C 66X _PB IST_ 1	LPS C_R 5FS S0_ CO RE0	LPS C_R 5FS S0_ CO RE1	LPS C_R 5FS S_P BIS T_0	LPS C_R 5FS S1_ CO RE0	LPS C_R 5FS S1_ CO RE1	LPS C_R 5FS S_P BIS T_1	LPS C_D ECO DE_ 0	LPS C_D ECO DE_ PBI ST	LPS C_E NC ODE _0	LPS C_E NC ODE _PB IST	LPS C_D MPA C	LPS C_S DE	LPS C_D MPA C_P BIS T	LPS C_V PAC	LPS C_V PAC _PB IST
LPSC_MMC4B_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MMC8B_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_UFS_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DSS	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DSS_PBI	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DSI	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_EDP_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_CSIRX_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_CSIRX_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_CSITX_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_ICSSG_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_ICSSG_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PCIE_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PCIE_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PCIE_2	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PCIE_3	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_9GSS	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_SAUL	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_I3C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_2	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_3	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_4	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_5	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_6	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_7	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_8	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_9	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Table 5-68. LPSC DEPENDENCES (PART 5) (continued)

	LPS C_A 72_ 1	LPS C_G PUC OM	LPS C_G PU PBI ST	LPS C_G PUC ORE	LPS C_C 66X _0	LPS C_C 66X _PB IST_ 0	LPS C_C 66X _1	LPS C_C 66X _PB IST_ 1	LPS C_R 5FS S0_ CO RE0	LPS C_R 5FS S0_ CO RE1	LPS C_R 5FS S_P BIS T_0	LPS C_R 5FS S1_ CO RE0	LPS C_R 5FS S1_ CO RE1	LPS C_R 5FS S_P BIS T_1	LPS C_D ECO DE_ 0	LPS C_D ECO DE_ PBI ST	LPS C_E NC ODE _0	LPS C_E NC ODE _PB IST	LPS C_D MPA C	LPS C_S DE	LPS C_D MPA C_P BIS T	LPS C_V PAC	LPS C_V PAC _PB IST
LPSC_MAIN_MCANSS_10	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_11	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_12	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_MCANSS_13	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DMTIMER_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DMTIMER_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DMTIMER_2	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DMTIMER_3	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_AUDIO	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_ATL	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_MLB	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_MOTOR	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_MISCIO	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_GPMC	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_VPFE	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_SPARE0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_PER_SPARE1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_MAIN_DEBUG	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_EMIF_DATA_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_EMIF_CFG_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_CC_TOP_PBI	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_C71X_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_C71X_0_PBI	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_A72_CLUSTER_0	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_A72_CLUSTER_0_PBI	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_A72_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_A72_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_GPUCOM	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_GPU_PBI	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Table 5-68. LPSC DEPENDENCES (PART 5) (continued)

	LPS C_A 72_ 1	LPS C_G PUC OM	LPS C_G PU_ PBI ST	LPS C_G PUC ORE	LPS C_C 66X _0	LPS C_C 66X _PB IST_ 0	LPS C_C 66X _1	LPS C_C 66X _PB IST_ 1	LPS C_R 5FS S0_ CORE0	LPS C_R 5FS S0_ CORE1	LPS C_R 5FS S_P BIS T_0	LPS C_R 5FS S1_ CORE0	LPS C_R 5FS S1_ CORE1	LPS C_R 5FS S_P BIS T_1	LPS C_D ECO DE_ 0	LPS C_D ECO DE_ PBI ST	LPS C_E NC ODE _0	LPS C_E NC ODE _PB IST	LPS C_D MPA C	LPS C_S DE	LPS C_D MPA C_P BIS T	LPS C_V PAC	LPS C_V PAC _PB IST
LPSC_GPUCORE	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_C66X_0	N	N	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_C66X_PBIST_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_C66X_1	N	N	N	N	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_C66X_PBIST_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_R5FSS0_CORE0	N	N	N	N	N	N	N	N	N	P	Y	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_R5FSS0_CORE1	N	N	N	N	N	N	N	N	P	N	Y	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_R5FSS_PBIST_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_R5FSS1_CORE0	N	N	N	N	N	N	N	N	N	N	N	N	P	Y	N	N	N	N	N	N	N	N	N
LPSC_R5FSS1_CORE1	N	N	N	N	N	N	N	N	N	N	N	P	N	Y	N	N	N	N	N	N	N	N	N
LPSC_R5FSS_PBIST_1	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DECODE_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N	N	N	N
LPSC_DECODE_PBIST	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_ENCODE_0	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N	N
LPSC_ENCODE_PBIST	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DMPAC	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N
LPSC_SDE	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_DMPAC_PBIST	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LPSC_VPAC	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y
LPSC_VPAC_PBIST	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

#### 5.2.2.2.1.4 PSC: Power Domain and Module States

The PSC organizes device modules into different power domains and into different module domains (clock domains). Note that there is no relationship between a power domain and a module domain. These are completely separate entities and may be controlled separately.

##### 5.2.2.2.1.4.1 Power Domain States

A power domain can be only in one of the two states (ON or OFF), defined as follows:

- **ON:** Power to the power domain is on. Logic/memories are awake.
- **OFF:** Power to the power domain is off. Logic/memories are turned off.

The AlwaysOn power domain in WKUP domain is always in the ON state when the device is powered on. The AlwaysOn power domain in MAIN domain is software controlled and is in OFF state in MCU-ONLY and DeepSleep modes. The other power domains can be in either the ON or OFF state; that is, the logic/memory for a specific module can remain powered down if it is not used.

##### 5.2.2.2.1.4.2 Module States

A module can be in one of the following states: *Disable*, *Enable*, *SwRstDisable*, or *SyncRst*. As shown in [Table 5-69](#), these states correspond to combinations of module reset asserted or de-asserted and module clock on or off. Note that module reset is defined as to a full reset of a given module, such that all hardware is put back into its default state.

**Table 5-69. Module States**

Module State	Module Reset	Module Clock	Disabled Request to Interconnect	VBUS Request
Disable	De-asserted	OFF	Asserted	Re-directed to Infrastructure NULL endpoint.
Enable	De-asserted	ON	De-asserted	VBUS requests are served.
SwRstDisable	Asserted	OFF	Asserted	Re-directed to Infrastructure NULL endpoint.
SyncRst	Asserted	ON	Asserted	Re-directed to Infrastructure NULL endpoint.

##### 5.2.2.2.1.4.3 Local Reset

In addition to module reset described in the previous section, some modules can be reset using a special local reset, see [Table 5-70](#) and [Table 5-71](#). When local reset is asserted, the internal memories (L1P, L1D, and L2) for the core are still accessible. The local reset resets only the corresponding module, not the rest of the chip. Local reset is intended to be used by the watchdog timers to reset the processor core in the event of an error. The procedures for asserting and de-asserting local reset are as follows (y denotes the module domain number):

1. Set WKUP\_PSC0\_MDCTL\_y[8] LRST or PSC0\_MDCTL\_y[8] LRST to 0x0 to assert local reset.
2. Set WKUP\_PSC0\_MDCTL\_y[8] LRST or PSC0\_MDCTL\_y[8] LRST to 0x1 to de-assert local reset.

**Table 5-70. Local Reset of Modules Controlled by WKUP\_PSC0**

LPSC Index	Modules	Local Reset
0	MCU_PDMA_G2_0, MCU_PDMA_ADC0, MCU_CLK8_ECC_AGGR0_CFG, MCU_PDMA_G1_0, MCU_PDMA_G0_0, MCU_CPT2_PROBEs, MCU_CBASS_FW0, MCU_ARM_ATB_FUNNEL0, MCU_CPT2_AGGREGATOR0, MCU_CBASS0, MCU_NAVSS0, MCU_CPSW_2GUSS0, MCU_FSS0, MCU_SPI0, MCU_SPI1, MCU_SPI2, MCU_UART0, MCU_I2C0, MCU_I2C1, MCU_TIMER0, MCU_TIMER1, MCU_TIMER2, MCU_TIMER3, MCU_TIMER4, MCU_TIMER5, MCU_TIMER6, MCU_TIMER7, MCU_TIMER8, MCU_TIMER9, MCU_DCC0, MCU_DCC1, MCU_DCC2, MCU_ESM0, MCU_PSRAM0, MCU_SRAM0, MCU_SEC_MMR0, MCU_PLL_CFG0, MCU_CTRL_MMR0, WKUP_DPPA0, WKUP_GPIOMUX_INTRTR0, WKUP_CLK4_ECC_AGGR0_CFG, WKUP_VTM0, WKUP_CBASS_FW0, WKUP_CBASS0, WKUP_CTRL_MMR0, WKUP_PSC0, WKUP_CTRL_MMR0, WKUP_ESM0	Y
1	WKUP_DMSC0	Y
2	-	Y
3	WKUP_GPIO0, WKUP_UART0, WKUP_I2C0	N
4	-	Y

**Table 5-70. Local Reset of Modules Controlled by WKUP\_PSC0 (continued)**

LPSC Index	Modules	Local Reset
5	-	Y
6	MCU_EFUSE0, MCU_PBIST0	N
7	-	Y
8	MCU_MCANSS0	N
9	MCU_MCANSS1	N
10	MCU_FSS_OSPI0	N
11	MCU_FSS_OSPI1	N
12	MCU_FSS0_HPBO	N
13	MCU_I3C0	N
14	MCU_I3C1	N
15	MCU_ADC0	N
16	MCU_ADC1	N
17	-	Y
18	-	Y
19	MCU_RTIO, MCU_R5FF0_CORE0	Y
20	MCU_RTIO, MCU_R5FF0_CORE1	Y
21	MCU_R5FF0_PBIST	N

**Table 5-71. Local Reset of Modules Controlled by PSC0**

LPSC Index	Modules	Local Reset
0	R5FSS0_INTRTR0, R5FSS1_INTRTR0, CSI_PSILSS0, CPT2_HC_AGGR0, CPT2_AC_AGGR0, C66SS1_INTRTR0, GPIO2, GPIO3, GPIO4, GPIO5, GPIO6, GPIO7, PDMA_UART_PSILSS0, MSRAM0, PDMA_MISC_PSILSS0, PDMA_AASRC_PSILSS0, PDMA_DEBUG_PSILSS0, PDMA_UART_G2, PDMA_UART_G1, PDMA_UART_G0, PDMA_G3, PDMA_G2, PDMA_G1, PDMA_G0, PDMA_MCASP_G1, PDMA_MCASP_G0, PDMA_AASRC0, C66SS0_INTRTR0, CSITXRX_CLK4_ECC_AGGR0, MAINCLK4_ECC_AGGR1, MAINCLK2_ECC_AGGR0, INFRACLK2_ECC_AGGR0_CFG, NAVSS0_ECC_AGGR0, DEBUG_SUSPENDRTR0, , MAIN2MCU_PLS_INTRTR0, TIMESYNC_INTRTR0, CMP_EVT_ROUTER0, GPIOMUX_INTRTR0, MAIN2MCU_LVL_INTRTR0, CPT2_PROBES, CBASS_FW0, ESM0, DCC0, DCC1, DCC2, DCC3, DCC4, DCC5, DCC6, DCC7, DCC8, DCC9, DCC10, DCC11, DCC12, PLL_CTRL0, PSC0, SEC_MMR0, PLL0_CFG, CTRL_MMR0, eFUSE0, GTC0, BIST, CPT2_AGGR0, NAVSS0, PSRAM_ARM_C71_BOOTVECTOR, PSRAM_ARM_C66_BOOTVECTOR, TIMER4, TIMER5, TIMER6, TIMER7, TIMER8, TIMER9, TIMER10, TIMER11, TIMER12, TIMER13, TIMER14, TIMER15, TIMER16, TIMER17, TIMER18, TIMER19, GPIO0, GPIO1, MSMC0, GIC0	Y
1	DFTSS0	N
2	PBIST_RC_NB, PBIST_RC_NAVSS, PBIST_HC, PBIST_INFRA0	N
3	AASRC0, MCASP0, MCASP1, MCASP2, MCASP3, MCASP4, MCASP5, MCASP6, MCASP7, MCASP8, MCASP9, MCASP10, MCASP11	N
4	ATL0	N
5	MLBSS0	N
6	ECAP0, ECAP1, ECAP2, EQEP0, EQEP1, EQEP2, EPWM0, EPWM1, EPWM2, EPWM3, EPWM4, EPWM5	N
7	UART0, UART1, UART2, UART3, UART4, UART5, UART6, UART7, UART8, UART9, MCSPI0, MCSPI1, MCSPI2, MCSPI3, MCSPI4, MCSPI5, MCSPI6, MCSPI7, I2C0, I2C1, I2C2, I2C3, I2C4, I2C5, I2C6	N
8	ELM0, GPMC0	N
9	VPFE0	N
10	-	Y
11	-	Y
12	-	Y

**Table 5-71. Local Reset of Modules Controlled by PSC0 (continued)**

LPSC Index	Modules	Local Reset
13	C66_DEBUG_CELL0, PDMA_DEBUG_C66, PDMA_DEBUG_CCMCU, DBG_CBASS0, C66_DEBUG_CELL1, DEBUG_CELL1, DEBUG_CELL0, CC_DEBUG_CELL0, CXSTM500SS0, DEBUGSS_CV0	Y
14	-	N
15	DDRSS0	N
16	-	N
17	-	N
18	-	Y
19	PBIST_CC_Top	N
20	USBSS0	N
21	USBSS1	N
22	-	N
23	MMCSD1	N
24	MMCSD2	N
25	MMCSD0	N
26	UFS0	N
27	-	N
28	PCIE0	N
29	PCIE1	N
30	PCIE2	N
31	PCIE3	N
32	-	N
33	I3C0	N
34	MCANSS0	N
35	MCANSS1	N
36	MCANSS2	N
37	MCANSS3	N
38	MCANSS4	N
39	MCANSS5	N
40	MCANSS6	N
41	MCANSS7	N
42	MCANSS8	N
43	MCANSS9	N
44	MCANSS10	N
45	MCANSS11	N
46	MCANSS12	N
47	MCANSS13	N
48	DSS0	N
49	PBIST_EDPDSI0	N
50	DSS_DSI0	N
51	EDP_PHY0, DSS_EDP0	N
52	-	N
53	CSI_RX_IF0	N
54	CSI_RX_IF1	N
55	-	N
56	CSI_TX_IF0	N



**Table 5-71. Local Reset of Modules Controlled by PSC0 (continued)**

LPSC Index	Modules	Local Reset
57	DPHY_TX0	N
58	DPHY_RX0	N
59	DPHY_RX0	N
60	-	N
61	PRU_ICSSG0	Y
62	PRU_ICSSG1	Y
63	CPSW_9G	N
64	SERDESS0	N
65	SERDESS1	N
66	SERDESS2	N
67	SERDESS3	N
68	SERDESS4	N
69	-	N
70	TIMER0	N
71	TIMER1	N
72	TIMER2	N
73	TIMER3	N
74	MMA0, C71SS0, RTI_16	Y
75	PBIST for C71SS0	N
76	-	Y
77	-	N
78	A72 cluster	Y
79	PBIST for A72 cluster	N
80	RTI_0	Y
81	RTI_1	Y
82	-	Y
83	-	N
84	-	Y
85	-	Y
86	GPU_COMMON, RTI_15	Y
87	PBIST_GPU0	N
88	GPU_CORE	Y
89	RTI_24, C66_DSPSS0	Y
90	PBIST_C66_DSPSS0	N
91	RTI_25, C66_DSPSS1	Y
92	PBIST_C66_DSPSS1	N
93	RTI_28, R5FSS0_CORE0	Y
94	RTI_29, R5FSS0_CORE1	Y
95	PBIST_R5FSS0	N
96	RTI_30, R5FSS1_CORE0	Y
97	RTI_31, R5FSS1_CORE1	Y
98	PBIST_R5FSS1	N
99	D5520MPx	Y
100	PBIST_DECODER	N
101	VXE384MP2	Y

**Table 5-71. Local Reset of Modules Controlled by PSC0 (continued)**

LPSC Index	Modules	Local Reset
102	PBIST_ENCODER	N
103	DMPAC_DOF0, DMPAC0	Y
104	DMPAC_SDE0	N
105	PBIST_DMPAC	N
106	VPAC_NF0, VPAC0_MSC, VPAC_LDC0, VPAC_VISS0, VPAC0	Y
107	PBIST_VPAC0	N

#### 5.2.2.2.1.5 PSC: Executing State Transitions

This section describes how to execute state transitions for power domains and modules. Examples show how to enable only power domains, only module domains, or a combination. Although the user has complete control of the sequencing, see [Section 5.2.2.2.1.5.4, Recommendations for Power Domain/Module Sequencing](#) for recommendations.

##### 5.2.2.2.1.5.1 Power Domain State Transitions

This section describes the basic procedure for transitioning the state of a power domain.

#### Note

The PD\_WKUP power domain is always in the ON state when the device is powered-on, and therefore it is not possible to transition this domain to the OFF state. Conversely, the other domains are in the OFF or ON state when the device is powered-on. Transitions between ON and OFF states need to follow the procedure below. See the device-specific Datasheet and respective module Technical Reference Manual chapter for any other considerations especially while turning off the power domain.

The procedure for power domain state transitions is shown below (x denotes the power domain number, y denotes the module domain number):

1. Wait for WKUP\_PSC0\_PTSTAT[x] GOSTAT or PSC0\_PTSTAT[x] GOSTAT to clear to 0. Wait for any previously initiated transitions to finish before initiating a new transition.
2. Set WKUP\_PSC0\_PDCTL\_y[0] NEXT or PSC0\_PDCTL\_y[0] NEXT for an ON (1) or OFF (0) transition. NOTE: When WKUP\_PSC0\_PTCMD[x] GO or PSC0\_PTCMD[x] GO is set to 1 in the next step, the PDCTLx[0] NEXT field of this power domain and the WKUP\_PSC0\_MDCTL\_y[4-0] NEXT or PSC0\_MDCTL\_y[4-0] NEXT field of the module in this power domain are evaluated. Therefore, the WKUP\_PSC0\_MDCTL\_y[4-0] NEXT or PSC0\_MDCTL\_y[4-0] NEXT field may be set for multiple modules before executing this step.
3. Set WKUP\_PSC0\_PTCMD[x] GO or PSC0\_PTCMD[x] GO to 1 to initiate the state transition(s). The PSC turns on or off the logic/memory for that particular domain.
4. Wait for WKUP\_PSC0\_PTSTAT[x] GOSTAT or PSC0\_PTSTAT[x] GOSTAT to self-clear to 0. The domain is safely in the new state only after WKUP\_PSC0\_PTSTAT[x] GOSTAT or PSC0\_PTSTAT[x] GOSTAT is cleared to 0.

##### 5.2.2.2.1.5.2 Module State Transitions

This section describes the procedure for transitioning the module state.

### Note

The following procedure is applicable for all LPSC-controlled modules. To transition the module state, one must be aware of several system considerations. Transitions between Enable and Disable states need to follow the procedure below. See the device-specific Datasheet and peripheral chapter for any other considerations especially while transitioning a module to Disable state. Also, before transitioning a module to Enable, if the logic/memories are not in the PD\_WKUP and GP\_CORE\_CTL power domains, they must be turned on before or in parallel with the transition. See [Section 5.2.2.2.1.5.1, Power Domain State Transitions](#).

The procedure for module state transitions is shown below (x denotes the power domain number, y denotes the module domain number):

1. Wait for WKUP\_PSC0\_PTSTAT[x] GOSTAT or PSC0\_PTSTAT[x] GOSTAT to clear to 0x0. Wait for any previously initiated transitions to finish before initiating a new transition.
2. Set WKUP\_PSC0\_MDCTL\_y[4-0] NEXT or PSC0\_MDCTL\_y[4-0] NEXT to Enable (0x3) or Disable (0x0). Note that transitions in multiple WKUP\_PSC0\_MDCTL\_y[4-0] NEXT or PSC0\_MDCTL\_y[4-0] NEXT fields may be set in this step as long as the corresponding power domain is on.
3. Set WKUP\_PSC0\_PTCMD[x] GO or PSC0\_PTCMD[x] GO to 1 to initiate the transition(s).
4. Wait for WKUP\_PSC0\_PTSTAT[x] GOSTAT or PSC0\_PTSTAT[x] GOSTAT to self-clear to 0. The module is safely in the new state only after WKUP\_PSC0\_PTSTAT[x] GOSTAT or PSC0\_PTSTAT[x] GOSTAT clears to 0.

#### 5.2.2.2.1.5.3 Concurrent Power Domain/Module State Transitions

This section describes the basic procedure for transitioning the state of a power domain and module domain for modules that are not in the AlwaysOn domain. This may be done separately as described in the sections above, if desired.

The procedure for concurrent power domain/module state transitions is shown below (x denotes the power domain number, y denotes the module domain number):

1. Wait for WKUP\_PSC0\_PTSTAT[x] GOSTAT or PSC0\_PTSTAT[x] GOSTAT to clear to 0. Wait for any previously initiated transitions to finish before initiating a new transition.
2. Set WKUP\_PSC0\_PDCTL\_y[0] NEXT or PSC0\_PDCTL\_y[0] NEXT for an ON (1) transition.
3. Set WKUP\_PSC0\_MDCTL\_y[4-0] NEXT or PSC0\_MDCTL\_y[4-0] NEXT to Enable (0x3). Note that transitions in multiple WKUP\_PSC0\_MDCTL\_y[4-0] NEXT or PSC0\_MDCTL\_y[4-0] NEXT fields may be set in this step as long as the corresponding power domain is on.
4. Set WKUP\_PSC0\_PTSTAT[x] GOSTAT or PSC0\_PTSTAT[x] GOSTAT to 1 to initiate the state transition(s). The PSC will turn on the logic/memory for that particular domain, starts the module clock, then de-asserts the module reset.
5. Wait for WKUP\_PSC0\_PTSTAT[x] GOSTAT or PSC0\_PTSTAT[x] GOSTAT to self-clear to 0. The module is safely in the new state only after WKUP\_PSC0\_PTSTAT[x] GOSTAT or PSC0\_PTSTAT[x] GOSTAT clears to 0.

#### 5.2.2.2.1.5.4 Recommendations for Power Domain/Module Sequencing

As noted in [Section 5.2.2.2.1.4, Power Domain and Module States Defined](#) and [Section 5.2.2.2.1.5, Executing State Transitions](#), a particular peripheral's power domain must be enabled before the module is enabled.

Unless there is a system-level reason to perform each function separately, the recommendation is to use the sequence listed in rather than individual sequencing. Even though a single write to the appropriate GO bit starts the power domain and module transition, it is still sequenced such that the memory/logic is certain to turn on before the module is enabled. Thus, there is no ill effect in performing these together.

It is important to know that when a power domain is enabled, all logic/memories for that module are immediately turned on and moved to an active state. This means that there will be a spike in current consumption at that particular time. In other words, di/dt will be affected. To minimize the effect on di/dt of enabling power domains, it is recommended to power up modules one at a time. It is also recommended that all modules be initialized early

in the application to avoid a large spike in  $di/dt$  during normal operation where the application may already be drawing a substantial amount of current from the supply.

#### 5.2.2.2.1.6 PSC: Emulation Support in the PSC

The PSC supports commands that allow emulation tools to have some control over the state of power domains and modules.

In particular, the PSC supports the following emulation commands:

- Power on and enable features:
  - **Force Power:** Allows emulation to force the power domain into an ON state.
  - **Force Active:** Allows emulation to force the power domain into an ON state and force the module into the Enable state.
- Reset features:
  - **Assert Reset:** Allows emulation to assert the module's local reset.
  - **Wait Reset:** Allows emulation to keep local reset asserted for an extended period of time after software initiates local reset de-assert.
  - **Block Reset:** Allows emulation to block software initiated local and module resets.

Local reset applies only to the domains listed in [Table 5-70](#) and [Table 5-71](#); module reset applies to all other domains.

#### 5.2.2.2.1.7 PSC: A72SS, MSMC, MCU Cortex-R5F, C71SS0, and C66SS Subsystem Power-Up and Power-Down Sequences

##### 5.2.2.2.1.7.1 ARMi\_COREn Power State Transition

Each ARMi\_COREn has its own power domain, see [Table 5-61](#). Each (PD\_A72\_k) of these power domains can be transitioned independently. [Table 5-72](#) lists the possible A72 Core power states. See also [Figure 5-23](#) for the power states of a core. No ARMi\_COREn can transition out of the ARMi\_COREn\_OFF or ARMi\_COREn\_RESET states unless respective CC\_ARMSS is in the A72SS\_ON state.

ARMi\_COREn can be in either WFI or WFE state after executing a WFI (Wait for Interrupt) or WFE (Wait For Event) Instruction, in which the block is On, but has additional clock gating applied to lower power.

**Table 5-72. A72 Core Power States**

Power Domain (PD_A72_n)	L2 Domain	L2 RAM Domain	Processors State
OFF	OFF	OFF	OFF
L2 Dormant Mode <b>Not supported in this family of devices.</b>	OFF	ON	OFF
		RET <sup>(1)</sup>	
L2 Retention Mode <b>Not supported in this family of devices.</b>	ON	RET	WFx <sup>(1)</sup>
			RET
			OFF
ON	ON	ON	ON
			WFx
			RET
			OFF

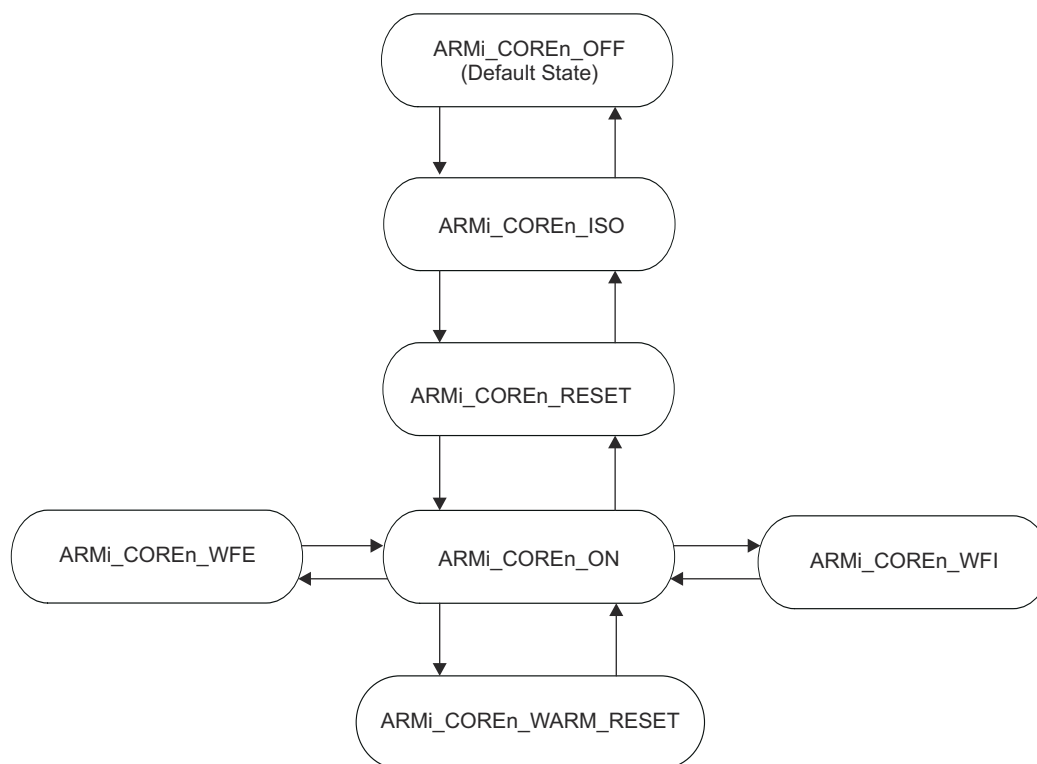
(1) RET = Retention; WFx is either WFI or WFE

#### Note

Note that this family of device does not support retention. For more information about the Arm A72 core transitions among the power states, see Power Management Section in Arm® Cortex®-A72 MPCore Processor Technical Reference Manual, available at <http://infocenter.arm.com/help/index.jsp>.

### Note

Cold reset signal of the ARMi\_COREn of A72SS is directly mapped to PORz reset on PSC0. The primary reset signals are directly mapped to MOD\_G\_RST on PSC0.



power-004

**Figure 5-23. ARMi\_COREn Power Transitions**

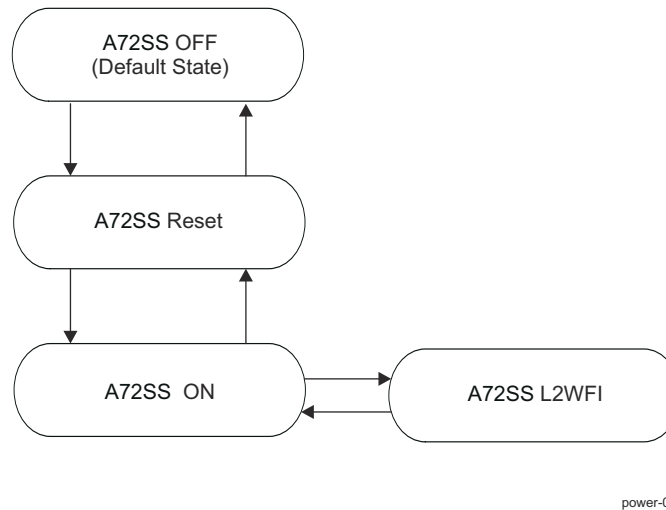
#### 5.2.2.1.7.2 A72SS Power State Transition

Each A72SS has its own power domain, see [Table 5-61](#). Each (PD\_A72\_Cluster\_i) of these power domains can be transitioned independently. See [Figure 5-23](#) for the power states of the A72SS.

If at least one ARMi\_COREn is ON, the respective CC\_ARMSS power domain has to be ON. The sequencing shall ensure that all cores except Core 0 are powered down (in ARMi\_COREn\_ISO or ARMi\_COREn\_OFF state) and CORE 0 is in ARMi\_CORE0\_RESET state before respective CC\_ARMSS domain can be powered down.

### Note

For more information about the Arm A72 cluster transitions among the power states, see Power Management Section in Arm® Cortex®-A72 MPCore Processor Technical Reference Manual, available at <http://infocenter.arm.com/help/index.jsp>.



**Figure 5-24. A72SS Power Transitions**

#### 5.2.2.1.7.3 GIC0 Sequencing to Support A72SS Power Management

When a CC\_ARMSS is to be powered down or with gated clocks, system shall disable sending any type of interrupts - LPI/PPI/SPI, to that A72SS. Also, sending any transactions over the AXI streaming interface to the A72SS shall be disabled.

GIC0 wake-up signal is connected to WKUP\_DMSC0. WKUP\_DMSC0 will have to wake-up the A72SS if a specific core is expected to service an interrupt and the core is powerdown or with gated clock.

#### 5.2.2.1.7.4 Power management features supported by C7x Corepac

Table 5-73 lists the supported power states of C7x Corepac.

**Table 5-73. Power Management Support by C7x**

Memory	Logic	Clock/Reset	C7x Corepac
OFF	OFF	Clockgated + reset	Supported
OFF	ON	Clockgated	Not supported
OFF	ON	Clockgated + reset	Supported
Retention (reton/retgood in tagrams's, not RTA)	ON	Clockgated	Supported
Retention	OFF	-	Not supported
Retention	ON	Reset	Not supported

#### 5.2.2.1.7.5 C7x CorePac Clkstop/Powerdown/Disconnect Sequencing

1. C7x PBIST CLKSTOP/ACK handshaking should be complete.
2. Software to program DRU to disable any L2 pre-warm or MMU pre-warm:
  - a. DRU DTR/pre-warm initiated by that core: Software in the C7x corepac should ensure completion.
  - b. DRU DTR/pre-warm initiated by other cores
3. Programmer executes specific sequence to initiate disconnect in response to a secure interrupt:
  - a. System software to reprogram CLEC router to disable events to C7x CorePac
  - b. Streaming Engine (SE) open streams should have been closed.
  - c. Flush L1D cache with MVC L1D ECR instructions.
  - d. Flush L2 cache with MVC L2 ECR instructions and set L2\$ size = 0 (any pre-warms from system will not generate MDMA traffic).
  - e. Execute IDLE instruction. The IDLE instruction monitor idle status of L1I, L1D, SE, CMMU, MMA blocks and send out an overall idle status to powerdown controller by monitoring the following:

- i. Loads/Stores: The CPU keeps track of load/store operations itself to know when they have completed.
  - ii. Instruction Fetch: The CPU tells the PMC to stop instruction prefetching and keeps track of when the PMC has stopped fetching new instructions.
  - iii. Streaming Engine: The CPU monitor the TSR register SE\_ACTIVE bits to ensure all streaming engines have been closed. Also, in the case of a CMO or cache pre-warm stream, the SE will send an “SCOHERENCE\_ACTIVE” signal to the CPU to let CPU know that all transactions that it initiates to L2 has completed.
  - iv. CPU\_IDLE asserted
  - v. CorePac level idle asserted. This signal is readable through the DMSC memory-mapped “corepac power management status register”. This signal represents a combination of the below idle signals:
    1. CPU\_IDLE
    2. UMC\_IDLE.
4. System asserts MSMC\_\*\_CPU\_PWR\_SCR\_DISABLE\_REQ to Compute Cluserter.
  5. Compute Cluserter level asserts MSMC MSMC\*\_CPU\_PWR\_SCR\_DISABLE\_REQ
    - This ensures that MSMC will not send any snoops/SDMA into C7x corepac
    - New transactions from MSMC into C7x CorePac will be stalled.
  6. System waits for MSMC\_\*\_CPU\_PWR\_SCR\_DISABLE\_ACK from MSMC\_CORE and asserts MSMC\_\*\_CPU\_PWR\_SCR\_DISABLE\_DONE
    - Ensures that all outstanding transactions to C7x are complete.
    - All new transactions to C7x CorePac are terminated with error status in MSMC.
  7. System asserts COREPAC\_CLKSTOP\_REQ (could be asserted anytime earlier as well).
  8. C7x CorePac asserts COREPAC\_CLKSTOP\_ACK
    - Qualified with idle of all corepac components and CLKSTOP\_ACK of debug components.
  9. C7x CorePac can be clockgated.

---

#### Note

Once the CorePac is disconnected from MSMC, it can be clockgated or powered-down. If clockgated, clocks can be restarted and CLEC events can be reprogrammed to wakeup the core.

System has to ensure that CLEC events to a C7x CorePac are disabled prior to clockgating or powerdown of that corepac.

---

#### Note

For more information about C7x signals and registers, see [Section 6.5, C71x DSP Subsystem](#).

---

##### 5.2.2.2.1.7.6 MSMC0 Clkstop/Powerdown/Disconnect Sequencing

The following sequence has to be performed in order to gate the MSMC0 clock:

1. All A72SS connected to MSMC0 shall be in clkstop (or) powerdown state:
  - a. Ensure that all Direct-TR initiated by the respective core is complete (by polling TR complete register).
  - b. A72SS powerdown sequencing are described in the respective sections in Arm® Cortex®-A72 MPCore Processor Technical Reference Manual, available at <http://infocenter.arm.com/help/index.jsp>.
  - c. This ensures that MSMC\_\*\_CPU\_PWR\_SCR\_DISABLE\_ACK is asserted for all CorePacs (and EN\_ACK if connected to an Arm CorePac are IDLE).
  - d. Note that L3 cache may have dirty lines at this point and needs to be written back to DDRSS0 RAM.
2. DRU has to be taken into idle state:
  - a. System software ensures that DRU does not initiate any new requests.
  - b. UDMA-P initiated TR need to be completed.
  - c. System de-asserts the PSIL link slave port of NAVSS0 to DRU.
3. System ensures that VBUSP\_CFG, VBUSP\_DBG, and VBUSP\_DMSC slave ports are disconnected in CBASS0.



4. System software guarantees that SoC0/SoC1 ports do not initiate any transactions to DDRSS0 or MSMC0 SRAM.
5. System software to disable MSMC0 scrubber.
6. System software to change L3 cache size to 0 by writing to MSMC0 CACHE\_CTRL register:
  - a. Note: This operation has to done by secure software running in WKUP\_DMSC0 only. User code shall not be allowed to change L3 cache size.
  - b. Results in L3 cache flush to DDRSS0.
  - c. Software to poll MSMC0 CACHE\_CTRL register to ensure that L3 cache size change is complete.
7. PSC0 disconnects DDRSS0 slave interfaces of MSMC0 and MSMC0 core asserts `MSMC_*_EMIF_PWR_SCR_DISABLE_ACK`.
8. PSC0 ensures SoC0 slave port from NAVSS0 to MSMC0 is disconnected at NAVSS0. Note that this will result in WKUP\_DMSC0 not being able to access MSMC0 control registers.
9. PSC0 disconnects SoC0/SoC1 slave interfaces of MSMC0 and MSMC0 core asserts `MSMC_*_SoC*_PWR_SCR_DISABLE_ACK`.
10. System asserts `MSMC_CLKSTOP_REQ`.
11. `MSMC_WRAP` asserts `MSMC_CLKSTOP_ACK`. This factors in idle of all the `MSMC0_WRAP` components as well:
  - a. `MSMC_CORE_IDLE`, `DRU_IDLE`
  - b. `VBUSP_DBG`, `VBUSP_CFG`, `VBUSP_DMSC` interfaces idle
  - c. `DRU_PSIL_LINK` (output of MSMC) is '0' – denotes disconnected.
12. System can gate the clocks to `MSMC_WRAP`.

#### 5.2.2.2.1.7.7 MCU Cortex-R5F Power Modes

Table 5-74 describes the possible power modes of MCU Cortex-R5F.

**Table 5-74. MCU Cortex-R5F Core Power States**

Mode	CPU Clock Gated	CPU Logic Powered	CPU RAMs powered	Exit to Run Mode Requires
RUN	NO	YES	YES	-
STANDBY	When Idle	YES	YES	Pipeline restart
DORMANT Not supported in this family of devices.	YES	NO	YES	Pipeline restart. Restore registers and configuration from memory
OFF	YES	NO	NO	Pipeline restart. Restore registers and configuration from memory. Invalidate caches and reinitialize caches and TCMs. DMSC intervention.

#### 5.2.2.2.1.7.8 C66x\_DSPSS Power Sequences

Power-on corepac sequence follows this order:

1. The processor C66x\_DSPSS is off.
2. The processor is first powered up by turning on the power switch. At this step, the clock remains off.
3. Asynchronous `clk_divalign` reset is asserted to reset the clocks.
4. `ASYNCLCK_DIVALIGN` reset is deserted, all clocks will start toggling at the desired frequencies.
  - For C66x\_DSPSS, the input clock is `C66_CLK1_CLK` and `PLL_CTRL_CLK`
5. After clock is brought up, POR reset should take place and set debug, PBIST logic in starting state before de-asserted. After POR reset is de-asserted for n cycle which guaranteed the reset took effect, the processor CorePac is powered up, therefore, `ISO_ON` can be de-asserted.
6. Then the processor or core functional logic should be brought to life by de-assert the `CLKSTOP_REQ`, `CLKSTOP_ACK` will follow
7. Finally, the warm `MOD_G_RST` is de-asserted a few cycles to start the processor function.

DSP clock, reset and Power management sequencing in detail:

Power-up:

1. The processor corepac is off.
2. The processor is first powered up by turning on the power switch.
3. ASYNC\_CLK\_DIVALIGN reset is asserted, all clocks will start toggling at the desired frequencies.
4. CLK\_EN should be asserted for a few cycles then deasserted while the corepac is still in reset to bring everything to reset values (acknowledge will follow).
5. After clocks have been brought up, POR\_RST should be deasserted for the corepac.
6. ISO\_EN should be deasserted to disable the isolation cells.

Corepac enable:

1. CLK\_EN should be asserted for a few cycles then deasserted while the corepac is still in reset to bring everything to reset values (acknowledge will follow)
2. MOD\_G\_RST should be deasserted to bring the corepac out of reset.
3. CLK\_EN should be asserted to give the module functional clocks (acknowledge will follow)
4. CLKSTOP\_REQ should be deasserted (and the acknowledge will follow).

SW\_RST\_DISABLE:

1. Processor is put in idle mode by IDLE instruction.
2. Assert CLKSTOP\_REQ in corepac (processor will ensure system is idle before responding with acknowledge).
3. Deassert CLK\_EN to clockgate the corepac (acknowledge will follow).
4. Assert MOD\_G\_RST to put the corepac in reset mode.

Power-down:

1. Configure the PDCCMD register to enable power down mode
2. Enable CPU interrupt(s) corresponding with wakeup and disable all others
3. Execute IDLE instruction
4. Assert CLKSTOP\_REQ in corepac (processor will ensure system is idle before responding with ack)

---

#### Note

C66x DSP Pre-fetch Should Be Disabled before Entering Power Down Mode.

---



---

#### Note

For more information about C66x signals and registers, see [Section 6.4, C66x DSP Subsystem](#).

---

The DSP may hang after multiple iterations of going into C66x Corepac Power-Down and wake up from external events.

The C66x XMC (External Memory Controller) can have outstanding pre-fetch requests when C66x Corepac transitions to a Power Down state. The XMC clocks are gated internally during this transition. While XMC clocks are gated, outstanding pre-fetch request responses are not seen by the XMC which leads to an inconsistent state between the XMC and the L3 Interconnect. When the DSP wakes up, this can manifest as different symptoms within the DSP subsystem, including Cache corruption, incorrect data being returned to the CPU, and can eventually lead to a DSP hang condition.

The steps to avoid this issue are as given below:

1. Ensure the code which places the DSP C66x Corepac to Power Down State (power down entry procedure shown below) is placed in the DSP C66x L2 RAM memory.
2. Set the IDLE bit in PDCCMD register during initialization.
3. Inside the power down entry procedure include the following software sequence:
  - a. Execute MFENCE instruction.

- b. Write 1 to XPF\_CMD.INV (address 0x0800\_0300).
- c. Read XPFACS (address 0x0800\_0304).
- d. Execute IDLE instruction.

While executing multi-threaded DSP software with C66x Corepac Power Down caution should be observed to not allow the power down entry sequence to be preempted and switch context.

#### 5.2.2.2.2 Integrated Power Management (DMSC)

##### 5.2.2.2.2.1 DMSC Power Management Overview

The device has one DMSC module located in WKUP domain - WKUP\_DMSC0, that controls the Power Management. [Figure 5-25](#) describes the top-level diagram of WKUP\_DMSC0 module, including the Power Management interfaces.



**Figure 5-25. Overview of Device Management and Security Control (WKUP\_DMSC0)**

#### 5.2.2.2.1.1 DMSC Power Management Features

DMSC power management supports:

- DMSC control:
  - Contains various control, configuration and status registers for power management functions
- Main features of oscillators control:
  - Controls the main oscillator power down
  - Controls the gating of the POR RC oscillator root clock of the main oscillator to avoid high frequency clock propagation during oscillator wake-up
- DMSC memory power management
  - Power-off of ROM during sleep modes
  - RAM retention during sleep modes
- IO power management
  - Controls IO isolation while going into standby mode
  - Controls IO wake-up capability
- External signals for power control
  - Support for 64 general purpose inputs which can be used by Cortex-M3
- Slave VBUSP port to directly connect to Power Sleep Controller (PSC).

---

**Note**

Power resource management of the device is done via firmware running on DMSC.

---

---

**Note**

For more information how to use DMSC, see TISCI API available at [ti.com](http://ti.com).

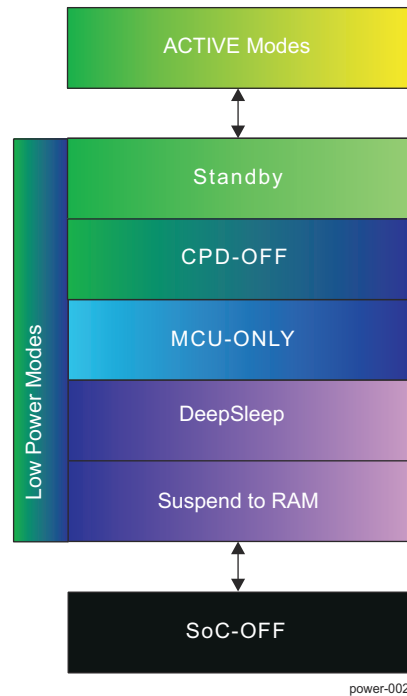
---

### 5.2.3 Device Power States

#### 5.2.3.1 Overview of Device Low-Power Modes

The device supports multiple power modes described in the sections below. The lower power consumption comes at the cost of longer time for recovery to a running mode.

Figure 5-26 depicts the valid power modes for the device.



**Figure 5-26. Power Modes**

Device supports the following power states:

- SoC-OFF: entire device is off by switching off voltage rails externally. External DDR memory may be optionally in Self-Refresh (SR).
- SuspendToRAM: all power rails are OFF except for DDR IO. External DDR memory may be optionally in self-refresh.
- DeepSleep: all processor core power domains are OFF (voltage domain ON). Only WKUP power domains are ON. Some top-level modules that does not have power domains, are ON.
- MCU-ONLY: WKUP and MCU voltage domains are ON, and MAIN SoC side voltage domains are OFF except for 1.8 V IO rail. External DDR memory is in SR optionally.
- Core Power Domain OFF (CPD-OFF): all voltage domains are ON, MAIN SoC side all power domains are OFF. CPD-OFF is alternative to the MCU-ONLY, in case there are restrictions to turn off MAIN SoC side voltage domains.
- Standby: Main processor cores in WFI/WFE, rest of active subsystems idles. .
- Active: main status and control registers are ON, processors and subsystems are ON, based on use cases.

See *Power Estimation Application Note*, for more information about power saved in each low power mode.

#### 5.2.3.2 Voltage Domains

Table 5-75 shows the state of power supply voltages in each low power mode.

**Table 5-75. Voltage Domains State in Low Power Modes**

Power Modes	Voltage domains						
	VD_WKUP/MCU	VD_CORE	VD_CC	Other WKUP (IO, Analog)	Other MCU (IO, Analog)	Other SoC (IO, Analog)	DDREMIF and PHY
SoC-OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
SuspendToRAM	ON	OFF	OFF	OFF	OFF	OFF	ON
DeepSleep	ON	ON	ON	ON	ON	ON	ON
MCU-ONLY	ON	OFF	OFF	ON	ON	ON/OFF	ON
CPD-OFF	ON	ON	ON	ON	ON	ON	ON
Standby	ON	ON	ON	ON	ON	ON	ON

### 5.2.3.3 Power Domains

Table 5-76 shows the state of power domains in each low power mode.

**Table 5-76. Power Domains State in Low Power Modes**

Power Modes	Power Domains															
	GP_COR_E_CTL_WKUP	PD_MCU_R5F0FS	GP_COR_E_CTL	PD_MCA_NSS	PD_DSS	PD_ICSS	PD_9GSS	PD_SERDES_0	PD_SERDES_1	PD_SERDES_2	PD_SERDES_3	PD_SE_RDES_4	PD_SE_RDES_5	PD_TIMER	PD_C7_1x_0	PD_C7_1x_1
SoC-OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
SuspendToRAM	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
DeepSleep	ON	OFF	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	OFF	OFF
MCU-ONLY	ON	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
CPD-OFF	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	OFF	OFF
Standby	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON

**Table 5-77. Power Domains State in Low Power Modes**

Power Modes	Power Domains															
	PD_A72_C_LUSTER_0	PD_A72_0	PD_A72_1	PD_A72_C_LUSTER_1	PD_A72_2	PD_A72_3	PD_GPU_COM	PD_GPU_CORE	PD_C66x_0	PD_C66x_1	PD_R5FSS_0	PD_R5FSS_1	PD_DECODE	PD_ENCODE	PD_DMPAC	PD_VPAC
SoC-OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
SuspendToRAM	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
DeepSleep	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
MCU-ONLY	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
CPD-OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
Standby	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON

### 5.2.3.4 Clock Sources States

Table 5-78 shows the state of global clocks in each low power mode.

**Table 5-78. Global Clock States in Low Power Modes**

Power Modes	Clocks		
	WKUP_HFOSC0	HFOSC1 (if available)	DPLLs
SoC-OFF	OFF	OFF	OFF
SuspendToRAM	OFF	OFF	OFF

**Table 5-78. Global Clock States in Low Power Modes (continued)**

Power Modes	Clocks		
	WKUP_HFOSC0	HFOSC1 (if available)	DPLLs
DeepSleep	OFF	OFF	OFF
MCU-ONLY	ON	ON	Single DPLL locked
CPD-OFF	ON	ON	Single DPLL locked
Standby	ON	ON	Bypass / Low Frequency

### 5.2.3.5 Wake-up Sources

Table 5-79 shows Wake-up Sources in each low power mode.

#### Note

MCU-ONLY state is not considered as a state that needs a wake-up source as processor is alive. A72SS and/or R5FSS can be in WFI and any interrupt which is routable to R5F or WKUP\_DMSC0 could cause exit from this state.

**Table 5-79. Wake-up Sources in Low Power Modes**

Power Modes	Wake-up source
SoC-OFF	MCU_PORz.
SuspendToRAM	TBD
DeepSleep	WKUP_DMSC0 Timers (any), WKUP_UART0, WKUP_GPIO0, WKUP_GPIO1, WKUP_I2C .
CPD-OFF	WKUP_GPIO0, WKUP_GPIO1, WKUP_UART0, WKUP_I2C0, WKUP_TIMERS. All instances of the following peripherals in MCU domain - UART, I2C, SPI, MCAN, Timers.
Standby	All peripherals with interrupts routed to GIC.

### 5.2.3.6 Device Power States and Transitions

The LPMs always transition from run state. If the device software decides to transition from one low power mode to another low power mode, it needs to go through the run state first.

Normally, the software initiates transition to low power mode and interrupts/events are the exit trigger.

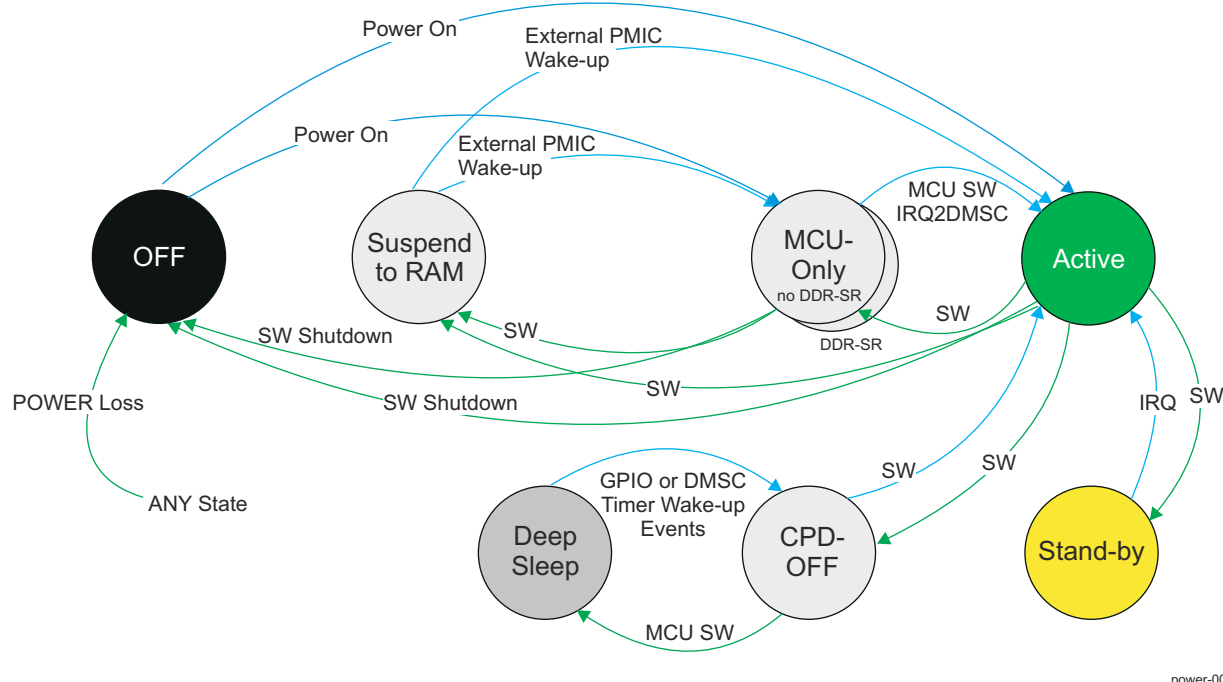
Figure 5-27 shows the state diagram of transitions between LPMs.

#### Note

The state diagram represents higher layer logical requesters, as any actual operation are actualized by WKUP\_DMSC0 firmware.

For more information about wake-up requesters, see Table 5-79, *Wake-up Sources and Debug Capability in Low Power Modes*.





**Figure 5-27. Transition between Low Power Modes**

#### Note

For details on power domain and module state transitions, please refer to [Section 5.2.2.2.1.5, Executing State Transitions](#).

#### Note

CTRLMMR\_WKUP\_DEEPSLEEP\_CTRL is used to force all WKUP and MAIN IOs into DeepSleep mode, for more information about control registers, see *Control Module (CTRL\_MMR)*.

The device supports the below transitioning schemes to low power states. No voltage domain manipulations. Only power domains:

- ACTIVE->Standby
- ACTIVE->CPD-OFF
- ACTIVE->DeepSleep, device effectively goes through CPD-OFF

Requiring voltage domain and supply management:

- ACTIVE->MCU-ONLY
- ACTIVE->SuspendToRAM
- MCU-ONLY->SuspendToRAM

#### 5.2.3.6.1 LPM Entry Sequences

[Table 5-80](#) lists all the possible steps in entering LPM. In the tables further down used are the following abbreviations:

- HLOS stands for High-Level Operational System.
- RTOS stands for Real-Time Operational System.
- PM stands for Power Management.
- FW stands for Firmware.
- PM-FW stands for Power Management - Firmware.

- DMSC-FW stands for DMSC - Firmware.
- SW stands for Software.

**Table 5-80. LPM Entry Sequence**

Step Label	Responsible HW/SW	Step description
ACT-0-0	Various	SoC fully ACTIVE
ACT-1-0	HLOS + PM - FW	Identify conditions to enter low power mode
ACT-1-0.1	HLOS + PM - FW	HLOS/RTOS + DMSC + DMSC-FW PM policy identifies conditions to enter Standby
ACT-1-0.2	HLOS + PM - FW	HLOS/RTOS + DMSC + DMSC-FW PM policy identifies conditions to enter CPD-OFF or DeepSleep
ACT-1-0.3	HLOS + PM - FW	HLOS/RTOS + DMSC + DMSC-FW PM policy: identifies conditions to enter MCU-ONLY
ACT-1-0.4	HLOS + PM - FW	HLOS/RTOS + DMSC + DMSC-FW PM policy: identifies conditions to enter SuspendToRAM
ACT-1-0.5	HLOS + PM - FW	HLOS/RTOS + DMSC + DMSC-FW PM policy: identifies conditions to enter Software Shutdown
ACT-2-0	SW	-Virtual-machines running in each core have completed all their tasks and any task involving other modules. - Virtual-machines running in each core have released all peripherals. - Virtual machines are either down or on standby.
ACT-6-1-1	SW(C71x)	- Save C71x core context in DDR. - Set C71x in IDLE instruction and memory system ready for c71x to be POWER-OFF. - Turn off power domain for C71x as per PSC dependency constrains (see <a href="#">Section 5.2.2.2.1.3.3, LPSC Dependences Overview</a> ).
ACT-6-2a	SW(A72) + FW	- Save A72 core context in DDR. - Set A72 core_0,1,2,3 into WFI and ready to be POWER-OFF as well as . - Set-up A72_CLUSTER_0 and A72_CLUSTER_1 ready to complete POWER-OFF of each of its 2 cores. - Turn off power domains PD_A72_0, PD_A72_1, PD_A72_2, PD_A72_3 as per PSC dependency constrains (see <a href="#">Section 5.2.2.2.1.3.3, LPSC Dependences Overview</a> ). - Set-up A72_CLUSTER_0 and A72_CLUSTER_1 ready to for both clusters to POWER-OFF. - Turn off power domains PD_A72_CLUSTER_0 and PD_A72_1 as per PSC dependency constrains (see <a href="#">Section 5.2.2.2.1.3.3, LPSC Dependences Overview</a> ).
ACT-6-4	SW	- When going from MCU-ONLY mode to SuspendToRAM no context can be saved to DDR. - Instead a limited amount of MCU context could be saved to external NOR-Flash via FSS.
ACT-7-2	SW(A72) + FW	- Set A72_CORE_0, A72_CORE_1, A72_CORE_2, A72_CORE_3 into WFI.
ACT-8-2	FW	Disable (SwRstDis), LPSC_MAIN_DEBUG and LPSC_MCU_DBG as per PSC dependency constrains (see <a href="#">Section 5.2.2.2.1.3.3, LPSC Dependences Overview</a> ).
ACT-10-1	FW	-Save acceleratorpac context to DDR, MAIN_ACCELLERATOR modules like ENCODER, DECODER, DMPAC. - Prepare accelerators for CLKSTOP and POWER-OFF state. - POWER-OFF power domains for acceleratorpac as per PSC dependency constrains (see <a href="#">Section 5.2.2.2.1.3.3, LPSC Dependences Overview</a> ).

**Table 5-80. LPM Entry Sequence (continued)**

Step Label	Responsible HW/SW	Step description
ACT-10-2-3	SW-FW	<ul style="list-style-type: none"> <li>- Save context from remaining MAIN power-domains that could be turned OFF depending on usage scenario.</li> <li>- POWER-OFF all other power-domains in MAIN domain that could be POWERED-OFF, excluding PD0 since that is always-on and locked.</li> <li>- For PD domains of type PDRIO, they also could be powered-OFF during Low-Power Modes as per PSC dependency constrains (see <a href="#">Section 5.2.2.2.1.3.3, LPSC Dependences Overview</a>).</li> <li>- To be able to be powered-OFF the reset isolated modules all their RST_ISO bits needs to be cleaned inside PSC for all the LPSC components of a given power domain.</li> <li>- At this point the LPSCs associated with PD0 are still enabled.</li> <li>- Power OFF domains and disable LPSCs as per PSC dependency constrains (see <a href="#">Section 5.2.2.2.1.3.3, LPSC Dependences Overview</a>).</li> </ul>
ACT-10-4-1	FW	<ul style="list-style-type: none"> <li>- Save MSMC-CFG to DDR.</li> <li>- Save any other context required to DDR from the top-of the Compute Cluster and MSMC.</li> <li>- Save limit set of WR register values from CTRL_MMR0 to DDR.</li> </ul>
ACT-10-4-2	FW	<ul style="list-style-type: none"> <li>- Save CTRLMMR_MAIN_DEVSTAT in DMSC RAM.</li> <li>- Save CTRLMMR_MAIN_BOOTCFG register from CTRL_MMR0 in DMSC RAM.</li> </ul>
ACT-11-01	SW(R5x) + FW	<ul style="list-style-type: none"> <li>- R5FSSs context save to DDR.</li> <li>- Perform cache manage and WFI sequence for R5FSSs in order to get it ready for POWER-OFF.</li> <li>- Turn off power domains for R5FSSs clusters PD_R5FSS_0 and PD_R5FSS_1 as per PSC dependency constrains (see <a href="#">Section 5.2.2.2.1.3.3, LPSC Dependences Overview</a>).</li> </ul>
ACT-11-4a	SW(C66x) + FW	<ul style="list-style-type: none"> <li>- C66x context save to DDR.</li> <li>- Set C66x in IDLE instruction and memory system ready for C66x to be POWER-OFF.</li> <li>- POWER-OFF C66x (PD_C66x_0 and PD_C66x_1) as per PSC dependency constrains (see <a href="#">Section 5.2.2.2.1.3.3, LPSC Dependences Overview</a>).</li> </ul>
ACT-11-05	FW	<ul style="list-style-type: none"> <li>- Save DDR config to DMSC-RAM.</li> <li>-Set LPSCs for DDR in SwRstDis state.</li> <li>- For PD0 (GP_CORE_CTL) (the only PD still ON) in PSC0, set ALL remaining LPSCs in SwRstDis state (except LPSC0 which is locked) as per PSC dependency constrains (see <a href="#">Section 5.2.2.2.1.3.3, LPSC Dependences Overview</a>).</li> </ul>
ACT-11-06	FW	<ul style="list-style-type: none"> <li>- Set the following LPSC to SwRstDis: LPSC_MAIN2WKUPMCU</li> <li>- Disable all the HSDIVs in MAIN. Except the one providing clock MAIN_PLL0_HSDIV0_CLKOUT to PLLCTRL , namely HSDIV0 of PLL0.</li> <li>- Disable all the PLLs in MAIN. Except PLL0.</li> <li>- Now set PLLCTRL (MAIN_PLL0_HSDIV0_CLKOUT) in bypass mode.</li> <li>- Disable PLL0.</li> <li>- Set all of the following LPSCs, in WKUP_PSC0, to SwRstDis as per PSC dependency constrains (see <a href="#">Section 5.2.2.2.1.3.3, LPSC Dependences Overview</a>): LPSC_DEBUG2DMSC, LPSC_WKUPMCU2MAIN.</li> <li>- If it wasn't already, switch WKUP_HFOSC0 to HFOSC1 as input reference to PLL0, by writing CTRLMMR_WKUP_MAIN_PLL0_CLKSEL[0] CLK_SEL = 1.</li> <li>- POWER-OFF HFOSC1 by writing to CTRLMMR_WKUP_HFOSC1_CTRL[7] PD_C = 1 in WKUP_CTRL_MMR0.</li> </ul>
ACT-12-00	FW	Place DDR in Self Refresh Mode
ACT-12-02	FW	MCU_ONLY.MCU_R5FF0FSS_ACTIVE >> Place DDR in Self Refresh Mode
ACT-13-01	FW(DMSC)	<p>Set all the other LPSCs of WKUP_PSC0 except PD0 (GP_CORE_CTL_WKUP) to SwRstDis state as per PSC dependency constrains (see <a href="#">Section 5.2.2.2.1.3.3, LPSC Dependences Overview</a>), except for the ones below:</p> <ul style="list-style-type: none"> <li>- Keep enable the LPSCs LPSC_WKUP_ALWAYSON, LPSC_DMSC, LPSC_WKUP_GPIO.</li> </ul>

**Table 5-80. LPM Entry Sequence (continued)**

Step Label	Responsible HW/SW	Step description
ACT-13-02	FW(DMSC)	<p>-If the WKUP_GPIOs need to detect wake-up events independently of the WKUP_IO-PM-DCH, then GPIOs need to remain fully functional and clocked during DeepSleep.</p> <p>- To do that do the following:</p> <ul style="list-style-type: none"> <li>Set WKUP_GPIO_CTRL[0] = 1'b1 . This will make sure that even when CLK_EN=0 is applied to GPIO the clock will still be running.</li> <li>Set LPSC_wkup_gpio in CLKSTOP-DISABLE state (NOT SwRstDis state).</li> <li>Go to WKUP_MMR module and switch the GPIO functional clock to 32 Khz.</li> <li>WKUP_GPIO_CLKSEL[1:0] = 2'b10.</li> <li>Or for a second type of test it could be switched to the 12.5Mhz RC clock.</li> <li>WKUP_GPIO_CLKSEL[1:0] = 2'b11.</li> </ul>
ACT-14-01	FW-DMSC	<p><b>For Modes with R5FSS OFF, except MCU-ONLY.R5FSS-OFF&gt;&gt;</b></p> <ul style="list-style-type: none"> <li>MCU_R5FF0FSS context save to DDR.</li> <li>Perform cache manage and WFI/WFE sequence for MCU_R5FF0FSS in order to get it ready for POWER-OFF.</li> <li>Turn off power domain - PD_MCU_R5FF0FSS, for MCU_R5FF0FSS as per PSC dependency constrains (see <a href="#">Section 5.2.2.2.1.3.3, LPSC Dependences Overview</a>).</li> </ul>
ACT-14-02	FW-DMSC	<p><b>For MCU_ONLY.R5FSS-OFF &gt;&gt;</b></p> <ul style="list-style-type: none"> <li>MCU_R5FF0FSS context save to Flash.</li> <li>Perform cache manage and WFI/WFE sequence for MCU_R5FF0FSS in order to get it ready for POWER-OFF.</li> <li>Turn off power domain - PD_MCU_R5FF0FSS, for MCU_R5FF0FSS as per PSC dependency constrains (see <a href="#">Section 5.2.2.2.1.3.3, LPSC Dependences Overview</a>).</li> </ul>
ACT-15-1	FW	MCU_R5FF0FSS enters WFI/WFE mode
ACT-14-3	FW	<p><b>ForModes with R5FSS OFF &gt;&gt;</b></p> <ul style="list-style-type: none"> <li>DMSC off power domain for MCU_R5FF0FSS as per PSC dependency constrains (see <a href="#">Section 5.2.2.2.1.3.3, LPSC Dependences Overview</a>).</li> </ul>
ACT-21	FW(DMSC)	DMSC sends commands to PMIC via WKUP_I2C0 to turn off all the supply, so device goes to OFF state.
ACT-22-1	FW(DMSC)	DMSC sends command to PMIC via WKUP_I2C0 to set desired wakeup events inside PMIC which will trigger wake-up from SuspendToRAM.
ACT-22-2	FW(DMSC)	Send command to PMIC via WKUP_I2C0 to assert from the board DDR_RET chip input.
ACT-22-3	FW(DMSC)	DMSC sends command to PMIC via WKUP_I2C0 to turn off all the supply except the supply to DDR IO (and except RTC supply inside PMIC), for PMIC/SoC to enter SuspendToRAM mode (PMIC-LP_State).
ACT-16-1	FW(DMSC)	If enable, disable the following LPSCs: LPSC_MCU_DEBUG, LPSC_MCU_TEST.
ACT-16-2	FW(DMSC)	Before entering MCU_ONLY mode disable all VTM temperature sensors that are on the SOC_MAIN portion of the SoC: VTM_TMPSENS1-4. At the same time make sure that the TEMP sensor in the VD_WKUP is enabled and running to keep monitoring temperature: VTM_TMPSENS0
ACT-17-1	FW	<ul style="list-style-type: none"> <li>Save to DMSC-RAM and disable POK modules related to SoC_MAIN core portion of the SoC via WKUP_CTRL_MMR0 module: POK_VDD_CORE_UV_CTRL, POK_VDD_CPU_UV_CTRL, POK_VDDR_CORE_UV_CTRL, POK_VMON_EXT_UV_CTRL, POK_VDD_CORE_OV_CTRL, POK_VDD_CPU_OV_CTRL, POK_VDDR_CORE_OV_CTRL, POK_VMON_EXT_OV_CTRL</li> <li>First save and disable their contribution to ESM events</li> <li>Next save and disable their contribution to PRG reset, CTRLMMR_WKUP_MAIN_PRG_CTRL all POK_*_EN = 0.</li> </ul>
ACT-17-2	FW	<ul style="list-style-type: none"> <li>Disable any still enable POK modules.</li> <li>Disable any ESM event generation going OFF-Chip.</li> </ul>

**Table 5-80. LPM Entry Sequence (continued)**

Step Label	Responsible HW/SW	Step description
ACT-17-2-2	FW	<p>- DMSC asserts PMIC_POWER_EN1 (the pin for SoC to communicate with PMIC) to turn-off the power supplies to VDD_CORE and VDD_CPU and IO-supplies that feed - USB, SERDES and PLLs in MAIN.</p> <p>or</p> <p>- DMSC sends commands to PMIC via WKUP_I2C0 to turn-off the power supplies to VDD_CORE and VDD_CPU and IO-supplies that feed - USB, SERDES and PLLs in MAIN.</p>
ACT-18	FW	DMSC informs PMIC to shut off supply to WKUP/MCU domain and only keep supply to DDR IO to enter SuspendToRAM state (PMIC-LP_State)
ACT-19-1	FW	<p>- Via WKUP_CTRL_MMR0 module, remove the reset-isolation blocking on the MAIN_PORZ propagation: CTRLMMR_WKUP_POR_RST_CTRL[0] POR_RST_ISO_DONE_Z = 0.</p> <p>- Via WKUP_CTRL_MMR0 module apply PORz to SoC_MAIN: CTRLMMR_WKUP_POR_RST_CTRL[19-16] SW_MAIN_POR = 4'b0110 .</p>
ACT-19-2	FW	Set CTRLMMR_WKUP_MAIN_VDOM_CTRL[0] MAIN_VD_OFF = 1 in WKUP_CTRL_MMR0 module prior to powering off the main voltage domain to ensure proper signal isolation.
ACT-25-00	SW+FW	START MAIN-IO-Power-Management daisy-chain (IO-PM-DCH) low power mode entry sequence via FW:
ACT-25-01	SW+FW	<p>MAIN-IO-Power-Management daisy-chain (IO-PM-DCH) low power mode entry sequence via FW:</p> <p>- Prior to the DMSC IO-PM-DCH sequence, Software-driver and R5F-FW determines what are the deep sleep (DS) configuration values to be set in the CTRLMMR_PADCONFIGi (i = 0 to 172) for all MAIN-IOs, and sets those registers.</p> <p>- Those same configuration values will apply regardless if it is any LowPowerMode entry that uses MAIN-IO-PM daisy-chain. There is only one DS setting in the PADCFG register.</p> <p>NOTE: If the END-TARGET LOW POWER MODE is either of MCU-ONLY or SuspendToRAM modes then:</p> <ul style="list-style-type: none"> <li>• Save all the CTRLMMR_PADCONFIGi (i = 0 to 172) for all MAIN-IOs to DDR (Context Save).</li> <li>• Additionally, for the very limited set of IOs that could be used as wake-up events, save all their registers context to DMSC-RAM.</li> </ul>
ACT-25-02a	FW(DMSC)	<p>Main-IO-PM-DCH:</p> <p>Kick-Off propagation of all the CTRLMMR_PADCONFIGi (i = 0 to 172) registers selections for Wake-up event enables ([29] WKUP_EN bit), and iso-bypass ([23] ISO_BYP bit) to all IOs in main-region IO-PM-DCH. Read-modify write to DMSC power registers, see DMSC addendum for more information.</p>
ACT-25-02b	FW(DMSC)	<p>Main-IO-PM-DCH: Kick-Off propagation of all the CTRLMMR_PADCONFIGi (i = 0 to 172) registers selections for Wake-up event enables ([29] WKUP_EN bit) to all IOs in main-region IO-PM-DCH. When the VDD_CORE is OFF NO-ISOBYPASS is supported. Read-modify write to DMSC power registers, see DMSC addendum for more information.</p>
ACT-25-03	FW(DMSC)	<p>Main-IO-PM-DCH: Kick-Off propagation of all the DeepSleep selection settings for IO inputs pupdsel, pi, gz from CTRLMMR_PADCONFIGi (i = 0 to 172) registers to all IOs in main-region IO-PM-DCH. Write WKUP_CTRL_MMR0 CTRLMMR_WKUP_DEEPSLEEP_CTRL[8] FORSE_DS_MAIN to force the deep sleep value onto pins.</p>
ACT-25-04	FW(DMSC)	<p>Main-IO-PM-DCH: Latch the wakeup event enabled in all IOs whose PADCFG register has WUEN bit set by toggling the WUCLK. WUCLK is toggled by writing '1' to a register in DMSC. WUCLK_CTRL_1 ( B[8] ) then poling until PMCTRL_IO_1.WUCLK_STATUS_1 ( B[9] ) = '1'. Next is to drive WUCLK=0 by writing '0' to MMR bit PMCTRL_IO_1.WUCLK_CTRL_1 ( B[8] ) then poling until PMCTRL_IO_1.WUCLK_STATUS_1 ( B[9] ) = '0'.</p>
ACT-25-05	FW(DMSC)	<p>Main-IO-PM-DCH: Kick-Off isolation and isolation sequence of all IOs in main-region IO-PM-DCH. Read-modify write to DMSC PMCTRL_IO_1, writing 1'b1 to bit PMCTRL_IO_1[24].</p>

**Table 5-80. LPM Entry Sequence (continued)**

Step Label	Responsible HW/SW	Step description
ACT-25-10	SW+FW	DMSC start to set WkupMcu IO into daisy-chain low power mode: - Prior to the DMSC's Wkup-MCU IO-PM-DCH sequence, Software-driver + R5-FW determines what are the deep sleep (DS) configuration values to be set in the Wkup PAD-CFG for all Wkup-IOs, and sets those registers. - Those same config values will apply regardless if it is any LowPowerMode entry that uses Wkup-IO-PM daisy-chain. There is only one DS setting in the PADCFG MMRs.
ACT-25-11	FW(DMSC)	Wkup-IO-Power-Management daisy-chain (Wkup-IO-PM-DCH) low power mode entry sequence via FW: Wait for completion of Isolation in Main-IO-PM-DCH. Keep polling bit PMCTRL_IO_1[25] until it shows '1' to indicate completion of isolation sequence for Main-IO-PM-DCH.
ACT-25-12	FW(DMSC)	Wkup-IO-PM-DCH: Kick-Off propagation of all the CTRLMMR_WKUP_PADCONFIGi (i = 0 to 93) registers selections for Wake-up event enables ([29] WKUP_EN bit), and iso-bypass ([23] ISO_BYP bit) to all IOs in Wkup-region IO-PM-DCH. Read-modify write to DMSC register PMCTRL_IO_0, writing 2'b11 to bits PMCTRL_IO_0[16, 6]
ACT-25-13	FW(DMSC)	Wkup-IO-PM-DCH: Kick-Off propagation of all the DeepSleep selection settings for IO inputs pupdsel, pi, gz from CTRLMMR_WKUP_PADCONFIGi (i = 0 to 93) to all IOs in Wkup-region IO-PM-DCH. Write 0x00000001 to DMSC register FW_CTRL_OUT0_SET.
ACT-25-14	FW(DMSC)	Wkup-IO-PM-DCH: Latch the wakeup event enabled in all IOs whose PADCFG has WUEN bit set by toggling the WUCLK. WUCLK is toggled by writing '1' to MMR bit PMCTRL_IO_0.WUCLK_CTRL_0 ( B[8] ) then poling until PMCTRL_IO_0.WUCLK_STATUS_0 ( B[9] ) = '1'. Next is to drive WUCLK=0 by writing '0' to MMR bit PMCTRL_IO_0.WUCLK_CTRL_0 ( B[8] ) then poling until PMCTRL_IO_0.WUCLK_STATUS_0 ( B[9] ) = '0'.
ACT-25-15	FW(DMSC)	Wkup-IO-PM-DCH: Kick-Off isolation and isolation sequence of all IOs in Wkup-region IO-PM-DCH. Configure WKUP_CTRL_MMR POR_RST_CTRL as needed Read-modify write to DMSC's MMR PMCTRL_IO_0, writing 1'b1 to bit PMCTRL_IO_0[24].
ACT-25-16	FW(DMSC)	Wkup-IO-PM-DCH: Wait for completion of Isolation in Wkup-IO-PM-DCH. Keep polling bit PMCTRL_IO_0[25] until it shows '1' to indicate completion of isolation sequence for Wkup-IO-PM-DCH. At this point the DMSC-M3 will finish/complete the low-power mode code sequence to the point it goes into WFI state. The entering of M3 into WFI causes the DMSC internal low power sequence to kick-in and DMSC will complete it and wait until a wake-up event.
ACT-26	FW	Set LPSC_DEBUG2DMSC, LPSC_WKUPMCU2MAIN, LPSC_MAIN2WKUPMCU to SwRstDis as per PSC dependency constrains (see <a href="#">Section 5.2.2.1.3.3, LPSC Dependencies Overview</a> ).
ACT-28-01	DMSC-FW	(If OFF-Mode or SuspendToRAM-Mode skip this) DMSC-FW config DMSC in LPM0. - Kick LPM Entry in DMSC by setting WFI in DMSC's M3.
ACT-29-01	DMSC-FW	(If OFF-Mode or SuspendToRAM-Mode skip this) -Configure DMSC for LPM4 or LPM3. - This should include enabling the interrupts and wake-up events that are desired to cause a wakeup out of the LPM. - The IO-PM daisy-chains for both Main-IO-PM-DCH and WKUP-IO-PM-DCH must be included among the wake-up events.
ACT-29-02	DMSC-FW	(If OFF-Mode or SuspendToRAM-Mode skip this) -Switch the input clock for WKUP_PLLCTRL0 from MCU_PLL0/HSDIV_CLKOUT to WKUP_HFOSC0_CLKOUT. - Set WKUP_PLL_CTRL register PLLCTL[0] PLEN = 0 and PLLCTL[5] PLENSRC = 1. - The IO-PM daisy-chains for both Main-IO-PM-DCH and WKUP-IO-PM-DCH must be included among the wake-up events. - Set MCU_PLL0 in bypass mode.
ACT-29-03	DMSC-FW	(If OFF-Mode or SuspendToRAM-Mode skip this) -Kick LPM Entry in DMSC by setting WFI in DMSC's M3.
BR-00-1	BRANCH	<<<< Steps-specific to MCU_ONLY.MCU_R5FF0FSS_Active Step-set-BR-00: BEGIN



**Table 5-80. LPM Entry Sequence (continued)**

Step Label	Responsible HW/SW	Step description
BR-00-2	BRANCH	>>>>Steps-specific to MCU_ONLY.MCU_R5FF0FSS_Active Step-set-BR-00:END
BR-10-1	BRANCH	<<<< Steps-specific to MCU_ONLY.MCU_R5FF0FSS_OFF Step-set-BR-10:BEGIN
BR-10-2	BRANCH	>>>>Steps-specific to MCU_ONLY.MCU_R5FF0FSS_OFF Step-set-BR-10:END

Table 5-81 through Table 5-85 present the steps to enter LPM states, using the Step Label from Table 5-80.

**Table 5-81. ACTIVE to Standby Mode**

Step	Step Label
1	ACT-0-0
2	ACT-1-0.1
3	ACT-2-0
4	ACT-7-2
Device in STANDBY mode	

**Table 5-82. ACTIVE to CPD-OFF to DeepSleep**

Step	Step Label
1	ACT-0-0
2	ACT-1-0.2
3	ACT-2-0
4	ACT-6-1-1
5	ACT-6-2a
6	ACT-11-4a
7	ACT-11-01
8	ACT-10-1
9	ACT-10-2-3
10	ACT-10-4-1
11	ACT-10-4-2
12	ACT-25-00
13	ACT-25-01
14	ACT-25-02a
15	ACT-25-03
16	ACT-25-04
17	ACT-25-05
18	ACT-16-1
19	ACT-12-00
20	ACT-22-2
21	ACT-11-05
22	ACT-11-06
Device enters CPD-OFF mode	
23	ACT-14-01
24	ACT-12-00
25	ACT-22-2
26	ACT-11-05
27	ACT-11-06
28	ACT-25-10
29	ACT-25-11



**Table 5-82. ACTIVE to CPD-OFF to DeepSleep  
(continued)**

Step	Step Label
30	ACT-25-12
31	ACT-25-13
32	ACT-25-14
33	ACT-25-15
34	ACT-25-16
35	ACT-13-01
36	ACT-13-02
37	ACT-29-01
38	ACT-29-02
39	ACT-29-03
Device enters Deep-Sleep mode	

### Note

NOTICE that there are a few steps in [Table 5-82](#) ACTIVE to CPD-OFF part of the table which should be skip if the transition is from ACTIVE to DeepSleep. If the device is already in CPD-OFF and need to transition to DeepSleep, then the content of R5F can't be save to DDR and will have to be saved to FLASH. Alternatively the context of R5F will be lost and the MCU\_R5FF0FSS in next wakeup if it stays in CPD-OFF state the R5Fwill start from scratch without context restored.

**Table 5-83. ACTIVE to MCU-ONLY to OFF**

Step	Step Label
1	ACT-0-0
2	ACT-1-0
3	ACT-1-0.3
4	ACT-2-0
5	ACT-6-1-1
6	ACT-6-2a
7	ACT-11-01
8	ACT-10-1
9	ACT-10-2-3
10	ACT-10-4-1
11	ACT-10-4-2
12	ACT-25-00
13	ACT-25-01
14	ACT-25-02b
15	ACT-25-03
16	ACT-25-04
17	ACT-25-05
18	ACT-12-00
19	ACT-22-2
20	ACT-11-05
21	ACT-11-06
22	ACT-17-1
23	ACT-19-2

**Table 5-83. ACTIVE to MCU-ONLY to OFF  
(continued)**

Step	Step Label
24	ACT-19-1
25	ACT-22-2
26	ACT-17-2-2
27	ACT-16-1
28	ACT-16-2
29	ACT-15-1
30	ACT-29-02
31	ACT-28-01
Device is now in MCU-ONLY.R5FSS_ACTIVE mode	
32	ACT-14-02
33	ACT-25-10
34	ACT-25-11
35	ACT-25-12
36	ACT-25-13
37	ACT-25-14
37	ACT-25-15
39	ACT-25-16
40	ACT-13-01
41	ACT-13-02
42	ACT29-02
43	ACT29-03
Device is now in MCU-ONLY.R5FSS_OFF mode	
44	ACT21
Device is now in OFF mode	

**Table 5-84. ACTIVE to SuspendToRAM**

Step	Step Label
1	ACT-0-0
2	ACT-1-0.4
3	ACT-2-0
4	ACT-6-1-1
5	ACT-6-2a
6	ACT-11-4a
7	ACT-11-01
8	ACT-10-1
9	ACT-10-2-3
10	ACT-10-4-1
11	ACT-10-4-2
12	ACT-13-01
13	ACT-14-01
14	ACT-12-00
15	ACT-13-01
16	ACT-17-2
17	ACT-22-1
18	ACT-22-2

**Table 5-84. ACTIVE to SuspendToRAM (continued)**

Step	Step Label
19	ACT-22-3
SoC and PMIC are now in SuspendToRAM mode	

**Table 5-85. MCU-ONLY to SuspendToRAM**

Step	Step Label
1	ACT-1-0.4
2	ACT-6-4
3	ACT-17-2
4	ACT-22-1
5	ACT-22-2
6	ACT-22-3
SoC and PMIC are now in SuspendToRAM mode	

### 5.2.3.6.2 LPM Exit Sequences

Table 5-86 lists all available steps used in exiting LPM.

**Table 5-86. LPM Exit Step Labels**

Step Label	Responsible HW/SW	Step description
LPM_Ex-01	SoC	SoC and PMIC are in SuspendToRAM mode.
LPM_Ex-02	PMIC	PMIC detects an external wakeup event from SuspendToRAM, it can be any, but not limited to the events below: <ul style="list-style-type: none"> <li>PMIC RTC timer event</li> <li>GPI wakeup event</li> <li>CAN-RX pin activity</li> </ul>
LPM_Ex-03	PMIC	PMIC wakeup and enables power supply to full SoC.
LPM_Ex-03a	PMIC	PMIC wakeup and enables power supply to MCU domain and MCU-I/Os.
LPM_Ex-04.1a	SoC	WKUP/MCU goes through PORz reset and starts ROM boot sequence.
LPM_Ex-04.1b	SW(MCU_R5FF0Fx)	R5FSS boots to SBL.
LPM_Ex-04.1c	SW(MCU_R5FF0Fx)	SBL sends command to PMIC via WKUP_I2C0 to check SuspendToRAM flag to identify if this is COLD or SuspendToRAM-boot.
LPM_Ex-04.2	SoC	Main domain goes through PORz reset
LPM_Ex-05	SW(MCU_R5FF0Fx)	SBL detects device recovered from SuspendToRAM.
LPM_Ex-05.1a	SW(MCU_R5FF0Fx)	SBL reads from boot-strap boot-mode registers that this is MCU-ONLY boot.
LPM_Ex-05.1b	SW(MCU_R5FF0Fx)	SBL reads from boot-strap boot-mode registers that this is full-SoC boot.
LPM_Ex-05.2	SW(MCU_R5FF0Fx)	SBL does restore to MCU domain (no DDR available).
LPM_Ex-06	FW/SW(MCU_R5FF0Fx)	<ul style="list-style-type: none"> <li>Restore required context to main domain from DDR and RESUME.</li> <li>DMSC (or potentially R5F) sets to power-up all PDs and ENABLE all the LPSCs (or as required per usage case and per as seen per context saving information).</li> <li>Sequence power-up and LPSC enable as per dependency constrains (see <a href="#">Section 5.2.2.2.1.3.3, LPSC Dependences Overview</a>).</li> <li>Restore required context to MAIN domain from DDR and RESUME.</li> </ul>
LPM_Ex-07-0	Start MCU-Only R5FSS-OFF	<ul style="list-style-type: none"> <li>PD_MCU_R5FF0FSS = OFF</li> <li>LPSC_WKUP_GPIO is in DISABLE (not SwRstDis) state, such that it could generate wake-up events</li> <li>LPSC_WKUP_ALWAYSON and LPSC_DMSC are in ENABLE state</li> <li>All other LPSCs are in SwRstDis state</li> <li>DMSC in LPM4 or LPM3.</li> </ul>
LPM_Ex-07-1	Start MCU-Only	MCU is in functional mode, PLLCNTRL in clock bypass mode, MCU_R5FF0F in WFI/WFE, DMSC in LPM0(WFI).

**Table 5-86. LPM Exit Step Labels (continued)**

Step Label	Responsible HW/SW	Step description
LPM_Ex-07-2	IO	<p>One of the IOs enable to detect wakeup events is toggled from the board.</p> <ul style="list-style-type: none"> <li>- Event will propagate asynchronous through all of the IO-PM daisy chain (DCH) and eventually reaches DMSC.</li> <li>- The IO-PM wake-up event corresponding to each of the 2 IO-PM-DCH were set prior to enter LPM4/3.</li> <li>- The DMSC ASYNC-starts driving WKUP_HFOSC0 to enable state and PWR-ON</li> <li>- The DMSC wake-up FSM running of the 32-kHz RC oscillator starts the wake-up of DMSC.</li> <li>- DMSC eventually exits LPM4/3 and the interrupt map to the IO-PM-DCHs will cause M3 to exit WFI.</li> <li>- M3 executes corresponding wakeup ISR</li> </ul>
LPM_Ex-10	FW	DMSC puts main IO back to normal state. When peripherals are initialized, CTRLMMR_WKUP_DEEPSLEEP_CTRL override for main domain can be released to allow IO to be controlled by core logic again.
LPM_Ex-11	DMSC-FW	<ul style="list-style-type: none"> <li>-DMSC Powers-ON PD_MCU_R5FF0FSS and enables R5x as per PSC dependency constrains (see <a href="#">Section 5.2.2.1.3.3, LPSC Dependences Overview</a>).</li> <li>-MCU-R5x reboot</li> </ul>
LPM_Ex-12	MCU SW	MCU_R5FF0F SW decides to SoC transition to active mode by requesting to DMSC.
LPM_Ex-13	FW	DMSC powers up the power domains in the main domain to enter active mode.
LPM_Ex-14	FW/SW(MCU_R5FF0Fx)	<p>Wake-up events CASE-1:</p> <ul style="list-style-type: none"> <li>- CASE1: An interrupt defined for the purpose of wake-up from active peripherals, as per the list below, is detected in the MCU_R5FF0FSS.</li> </ul> <p>If MCU_R5FF0F is in WFI/WFE mode, it exits WFI/WFE mode and respond to interrupt. When MCU_R5FF0F is ACTIVE or WFI the interrupt can be from any of the peripherals or timer events in MCU map to MCU_R5FF0F:</p> <p>Examples - I2C, UART, ADC, SPI etc.</p>
LPM_Ex-14-2	DMSC-FW	<p>Wake-up events CASE-2 and 3: For the below wakeup cases only DMSC can detect the wake-up condition.</p> <ul style="list-style-type: none"> <li>- CASE2: When R5F is OFF or in CLKSTOP only interrupts map to DMSC could cause wakeup.</li> <li>- CASE3: Even if R5F is ON, when the wake-up event is map to be detected by the IO-PM daisy chain either in WKUP/MCU or MAIN, only DMSC can detect such event.</li> </ul>
LPM_Ex-15-2-1	FW/SW(MCU_R5FF0Fx)	<p>Wake-up events CASE-1:</p> <p>R5F ON and enabled and out of WFI/WFE executing ISR. The ISR will also ID if SoC_MAIN needs to be ON.</p>
LPM_Ex-15-2-2	DMSC-FW	<p>Wake-up events CASE-3:</p> <ul style="list-style-type: none"> <li>- R5F is ON, but wake-up event is only map as an interrupt and wake-up event for DMSC to detect it.</li> <li>- DMSC runs ISR which depending on the WKUP source might not require to turn ON SoC_MAIN.</li> <li>- If wakeup source is from WKUP-IO-PM-DCH then DMSC needs to check all CTRLMMR_WKUP_PADCONFIGi (i = 0 to 93) registers to figure out the peripheral whose IO caused the wkup event</li> <li>- If wakeup source is from MAIN-IO-PM-DCH then DMSC will need to power-ON SoC_MAIN.</li> </ul>
LPM_Ex-15-2-3	DMSC-FW	<p>Wake-up events CASE-2:</p> <ul style="list-style-type: none"> <li>- R5F is OFF, wake-up event is only map as an interrupt and wake-up event for DMSC to detect it.</li> <li>- DMSC runs ISR which depending on the WKUP source might not require to turn ON SoC_MAIN.</li> <li>- If wakeup source is from MAIN-IO-PM-DCH then DMSC will have to power-on SoC_MAIN.</li> </ul>
LPM_Ex-15-2-5	FW/SW(MCU_R5FF0Fx)	<p>If SoC_MAIN needs to be ON there are 2 cases:</p> <ul style="list-style-type: none"> <li>-If PMIC_PIN_CTRL: PMIC control is via pin MCU_R5FF0x core will send request message to DMSC to power-ON main.</li> <li>-Else If PMIC_I2C_CTRL: then MCU_R5FF0x will run the PMIC driver itself and message to DMSC once main is ON for the FW to stay in sync.</li> </ul>

**Table 5-86. LPM Exit Step Labels (continued)**

Step Label	Responsible HW/SW	Step description
LPM_Ex-15-2-6	DMSC-FW	<ul style="list-style-type: none"> <li>- Set MMR inside WKUP_PLLCTRL0 PLLCTL[0] PLEN = 1 and PLLCTL[5] PLENSRC = 1.</li> <li>- Switch the input clock for WKUP_PLLCTRL0 from WKUP_HFOSC0_CLKOUT to MCU_PLL0/HSDIV_CLKOUT.</li> <li>- By now WKUP_PLLCTRL0 should be out of bypass mode and running of MCU_PLL0.</li> <li>- If required as part of the context restore, turn-ON/Enable HFOSC1.</li> <li>- If required as part of the context restore, turn-ON/Enable MCU_PLL1 and MCU_PLL2.</li> </ul>
LPM_Ex-15-2-7	FW	DMSC (or potentially R5F) sets to ENABLE all the LPSCs (or as required per usage case) in the WKUP_PSC based on dependency constrains (see <a href="#">Section 5.2.2.2.1.3.3, LPSC Dependencies Overview</a> ). With the exception of the LPSCs listed below, which will be enable at a later step: <ul style="list-style-type: none"> <li>- LPSC_WKUPMCU2MAIN, LPSC_MAIN2WKUPMCU, LPSC_DEBUG2DMSC, LPSC_MCU_DEBUG.</li> </ul>
LPM_Ex-15-2-8	DMSC-FW	DMSC to remove wkup-IO-PM-DCH from isolation mode and move them to functional mode. When peripherals are initialized, CTRLMMR_WKUP_DEEPSLEEP_CTRL override for WKUP can be released to allow IO to be controlled by core logic again.
LPM_Ex-15-2-9	FW	<ul style="list-style-type: none"> <li>-Via WKUP_CTRL_MMR0 module, confirm removal of the reset-isolation blocking on the main_porz propagation: CTRLMMR_WKUP_POR_RST_CTRL[0] POR_RST_ISO_DONE_Z = 0.</li> <li>-Via WKUP_CTRL_MMR0 module apply PORz to SoC_MAIN: CTRLMMR_WKUP_POR_RST_CTRL[19-16] SW_MAIN_POR = 4'b0110 .</li> </ul>
LPM_Ex-16-1	FW	DMSC sets CTRLMMR_WKUP_MAIN_PWR_CTRL[0] PWR_EN to turn on the main voltage domain. This information is further transported through pin to PMIC to turn on the main voltage domain supply.
LPM_Ex-16-2	FW	<ul style="list-style-type: none"> <li>-Clear PGOOD status bit in POK modules related to SoC_MAIN core portion of the SoC via WKUP_CTRL_MMR0 module: CTRLMMR_WKUP_MAIN_PRG_STAT[31] POK_CLR = 1.</li> <li>POK_VDD_CORE_UV_CTRL, POK_VDD_CPU_UV_CTRL,</li> <li>POK_VDDR_CORE_UV_CTRL, POK_VMON_EXT_UV_CTRL</li> <li>POK_VDD_CORE_OV_CTRL, POK_VDD_CPU_OV_CTRL,</li> <li>POK_VDDR_CORE_OV_CTRL, POK_VMON_EXT_OV_CTRL</li> </ul>
LPM_Ex-16-3	FW	<ul style="list-style-type: none"> <li>-Poll until all POK UV and OV status bitts indicate PGOOD status for all POK modules related to SoC_MAIN core portion of the SoC via WKUP_CTRL_MMR0 module: CTRLMMR_WKUP_MAIN_PRG_STAT corresponding POK_*_OV/UV = 1.</li> <li>POK_VDD_CORE_UV_CTRL, POK_VDD_CPU_UV_CTRL,</li> <li>POK_VDDR_CORE_UV_CTRL, POK_VMON_EXT_UV_CTRL</li> <li>POK_VDD_CORE_OV_CTRL, POK_VDD_CPU_OV_CTRL,</li> <li>POK_VDDR_CORE_OV_CTRL, POK_VMON_EXT_OV_CTRL</li> </ul>
LPM_Ex-16-4	FW	<ul style="list-style-type: none"> <li>-Restore the ESM and RESET</li> <li>-Gating values as they were prior to enter MCU_ONLY* mode for POK modules related to SoC_MAIN core portion of the SoC via WKUP_CTRL_MMR0 module.</li> <li>POK_VDD_CORE_UV_CTRL, POK_VDD_CPU_UV_CTRL,</li> <li>POK_VDDR_CORE_UV_CTRL, POK_VMON_EXT_UV_CTRL</li> <li>POK_VDD_CORE_OV_CTRL, POK_VDD_CPU_OV_CTRL,</li> <li>POK_VDDR_CORE_OV_CTRL, POK_VMON_EXT_OV_CTRL</li> <li>- First restore their contribution to ESM events.</li> <li>- Next restore their contribution to PRG reset,</li> <li>CTRLMMR_WKUP_MAIN_PRG_CTRL corresponding POK_*_GATERST_EN = 0.</li> </ul>
LPM_Ex-17-1	FW	Clear CTRLMMR_WKUP_MAIN_VDOM_CTRL[0] MAIN_VD_OFF = 0 in WKUP_CTRL_MMR0 module prior to powering off the main voltage domain to remove proper signal isolation.
LPM_Ex-17-2	SoC	Keep polling WKUP_CTRL_MMR0 register module bit RST_STAT[0] until it is '1' to make sure maing has gone through cold reset.
LPM_Ex-17-3	SoC	After main domain is out of reset and to enable the communication from main domain to WKUP/MCU domain, DMSC must enable LPSC_WKUPMCU2MAIN, LPSC_MAIN2WKUPMCU, LPSC_DEBUG2DMSC, LPSC_MCU_DEBUG as per dependency constrains (see <a href="#">Section 5.2.2.2.1.3.3, LPSC Dependencies Overview</a> ).

**Table 5-86. LPM Exit Step Labels (continued)**

Step Label	Responsible HW/SW	Step description
LPM_Ex-17.5	FW	<ul style="list-style-type: none"> <li>- Enable LPSC_EMIF_CFG0 as per PSC dependency constrains (see <a href="#">Section 5.2.2.2.1.3.3, LPSC Dependences Overview</a>).</li> <li>- Re-configure DDR EMIF.</li> <li>- Send command to PMIC via WKUP_I2C0 to drive to 0 the board DDR_RET chip input.</li> <li>- Enable LPSC_EMIF_DATA0 as per PSC dependency constrains (see <a href="#">Section 5.2.2.2.1.3.3, LPSC Dependences Overview</a>).</li> <li>- Do a DDR read to wake-up from self-refresh.</li> </ul>
LPM_Ex-18-01	FW	Restore CTRLMMR_MAIN_DEVSTAT from DMSC RAM. NOTE: CTRLMMR_MAIN_BOOTCFG is not restorable.
LPM_Ex-18-03a	DMSC-FW	Ensure all the PLLs and HSDIVs used in MAIN prior to entering MCU_ONLY mode are running.
LPM_Ex-18-03b	DMSC-FW	<ul style="list-style-type: none"> <li>- Restore the contents of all CTRLMMR_PADCONFIGi (i = 0 to 172) context from DDR.</li> <li>- Check in DMSC-RAM, the look-up table that show which CTRLMMR_PADCONFIGi (i = 0 to 172) were enable to detect wake-up event prior to enter MCU-ONLY mode.</li> <li>- DMSC (or potentially R5F) sets to power-up all PDs and ENABLE all the LPSCs (or as required per usage case and per as seen per context saving information).</li> <li>- Sequence power-up and LPSC enable as per dependency constrains (see <a href="#">Section 5.2.2.2.1.3.3, LPSC Dependences Overview</a>).</li> <li>- Restore required context to MAIN domain from DDR and RESUME.</li> </ul>
LPM_Ex-18-04	DMSC-FW	Check in DMSC-RAM, the look-up table and power-up the PDs and ENABLE LPSCs associated with all the potential wake-up events. Do this as per dependency constrains (see <a href="#">Section 5.2.2.2.1.3.3, LPSC Dependences Overview</a> ).
LPM_Ex-18-06	FW	<ul style="list-style-type: none"> <li>-Remove self-refresh from DDR.</li> <li>- Send read command via DDR to board SDRAM for DDR-self-refresh to be removed.</li> </ul>
LPM_Ex-18-07	FW	-DMSC to remove MAIN-IO-PM-DCH from isolation mode and move them to functional mode.
LPM_Ex-24	FW	The SoC can awake from Standby by any interrupt routed to A72 through GIC
LpEx300	SoC	
LpEx301	Ext-Dbg	Debugger is attached externally to the SoC JTAG pins.
LpEx302	SoC	<ul style="list-style-type: none"> <li>a) The SoC Icemelter detects the activity in TCK/TRST pins in the wake-up domain.</li> <li>b) Icemelter will drive "Debug Activated" condition into DMSC, PSC.</li> <li>c) DMSC drives pin PMIC_POWER_EN1 = 1. Resulting in PMIC driving all SOC_MAIN supplies to ON.</li> <li>d) PSC starts the sequence to ENABLE DEBUGSS.</li> <li>e) "Debug Activated" also maps as Cortex.INT[24] in DMSC.</li> </ul>
LpEx303	Ext-Dbg	<ul style="list-style-type: none"> <li>- DMSC takes Cortex-INT[24] and executes the LP-MCU-Only-DBG ISR.</li> <li>- As part of the LP-MCU-Only-DBG ISR all the steps listed after this are performed.</li> </ul>
LpEx304	-	<ul style="list-style-type: none"> <li>- By now the PSC should have completed the DEBUGSS ENABLE process.</li> <li>- DAP accesses should be possible now.</li> </ul>

Table 5-87 through Table 5-92 present the sequences to exit the device LPM using the Step Label from Table 5-86.

**Table 5-87. SuspendToRAM to ACTIVE**

Step	Step Label
1	LPM_Ex-01
2	LPM_Ex-02
3	LPM_Ex-03
4	LPM_Ex-04.1a
5	LPM_Ex-04.2
7	LPM_Ex-04.1c

**Table 5-87. SuspendToRAM to ACTIVE (continued)**

Step	Step Label
8	LPM_Ex-05
9	LPM_Ex-05.1b
10	LPM_Ex-17-2
11	LPM_Ex-17-3
12	LPM_Ex-17.5
13	LPM_Ex-06
Device is now in ACTIVE mode	

**Table 5-88. SuspendToRAM to MCU-ONLY**

Step	Step Label
1	LPM_Ex-01
2	LPM_Ex-02
3	LPM_Ex-03a
4	LPM_Ex-04.1a
5	LPM_Ex-04.1b
6	LPM_Ex-04.1c
7	LPM_Ex-05
8	LPM_Ex-05.1a
9	LPM_Ex-05.2
Device is now in MCU-ONLY.R5FSS_ACTIVE mode	

**Table 5-89. MCU-ONLY to ACTIVE**

Step	Step Label
1	LPM_Ex-07-1
2	LPM_Ex-14
3	LPM_Ex-14-2
4	LPM_Ex-15-2-1
5	LPM_Ex-15-2-2
6	LPM_Ex-15-2-6
7	LPM_Ex-15-2-7
8	LPM_Ex-15-2-8
9	LPM_Ex-15-2-9
10	LPM_Ex-15-2-5
11	LPM_Ex-16-2
12	LPM_Ex-16-3
13	LPM_Ex-16-4
14	LPM_Ex-17-1
15	LPM_Ex-17-2
16	LPM_Ex-17-3
17	LPM_Ex-18-01
18	LPM_Ex-18-03a
19	LPM_Ex-17.5
20	LPM_Ex-18-03b
21	LPM_Ex-18-07
22	LPM_Ex-06



**Table 5-89. MCU-ONLY to ACTIVE (continued)**

Step	Step Label
	The device is now in ACTIVE mode

**Table 5-90. DeepSleep to CPD-OFF to ACTIVE**

Step	Step Label
1	LPM_Ex-07-2
2	LPM_Ex-15-2-6
3	LPM_Ex-15-2-8
4	LPM_Ex-11
The device is now in CDP-OFF mode	
5	LPM_Ex-12
6	LPM_Ex-13
7	LPM_Ex-17.5
8	LPM_Ex-18-07
9	LPM_Ex-10
10	LPM_Ex-06
The device is now in ACTIVE mode	

**Table 5-91. Standby to ACTIVE**

Step	Step Label
1	LPM_Ex-24
The device is now in ACTIVE mode	

**Table 5-92. MCU-ONLY-MAIN-OFF to MCU-ONLY-DEBUG**

Step	Step Label
1	LpEx00.2
2	LpEx301
3	LpEx302
4	LpEx303
5	LpEx-16-2
6	LpEx-16-3
7	LpEx-15-2-9
8	LpEx-17-1
9	LpEx-17-2
10	LpEx-17-3
11	LpEx304

### 5.2.3.6.3 IO Retention

IO retention is supported on MCU\_GENERAL and CANUART IOs, MCU Scratch Memory, SerDes PHY, and DDR PHY.

- The control of the retention of LVCMOS IOs is performed via the corresponding PADCONFIG register listed in *Control Module (CTRL\_MMR)*. For more information about the exact PADCONFIG register see Pin Multiplexing Table in the device-specific Datasheet.
- MCU-Scratch-Memory - TBD
- The control of the retention of SERDES PHY is performed via CTRLMMR\_SERDESx\_CTRL[8] RET\_EN, x = 0 to 4. For more information about control registers, see *Control Module (CTRL\_MMR)*.
- The control of DDRSS retention is controlled via external DDR\_RET pin.

## 5.2.4 Dynamic Power Management

### 5.2.4.1 AVS Support

The device supports only Adaptive Voltage Scaling (AVS) Class 0. This is a procedure for lowering the voltage on certain device power rails. AVS-Class0 attempts to normalize the power consumption across all devices. The optimal voltage for each AVS supported rail of each device is determined after analysis in the factory, based on the strength of the device during manufacturing. This value is written in the device eFuse where it can be read through dedicated registers.

The following supplies support AVS-Class0:

- VDD\_CC

The AVS voltage for a supply is an E-fuse VID value located in the corresponding WKUP\_VTM\_VD\_OPPVID\_j register in the VTM module. [Table 5-56](#) lists the mapping of the supplies to the VTM control registers. Separate VID values are fused for the different OPPs that the supply supports. For more information about OPPs supported by a supply in the device, see the device-specific DataSheet.

The AVS functionality can be enabled and performed by software after the primary boot of the device is finished. Typically, the voltage adjustment of AVS operation is done as secondary boot software.

### 5.2.4.2 Dynamic Frequency Scaling (DFS) Operations

DFS is supported for all major cores (no change in voltage) such as MPU/GPU/C7x/C66x DSP.

Processor cores that have dedicated voltage supply and asynchronous operating frequency from the main SoC support:

- Independent AVS voltage setting based on Vmin of cores on the domain (Compute Cluster / VDD\_CC components only).
- Dynamic scaling of processor operating frequency without change of operating voltages.

**Table 5-93. Device Family OPP**

Voltage Domain	Module	OPP_NOM Frequency, MHz
VDD_CC (Fixed/AVS/DFS)	MPU0	2000
	C71SS0	1000-1200
	MSMC0	
VDD_CORE(Fixed)	GPU0	750-800
	C66x_DSPSS0,1	1200-1500
	VXE384MP2 (VENC)	660-687
	D5520MPx (DEC)	550
	VPAC	720
	DMPAC	520
	CBASS0	500/250
	MCU_1/2	1000
	Rest of chip	various
	DDRSS	4266
	MCU_0	1000333
VDD_MCU(Fixed)	WKUP_DMSC0 - M3	333

**Table 5-94. Valid OPP Combination**

Voltage Domain	Power State		
	OFF	MCU ONLY	Active/Deep/CPD-OFF
VDD_CC	OFF	OFF	OFF
VDD_CORE	OFF	OFF	OPP-NOM

**Table 5-94. Valid OPP Combination (continued)**

Voltage Domain	Power State		
	OFF	MCU ONLY	Active/Deep/CPD-OFF
VDD_MCU	OFF	OPP-NOM Fixed	OPP-NOM Fixed

### 5.2.5 Thermal Management

The modules that perform thermal management are:

- On-die temperature Monitor with Integrated multiple sensors
- One thermal diode
- VTM
- DMSC firmware.

For information about VTM, see [Voltage and Thermal Manager \(VTM\)](#).

For more information how to use DMSC, see the [TISCI API](#).

## 5.3 Reset

This chapter describes the device reset management.

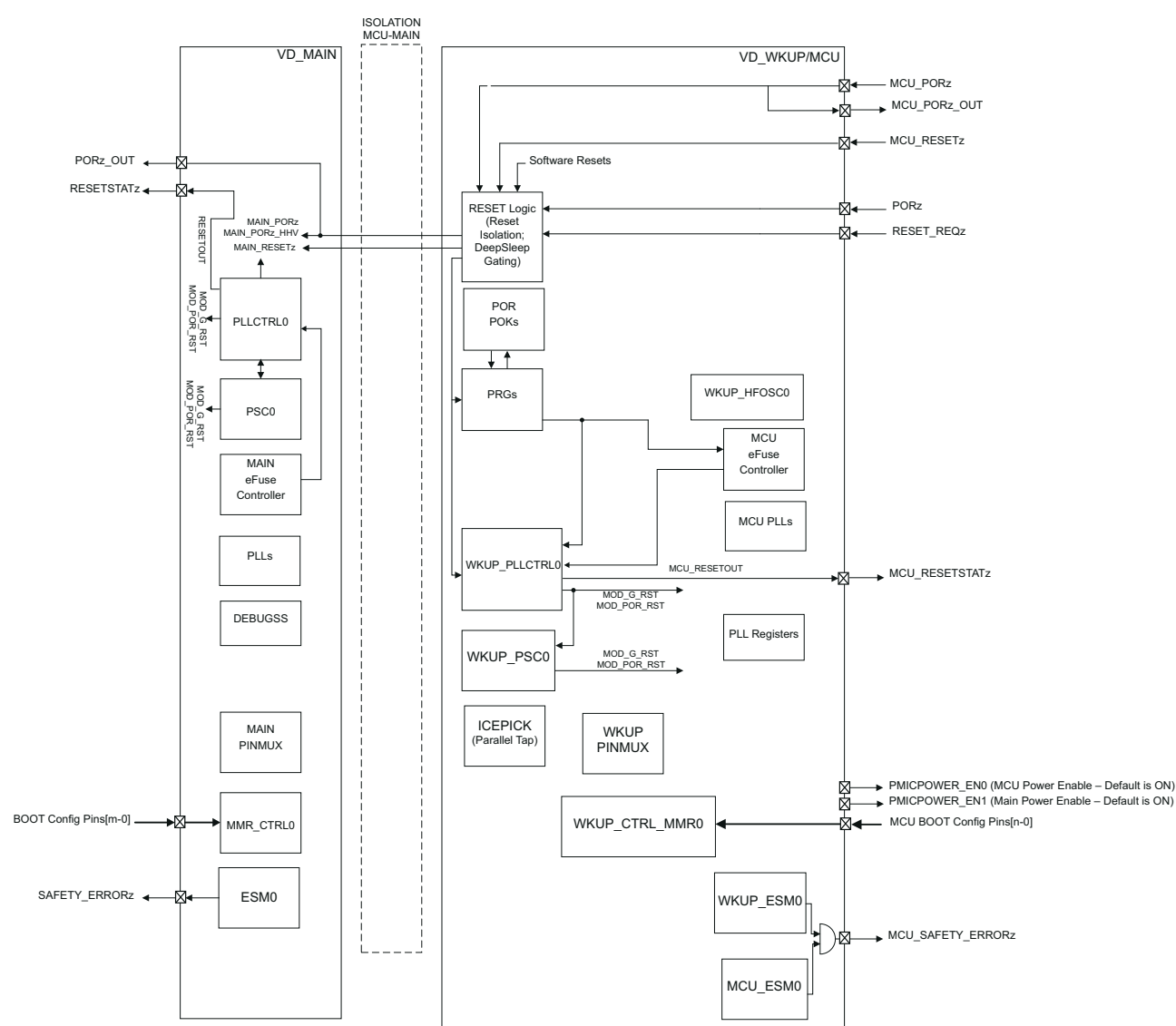
### Terminology

- DMSC - Device Management Security Controller
- PSC - Power Sleep Controller
- LPSC - Local Power Sleep Controller
- ESM - Error Signaling Module
- Isolation – the module is exempted from reset

#### 5.3.1 Reset Overview

Resets bring the device or part of the device to a known state after events such as power-up or hardware or software requests.

Figure 5-28 shows an overview of the reset architecture.



**Figure 5-28. Reset Architecture**

### Note

For power modules used in the reset see:

- For internal PORz detection, see *Power on Reset (POR) Module*, in *Power*.
- For Power Reset Generator (PRG), see *Power Reset Generator (PRG) Modules*, in *Power*.
- For POK, see *POK Modules*, in *Power*.
- For Power Glitch Detector (PGD), see *Power Glitch Detector (PGD) Modules*, in *Power*.

### Note

For information about Boot modes and Boot configuration pins, see *Initialization*.

### Note

MOD\_POR\_RST[n] and MOD\_G\_RST[n] are connected to certain modules. For more information which is the number of the LPSC of a certain module, see its specific chapter.

## 5.3.2 Reset Sources

Table 5-95 summarizes the resets supported by device.

**Table 5-95. Reset Supported in the Device**

Reset Source	Type	Sequence (Effect)
MCU_PORz	Input (control) pin Master Power-on-Reset input (active low) to the entire device. Effects all domains – WKUP, MCU, and MAIN No isolation possible.	See <a href="#">Section 5.3.6.2, MCU_PORz Sequence</a> .
MCU_RESEZz	Input (control) pin Master Warm Reset input (active low) to the entire device. Effects all domains – WKUP, MCU, and MAIN Modules configured for isolation in the MAIN domain are isolated although the WKUP/MCU domain is not isolated from the MAIN domain.	See <a href="#">Section 5.3.6.3, MCU_RESEZz Sequence</a> .
PORz	Input (control) pin Power-on-Reset input (active low) to the MAIN domain. Effects MAIN domain WKUP/MCU should be isolated from the MAIN domain	See <a href="#">Section 5.3.6.4, PORz Sequence</a> .
RESET_REQz	Input (control) pin Warm Reset Request input (active low) to the MAIN domain Effects MAIN domain WKUP/MCU should be isolated from the MAIN domain. Additionally, modules within the MAIN domain may be isolated.	See <a href="#">Section 5.3.6.5, RESET_REQz Sequence</a> .
SW_MCU_WARMRST	Software reset: equivalent to MCU_RESEZz pin CTRLMMR_WKUP_MCU_WARM_RST_CTRL <sup>(1)</sup>	See <a href="#">Section 5.3.6.3, MCU_RESEZz Sequence</a> .
SW_MAIN_POR	Software reset: equivalent to PORz pin CTRLMMR_WKUP_POR_RST_CTRL <sup>(1)</sup>	See <a href="#">Section 5.3.6.4, PORz Sequence</a> .
SW_MAIN_WARMRST	Software reset: equivalent to RESET_REQz pin CTRLMMR_WKUP_MAIN_WARM_RST_CTRL <sup>(1)</sup>	See <a href="#">Section 5.3.6.5, RESET_REQz Sequence</a> .
WKUP_DMSC0 COLD RST	Internal event: equivalent to MCU_RESEZz	See <a href="#">Section 5.3.6.3, MCU_RESEZz Sequence</a> .
WKUP_DMSC0 WARM RST	Internal event: equivalent to MCU_RESEZz	See <a href="#">Section 5.3.6.3, MCU_RESEZz Sequence</a> .

**Table 5-95. Reset Supported in the Device (continued)**

Reset Source	Type	Sequence (Effect)
Thermal Over-temperature (VTM)	Internal event: equivalent to MCU_RESETz (except that PLL behavior is different)	See <a href="#">Section 5.3.6.3, MCU_RESETz Sequence</a> .
Debug Reset		
Local Resets	Local reset is a reset that is applied to any given module in a module domain or to a CPU and/or its megamodule (any tightly coupled controllers delivered with the CPU) and their associated local watchdog timers.	

(1) These registers are part of the WKUP CTRL\_MMR domain, see *Control Module (CTRL\_MMR)*.

### 5.3.3 Reset Status

[Table 5-96](#) summarizes the reset status of the device.

**Table 5-96. Reset Status**

Reset Status	Type/Description
MCU_PORz_OUT	Output pin Propagated from the MCU_PORz input
MCU_RESETSTATz	Output pin Indicates release of internal reset on the WKUP/MCU domain Released synchronous with start of WKUP_DMSC0 ROM code execution.
PORz_OUT	Output pin Indicates PORz state in MAIN domain (delayed from PORz)
RESETSTATz	Output pin Indicates release of internal reset on the MAIN domain
THERMAL_RST	Software bit in CTRLMMR_WKUP_RESET_SRC_STAT <sup>(1)</sup>
DEBUGSS_RST	
COLD_OUT_RST	
WARM_OUT_RST	
PORZ_PIN	
RESET_REQZ_PIN	
MCU_RSTZ_PIN	
SW_MAIN_POR	
SW_MAIN_WARMRST	
SW_MCU_WARMRST	
MAIN_RST_DONE	Software bit in
MCU_RST_DONE	CTRLMMR_WKUP_RST_STAT <sup>(1)</sup>

(1) These registers are part of the WKUP CTRL\_MMR domain, see *Control Module (CTRL\_MMR)*.

### 5.3.4 Reset Control

[Table 5-97](#) lists the reset control registers.

**Table 5-97. Reset Control**

Reset Control	Control Register <sup>(1)</sup>
WKUP_DMSC0 Timeout control	CTRLMMR_WKUP_MAIN_POR_TO_CTRL (0 – 500 $\mu$ s in 100 $\mu$ s increments)
POR_RST_ISO_DONE_Z	CTRLMMR_WKUP_POR_RST_CTRL
SOC_WARMRST_ISO_DONE_Z	CTRLMMR_WKUP_MAIN_WARM_RST_CTRL

(1) These registers are part of the WKUP CTRL\_MMR domain, see *Control Module (CTRL\_MMR)*.



### 5.3.5 BOOTMODE Pins

The MCU\_BOOTMODE pins and BOOTMODE direct the MCU\_R5FSS boot (first stage boot). MCU\_BOOTMODE pins are latched based upon the rising edge of MCU\_PORz. BOOTMODE pins are latched based upon the rising edge of PORz\_OUT.

For more details on Bootmode pins, see *Initialization*.

### 5.3.6 Reset Sequences

#### 5.3.6.1 MCU\_PORz Overview

On MCU\_PORz,

- the WKUP\_DMSC0 is powered and clocked;
- the MCU\_R5FSS0 is powered but not clocked;
- all processors (except PRU\_ICSSG) on the MAIN side are powered down (and clock-gated):
  - C71x cores
  - C66x cores
  - Dual A72 MPU Sub-systems
  - GPU
  - R5FSS
  - HD Video Encoder/Decoder
  - Vision Pre-processing accelerator
  - Depth and Motion perception accelerator
- PRU\_ICSSG is in an Always On domain but is clock gated.

The general behavior for coming out of an MCU\_PORz reset is:

- Hardware initialization

**ROM Boot Loader (or ROM Code):** the purpose of the ROM Boot Loader is to load and check integrity of the Secondary Boot Loader (or application image).

- WKUP\_DMSC0 Code executes, setting up MCU\_R5FSS0 clocks and configuring security. WKUP\_DMSC0 code passes Boot info to MCU\_R5FSS0 and releases the MCU\_R5FSS0 clocks.
- MCU\_R5FSS0 ROM reads MCU\_BOOTMODE pins and if the boot is not specified as MCU-Only, the MCU\_R5FSS0 ROM also reads the BOOTMODE pins. The Bootmode pins define the peripherals involved in the boot and the associated media (see ROM Code boot modes). The media holds the secondary boot loader (SBL) or the entire application image. The WKUP\_DMSC0 firmware needs to be incorporated in this code. For MCU\_R5FSS ROM functional description, refer to *Initialization*.
- Once the Secondary Boot Loader has been verified, the WKUP\_DMSC0 issues a clock stop to the MCU\_R5FSS0 and resets the MCU\_R5FSS0.

**Secondary Boot Loader:** the secondary boot loader configures the device for application.

The previous description gives the high-level behavior for the MCU\_PORz reset. Other resets are almost a subset of this procedure. [Section 5.3.6.2](#), [Section 5.3.6.3](#), [Section 5.3.6.4](#), and [Section 5.3.6.5](#) explain this sequence in more detail.

#### 5.3.6.2 MCU\_PORz Sequence

MCU\_PORz and PORz must be held low until:

- All supplies have ramped to proper levels
- WKUP\_HFOSC0 has a stable amplitude that propagates clocks into the core

If the supplies and clock are already stable, MCU\_PORz and/or PORz must be held active (low) for a minimum of 1.2  $\mu$ s.

1. When MCU\_PORz is asserted low, WKUP\_CTRL\_MMR0 and MCU\_CTRL\_MMR0 registers are reset; a bit within the WKUP\_CTRL\_MMR0 registers is reset which allows PORz to propagate to the MAIN domain without delay. In the MAIN domain, CTRL\_MMR0 registers are reset.
2. The MCU\_BOOTMODE pins are latched on the rising edge of MCU\_PORz. The BOOTMODE pins are latched on the rising edge of PORz.
3. The release of MCU\_PORz begins a read of fuse values to register files; several different chains of e-fuse values are registered in parallel read sequences. Values are consumed by different blocks at various points in this sequence. In parallel and based upon the release of PORz, the MAIN domain e-fuse controller initiates a read of its e-fuse values.

4. LBIST/PBIST checks of the WKUP\_DMSC0 and MCU\_R5FSS0 cores and memories are run in the WKUP/MCU domain; in the MAIN domain, the BIST engines are controlled by the application software.
5. WKUP\_PSC0 is initialized (and PSC0 in the MAIN domain is initialized); concretely, the PSC modules (both WKUP\_PSC0 and PSC0) configure default power states and LPSC (clocking) states.
6. After power domain and clock domain initialization is completed, PSC modules release resets to the device (MOD\_POR\_RST, MOD\_G\_RST).
7. WKUP\_DMSC0 is removed from reset; WKUP\_DMSC0 ROM code execution begins. At this time MCU\_RESETSTATz is de-asserted.
8. WKUP\_DMSC0 executes WKUP\_DMSC0 ROM.
  - a. Configures MCU\_PLL0, MCU\_PLL0\_HSDIV1, and WKUP\_PLLCTRL0 (proper clock frequency for MCU\_R5FSS0).
  - b. Configures firewalls and message manager.
  - c. Uses the message manager to pass Boot info to MCU\_R5FSS0 ROM.
9. Based upon MCU\_BOOTMODE[06],
  - a. if MCU\_BOOTMODE[06] = 0, the user has requested a normal boot. Wait for MAIN to be released from reset (duration limited by WKUP\_DMSC0 timeout).
  - b. if MCU\_BOOTMODE[06] = 1, MCU-Only boot is requested. There is no timeout nor dependency upon MAIN release from power on reset.
10. MCU\_R5FSS0\_CORE0 is released from reset and begins to execute ROM
  - a. Based upon the value of MCU\_BOOTMODE pins (and MAIN domain BOOTMODE pins if MCU\_BOOTMODE[06] = 0), the R5F ROM code configures peripherals and PLLs to enable loading the external code.
  - b. Secondary boot-loader is loaded from external memory
  - c. MCU ROM requests WKUP\_DMSC0 for loaded code authentication
  - d. WKUP\_DMSC0 stops clocks to MCU\_R5FSS0
  - e. WKUP\_DMSC0 issues an MCU\_R5FSS0 reset
  - f. MCU\_R5FSS0\_CORE0 begins executing secondary boot-loader.

At the end of this sequence, MCU\_R5FSS0 is executing the secondary bootloader. The control of the device now passes to the customer code.

---

#### Note

On warm resets (except those caused by a VTM Thermal over-temperature), the output of various PLLs are either bypassed or left unaffected. For more details, see [Section 5.3.7, PLL Behavior on Reset](#).

---



---

#### Note

Reset Isolated domains are isolated during a MCU\_RESETz (or any warm reset) event.

---

### 5.3.6.3 MCU\_RESETz Sequence

MCU\_RESETz release requires:

- All supplies have ramped to proper levels
- WKUP\_HFOSC0 has a stable amplitude that propagates clocks into the core

MCU\_RESETz must be held active (low) for a minimum of 1.2  $\mu$ s.

When MCU\_RESETz is asserted low, it is also immediately passed to the MAIN domain (similar to MCU\_PORz asserting PORz).

1. Much of the device is reset. Blocks that are not reset:
  - a. Internal 12.5 MHz oscillator
  - b. External HF oscillator
  - c. External LF oscillator

- d. Power monitoring circuitry (Bandgap, POR, POK, PGD, PRG)
- e. I/O cells and I/O pad configuration registers
- f. SERDES
- g. efuse Controller/storage
- h. PLLs
- i. Voltage Temperature Module
- j. Global Timebase Counter (GTC)
- k. Boot Mode Latches
- l. Debug Components
- m. PSRAM

Another group of modules is partially reset:

- a. PSC
  - b. WKUP\_DMSC0
  - c. Debug
  - d. STM
  - e. MCU\_CPSW0 (CPSW2G)
  - f. CPSW0 (CPSW9G)
  - g. Probe
2. When MCU\_RESETz is released, PSC modules release MOD\_G\_RST to the device.
  3. WKUP\_DMSC0 is removed from reset; WKUP\_DMSC0 ROM code execution begins. At this time MCU\_RESETSTATz is de-asserted.
  4. WKUP\_DMSC0 executes ROM.
    - a. Configures MCU PLL0, MCU PLL0 HSDIV1, and WKUP PLLCTRL0 (proper clock frequency for MCU\_R5FSS0).
    - b. Configures firewalls and message manager.
    - c. Uses the message manager to pass Boot info to MCU
  5. Based upon MCU\_BOOTMODE[06],
    - a. if MCU\_BOOTMODE[06] = 0, the user has requested a normal boot. Wait for MAIN to be released from reset (duration limited by WKUP\_DMSC0 timeout).
    - b. if MCU\_BOOTMODE[06] = 1, MCU-Only boot is requested. There is no timeout nor dependency upon MAIN release from power on reset.
  6. MCU\_R5FSS0\_CORE0 is released from reset and begins to execute ROM
    - a. Based upon the value of MCU\_BOOTMODE pins (and MAIN domain BOOTMODE pins if MCU\_BOOTMODE[06] = 0), the MCU ROM code configures peripherals and PLLs to enable loading the external code.
    - b. Secondary boot-loader is loaded from external memory
    - c. MCU ROM requests WKUP\_DMSC0 for loaded code authentication
    - d. WKUP\_DMSC0 stops clocks to MCU\_R5FSS0
    - e. WKUP\_DMSC0 issues MCU reset
    - f. MCU\_R5FSS0\_CORE0 begins executing secondary boot-loader.

At the end of this sequence, MCU\_R5FSS0 is executing the secondary bootloader. The control of the device now passes to the customer code.

#### 5.3.6.4 PORz Sequence

PORz release requires:

- All supplies have ramped to proper levels
- WKUP\_HFOSC0 has a stable amplitude that propagates clocks into the core

PORz must be held active (low) for a minimum of 1.2  $\mu$ s.

If PORz is asserted with MCU\_PORz, then PORz propagates to the MAIN domain immediately.

If PORz is asserted independent of MCU\_PORz, then the user code must have setup:

- CTRLMMR\_WKUP\_POR\_RST\_CTRL[0] POR\_RST\_ISO\_DONE\_Z (high) in order to block the PORz from immediate propagation and
- CTRLMMR\_WKUP\_MAIN\_POR\_TO\_CTRL[2:0] TIMEOUT\_PER in order to configure the timeout, after which PORz is unconditionally propagated to MAIN domain.

PORz sequence is:

1. When PORz is asserted low, hardware in the WKUP/MCU domain recognizes the PORz and generates an interrupt to WKUP\_DMSC0 and MCU. Reset hardware latches the PORz low level.
2. User software is responsible for isolating WKUP/MCU domain from the MAIN domain. The details of the isolation are application specific and are carried out in software. Connections between the WKUP/MCU domain and the MAIN domain include:
  - LPSC\_WKUPMCU2MAIN - LPSC4 (WKUP\_PSC0)
  - LPSC\_MAIN2WKUPMCU - LPSC5 (WKUP\_PSC0)
  - Communications channel: MCSPI3 is connected as a master to MCU\_MCSPI1; MCSPI4 is directly connected as a slave to MCU\_MCSPI2.
  - WKUP\_GPIOMUX\_INTRTR0
  - MAIN2MCU\_LVL\_INTRTR0
  - MAIN2MCU\_PLS\_INTRTR0
  - WKUP\_ESM0
  - MCU\_ESM0

When the isolation is complete, MCU\_R5FSS0 or WKUP\_DMSC0 must write CTRLMMR\_WKUP\_POR\_RST\_CTRL[0] POR\_RST\_ISO\_DONE\_Z low to allow the PORz signal to propagate.

3. The BOOTMODE pins are latched on the rising edge of PORz.
4. The release of PORz begins a read of eFuse values to register files; several different chains of e-fuse values are registered in parallel read sequences. Values are consumed by different blocks at various points in this sequence.
5. PSC0 is initialized; specifically, PSC0 module configures default power states and LPSC (clocking) states.
6. After power domain / clock domain initialization is completed, PSC modules release resets to the device (MOD\_POR\_RST, MOD\_G\_RST).
7. CTRLMMR\_WKUP\_RST\_STAT[0] MAIN\_RST\_DONE is asserted.

At the end of this sequence, MCU\_R5FSS can read the CTRLMMR\_WKUP\_RST\_STAT[0] MAIN\_RST\_DONE bit and begin re-configuring the MAIN domain.

### 5.3.6.5 RESET\_REQz Sequence

RESET\_REQz release requires:

- All supplies have ramped to proper levels
- WKUP\_HFOSC0 has a stable amplitude that propagates clocks into the core

RESET\_REQz must be held active (low) for a minimum of 1.2  $\mu$ s.

Like the PORz sequence, it is necessary to have configured CTRLMMR\_WKUP\_MAIN\_WARM\_RST\_CTRL[0] SOC\_WARMRST\_ISO\_DONE\_Z high in order to block the reset request from immediately propagating. Unlike the PORz sequence, there is no timeout. Also, unlike the PORz sequence, some clock domains within given power domains can be configured to ignore the RESET\_REQz signal.

1. When RESET\_REQz is asserted low, hardware in the WKUP/MCU domain recognizes the RESET\_REQz and generates an interrupt to WKUP\_DMSC0 and MCU\_R5FSS0. Reset hardware latches the RESET\_REQz low level.
2. User software is responsible for isolating WKUP/MCU domain from the MAIN domain. The details of the isolation are application specific and are carried out in software. Connections between the WKUP/MCU domain and the MAIN domain include:
  - LPSC\_WKUPMCU2MAIN - LPSC4 (WKUP\_PSC0)
  - LPSC\_MAIN2WKUPMCU - LPSC5 (WKUP\_PSC0)

- Communications channel: MCSPI3 is connected as a master to MCU\_MCSP11; MCSPI4 is directly connected as a slave to MCU\_MCSP12.
- WKUP\_GPIOMUX\_INTRTR0
- MAIN2MCU\_LVL\_INTRTR0
- MAIN2MCU\_PLS\_INTRTR0
- WKUP\_ESM0
- MCU\_ESM0

When the isolation is complete, MCU\_R5FSS0 or WKUP\_DMSC0 must write CTRLMMR\_WKUP\_MAIN\_WARM\_RST\_CTRL[0] SOC\_WARMRST\_ISO\_DONE\_Z low to allow the RESET\_REQz signal to propagate.

3. Power domains are not reset by RESET\_REQz if-and-only-if a LPSC in the power domain is configured to be reset isolated. Otherwise, power domains and clock domains are reset by RESET\_REQz.

#### Note

As an example of how to read tables in the PSC: Device Power-Management Layout Section, in *Power*, PSC0 Power Management Device-Level Layout shows that PD\_R5FSS\_0 is assigned Power Domain Index 24 with LPSCs 93, 94, and 95 controlling clocks/resets to circuitry within this power domain. PSC0 Power Domain Features shows that power domain 24 is OFF by default (if isolation does not block the reset). PSC0 LPSC Features shows that LPSC 93, 94, and 95 are OFF by default *but* LPSC 93 and 94 can be configured for reset isolation.

If Reset isolation is selected for LPSC 93 and 94, resets and clocks are not affected by the RESET\_REQz signal. Power Domain 24 cannot be reset when the LPSCs are active. Note that LPSC 95 **is** reset.

If Reset isolation is not selected for either LPSC 93 or 94, resets propagate to LPSC 93, 94, and 95; the clocks are stopped to these LPSCs (since their default state is OFF). Power Domain 24 is reset to its OFF state.

4. After power domain / clock domain initialization is completed, PSC modules release resets to the non-isolated parts of the device (MOD\_POR\_RST, MOD\_G\_RST).
5. CTRLMMR\_WKUP\_RST\_STAT[0] MAIN\_RST\_DONE is asserted.

At the end of this sequence, MCU\_R5FSS0 can read the CTRLMMR\_WKUP\_MAIN\_WARM\_RST\_CTRL[0] SOC\_WARMRST\_ISO\_DONE\_Z bit and begin re-configuring the MAIN domain.

#### Note

On warm resets (except those caused by a VTM Thermal over-temperature), the output of various PLLs are either bypassed or left unaffected. For more details, see [Section 5.3.7, PLL Behavior on Reset](#).

### 5.3.7 PLL Behavior on Reset

[Table 5-98](#) summarizes the PLL behavior on reset.

**Table 5-98. PLL Behavior on Reset**

Reset Type	PLL Behavior	
	MCU Domain PLLs	MAIN Domain PLLs
MCU_PORz	All PLLs are reset	All PLLs are reset
PORz	All PLLs are unaffected	All PLLs are reset
MCU_RESETz or RESET_REQz (or other warm resets except VTM Thermal over-temperature event)	All PLL outputs are bypassed (to the reference input frequency into the PLL), see <a href="#">Table 5-99</a> .	Some PLL outputs are bypassed (to the reference input frequency into the PLL) and other PLLs are unaffected, see <a href="#">Table 5-99</a> .
Warm reset triggered by a VTM thermal over-temperature event	All PLL outputs are bypassed	All PLL outputs are bypassed

**Table 5-99. PLL Behavior on Warm Resets, Except VTM Event**

Not-bypassed (unaffected)	Bypassed
PLL1	MCU_PLL0
PLL2	MCU_PLL1
PLL3	MCU_PLL2
PLL16	PLL0
PLL17	PLL4
PLL18	PLL5
PLL19	PLL6
PLL23	PLL7
	PLL8
	PLL12
	PLL13
	PLL14
	PLL15
	PLL24
	PLL25

The warm reset event bypasses the PLL (that are to be bypassed) while the warm reset is asserted; after the warm reset is released, the PLL switches from bypass to using the PLL again. If the user wants to maintain bypassed PLLs in the bypass mode (for example, to control current slews on the board in a reset event); there are control bits to configure the PLL in this way:

- For MCU\_PLL\_[2:0]:  
CTRLMMR\_WKUP\_MCU\_PLL\_CLKSEL[23] BYP\_WARM\_RST = 0  
CTRLMMR\_WKUP\_MCU\_PLL\_CLKSEL[31] BYPASS\_SW\_OVRD = 1
- For bypassed PLLs in the MAIN domain:  
CTRLMMR\_WKUP\_MAIN\_PLLn\_CLKSEL[23] BYP\_WARM\_RST = 0  
CTRLMMR\_WKUP\_MAIN\_PLLn\_CLKSEL[31] BYPASS\_SW\_OVRD = 1

The behavior with PLLs configured as defined above is:

- the BYPASS\_SW\_OVRD bit ( = 1) enables the following behavior:
  - the PLL currently is not in bypass mode (for example, the PLL clock propagates) because BYP\_WARM\_RST is 0
  - when a warm reset occurs, the PLLs configured to bypass automatically set the BYP\_WARM\_RST to 1. This setting (with BYPASS\_SW\_OVRD = 1) maintains the bypass state after the warm reset is released.

When software is ready to use the PLL, the BYP\_WARM\_RST is cleared back to 0.



## 5.4 Clocking

This chapter describes the clock architecture of the device.

---

### Note

For a detailed visual representation of the distribution and management of the device clocks at the device level, see the clock interactive software for the device - Clock Tree Tool (CTT) .

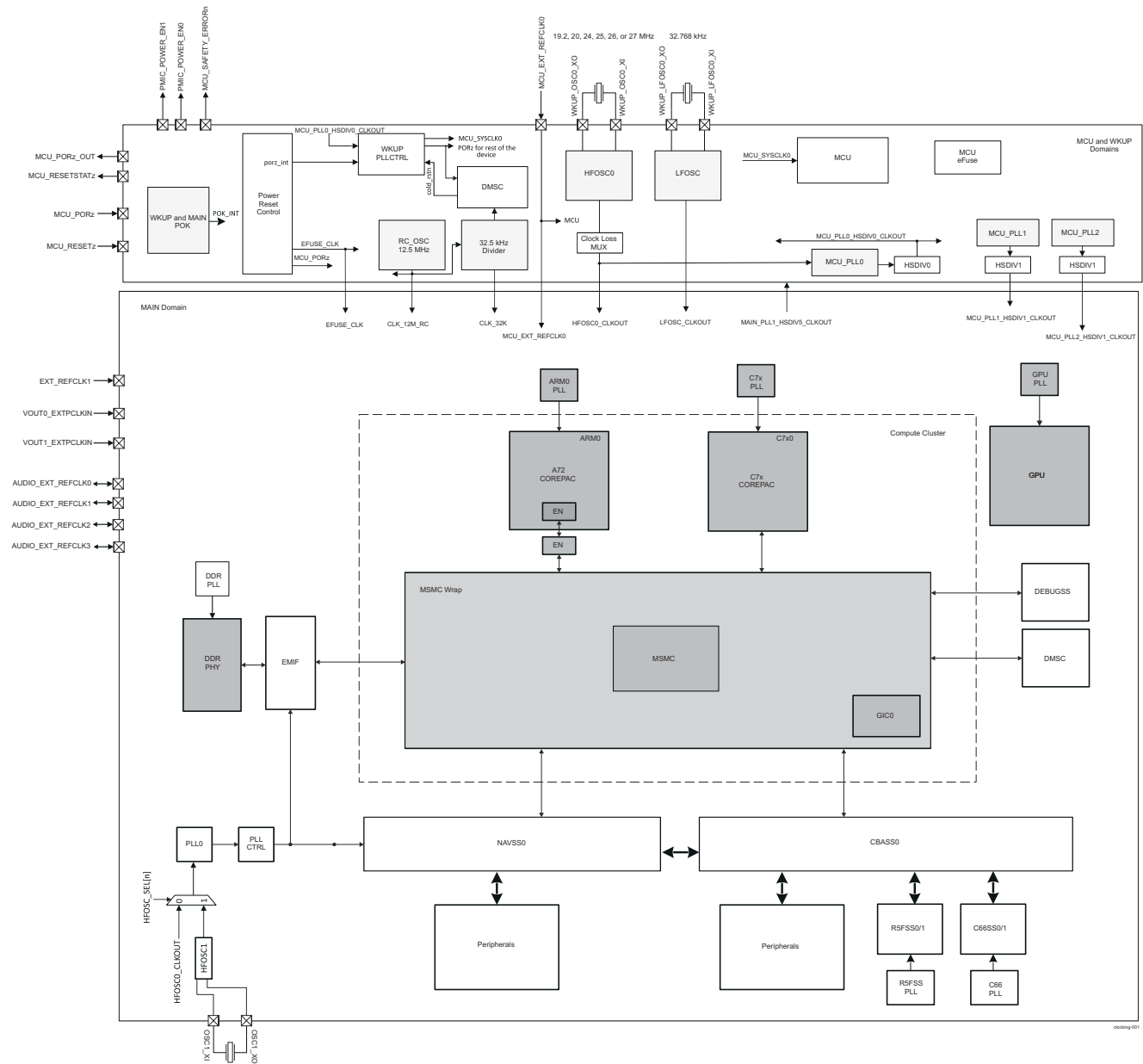
---

### 5.4.1 Overview

To satisfy the various subsystems requirements, the device features multiple clock sources and clock generators:

- External Crystal Drivers
- Internal Oscillator
- Phase-Locked Loop circuits (PLLs)
- Dividers

[Figure 5-29](#) shows the device top-level clock diagram. The clocking is divided into two clocking domains - WKUP/MCU, and MAIN.



### Figure 5-29. Top-Level Clock Diagram

## 5.4.2 Clock Inputs

### 5.4.2.1 Overview

Various external clock inputs are needed to drive the device. Summary of these input clock signals is given below. Some of those inputs are bidirectional and are marked as inputs/outputs in the summary:

- High frequency oscillators
  - WKUP\_OSC0\_XO/WKUP\_OSC0\_XI — external main crystal interface pins of the internal oscillator which sources a reference clock. Provides reference clock to PLLs within WKUP/MCU and MAIN domain.
  - OSC1\_XO/OSC1\_XI — external main crystal interface pins connected to internal oscillator which sources reference clock. Provides reference clock to PLLs within MAIN domain. This high-frequency oscillator is used to provide audio clock frequencies to MCASPs.
- Low frequency oscillator
  - WKUP\_LFOSC\_XO/WKUP\_LFOSC\_XI — external main crystal interface pins connected to internal oscillator which sources reference clock provides a clock for low power operation.
- General purpose clock inputs
  - MCU\_EXT\_REFCLK0 — optional external system clock input (MCU domain).
  - EXT\_REFCLK1 — optional external system clock input (MAIN domain). Optionally PLL4 (AUDIO0 PLL), PLL15 (AUDIO0 PLL) and MCASP can be sourced by EXT\_REFCLK1 (sourced externally).
- External video pixel clock inputs
  - VOUT0\_EXTPCLKIN — optional for the DPI0 port of DSS.
  - VOUT1\_EXTPCLKIN — optional for the DPI1 port of DSS.
- External CPTS reference clock inputs
  - MCU\_CPTS0\_RFT\_CLK — CPTS reference clock inputs for MCU\_CPTS\_RFT\_CLK.
  - CPTS0\_RFT\_CLK — CPTS reference clock inputs for CPTS\_RFT\_CLK.
- External audio reference clock input/output pins. For more information which clocks may be sourced as an outputs to the pins, see [Section 12.5.2, Multichannel Audio Serial Port \(MCASP\)](#).
  - AUDIO\_EXT\_REFCLK0
  - AUDIO\_EXT\_REFCLK1
  - AUDIO\_EXT\_REFCLK2
  - AUDIO\_EXT\_REFCLK3

#### Note

Other clock inputs that are routed directly to the subsystems are described in the respective subsystem chapters.

### 5.4.2.2 Mapping of Clock Inputs

[Table 5-100](#) lists the mapping for the device clock inputs.

**Table 5-100. Mapping for Input Sources**

Domain	Clock	Ball Mapping	Frequency List / Range	Type
WKUP	WKUP_HFOSC0_CLK	WKUP_OSC0_XI WKUP_OSC0_XO	19.2, 20, 24, 25, 26, or 27 MHz	External Clock / Crystal Pins
	WKUP_LFOSC0_CLKOUT	WKUP_LFOSC_XI WKUP_LFOSC_XO	32.768 kHz	External Clock / Crystal Pins
	MCU_EXT_REFCLK0	MCU_EXT_REFCLK0	up to 100 MHz	LVC MOS
	MCU_CPTS_RFT_CLK	MCU_CPTS0_RFT_CLK	up to 100 MHz	LVC MOS
MAIN	HFOSC1_CLK	OSC1_XI OSC1_XO	19.2, 20, 24, 25, 26, or 27 MHz For audio applications: 22.5792 MHz or 24.576 MHz	External Clock / Crystal Pins
	EXT_REFCLK1	EXT_REFCLK1	up to 100 MHz	LVC MOS
	VOUT0_EXTPCLKIN	VOUT0_EXTPCLKIN	up to 150 MHz	LVC MOS

**Table 5-100. Mapping for Input Sources (continued)**

Domain	Clock	Ball Mapping	Frequency List / Range	Type
	VOUT1_EXTCLKIN	VOUT1_EXTCLKIN	up to 150 MHz	LVC MOS
	CPTS_RFT_CLK	CPTS0_RFT_CLK	up to 150 MHz	LVC MOS
	AUDIO_EXT_REFCLK0	AUDIO_EXT_REFCLK0	up to 80 MHz	LVC MOS
	AUDIO_EXT_REFCLK1	AUDIO_EXT_REFCLK1	up to 80 MHz	LVC MOS
	AUDIO_EXT_REFCLK2	AUDIO_EXT_REFCLK2	up to 80 MHz	LVC MOS
	AUDIO_EXT_REFCLK3	AUDIO_EXT_REFCLK3	up to 80 MHz	LVC MOS

[Table 5-101](#) lists the internal RC-oscillator sources.

**Table 5-101. Internal RC Oscillator Input Sources**

Domain	RC Oscillator	Clock	Frequency
WKUP	WKUP_RC_OSC_12M	CLK_12M_RC	12.5 MHz

### 5.4.3 Clock Outputs

The device provides several system clock outputs. Summary of these output clock signals is as follows:

- MCU\_SYCLKOUT0
- MCU\_OBSCLK0
- SYCLKOUT0
- OBSCLK0, OBSCLK1, OBSCLK2

Other clock outputs, that are routed directly from subsystems to device pins, are described in the respective module chapter.

Observation clock pins - MCU\_OBSCLK0, OBSCLK0, OBSCLK1, and OBSCLK2 serve the following purposes:

- During testing, PLLs on the device are configured to output all operating frequencies desired by each module to characterize PLL performance.
- System debug: during debug, clocks from various PLLs can be inspected for possible clues. Example clock glitches, lock loss etc.

#### Note

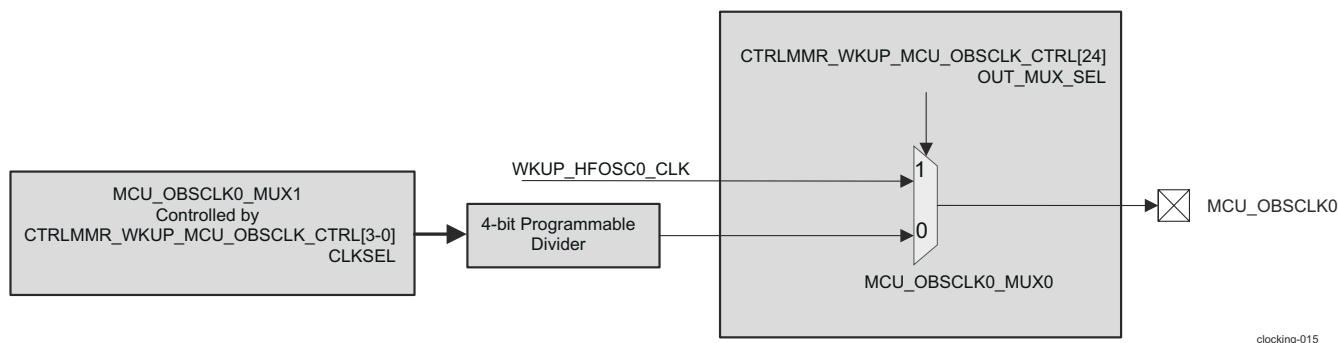
Maximum frequency supported on both MCU\_OBSCLK0, OBSCLK0, OBSCLK1, and OBSCLK2 pins is 200 MHz. Hence the divider at the output of each OBSCLK mux must be programmed to meet this limitation.

Unlike the OBSCLK pins which can select several different clocks for output, system clock pins (MCU\_SYCLKOUT0 and SYCLKOUT0) are hardwired to dedicated clock resources (see [Section 5.4.3.2](#))

#### 5.4.3.1 Observation Clock Pins

##### 5.4.3.1.1 MCU\_OBSCLK0 Pin

MCU\_OBSCLK0 output is controlled by CTRLMMR\_WKUP\_MCU\_OBSCLK\_CTRL register in the WKUP\_CTRL\_MMR0 module; for more information about control registers, see *Control Module (CTRL\_MMR)*. Two muxes are connected in series - MCU\_OBSCLK0\_MUX0 and MCU\_OBSCLK0\_MUX1, see [Figure 5-30](#).



**Figure 5-30. MCU\_OBSCLK0 Muxes Diagram**

How to select the output of MCU\_OBSCLK0\_MUX1 is described in [Table 5-102](#).

**Table 5-102. MCU\_OBSCLK0\_MUX1 Output Clock Selection**

CTRLMMR_WKUP_MCU_OBSCLK_CTRL <sup>(2)</sup> [3-0] CLK_SEL	MCU_OBSCLK0_MUX1 Output Clock Selection <sup>(1)</sup>
0x0	CLK_12M_RC
0x1	0 (GND) <sup>(3)</sup>
0x2	MCU_PLL0_HSDIV0_CLKOUT
0x3	WKUP_PLLCTL_OBSCLK (MCU_PLL0 input reference clock)
0x4	MCU_PLL1_HSDIV1_CLKOUT

**Table 5-102. MCU\_OBSCLK0\_MUX1 Output Clock Selection (continued)**

CTRLMMR_WKUP_MCU_OBSCLK_CTRL <sup>(2)</sup> [3-0] CLK_SEL	MCU_OBSCLK0_MUX1 Output Clock Selection <sup>(1)</sup>
0x5	MCU_PLL1_HSDIV2_CLKOUT
0x6	MCU_PLL1_HSDIV3_CLKOUT
0x7	MCU_PLL1_HSDIV4_CLKOUT
0x8	MCU_PLL2_HSDIV0_CLKOUT
0x9	CLK_32K
0xA	MCU_PLL2_HSDIV1_CLKOUT
0xB	MCU_PLL2_HSDIV2_CLKOUT
0xC	MCU_PLL2_HSDIV3_CLKOUT
0xD	MCU_PLL2_HSDIV4_CLKOUT
0xE	WKUP_HFOSC0_CLKOUT
0xF	WKUP_LFOSC0_CLKOUT

(1) Software-controlled 4-bit divider is used to obtain the divided versions of the clocks passed to MCU\_OBSCLK1 mux

(2) For more information about control registers, see *Control Module (CTRL\_MMR)*.

(3) GND - ground

The value of the software-controlled 4-bit divider is determined by register CTRLMMR\_WKUP\_MCU\_OBSCLK\_CTRL[11-8] MCU\_OBSCLK\_CTRL\_CLK\_DIV in the WKUP\_CTRL\_MMR0 module; for more information about control registers, see *Control Module (CTRL\_MMR)*.

#### Note

MCU\_OBSCLK1\_MUX0 is provided as a low jitter output for WKUP\_HFOSC0\_CLK. In this configuration, CTRLMMR\_WKUP\_MCU\_OBSCLK\_CTRL[3:0] should be configured as 0001b (1, selecting a logical low signal) and CTRLMMR\_WKUP\_MCU\_OBSCLK\_CTRL[24] should be configured as 1.

#### 5.4.3.1.2

#### Note

MCU\_OBSCLK1\_MUX0 is provided as a low jitter output for WKUP\_HFOSC0\_CLK. In this configuration, CTRLMMR\_WKUP\_MCU\_OBSCLK\_CTRL[3:0] should be configured as 0001b (1, selecting a logical low signal) and CTRLMMR\_WKUP\_MCU\_OBSCLK\_CTRL[24] should be configured as 1.

#### 5.4.3.1.3 OBSCLK0, OBSCLK1, and OBSCLK2 Pins

The OBSCLK0, OBSCLK1, and OBSCLK2 output pins are controlled simultaneously, so that the three pins are connected to the same signal. OBSCLK0, OBSCLK1, and OBSCLK2 outputs are controlled by CTRLMMR\_OBSCLK0\_CTRL register in the CTRL\_MMR0 module; for more information about control registers, see *Control Module (CTRL\_MMR)*. [Figure 5-31](#) shows a block diagram of internal OBSCLK0 mux connections.

#### Note

OBSCLK2 has lot more jitter due to PD limitations/long routing. If the intent is to use OBSCLK2 to do anything beyond just observe clock toggle, customers should switch to OBSCLK0 or OBSCLK1.


**Figure 5-31. OBSCLK0 Muxes Diagram**

**Table 5-103. OBSCCLK0\_MUX1\_CLKOUT**

CTRLMMR_OBSCCLK1_CTRL <sup>(1)</sup> [1-0] CLK_SEL	OBSCCLK0_MUX1_CLKOUT Selection
0x0	MAIN_PLL7_HSDIV0_CLKOUT / 4
0x1	MAIN_PLL8_HSDIV0_CLKOUT / 8
0x2	MAIN_PLL13_HSDIV0_CLKOUT / 4
0x3	0 (GND) <sup>(2)</sup>

(1) For more information about control registers, see *Control Module (CTRL\_MMR)*.

(2) GND - ground

**Table 5-104. OBSCCLK0, OBSCCLK1, and OBSCCLK2 Clock Selection**

CTRLMMR_OBSCCLK0_CTRL <sup>(2)</sup> [4-0] CLK_SEL	OBSCCLK0, OBSCCLK1, and OBSCCLK2 Selection <sup>(1)</sup>
0x0	MAIN_PLL0_HSDIV0_CLKOUT
0x1	MAIN_PLL1_HSDIV0_CLKOUT
0x2	MAIN_PLL2_HSDIV0_CLKOUT
0x3	MAIN_PLL3_HSDIV0_CLKOUT
0x4	MAIN_PLL4_HSDIV0_CLKOUT
0x5	MAIN_PLL5_HSDIV0_CLKOUT
0x6	MAIN_PLL6_HSDIV0_CLKOUT
0x7	0 (GND) <sup>(3)</sup>
0x8	0 (GND) <sup>(3)</sup>
0x9	0 (GND) <sup>(3)</sup>
0xA	0 (GND) <sup>(3)</sup>
0xB	0 (GND) <sup>(3)</sup>
0xC	MAIN_PLL12_HSDIV0_CLKOUT
0xD	Input from OBSCCLK0_MUX1_CLKOUT
0xE	MAIN_PLL14_HSDIV0_CLKOUT
0xF	MAIN_PLL15_HSDIV0_CLKOUT
0x10	MAIN_PLL16_HSDIV0_CLKOUT
0x11	MAIN_PLL17_HSDIV0_CLKOUT
0x12	MAIN_PLL18_HSDIV0_CLKOUT
0x13	MAIN_PLL19_HSDIV0_CLKOUT
0x14	UFS MPHY_TX_REF_SYMBOLCLK
0x15	UFS MPHY_M31_VCO_19P2M_CLK
0x16	UFS MPHY_M31_VCO_26M_CLK
0x17	MAIN_PLL23_HSDIV0_CLKOUT
0x18	MAIN_PLL24_HSDIV0_CLKOUT
0x19	MAIN_PLL25_HSDIV0_CLKOUT
0x1A	CPTS_GENF3
0x1B	CLK_12M_RC
0x1C	LFXOSC_CLKOUT
0x1D	PLLCTRL_OBSCCLK (PLL0 input reference clock)
0x1E	HFOSC1_CLK
0x1F	WKUP_HFOSC0_CLKOUT

(1) Software-controlled 8-bit divider is used to obtain the divided versions of the clocks passed to output pins

(2) For more information about control registers, see *Control Module (CTRL\_MMR)*.

(3) GND - ground

The value of the software-controlled 8-bit divider is determined by register CTRLMMR\_OBSCCLK0\_CTRL[15-8] OBSCCLK\_CTRL\_CLK\_DIV; for more information about control registers, see *Control Module (CTRL\_MMR)*.



### 5.4.3.2 System Clock Pins

#### 5.4.3.2.1 MCU\_SYSCLKOUT0

MCU\_SYSCLK0 is divided by 4 and then send out of the device as a LVCMOS clock signal (MCU\_SYSCLKOUT0).

#### Note

MCU\_SYSCLKOUT0 cannot be used as a clock source for external devices on the board.

#### 5.4.3.2.2 SYSCLKOUT0

SYSCLK0 is divided by 4 and then send out of the device as a LVCMOS clock signal (SYSCLKOUT0). This signal can be used to test if the specific device clock is functioning or not.

#### Note

SYSCLKOUT0 cannot be used as a clock source for external devices on the board.

### 5.4.4 Device Oscillators

The device has the possibility to source clocks of two external high-frequency oscillators - WKUP\_HFOSC0 and HFOSC1 and one external low-frequency oscillator - WKUP\_LFOSC0. The device has one internal RC oscillator - WKUP\_RC\_OSC\_12M.

#### 5.4.4.1 Device Oscillators Integration

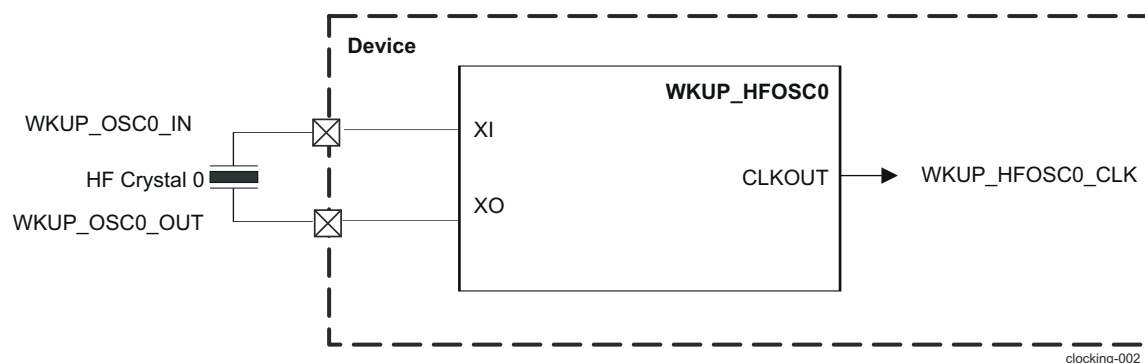
##### 5.4.4.1.1 Oscillators with External Crystal

By default, WKUP\_LFOSC0 and HFOSC1 are disabled, while WKUP\_HFOSC0 is enabled after PORz.

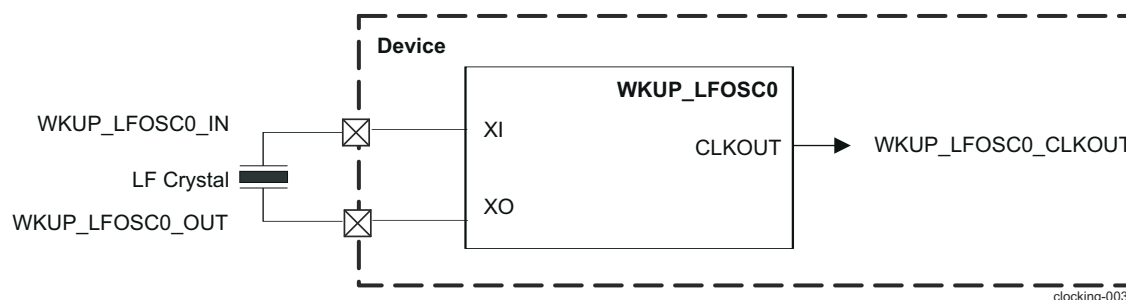
Program CTRLMMR\_WKUP\_LFXOSC\_CTRL register to enable WKUP\_LFOSC0 oscillator.

CTRLMMR\_WKUP\_LFXOSC\_TRIM[18-16] I\_MULT requires configuration based upon the load capacitance of the low-frequency crystal circuit.

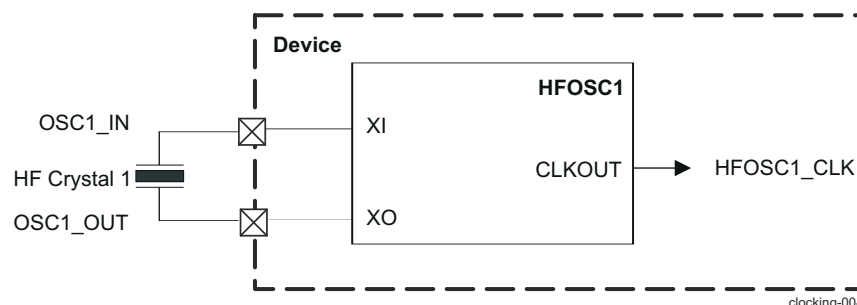
All device oscillators with external crystals support a bypass mode. In this mode, an external clock can be driven on the XI Pin.



**Figure 5-32. WKUP\_HFOSC0 Integration Diagram**



**Figure 5-33. WKUP\_LFOSC0 Integration Diagram**



**Figure 5-34. HFOSC1 Integration Diagram**

#### 5.4.4.1.2 Internal RC Oscillator

WKUP\_RC\_OSC\_12M is always oscillate mode.

The RC clock output (WKUP\_RC\_OSC\_12M) is used by the on-chip Reset Glue logic, Power Reset Generator (PRG) circuits, WKUP\_HFOSC0 Loss Circuits, Dual Clock Comparator (DCC) and Timer Modules. Its frequency is targeted at 12.55MHz +/-10%.

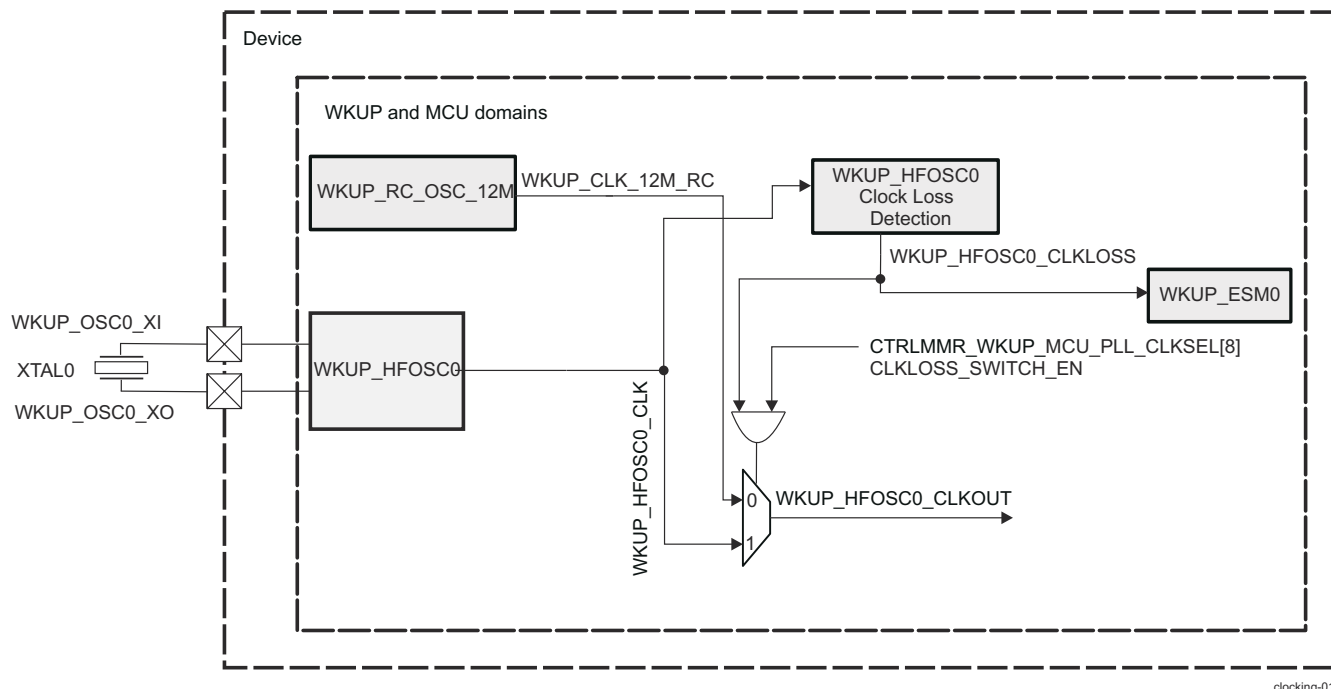
#### Note

An output clock of WKUP\_RC\_OSC\_12M is further divided down to generate a 32-kHz clock (CLK\_32K). This clock is used by WKUP\_DMSC0 and Timer Modules.

If an application needs accurate 32.768-kHz clock, then use WKUP\_LFOSC0 (32.768 kHz) output clock.

#### 5.4.4.2 Oscillator Clock Loss Detection

The device supports WKUP\_HFOSC0 clock loss circuitry to detect when WKUP\_HFOSC0\_CLK stops toggling. If CTRLMMR\_WKUP\_MCU\_PLL\_CLKSEL[8] CLKLOSS\_SWCH\_EN is set and WKUP\_HFOSC0\_CLK stops toggling condition is detected, the reference clock is switched from WKUP\_HFOSC0\_CLKOUT to WKUP\_CLK\_12M\_RC to allow the device to operate with a slower clock. The WKUP\_HFOSC0\_CLK stops toggling condition is reported as an error to WKUP\_ESM0 regardless of the value of CTRLMMR\_WKUP\_MCU\_PLL\_CLKSEL[8] CLKLOSS\_SWCH\_EN. Integration diagram of WKUP\_HFOSC0 clock loss detection is presented in [Figure 5-35](#).



**Figure 5-35. WKUP\_HFOSC0 Clock Loss Detection Integration Diagram**

WKUP\_ESM0 can optionally generate an interrupt to WKUP\_DMSC0 and MCU\_R5FSS so they can save some critical contents such as error logging to scratch pad memory or external flash. ESM must also be configured to report this error on the SAFETY\_ERRORn pin.

WKUP\_HFOSC0 clock loss is a catastrophic failure since this clock is the most critical system clock. During the clock loss condition an external system intervention is required.

The clock loss mux is not a glitch free mux. The mux control is synchronized to the CLK\_12M\_RC clock. Since WKUP\_HFOSC0\_CLK has stopped switching the mux should transition to RC Clock without producing glitches.

Hence it is important to ensure that MCU\_SAFETY\_ERRORn is pulled low during this error condition. In clock-loss condition, the device reports the error to the external device through MCU\_SAFETY\_ERRORn pin - the pin is driven Low. The recovery mechanism is up to the external system (such as a PMIC to take action). For example, it can try a full system power cycle to see if the system recovers. If the system does not recover then, it has to take some other action such as to check system clocks, external crystal, supply rails.

PLL will lose the lock when clock loss is detected. During this time 12.5-MHz RC clock is output through the bypass muxes. The PLL will relock to a new frequency based on 12.5 MHz.

In an event of clock glitch which may potentially hang the device, then the WKUP\_DMSC0 watchdog timer will expire and will generate an internal reset for the whole device.

### 5.4.5 PLLs

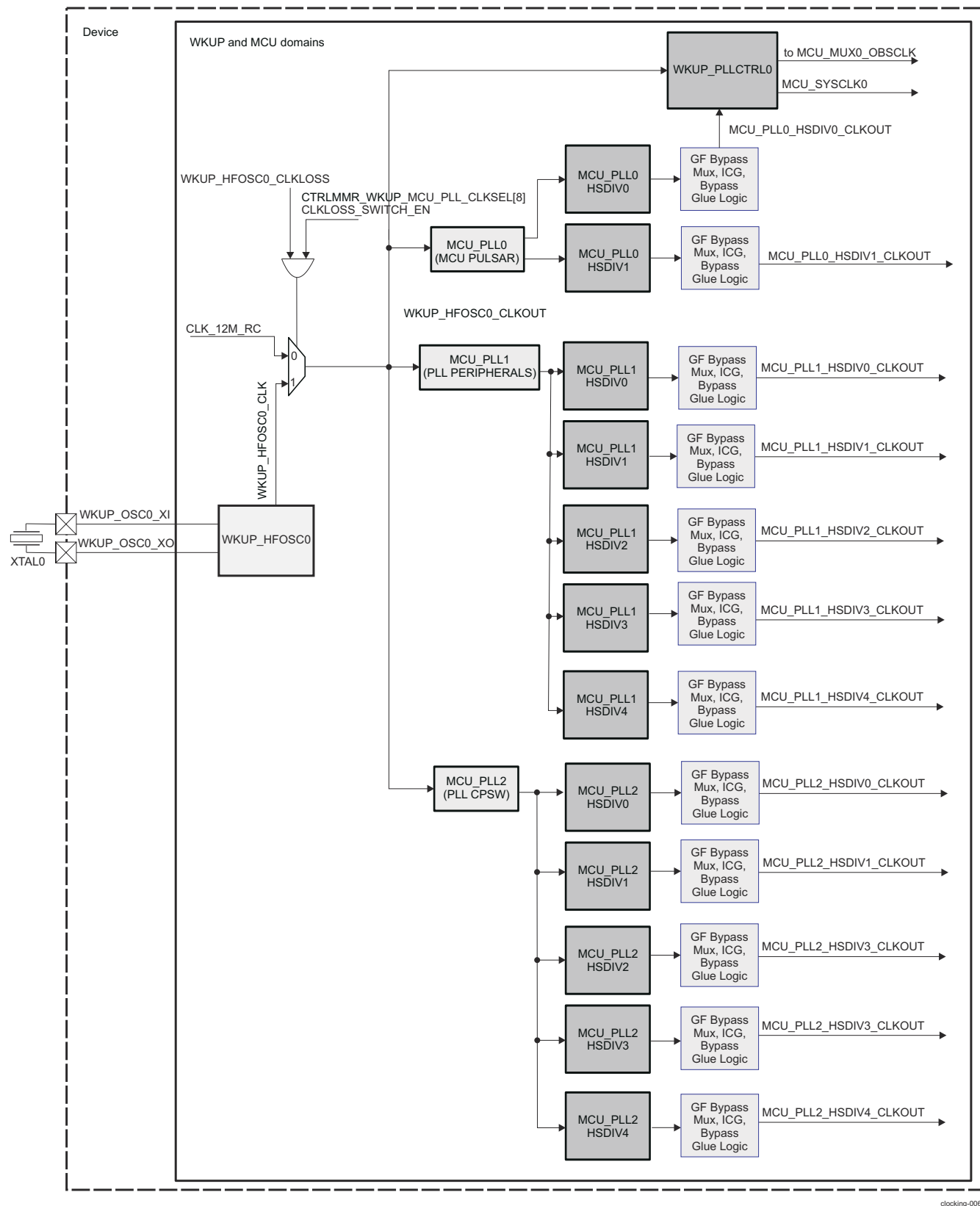
Phase-Locked Loop circuits (PLLs) in the device are clock generator PLLs, which multiply the lower-frequency reference clock up to the operating frequency of the respective subsystem(s).

#### 5.4.5.1 WKUP and MCU Domains PLL Overview

There are total three PLLs in the device in WKUP/MCU domain:

- MCU\_PLL0 (MCU R5FSS PLL) with WKUP\_PLLCTRL0
- MCU\_PLL1 (MCU PERIPHERAL PLL)
- MCU\_PLL2 (MCU Command Platform Ethernet Switch (CPSW) PLL).

Overview of the device PLLs with their reference clock options in WKUP/MCU domain is shown on [Figure 5-36](#). For more specific information about PLLs see [Section 5.4.5.5, PLLs Device-Specific Information](#).



docking-006

Figure 5-36. WKUP/MCU Domain PLLs Integration

#### 5.4.5.2 MAIN Domain PLLs Overview

There are total twenty PLLs in the device in MAIN domain:

- PLL0 (MAIN PLL) with PLLCTRL0
- PLL1 (PER0 PLL)
- PLL2 (PER1 PLL)
- PLL3 (CPSW9G PLL)
- PLL4 (AUDIO0 PLL)
- PLL5 (VIDEO PLL)
- PLL6 (GPU PLL)
- PLL7 (C7x PLL)
- PLL8 (ARM0 PLL)
- PLL12 (DDR PLL)
- PLL13 (C66 PLL)
- PLL14 (R5FSS PLL)
- PLL15 (AUDIO1 PLL)
- PLL16 (DSS PLL0)
- PLL17 (DSS PLL1)
- PLL18 (DSS PLL2)
- PLL19 (DSS PLL3)
- PLL23 (DSS PLL7)
- PLL24 (MLB PLL)
- PLL25 (VISION PLL)

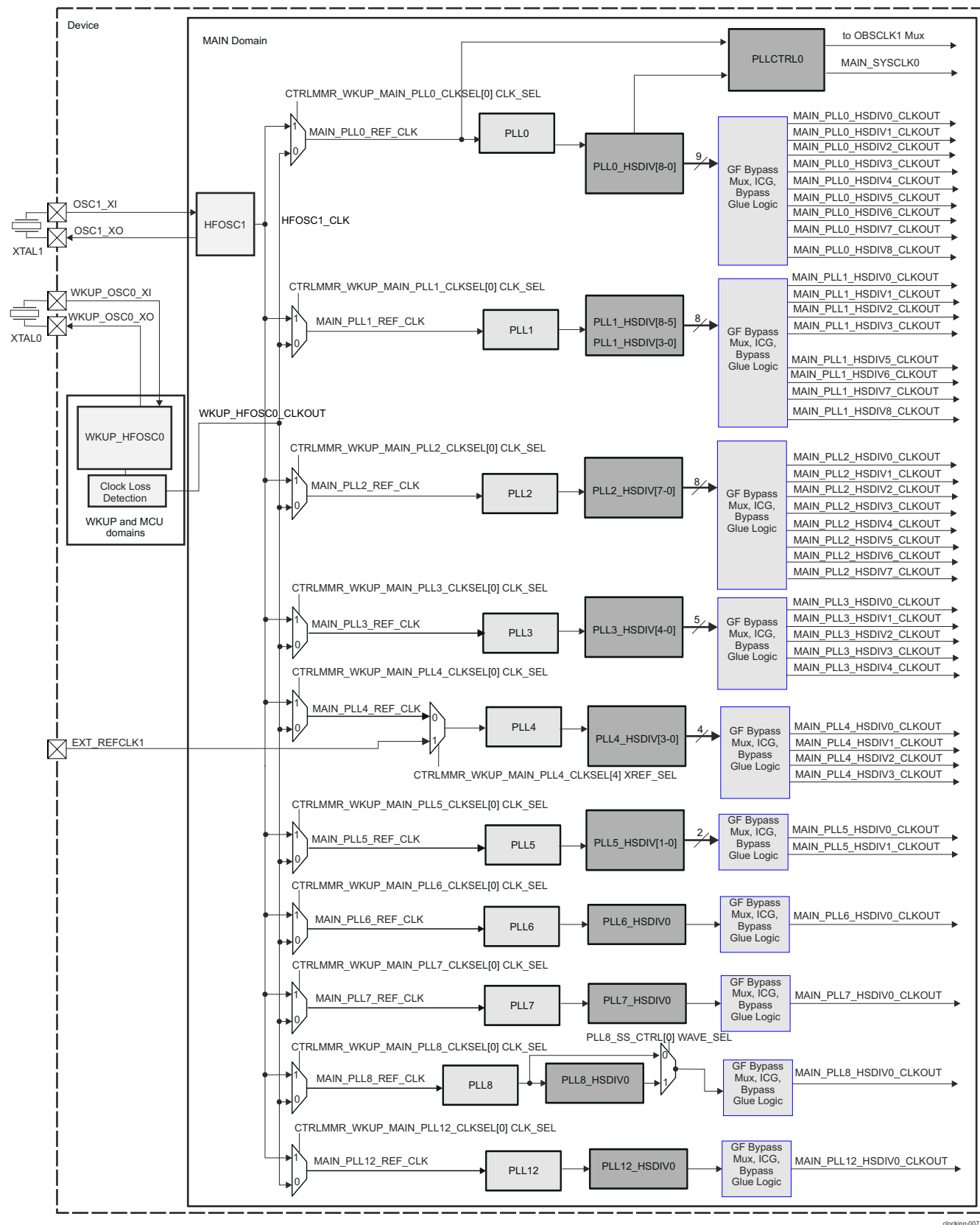
Overview of the device PLLs with their reference clock options in MAIN domain is shown on [Figure 5-37](#) and [Figure 5-38](#). For more specific information about PLLs see [Section 5.4.5.5, PLLs Device-Specific Information](#).

---

#### Note

The external muxes of choosing the reference clocks are glitch-free muxes.

---



**Figure 5-37. MAIN Domain PLLs Integration - Part 1**

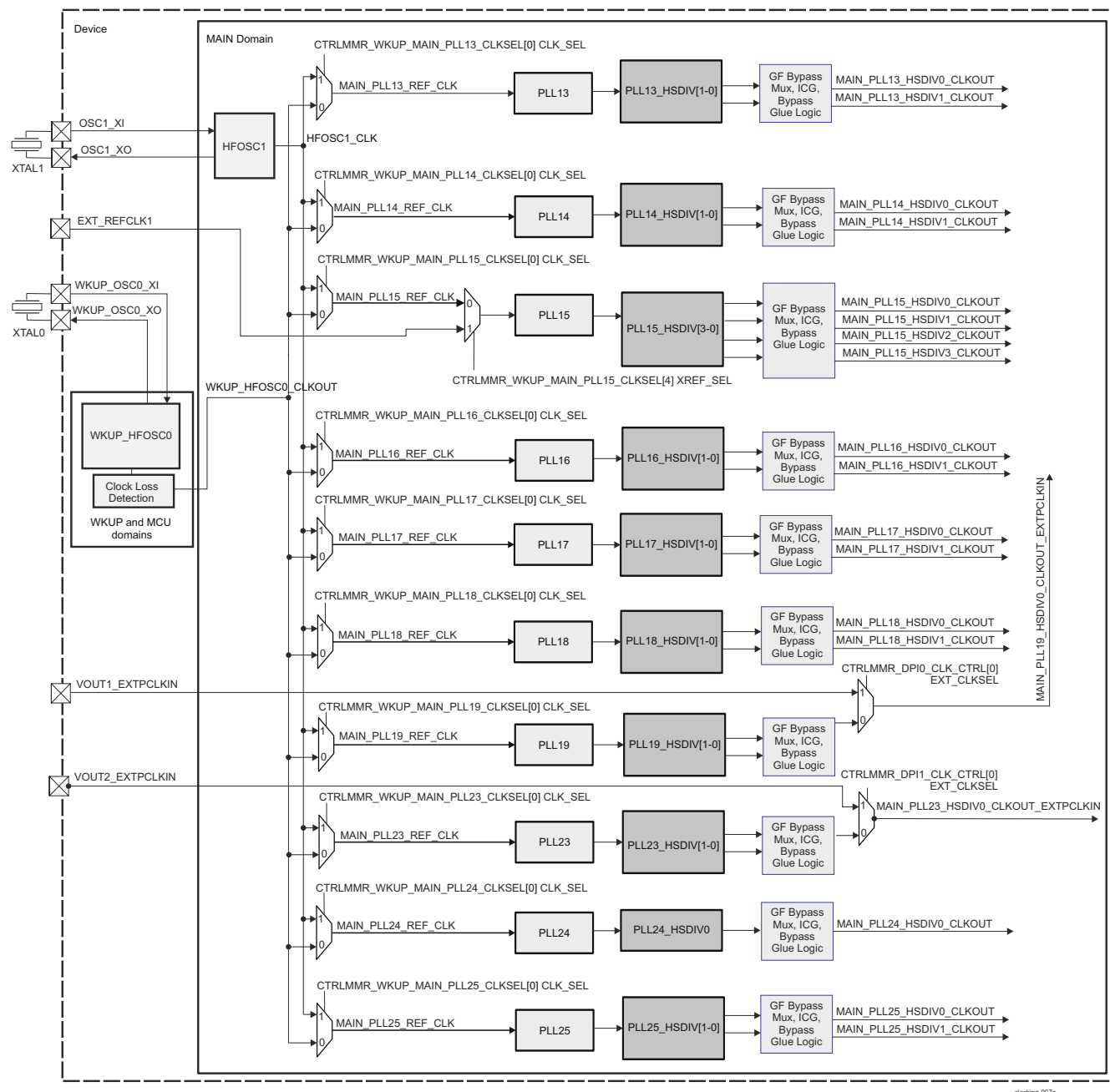


Figure 5-38. MAIN Domain PLLs Integration - Part 2

### 5.4.5.3 PLL Reference Clocks

#### 5.4.5.3.1 PLLs in MCU Domain

The reference clocks MCU\_PLL0\_REF\_CLK, MCU\_PLL1\_REF\_CLK and MCU\_PLL2\_REF\_CLK for the PLLs in MCU domain is chosen between the internal high-frequency (HF) oscillator with external crystal, WKUP\_HFOSC0, and 12.5-MHz free-running RC oscillator. The selection for all PLLs is made through CTRLMMR\_WKUP\_MCU\_PLL\_CLKSEL[8] CLKLOSS\_SWCH\_EN, see [Table 5-105](#).

Table 5-105. PLL MCU Domain Reference Clock Selection - WKUP\_HFOSC0\_CLKOUT Selection

CTRLMMR_WKUP_MCU_PLL_CLKSEL <sup>(2)</sup> [8] CLKLOSS_SWCH_EN	MCU_PLLn_REF_CLK <sup>(1)</sup>
0 (default)	WKUP_HFOSC0_CLK



**Table 5-105. PLL MCU Domain Reference Clock Selection - WKUP\_HFOSC0\_CLKOUT Selection (continued)**

CTRLMMR_WKUP_MCU_PLL_CLKSEL <sup>(2)</sup> [8] CLKLOSS_SWCH_EN	MCU_PLLn_REF_CLK <sup>(1)</sup>
1	WKUP_CLK_12M_RC if clock loss is detected or WKUP_HFOSC0_CLK if clock loss is not detected

(1) n = 0, 1, 2

(2) For more information about control registers, see *Control Module (CTRL\_MMR)*.

#### 5.4.5.3.2 PLLs in MAIN Domain

Each PLL in MAIN domain has a dedicated register CTRLMMR\_WKUP\_MAIN\_PLLn\_CLKSEL in WKUP\_CTRL\_MMR0 to choose its reference clock.

MAIN\_PLLn\_REF\_CLK (n = 0 to 8, 12 to 19, 23 to 25) clock source is selected by means of CTRLMMR\_WKUP\_MAIN\_PLLn\_CLKSEL[0] CLK\_SEL bit; for more information about control registers, see *Control Module (CTRL\_MMR)*. The encoding of CLK\_SEL is shown in [Table 5-106](#).

**Table 5-106. PLL MAIN Domain Reference Clock Selection**

CTRLMMR_WKUP_MAIN_PLLn_CLKSEL <sup>(1)</sup> [0] CLK_SEL	MAIN_PLLn_REF_CLK Selection <sup>(1)</sup>
0 (default)	WKUP_HFOSC0_CLK
1	HFOSC1_CLK

(1) n = 0 to 8, 12 to 19, 23 to 25

PLL4 (AUDIO0 PLL) and PLL15 (AUDIO1 PLL) input reference clocks have additional selection via mux configured in the CTRLMMR\_WKUP\_MAIN\_PLL4\_CLKSEL[4] XREF\_SEL and the CTRLMMR\_WKUP\_MAIN\_PLL15\_CLKSEL[4] XREF\_SEL, respectively, in WKUP\_CTRL\_MMR0 (see [Table 5-107](#) and [Table 5-108](#)); for more information about control registers, see *Control Module (CTRL\_MMR)*.

**Table 5-107. PLL4 (AUDIO0 PLL) Clock Selection**

CTRLMMR_WKUP_MAIN_PLL4_CLKSEL[4] XREF_SEL	PLL4 (AUDIO0 PLL) Ref Selection
0 (default)	MAIN_PLL4_REF_CLK already selected via MAIN_PLL4_REF_CLK[0] CLK_SEL
1	EXT_REFCLK1 (external clock input on device pins)

**Table 5-108. PLL15 (AUDIO1 PLL) Clock Selection**

CTRLMMR_WKUP_MAIN_PLL15_CLKSEL[4] XREF_SEL	PLL15 (AUDIO1 PLL) Ref Selection
0 (default)	MAIN_PLL15_REF_CLK already selected via MAIN_PLL15_REF_CLK[0] CLK_SEL
1	EXT_REFCLK1 (external clock input on device pins)

#### 5.4.5.4 Generic PLL Overview

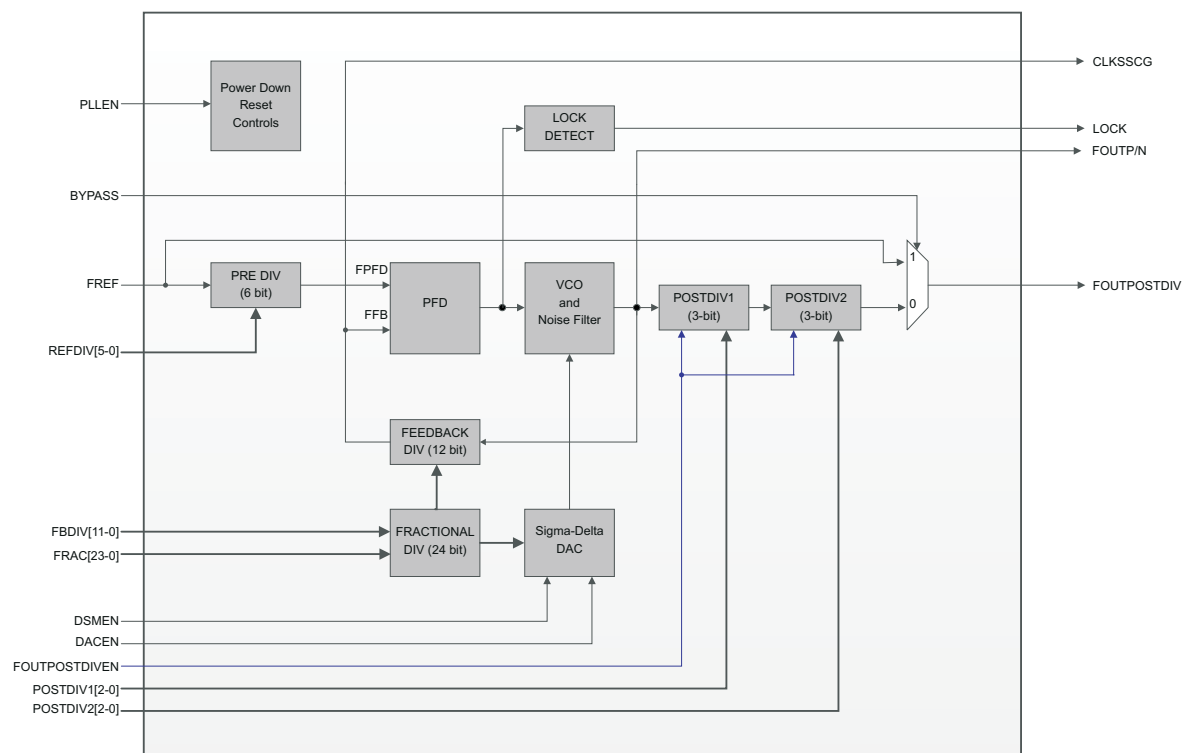
To generate high-frequency clocks, the device supports multiple on-chip PLLs controlled directly by the Top-level Clocking. They are of three types: Standard fractional PLL (PLLTS16FFCLAFRAC2), Fractional PLL with Calibration (PLLTS16FFCLAFRACF) and De-skew PLL with Calibration (PLLTS16FFCLVDESKEWC) PLLs.

#### Note

This chapter discusses only the PLLs that are directly controlled by the Top-level Clocking. The other PLLs embedded in and managed by other subsystems are described in their respective subsystems.

#### 5.4.5.4.1 PLLs Output Clocks Parameters

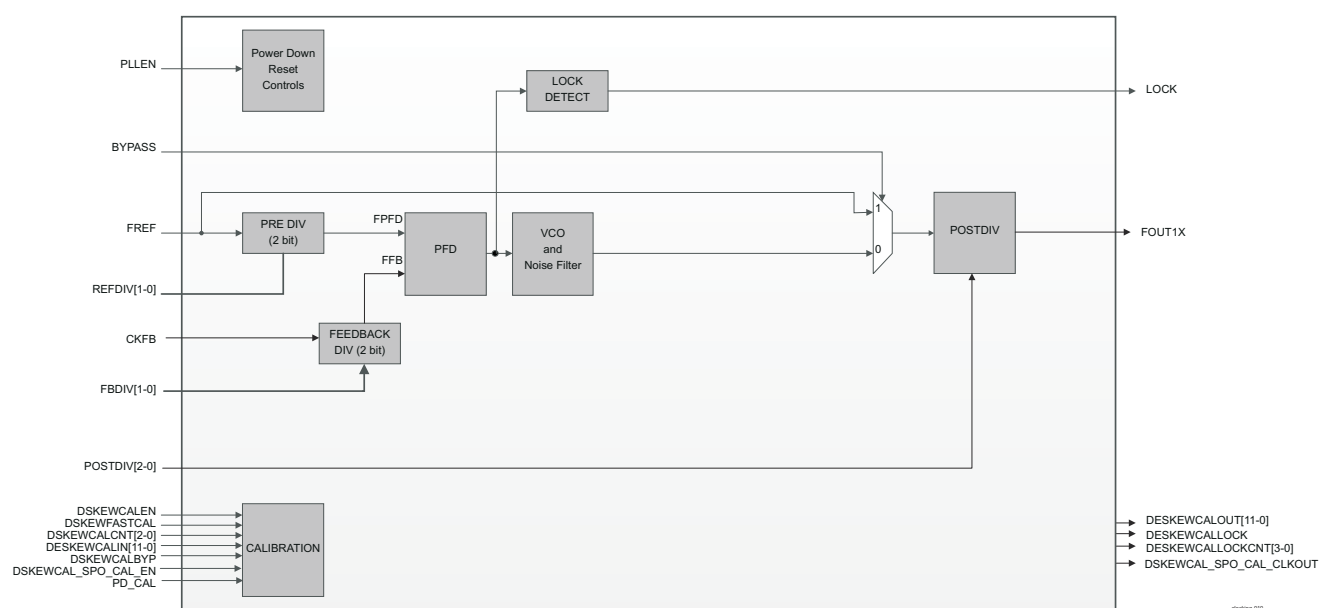
Figure 5-39, Figure 5-40, and Figure 5-41 show the functional architecture of a generic PLL for PLLTS16FFCLAFRAC2, PLLTS16FFCLAFRACF, and PLLTS16FFCLVDESKEWC types, respectively.



clocking-008

**Figure 5-39. Generic PLL Functional Diagram for PLLTS16FFCLAFRAC2 Type**

**Figure 5-40. Generic PLL Functional Diagram for PLLTS16FFCLAFRACF Type**



clocking-010

**Figure 5-41. Generic PLL Functional Diagram for PLLTS16FFCLVDESKEWC Type**

#### 5.4.5.4.1.1 PLLs Input Clocks

As shown in [Figure 5-40](#) and [Figure 5-41](#) for each type of PLL, the two possible input clocks are:

- FREF: Reference clock input is used to generate the synthesized clock but can also be used as the bypass clock for some outputs of the PLL whenever the PLL enters bypass mode. It is mandatory for the PLL clock synthesis.
- CKFB: Feedback clock input. Used only in PLLTS16FFCLVDESKEWC type. The CKFB input is internally connected, see [Figure 5-41](#).

#### 5.4.5.4.1.2 PLL Output Clocks

##### 5.4.5.4.1.2.1 PLLTS16FFCLAFRAC2 Type Output Clocks

[Table 5-109](#) describes the output clocks of PLLTS16FFCLAFRAC2.

**Table 5-109. PLLTS16FFCLAFRAC2 Output Clocks**

Output	Description	Frequency
FOUTP	Positive phase VCO output (no post divider)	$((FREF / REFDIV) * (FBDIV + FRAC))$
FOUTN	Negative phase VCO output (no post divider)	$((FREF / REFDIV) * (FBDIV + FRAC))$
FOUTPOSTDIV	VCO-divided clock output.	$FOUTP / (POSTDIV1 * POSTDIV2)$
CLKSSCG	Clock to SSMOD	$(FREF / REFDIV)$

Where:

- REFDIV is the software-configured division ratio binary value.
- FBDIV is the software-configured division ratio binary value.
- FRAC is the software-configured fractional division.
- POSTDIV1 is the software-configured division ratio binary value.
- POSTDIV2 is the software-configured division ratio binary value.

#### Note

POSTDIV1 and POSTDIV2 valid values are from 1 to 7. To ensure correct operation, POSTDIV1 must always be programmed to a value equal to or greater than POSTDIV2.

#### Note

For device-specific information about clock output parameters and synthesized clocks, see [Table 5-117](#) and [Table 5-119](#).

##### 5.4.5.4.1.2.2 PLLTS16FFCLAFRACF Type Output Clocks

[Table 5-110](#) describes the output clocks of PLLTS16FFCLAFRACF.

**Table 5-110. PLLTS16FFCLAFRACF Output Clocks**

Output	Description	Frequency
FOUTP	Positive phase VCO output (no post divider)	$(FREF / REFDIV) * (FBDIV + FRAC)$
FOUTN	Negative phase VCO output (no post divider)	$(FREF / REFDIV) * (FBDIV + FRAC)$
FOUTPOSTDIV	VCO-divided clock output.	$FOUTP / (POSTDIV1 * POSTDIV2)$
CLKSSCG	Clock to SSMOD	$(FREF / REFDIV)$

Where:

- REFDIV is the software-configured division ratio binary value.
- FBDIV is the software-configured division ratio binary value.
- FRAC is the software-configured fractional division.
- POSTDIV1 is the software-configured division ratio binary value.
- POSTDIV2 is the software-configured division ratio binary value.

---

**Note**

POSTDIV1 and POSTDIV2 valid values are from 1 to 7. To ensure correct operation, POSTDIV1 must always be programmed to a value equal to or greater than POSTDIV2.

---



---

**Note**

For device-specific information about clock output parameters and synthesized clocks, see [Table 5-117](#) and [Table 5-119](#).

---

#### 5.4.5.4.1.2.3 PLLTS16FFCLVDESKEWC Type Output Clocks

[Table 5-111](#) describes the output clocks of PLLTS16FFCLVDESKEWC.

**Table 5-111. PLLTS16FFCLVDESKEWC Output Clocks**

Output	Description	Frequency
FOUT1X	Output Clock	(FREF / REFDIV * FBDIV)

Where:

- REFDIV is the software-configured division ratio binary value.
- FBDIV is the software-configured division ratio binary value.

---

**Note**

REFDIV and FBDIV valid values are 1, 2, and 4.

POSTDIV valid values are 1, 2, 4, 8, 16, 32, 64, and 128.

---



---

**Note**

For device-specific information about clock output parameters and synthesized clocks, see [Table 5-118](#) and [Table 5-120](#).

---

#### 5.4.5.4.1.2.4 PLL Lock

PLL module outputs a lock status signal to indicate that the PLL has achieved frequency lock. When the PLL detects no cycle slips between the feedback clock (FFB) and reference clock FPDF (FREF/ REFDIV[5-0]) for 128 consecutive cycles, it asserts the lock signal high. When the PLL detects any cycle slip, lock signal will go low and stays low until it detects no cycle slip for 128 consecutive cycles. This lock signal is captured in <PLL\_name>\_STAT[0] LOCK bit. Software can read this bit to determine if the PLL has achieved frequency lock before selecting the PLL clock through external bypass mux.

When PLL losses lock, there is a hardware mechanism to automatically bypass the PLL clock to the reference clock (FREF) using the external glitch free mux. BYP\_ON\_LOCKLOSS bit in <PLL\_name>\_CTRL register enables this automatic bypass mode on PLL lock loss. When the PLL re-locks, the glitch free mux switches to PLL clock out.

The PLL lock signal is also routed to ESM module for error reporting. ESM can be configured to generate interrupts to MCU\_R5FSS/R5FSS and WKUP\_DMSC0 and assert Low on SAFETY\_ERRORn pin on PLL Lock loss for further action.

Table 5-112 summarizes the PLL lock signals in the device (all level type).

**Table 5-112. PLL Lock Signal Summary**

PLL Instance	PLL Lock Signal (Source)	ESM Error Input (Destination)
PLL0 (MAIN PLL)	PLLFRAC2_SSMOD0_LOCKLOSS_IPCFG_0	ESM0_LVL_IN_0
PLL1 (PER0 PLL)	PLLFRAC2_SSMOD1_LOCKLOSS_IPCFG_0	ESM0_LVL_IN_1
PLL2 (PER1 PLL)	PLLFRAC2_SSMOD2_LOCKLOSS_IPCFG_0	ESM0_LVL_IN_2
PLL3 (CPSW9G PLL)	PLLFRAC2_SSMOD3_LOCKLOSS_IPCFG_0	ESM0_LVL_IN_3
PLL4 (AUDIO0 PLL)	PLLFRAC2_SSMOD4_LOCKLOSS_IPCFG_0	ESM0_LVL_IN_4
PLL5 (VIDEO PLL)	PLLFRAC2_SSMOD5_LOCKLOSS_IPCFG_0	ESM0_LVL_IN_5
PLL6 (GPU PLL)	PLLFRAC2_SSMOD6_LOCKLOSS_IPCFG_0	ESM0_LVL_IN_6
PLL7 (C7x PLL)	PLLFRAC2_SSMOD7_LOCKLOSS_IPCFG_0	ESM0_LVL_IN_7
PLL8 (ARM0 PLL)	PLLFRAC2_SSMOD8_LOCKLOSS_IPCFG_0	ESM0_LVL_IN_8
PLL12 (DDR PLL)	PLLFRAC2_SSMOD12_LOCKLOSS_IPCFG_0	ESM0_LVL_IN_12
PLL13 (C66 PLL)	PLLFRAC2_SSMOD13_LOCKLOSS_IPCFG_0	ESM0_LVL_IN_13
PLL14 (R5FSS PLL)	PLLFRAC2_SSMOD14_LOCKLOSS_IPCFG_0	ESM0_LVL_IN_14
PLL15 (AUDIO1 PLL)	PLLFRAC2_SSMOD15_LOCKLOSS_IPCFG_0	ESM0_LVL_IN_15
PLL16 (DSS PLL0)	PLLFRAC2_SSMOD16_LOCKLOSS_IPCFG_0	ESM0_LVL_IN_16
PLL17 (DSS PLL1)	PLLFRAC2_SSMOD17_LOCKLOSS_IPCFG_0	ESM0_LVL_IN_17
PLL18 (DSS PLL2)	PLLFRAC2_SSMOD18_LOCKLOSS_IPCFG_0	ESM0_LVL_IN_18
PLL19 (DSS PLL3)	PLLFRAC2_SSMOD19_LOCKLOSS_IPCFG_0	ESM0_LVL_IN_19
PLL23 (DSS PLL7)	PLLFRAC2_SSMOD23_LOCKLOSS_IPCFG_0	ESM0_LVL_IN_23
PLL24 (MLB PLL)	PLLDDESKEW24_LOCKLOSS_IPCFG_0	ESM0_LVL_IN_24
PLL25 (VISION PLL)	PLLFRAC2_SSMOD25_LOCKLOSS_IPCFG_0	ESM0_LVL_IN_25

#### 5.4.5.4.1.2.5 HSDIVIDER

A PLL may contain up to 16 HSDIVIDER modules to produce more clocks with divided ratio based on the PLL synthesized clock frequency. HSDIVIDER provides only one output. The HSDIVIDER output clock frequency is given by the equations in Table 5-113.

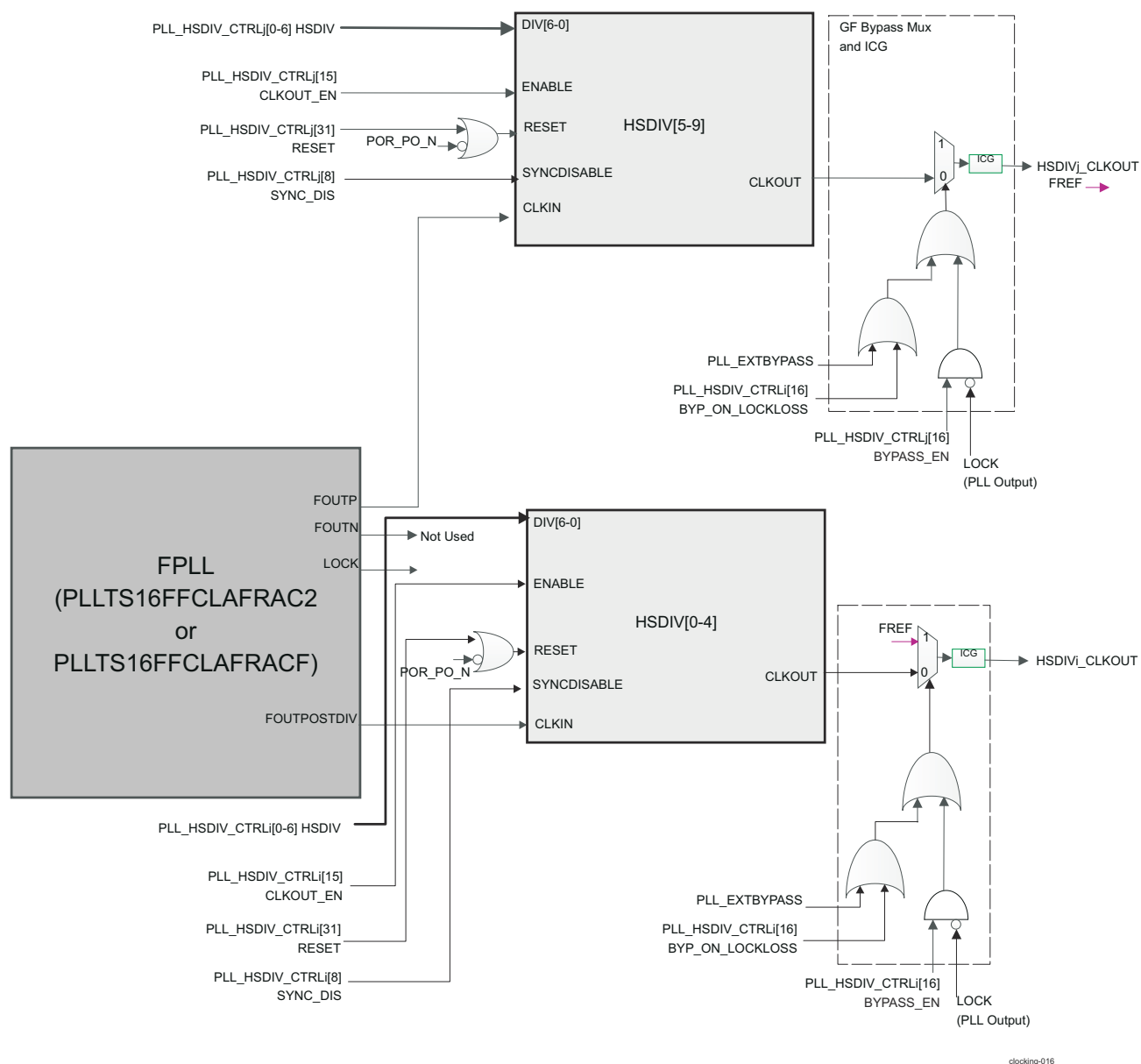
**Table 5-113. HSDIV\_CLKOUT Frequency With PLL State**

Equation	PLL Mode
$HSDIV\_CLKOUT = FOUTP / (HSDIV + 1)$	Locked
$HSDIV\_CLKOUT = FREF$	Before lock or during reload

Where:

- FOUTP is the PLL lock frequency. In case of PLLTS16FFCLVDESKEWC PLL, FOUTP is substituted with FOUT1X.
- HSDIV is the software-configured division ratio binary value.

Figure 5-42 and Figure 5-43 describe the connection between a HSDIV and a specific type of PLL. Signals related only to this connections are shown in both figures.



(1)<sub>i</sub> = 0 to 4; <sub>j</sub> = 5 to 9

**Figure 5-42. PLLTS16FFCLAFRAC2 / PLLTS16FFCLAFRACF and HSDIV Connection**

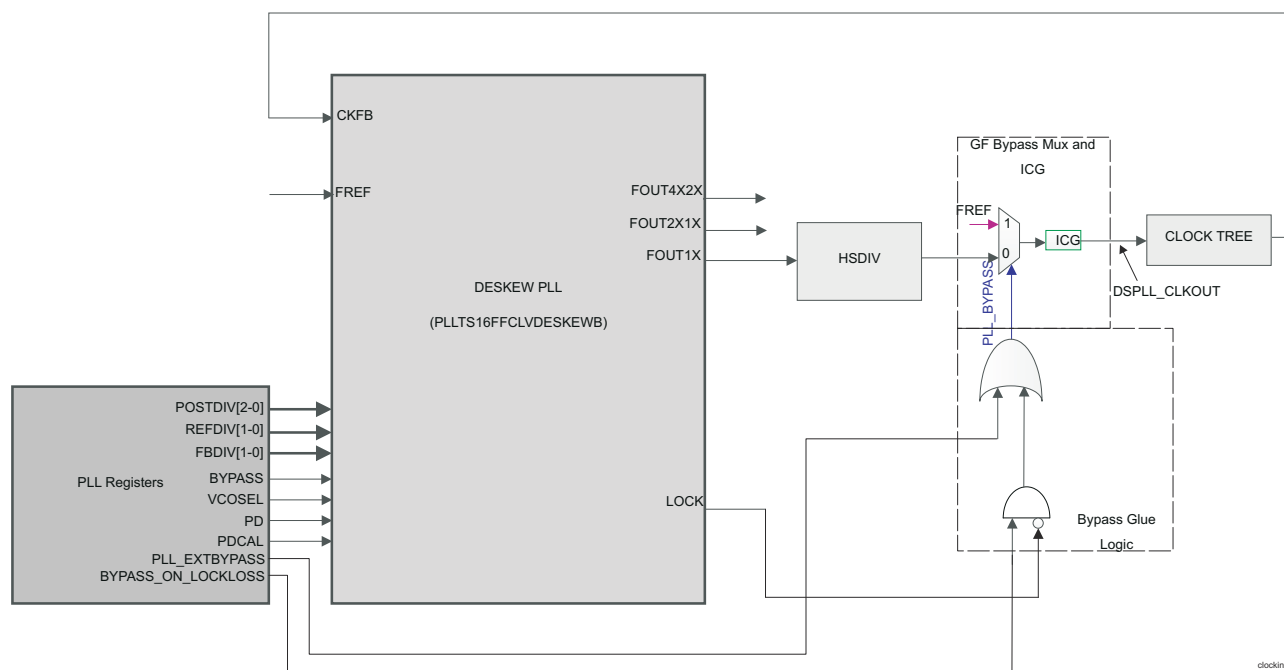


Figure 5-43. PLLTS16FFCLVDESKEWC and HSDIV Connection

#### 5.4.5.4.1.2.6 ICG Module

ICG module ensures that when a particular HSDIV is not enabled the HSDIV clock output is gated off. This is a hardware module with no software control.

#### 5.4.5.4.1.2.7 PLL Power Down

PLLTS16FFCLVDESKEWC has a power down feature. The PLL can be powered down via DSPLL24\_CTRL[4] PD\_EN.

PLLTS16FFCLAFRAC2 and PLLTS16FFCLAFRACF are disabled with MCU\_PLLn\_CTRL[15] PLL\_EN or PLLn\_CTRL[15] PLL\_EN.

#### 5.4.5.4.1.2.8 PLL Calibration

PLLTS16FFCLAFRACF and PLLTS16FFCLVDESKEWC have calibration feature. The calibration settings are done via <PLL\_Name>\_CAL\_CTRL register. The calibration status can be monitored by reading <PLL\_Name>\_CAL\_STAT register. For more information about device-specific information about calibration feature, see [Table 5-121](#).

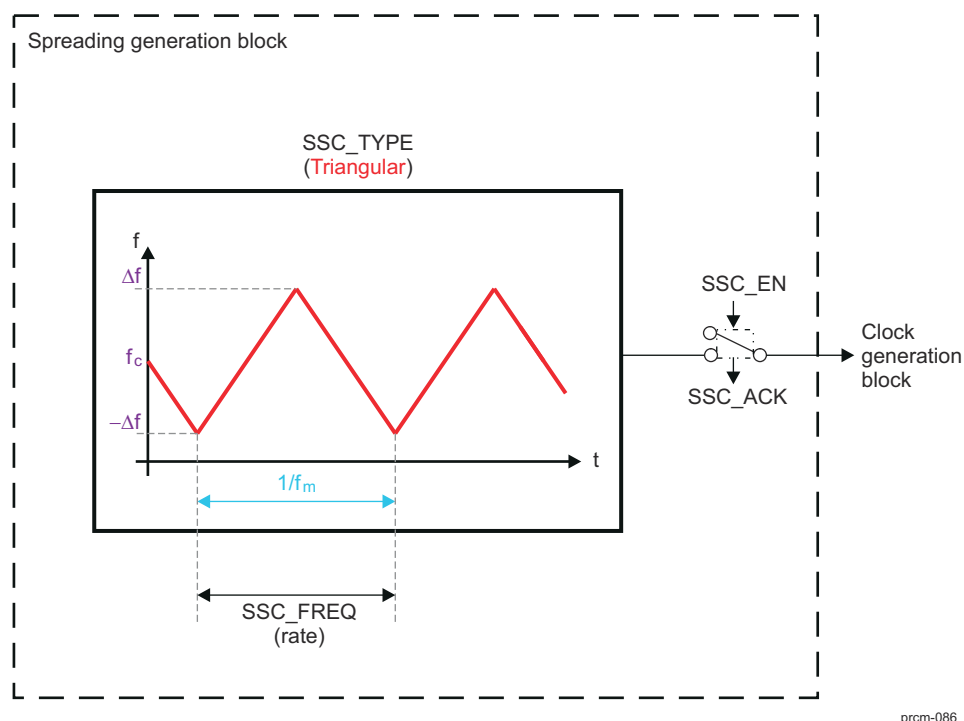
#### 5.4.5.4.2 PLL Spread Spectrum Modulation Module

Spread Spectrum Modulation (SSMOD) modules on the device reduces Electro Magnetic Interference (EMI). It is a fully-digital circuit, used to modulate the frequency of the selected fractional PLLs. Programming options include selection of center spread or down spread, modulation depth, and modulation shape. The default modulation profile is triangular.

All PLLs in this family of device have SSMOD module, except for PLL24, which is the only PLL of PLLTS16FFCLVDESKEWC type.

SSMOD in the PLL is performed by changing the feedback divider (FRAC and FBDIV) in a triangular pattern. This varies the frequency of the output clock in a triangular pattern. The frequency of this pattern is the modulation frequency ( $f_m$ ). It is programmed as a ratio of the CLKIN/4.

[Figure 5-44](#) shows a diagram of the spreading generation block.



prcm-086

**Figure 5-44. Spreading Generation Block Diagram**

**Note**

$f_c$  is the original output clock frequency.

$f_m$  is the spreading frequency.

This additional block generates the required waveform used to reduce EMI. This waveform is then modulated with the initial signal to add some controlled deviation to the clock signal frequency, which spreads the energy of the clock and its harmonics into a band of frequencies, and then reduces EMI. SSMOD\_DOWNSPREAD can control the position of the generated signal. It is controlled by the <PLL\_name>\_CTRL[4] DOWNSPREAD\_EN bit of the corresponding registers. If the DOWNSPREAD\_EN bit is set to 1, the frequency spread on the lower side is twice the programmed value. The frequency spread on the higher side is 0.

The value of SSMOD\_FREQ controls the rate of the generated signal. It can be programmed as a ratio of the reference clock:  $f_{ref} / 4$ . The value that must be programmed is calculated as follows:  
ModFreqDivider =  $f_{ref} / (4 \times f_m)$ , where  $f_m < f_{ref} / 70$ ,  $f_{ref} = f_{inp} / (1+N)$ , and  $f_{inp}$  is the input clock for the PLL.

**5.4.5.4.2.1 Definition of SSMOD**

The aim of the SSMOD is to add a variation to the frequency of an original clock, which spreads the generated interference over a larger band of frequencies.

In theory, SSMOD means that the clock signal is varied around the desired frequency. For example, for a 1 GHz clock, the frequency might be 999.5 MHz at one time and 1.0005 GHz at another. Doing this constantly causes the power of the tone to be spread out more over a broader band of tight frequencies (centered at the desired tone). To realize this constant variation on the original signal, a modulation with an additional signal (called spreading waveform) is realized.

Creating an SSMOD by spreading the initial clock frequency is done by defining the following parameters:

- The spreading frequency (deviation), which is the ratio of the range of spreading frequency over the original clock frequency.



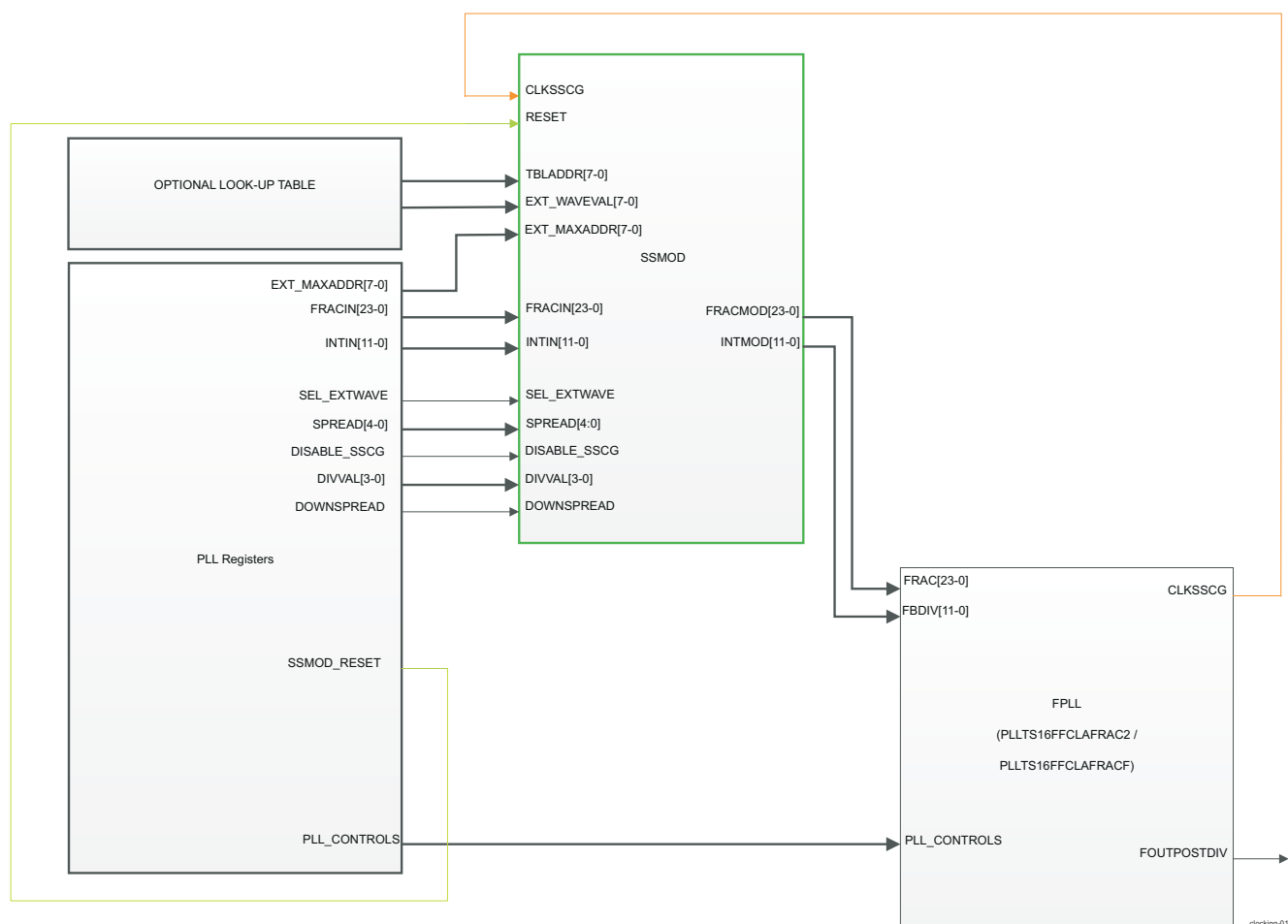
#### 5.4.5.4.2.2 SSMOD Configuration

Table 5-114 describes the PLL SSMOD control bitfields. Figure 5-45 describes the connection between a SSMOD and a PLL.

**Table 5-114. SSMOD related bitfields**

Parameter	Register	Description
SSMOD enable control	<PLL_name>n_SS_CTRL[31] <sup>(1)</sup> BYPASS_EN (For example, MCU_PLL0 - MCU_PLL0_SS_CTRL[31] BYPASS_EN)	Enable/disable the PLL SSMOD feature.
Wave table maximum address	<PLL_name>n_SS_CTRL[25-18] <sup>(1)</sup> WV_TBL_MAXADDR (For example, MCU_PLL0 - MCU_PLL0_SS_CTRL[25-18] WV_TBL_MAXADDR)	Wave table maximum address. Indicates the maximum number of address bits used to access the external wave table. These bits are not used if <PLL_name>n_SS_CTRL[0] WAVE_SEL = 0
Type of spread	<PLL_name>n_SS_CTRL[4] <sup>(1)</sup> DOWNSPREAD_EN (For example, MCU_PLL0 - MCU_PLL0_SS_CTRL[4] DOWNSPREAD_EN)	Selects center spread or down spread clock variance
Wave table selection	<PLL_name>n_SS_CTRL[0] <sup>(1)</sup> WAVE_SEL (For example, MCU_PLL0 - MCU_PLL0_SS_CTRL[0] WAVE_SEL)	Wave pattern select External wave table should only be selected
Modulation divider	<PLL_name>n_SS_SPREAD[19-16] <sup>(1)</sup> MOD_DIV (For example, MCU_PLL0 - MCU_PLL0_SS_SPREAD[19-16] MOD_DIV)	Input clock divider. This divider sets the modulation frequency.
Spread Depth	<PLL_name>n_SS_SPREAD[4-0] <sup>(1)</sup> SPREAD (For example, MCU_PLL0 - MCU_PLL0_SS_SPREAD[4-0] SPREAD)	Sets the spread modulation depth.

(1) n = 0, 1, 2 for MCU domain and n = 0 - 8, 12 - 19, 23, and 25 for MAIN domain



**Figure 5-45. PLL and SSMOD Connection**

### 5.4.5.5 PLLs Device-Specific Information

Table 5-115 lists the basic information of each of the three MCU PLLs.

**Table 5-115. Basic Information of MCU\_PLL0, MCU\_PLL1, and MCU\_PLL2**

PLL name	Type	SSMOD Available	HSDIV Available	Input clock	Comments
MCU_PLL0	PLLTS16FFCLAFRAC2	Yes, see <a href="#">Table 5-114</a> for more information on SSMD related bitfields/bits	HSDIV0 and HSDIV1	WKUP_HFOSC0_CLKOUT	See <a href="#">(1)</a> , <a href="#">(2)</a> , and <a href="#">(3)</a>
MCU_PLL1	PLLTS16FFCLAFRAC2	Yes, see <a href="#">Table 5-114</a> for more information on SSMD related bitfields/bits	HSDIV0, HSDIV1, HSDIV2, HSDIV3, and HSDIV4	WKUP_HFOSC0_CLKOUT	See <a href="#">(1)</a> , <a href="#">(2)</a> , and <a href="#">(3)</a>
MCU_PLL2	PLLTS16FFCLAFRAC2	Yes, see <a href="#">Table 5-114</a> for more information on SSMD related bitfields/bits	HSDIV0, HSDIV1, HSDIV2, HSDIV3, and HSDIV4	WKUP_HFOSC0_CLKOUT	See <a href="#">(1)</a> , <a href="#">(2)</a> , and <a href="#">(3)</a>

(1) See *Generic PLL Functional Diagram for PLLTS16FFCLAFRAC2 type* for overview of the PLL.

(2) See [Section 5.4.5.5.2](#) for synthesised clock parameters.

(3) See [Section 5.4.5.5.3](#) for clock output parameters.

Table 5-116 lists the basic information of each of the MAIN PLLs.

**Table 5-116. Basic Information of MAIN Domain PLLs**

PLL name	Type	SSMOD Available	HSDIV Available	Input clock	Comments
PLL0	PLLTS16FFCLAFRAC2	Yes, see <a href="#">Table 5-114</a> for more information on SSMD related bitfields/bits	HSDIV0, HSDIV1, HSDIV2, HSDIV3, HSDIV4, HSDIV5, HSDIV6, HSDIV7, and HSDIV8	MAIN_PLL0_REF_CLK	See <a href="#">(1)</a> , <a href="#">(2)</a> , and <a href="#">(3)</a>
PLL1	PLLTS16FFCLAFRAC2	Yes, see <a href="#">Table 5-114</a> for more information on SSMD related bitfields/bits	HSDIV0, HSDIV1, HSDIV2, HSDIV3, HSDIV5, HSDIV6, HSDIV7, and HSDIV8	MAIN_PLL1_REF_CLK	See <a href="#">(1)</a> , <a href="#">(2)</a> , and <a href="#">(3)</a>
PLL2	PLLTS16FFCLAFRAC2	Yes, see <a href="#">Table 5-114</a> for more information on SSMD related bitfields/bits	HSDIV0, HSDIV1, HSDIV2, HSDIV3, HSDIV4, HSDIV5, HSDIV6, and HSDIV7	MAIN_PLL2_REF_CLK	See <a href="#">(1)</a> , <a href="#">(2)</a> , and <a href="#">(3)</a>
PLL3	PLLTS16FFCLAFRAC2	Yes, see <a href="#">Table 5-114</a> for more information on SSMD related bitfields/bits	HSDIV0, HSDIV1, HSDIV2, HSDIV3, and HSDIV4	MAIN_PLL3_REF_CLK	See <a href="#">(1)</a> , <a href="#">(2)</a> , and <a href="#">(3)</a>
PLL4	PLLTS16FFCLAFRAC2	Yes, see <a href="#">Table 5-114</a> for more information on SSMD related bitfields/bits	HSDIV0, HSDIV1, HSDIV2, and HSDIV3	MAIN_PLL4_REF_CLK	See <a href="#">(1)</a> , <a href="#">(2)</a> , and <a href="#">(3)</a>
PLL5	PLLTS16FFCLAFRAC2	Yes, see <a href="#">Table 5-114</a> for more information on SSMD related bitfields/bits	HSDIV0 and HSDIV1	MAIN_PLL5_REF_CLK	See <a href="#">(1)</a> , <a href="#">(2)</a> , and <a href="#">(3)</a>
PLL6	PLLTS16FFCLAFRAC2	Yes, see <a href="#">Table 5-114</a> for more information on SSMD related bitfields/bits	HSDIV0	MAIN_PLL6_REF_CLK	See <a href="#">(1)</a> , <a href="#">(2)</a> , and <a href="#">(3)</a>
PLL7	PLLTS16FFCLAFRAC2	Yes, see <a href="#">Table 5-114</a> for more information on SSMD related bitfields/bits	HSDIV0	MAIN_PLL7_REF_CLK	See <a href="#">(1)</a> , <a href="#">(2)</a> , and <a href="#">(3)</a>

**Table 5-116. Basic Information of MAIN Domain PLLs (continued)**

PLL name	Type	SSMOD Available	HSDIV Available	Input clock	Comments
PLL8	PLLTS16FFCLAFRAC2	Yes, see <a href="#">Table 5-114</a> for more information on SSMOD related bitfields/bits	HSDIV0	MAIN_PLL8_REF_CLK	See <a href="#">(1)</a> , <a href="#">(2)</a> , and <a href="#">(3)</a>
PLL12	PLLTS16FFCLAFRAC2	Yes, see <a href="#">Table 5-114</a> for more information on SSMOD related bitfields/bits	HSDIV0	MAIN_PLL12_REF_CLK	See <a href="#">(4)</a> , <a href="#">(2)</a> , and <a href="#">(3)</a>
PLL13	PLLTS16FFCLAFRAC2	Yes, see <a href="#">Table 5-114</a> for more information on SSMOD related bitfields/bits	HSDIV0 and HSDIV1	MAIN_PLL13_REF_CLK	See <a href="#">(1)</a> , <a href="#">(2)</a> , and <a href="#">(3)</a>
PLL14	PLLTS16FFCLAFRAC2	Yes, see <a href="#">Table 5-114</a> for more information on SSMOD related bitfields/bits	HSDIV0 and HSDIV1	MAIN_PLL14_REF_CLK	See <a href="#">(1)</a> , <a href="#">(2)</a> , and <a href="#">(3)</a>
PLL15	PLLTS16FFCLAFRAC2	Yes, see <a href="#">Table 5-114</a> for more information on SSMOD related bitfields/bits	HSDIV0, HSDIV1, HSDIV2, and HSDIV3	MAIN_PLL15_REF_CLK	See <a href="#">(1)</a> , <a href="#">(2)</a> , and <a href="#">(3)</a>
PLL16	PLLTS16FFCLAFRAC2	Yes, see <a href="#">Table 5-114</a> for more information on SSMOD related bitfields/bits	HSDIV0 and HSDIV1	MAIN_PLL16_REF_CLK	See <a href="#">(1)</a> , <a href="#">(2)</a> , and <a href="#">(3)</a>
PLL17	PLLTS16FFCLAFRAC2	Yes, see <a href="#">Table 5-114</a> for more information on SSMOD related bitfields/bits	HSDIV0 and HSDIV1	MAIN_PLL17_REF_CLK	See <a href="#">(1)</a> , <a href="#">(2)</a> , and <a href="#">(3)</a>
PLL18	PLLTS16FFCLAFRAC2	Yes, see <a href="#">Table 5-114</a> for more information on SSMOD related bitfields/bits	HSDIV0 and HSDIV1	MAIN_PLL18_REF_CLK	See <a href="#">(1)</a> , <a href="#">(2)</a> , and <a href="#">(3)</a>
PLL19	PLLTS16FFCLAFRAC2	Yes, see <a href="#">Table 5-114</a> for more information on SSMOD related bitfields/bits	HSDIV0	MAIN_PLL19_REF_CLK	See <a href="#">(1)</a> , <a href="#">(2)</a> , and <a href="#">(3)</a>
PLL23	PLLTS16FFCLAFRAC2	Yes, see <a href="#">Table 5-114</a> for more information on SSMOD related bitfields/bits	HSDIV0	MAIN_PLL23_REF_CLK	See <a href="#">(1)</a> , <a href="#">(2)</a> , and <a href="#">(3)</a>
PLL24	PLLTS16FFCLAFRAC2	No	HSDIV0	MAIN_PLL24_REF_CLK MAIN_PLL24 HSDIV0_CLK_OUT	See <a href="#">(5)</a> , <a href="#">(2)</a> , and <a href="#">(3)</a>
PLL25	PLLTS16FFCLAFRAC2	Yes, see <a href="#">Table 5-114</a> for more information on SSMOD related bitfields/bits	HSDIV0 and HSDIV1	MAIN_PLL25_REF_CLK	See <a href="#">(1)</a> , <a href="#">(2)</a> , and <a href="#">(3)</a>

- (1) See *Generic PLL Functional Diagram for PLLTS16FFCLAFRAC2 type* for overview of the PLL.
- (2) See [Section 5.4.5.5.2](#) for synthesized clock parameters.
- (3) See [Section 5.4.5.5.3](#) for clock output parameters.
- (4) See [Figure 5-40](#), *Generic PLL Functional Diagram for PLLTS16FFCLAFRAC2F type* for overview of the PLL.
- (5) See *Generic PLL Functional Diagram for PLLTS16FFCLVDESKEWC type* for overview of the PLL.

#### 5.4.5.5.1 SSMOD Related Bitfields Table

[Table 5-114](#) lists SSMOD related bitfields for all three types of PLLs.

#### 5.4.5.5.2 Clock Synthesis Inputs to the PLLs

Table 5-117 lists the clock synthesis parameters for PLLTS16FFCLAFRAC2 and PLLTS16FFCLAFRACF types.

**Table 5-117. Clock synthesis Parameters for PLLTS16FFCLAFRAC2 and PLLTS16FFCLAFRACF Types**

Parameter Name	Register
FBDIV	<PLL_name>_FREQ_CTRL0[11-0] FB_DIV_INT (For example, MCU_PLL0 - MCU_PLL0_FREQ_CTRL0[11-0] FB_DIV_INT)
FRAC	<PLL_name>_FREQ_CTRL1[23-0] FB_DIV_FRAC (For example, MCU_PLL0 - MCU_PLL0_FREQ_CTRL1[23-0] FB_DIV_FRAC)
POSTDIV2	<PLL_name>_DIV_CTRL[26-24] POST_DIV2 (For example, MCU_PLL0 - MCU_PLL0_DIV_CTRL[26-24] POST_DIV2)
POSTDIV1	<PLL_name>_DIV_CTRL[18-16] POST_DIV1 (For example, MCU_PLL0 - MCU_PLL0_DIV_CTRL[18-16] POST_DIV1)
REFDIV	<PLL_name>_DIV_CTRL[5-0] REF_DIV (For example, MCU_PLL0 - MCU_PLL0_DIV_CTRL[5-0] REF_DIV)

Table 5-118 lists the clock synthesis parameters for PLLTS16FFCLVDESKEWC type.

#### Note

For PLLTS16FFCLAFRAC2 and PLLTS16FFCLAFRACF types - POSTDIV1 and POSTDIV2 values are from 1 to 7. To ensure correct operation, POSTDIV1 must always be programmed to a value equal to or greater than POSTDIV2.

**Table 5-118. Clock synthesis Parameters for PLLTS16FFCLVDESKEWC Type**

Parameter Name	Register
FBDIV	<PLL_name>_DIV_CTRL[13-12] FB_DIV (For example, PLL24 - PLL24_PLL_DIV_CTRL[13-12] FB_DIV)
POSTDIV	<PLL_name>_DIV_CTRL[10-8] POST_DIV (For example, PLL24 - PLL24_PLL_DIV_CTRL[10-8] POST_DIV)
REFDIV	<PLL_name>_DIV_CTRL[1-0] REF_DIV (For example, PLL24 - PLL24_PLL_DIV_CTRL[1-0] REF_DIV)

#### Note

For PLLTS16FFCLVDESKEWC type:

- REFDIV and FBDIV valid values are 1, 2, and 4.
- POSTDIV valid values are 1, 2, 4, 8, 16, 32, 64, and 128.

#### 5.4.5.5.3 Clock Output Parameter

Table 5-119 lists the control/status bit fields for output clocks.

**Table 5-119. Clock Output Parameter for PLLTS16FFCLAFRAC2 and PLLTS16FFCLAFRACF Types**

Clock Output/Divider	Parameter Name	Control/Status Bit Field
<PLL_name>_CLKOUT, <PLL_name>_HSDIV_CLKOUT <sup>(1)</sup>	Control	<PLL_name>_CTRL[31] BYPASS_EN (For example, MCU_PLL0 - MCU_PLL0_CTRL[31] BYPASS_EN)

**Table 5-119. Clock Output Parameter for PLLTS16FFCLAFRAC2 and PLLTS16FFCLAFRACF Types (continued)**

Clock Output/Divider	Parameter Name	Control/Status Bit Field
<PLL_name>_HSDIVj_CLKOUT <sup>(1)</sup>	Control	<PLL_name>_HSDIV_CTRLj[15] CLKOUT_EN (For example, MCU_PLL0 - MCU_PLL0_HSDIV_CTRL0[5] CLKOUT_EN and MCU_PLL0_HSDIV_CTRL1[5] CLKOUT_EN)
<PLL_name>_HSDIVj_CLKOUT <sup>(1)</sup>	Divider control	<PLL_name>_HSDIV_CTRLj[6-0] HSDIV (For example, MCU_PLL0 - MCU_PLL0_HSDIV_CTRL0[6-0] HSDIV and MCU_PLL0_HSDIV_CTRL1[6-0] HSDIV)
PLL bypass mux	Control	<PLL_name>_CTRL[16] BYP_ON_LOCKLOSS (For example, MCU_PLL0 - MCU_PLL0_CTRL[16] BYP_ON_LOCKLOSS)
Enable 4-phase clock generator	Control	<PLL_name>_CTRL[5] CLK_4PH_EN (ignored if <PLL_name>_CTRL[4] CLK_POSTDIV_EN = 0) (For example, MCU_PLL0 - MCU_PLL0_CTRL[5] CLK_4PH_EN)
Post divide CLK enable	Control	<PLL_name>_CTRL[4] CLK_POSTDIV_EN (For example, MCU_PLL0 - MCU_PLL0_CTRL[4] CLK_POSTDIV_EN)
Delta-Sigma modulator enable	Control	<PLL_name>_CTRL[1] DSM_EN (For example, MCU_PLL0 - MCU_PLL0_CTRL[1] DSM_EN)
Enable fractional noise canceling DAC	Control	<PLL_name>_CTRL[0] DAC_EN (For example, MCU_PLL0 - MCU_PLL0_CTRL[0] DAC_EN)

(1) j is the number of HSDIV for the current PLL.

**Table 5-120. Clock Output Parameter for PLLTS16FFCLVDESKEWC Type**

Clock Output/Divider	Parameter Name	Control/Status Bit Field
<PLL_name>_CLK_OUT, <PLL_name>_HSDIVj_CLK_OUT	Control	<PLL_name>_CTRL[31] BYPASS_EN (For example, PLL24 - PLL24_CTRL[31] BYPASS_EN)

#### 5.4.5.5.4 Calibration Related Bitfields

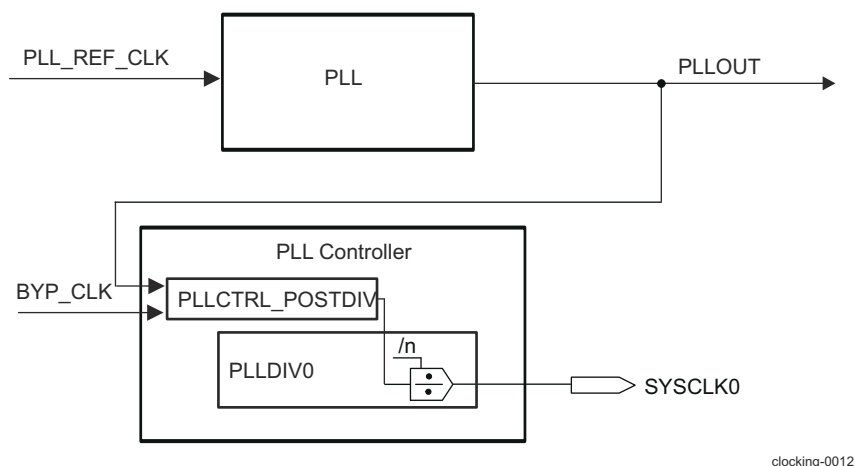
Table 5-121 lists the calibration related bitfields/bits.

**Table 5-121. Recalibration Feature Parameters**

Recalibration feature	Parameter Name	Control/Status Bit Field
Calibration enable to actively adjust for input skew	Control	<PLL_name>_CAL_CTRL[31] CAL_EN (For example, PLL12 - PLL12_CAL_CTRL[31] CAL_EN)
Fast calibration enabled	Control	<PLL_name>_CAL_CTRL[20] FAST_CAL (For example, PLL12 - PLL12_CAL_CTRL[20] FAST_CAL)
Calibration loop programmable counter	Control	<PLL_name>_CAL_CTRL[18-16] CAL_CNT (For example, PLL12 - PLL12_CAL_CTRL[18-16] CAL_CNT)
Calibration bypass	Control	<PLL_name>_CAL_CTRL[15] CAL_BYP (For example, PLL12 - PLL12_CAL_CTRL[15] CAL_BYP)
Calibration input	Control	<PLL_name>_CAL_CTRL[11-0] CAL_IN (For example, PLL12 - PLL12_CAL_CTRL[11-0] CAL_IN)
Output of the calibration block	Status	<PLL_name>_CAL_STAT[11-0] CAL_OUT (For example, PLL12 - PLL12_CAL_STAT[11-0] CAL_OUT)

#### 5.4.5.6 PLL and PLL Controller Connection

Two PLL Controllers are implemented in the device - PLLCTRL0 and WKUP\_PLL\_CTRL0. They are respectively related to PLL0 and MCU\_PLL0. The PLL Controllers manage the clock ratios, alignment, and gating for the system clocks.



**Figure 5-46. PLL and PLL Controller**

#### Note

PLLCTRL\_POSTDIV is not supported in this family of devices.

#### Note

The PLL Controllers registers can be accessed by any master in the device.

A SYSCLK0 clock out of a PLLCTRL is the only synchronous clock in that domain. That means  
- MCU\_SYSCLK0 out of WKUP\_PLLCTRL is the synchronous CBASS clock in MCU domain and  
SYSCLK0 out of Main PLLCTRL is the synchronous CBASS clock in Main domain.

#### 5.4.5.7 System Clocks Operating Frequency Ranges

Table 5-122 lists the operating frequency ranges for the system clocks of the device.

**Table 5-122. System Clocks Operating Frequency Range**

System Clocks	Minimum Operating Frequency (MHz)
MCU_PLL0 (MCU TBD PLL) with WKUP_PLLCTRL0	TBD
MCU_PLL1 (MCU PERIPHERAL PLL)	TBD
MCU_PLL2 (MCU CPSW PLL)	TBD
PLL0 (MAIN PLL) with PLLCTRL0	TBD
PLL1 (PER0 PLL)	TBD
PLL2 (PER1 PLL)	TBD
PLL3 (CPSW9G PLL)	TBD
PLL4 (AUDIO0 PLL)	TBD
PLL5 (VIDEO PLL)	TBD
PLL6 (GPU PLL)	TBD
PLL7 (C7x PLL)	TBD
PLL8 (ARM0 PLL)	TBD
PLL12 (DDR PLL)	TBD



**Table 5-122. System Clocks Operating Frequency Range (continued)**

System Clocks	Minimum Operating Frequency (MHz)
PLL13 (C66 PLL)	TBD
PLL14 (PULSAR PLL)	TBD
PLL15 (AUDIO1 PLL)	TBD
PLL16 (DSS PLL0)	TBD
PLL17 (DSS PLL1)	TBD
PLL18 (DSS PLL2)	TBD
PLL19 (DSS PLL3)	TBD
PLL23 (DSS PLL7)	TBD
PLL24 (MLB PLL)	TBD
PLL25 (VISION PLL)	TBD

(1) Supported input reference clock frequencies to the PLL are 19.2/24/25/26 MHz only.

(2) Interconnect clock on DSS is CPU/4. This will range from 100 MHz to 250 MHz.

(3) When Main PLL is configured to 400 MHz mode, DSS can only support a max pixel clock of 74.25 MHz. For lower resolution displays the DSS clock can be lower than 74.25 MHz.

#### 5.4.5.8 Recommended Clock and Control Signal Transition Behavior

All clocks and strobe signals must transition between  $V_{IH}$  and  $V_{IL}$  (or between  $V_{IL}$  and  $V_{IH}$ ) in a monotonic manner. Monotonic transitions are more easily ensured with faster switching signals. Slower input transitions are more susceptible to glitches due to noise, and special care must be taken for slow input clocks.

#### 5.4.5.9 Interface Clock Specifications

##### Interface Clock Terminology

The interface clock is used at the system level to sequence the data and to control transfers accordingly with the interface protocol.

##### Interface Clock Frequency

The two interface clock characteristics are:

- The maximum clock frequency
- The maximum operating frequency

The interface clock frequency documented here is the maximum clock frequency, which corresponds to the maximum frequency programmable on this output clock. This frequency defines the maximum limit supported by the Device IC and does not take into account any system consideration (PCB, peripherals).

The system designer must take into account these system considerations and the Device IC timing characteristics to properly define the maximum operating frequency that corresponds to the maximum frequency supported to transfer the data on this interface.

#### 5.4.5.10 PLL, PLLCTRL, and HSDIV Controllers Programming Guide

##### 5.4.5.10.1 PLL Initialization

##### 5.4.5.10.1.1 Kick Protection Mechanism

#### Note

PLL configuration registers are write-protected at power-up. Software must first un-lock the PLLn\_LOCKKEY0 and PLLn\_LOCKKEY1 registers prior to writing to any chip-level registers.

The un-lock process shall follow the steps:

1. Write value 0x68EF3490 to PLLn\_LOCKKEY0 register.
2. Write value 0xD172BC5A to PLLn\_LOCKKEY1 register.

After these two steps a write access to the PLL registers is allowed. Writing any other data value to either of these two registers locks the kicker mechanism and blocks any writes to the PLL registers.

---

#### Note

In order to ensure that all PLL registers are write protected, software must always re-lock the kicker mechanism after completing the register writes.

---

#### 5.4.5.10.1.2 PLL Initialization to PLL Mode

---

#### Note

For validated set of parameters that can be programmed to PLLs, please refer to the device-specific Datasheet.

---

Initially, the device powers up in PLL bypass mode. The bypass mux is a glitch free mux and is located outside the PLL module. PLL\_EXTBYPASS bit controls the bypass mux section. During Power-up, the bypass mux defaults to select FREF clock (PLL clock bypass mode).

During boot, WKUP\_DMSC0 ROM programs the MCU\_PLL0 to a valid frequency based on the crystal frequency and BOOTMODE pin settings. WKUP\_DMSC0 then waits for PLL to be locked by checking the PLL LOCK status bit. WKUP\_DMSC0 ROM programs PLL\_EXTBYPASS bit to '0' to disable the bypass mode to propagate MCU\_PLL0 clock to WKUP\_PLLCTRL0. At this point WKUP\_DMSC0 brings MCU\_R5FSS out of reset to complete the reset of the boot process. R5FSS software configures the remaining PLLs and bring them out of bypass mode.

The following shows the PLL initialization sequence.

1. Wait until supplies are stable and PORz is de-asserted. By this time the reference clock source must be up and running.
2. Default controls settings:
  - By default PLEN signal is Low provided by <PLL\_name>\_CTRL register default state. This signal behaves like a reset control to the PLL.
  - By default PLL\_INTBYPASS signal is low provided by <PLL\_name>\_CTRL register default state.
  - By default PLL\_EXTBYPASS signal is high provided by PLL MMR CTRL default state. PLL clock will be bypassed externally by a glitch free mux. The output of the mux will be the reference clock FREF.
  - LOCK output of PLL will be Low.
3. Program the PLL to a valid setting that runs the VCO within the specified range. The following bits/bitfields must be programmed appropriately by software: DSMEN, DACEN, FOUTPOSTDIVEN, FOUT4PHASEEN, REFDIV[5-0], FBDIV[11-0], FRAC[23-0], POSTDIV1[2-0], POSTDIV2[2-0].
4. Wait for 1  $\mu$ s to allow the PLL internal reset to complete (PLL powerdown switch pulls the loop filter voltage from rail-to-rail, which ensures the PLL will be completely powered down from any state).
5. Assert PLEN to a High (by writing a '1' into PLEN bit in <PLL\_name>\_CTRL register).
6. Wait for PLL Lock output to go high. Software can read the LOCK bit in <PLL\_name>\_STATS register to check if PLL has locked.
7. De-assert PLL\_EXTBYPASS signal to a Low by writing 0 into BYPASS\_EN bit in <PLL\_name>\_CTRL register.
8. Glitch free mux safely switches to the PLL clock output.

**Changing PLL setting:** During normal operation, before changing PLL settings, the PLL clock must be bypassed by setting PLL\_EXTBYPASS control to a high. Then follow steps 2 to 8 from the above sequence.

#### 5.4.5.10.1.3 PLL Programming Requirements

- A PLL VCO frequency must be set to > 1500 MHz
  - For best PLL performance, maximize VCO frequency where possible.
- A PLL must always operate in Fractional Mode (DACEN and DSMEN set to '1'). This is true even when the FBDIV is a integer.
- A PLL Reference clock frequency must be > 10 MHz.
- REFDIV[5:0] must be set to "000001".
- DESKEWC and FRACF PLLs must always be used with continuous calibration mode turned on and follow the calibration procedure.

#### 5.4.5.10.2 HSDIV PLL Programming

<PLL\_Name>\_HSDIV\_CTRLK[15] CLKOUT\_EN = 0 cleanly disables the HSDIV output clock

<PLL\_Name>\_HSDIV\_CTRLK[6:0] HSDIV = <value> for divider

<PLL\_Name>\_HSDIV\_CTRLK[8] SYNC\_DIS = 0 insures that the divider changes occur synchronously to the internal divider of the HSDIV block.

when the clock is required,

<PLL\_Name>\_HSDIV\_CTRLK[15] CLKOUT\_EN = 1 cleanly enables the HSDIV output clock

#### 5.4.5.10.3 PLL Controllers Programming - Dividers PLLDIVn and GO Operation

The GO operation is required to change the divider ratios of the PLLDIVn PLLDIVn registers. [Section 5.4.5.10.3.1, GO Operation](#), discusses the GO operation. [Section 5.4.5.10.3.2, Software Steps to Modify PLLDIVn Ratios](#), provides the software steps required to change the divider ratios.

##### 5.4.5.10.3.1 GO Operation

The GO operation writes to the RATIO field in the PLLDIVn PLLDIVn. Registers do not change the dividers' divide ratios immediately. The PLLDIV dividers change to the new RATIO rates only during a GO operation. This section discusses the GO operation and alignment of the SYSCLKs.

The PLL Controller clock align control register (ALNCTL) determines which SYSCLKs must be aligned. Before a GO operation, program ALNCTL so that the appropriate clocks are aligned during the GO operation.

A GO operation is initiated by setting the GOSET bit in PLLCMD to 1. During a GO operation:

- Any SYSCLK *n* with the corresponding ALN *n* bit in ALNCTL and SYS *n* bit in DCHANGE set to 1 is paused at the low edge. Then the PLL Controller restarts all these SYSCLKs simultaneously, aligned at the rising edge. When the SYSCLKs are restarted, SYSCLK *n* toggles at the rate programmed in the RATIO field in PLLDIVn PLLDIVn.
- Any SYSCLK *n* with the corresponding ALN *n* bit in ALNCTL cleared and the SYS *n* bit in DCHANGE set immediately changes to the new rate programmed in the RATIO field.
- The GOSTAT bit in PLLSTAT is set throughout the duration of a GO operation.

#### CAUTION

To help prevent errors, all device operation must be stopped before the GO operation.

#### 5.4.5.10.3.2 Software Steps to Modify PLLDIV Ratios

Perform the following steps to modify PLLDIV.

1. Check that the GOSTAT bit in PLLSTAT is cleared to show that no GO operation is currently in progress.
2. Program the RATIO field in PLLDIVn to the desired new divide-down rate. If the RATIO field changed, the PLL Controller will flag the change in the corresponding bit of DCHANGE.
3. Set the respective ALN *n* bits in ALNCTL to align any SYSCLKs after the GO operation.
4. Set the GOSET bit in PLLCMD to initiate the GO operation to change the divide values and align the SYSCLKs as programmed.

5. Read the GOSTAT bit in PLLSTAT to make sure the bit returns to 0 to indicate that the GO operation has completed.

#### 5.4.5.10.4 Entire Sequence for Programming PLLCTRL, HSDIV, and PLL

Table 5-123 shows the entire sequence of programming PLLCTRL, HSDIV, and PLL from an unknown state to a defined non-spread-spectrum state.

**Table 5-123. Programming Sequence of PLLCTRL, HSDIV, and PLL**

Step	Description
Unlock PLL registers	<PLL_Name>_LOCKKEY0 (= 0x68EF3490) <PLL_Name>_LOCKKEY1 (= 0xD172BC5A)
If PLL0	Configure PLLCTRL block into bypass mode PLLCTL[0] PPLEN = 0 PLLCTL[5] PPLENSRC = 0
Configure external bypass so that no transient clock propagates	<PLL_Name>_CTRL[31] BYPASS_EN = 1
Delay	
Disable HSDIV(s)	<PLL_Name>_HSDIV_CTRLk[15] CLKOUT_EN = 0
Delay	
Disable PLL	<PLL_Name>_CTRL[15] PLL_EN = 0
Delay	
Reset HSDIV(s)	<PLL_Name>_HSDIV_CTRLk[31] RESET = 1
If PLL0	Check PLLSTAT[0] GOSTAT to make sure that divider change is not currently in-progress GOSTAT should equal 0. Clear GOSET bit PLLCMD[0] GOSET = 0 Configure divider in PLLCTRL to /1 PLLDIV1 = 0x8000 (enable divider and divide by 1) PLLDIV2 = 0x00 (disable divider and divide by 1) Set alignment control bits ALNCTL = 3 Set GOSET bit to initiate the divider change PLLCMD[0] GOSET = 1 Check PLLSTAT[0] GOSTAT to make sure that divider change has completed GOSTAT should equal 0.
Configure HSDIV(s) divider value	<PLL_Name>_HSDIV_CTRLk[6-0] HSDIV (=val)
Clear HSDIV(s) SYNC_DIS	<PLL_Name>_HSDIV_CTRLk[8] SYNC_DIS = 0
Delay	
Clear Reset HSDIV(s)	<PLL_Name>_HSDIV_CTRLk[31] RESET = 0
Configure PLL multiplier	Integer portion of divider <PLL_Name>_FREQ_CTRL0[11-0] (=val) Fractional portion of divider <PLL_Name>_FREQ_CTRL1[23-0] (=val)
Configure PLL dividers	Reference clock divider <PLL_Name>_DIV_CTRL[5-0] REF_DIV = 1 Post divider 1 <PLL_Name>_DIV_CTRL[18-16] POST_DIV1 = 1 Post divider 2 <PLL_Name>_DIV_CTRL[26-24] POST_DIV2 = 1
Configure "random" PLL controls	<PLL_Name>_CTRL[8] INTL_BYP_EN = 0 <PLL_Name>_CTRL[5] CLK_4PH_EN = 0 <PLL_Name>_DIV_CTRL[26-24] POST_DIV2 = 1

**Table 5-123. Programming Sequence of PLLCTRL, HSDIV, and PLL (continued)**

Step	Description
	<PLL_Name>_DIV_CTRL[18-16] POST_DIV1 = 1
	<PLL_Name>_DIV_CTRL[5-0] REF_DIV = 1
	<PLL_Name>_CTRL[1] DSM_EN = 1
	<PLL_Name>_CTRL[0] DAC_EN = 1
	<PLL_Name>_SS_CTRL[31] BYPASS_EN (SSMOD) = 1
	<PLL_Name>_CTRL[4] CLK_POSTDIV_EN – if PLL has more than 5 HSDIV blocks, 1; else 0.
	<PLL_Name>_CTRL[16] BYP_ON_LOCKLOSS (= {0, 1})
Delay	
Enable PLL	<PLL_Name>_CTRL[15] PLL_EN = 1
Wait for Lock	Wait for MCU_PLL0_STAT[0] LOCK = 1
Enable HSDIV(s)	<PLL_Name>_HSDIV_CTRLk[15] CLKOUT_EN = 1
Delay	
Configure external bypass to pass PLL	<PLL_Name>_CTRL[31] BYPASS_EN = 0
Delay	
If PLL0	Configure PLLCTRL block into PLL mode PLLCTL[0] PLEN = 0
Lock PLL registers	<PLL_Name>_LOCKKEY0 (= any value) <PLL_Name>_LOCKKEY1 (= any value)

Chapter 6

**Processors and Accelerators**

---



6.1 Compute Cluster.....	509
6.2 Dual-A72 MPU Subsystem.....	517
6.3 Dual-R5F MCU Subsystem.....	531
6.4 C66x DSP Subsystem.....	563
6.5 C71x DSP Subsystem.....	598
6.6 Graphics Accelerator (GPU).....	609
6.7 Multi-Standard HD Video Decoder (DECODER).....	616
6.8 Multi-Standard HD Video Encoder (ENCODER).....	621
6.9 Vision Pre-processing Accelerator (VPAC).....	626
6.10 Depth and Motion Perception Accelerator (DMPAC).....	779

## 6.1 Compute Cluster

This section describes the Compute Cluster (COMPUTE\_CLUSTER0).

### 6.1.1 Compute Cluster Overview

The COMPUTE\_CLUSTER0 encompasses the Arm® Cortex®-A72 subsystem, C71x DSP subsystem, Cluster Level Event Controller (CLEC), Generic Interrupt Controller (GIC), Multicore Shared Memory Controller (MSMC), Data Routing Unit (DRU), AXI to VBUSM.C bridge, ECC aggregators and debug components. Figure 6-1 shows an overview of the COMPUTE\_CLUSTER0 and its surrounding modules.

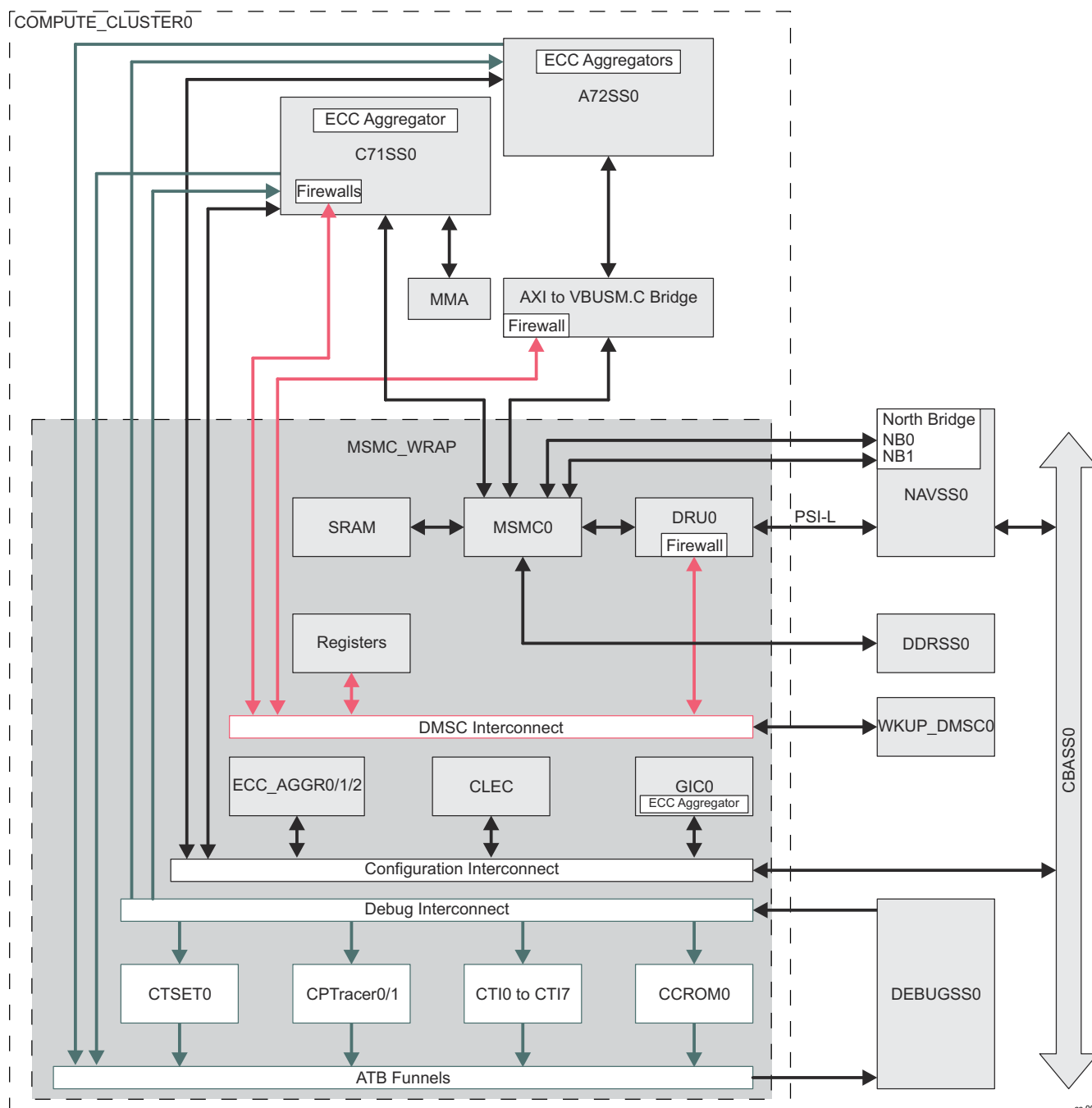


Figure 6-1. COMPUTE\_CLUSTER0 Overview

## 6.1.2 Compute Cluster Functional Description

For information about supported features, integration and functional description of the main COMPUTE\_CLUSTER0 modules, see:

- [Section 6.2 Dual-A72 MPU Subsystem](#)
- [Section 6.5 C71x DSP Subsystem](#)
- [Section 8.1 Multicore Shared Memory Controller \(MSMC\)](#)
- [Section 10.4 Data Routing Unit \(DRU\)](#)
- [Chapter 9 Interrupts](#)

### 6.1.2.1 Compute Cluster Memory Regions

[Table 6-1](#) provides a summary of the COMPUTE\_CLUSTER0 memory regions as seen from the system, WKUP\_DMSC0 and debug external APB.

**Table 6-1. COMPUTE\_CLUSTER0 Memory Regions**

Region	Start Address	End Address	Region Size
System View			
COMPUTE_CLUSTER0_GIC_TRANSLATER	0x00 0100 0000	0x00 013F FFFF	4 MB
COMPUTE_CLUSTER0_GIC_DISTRIBUTOR	0x00 0180 0000	0x00 0180 FFFF	64 KB
COMPUTE_CLUSTER0_GIC_MESSAGE_BASED_SPIS	0x00 0181 0000	0x00 0181 FFFF	64 KB
COMPUTE_CLUSTER0_GIC_ITS	0x00 0182 0000	0x00 0182 FFFF	64 KB
COMPUTE_CLUSTER0_GIC_REDISTRIBUTOR_CONTR OL_LPI_0	0x00 0190 0000	0x00 0190 FFFF	64 KB
COMPUTE_CLUSTER0_GIC_REDISTRIBUTOR_SGI_PPI _0	0x00 0191 0000	0x00 0191 FFFF	64 KB
COMPUTE_CLUSTER0_GIC_REDISTRIBUTOR_CONTR OL_LPI_1	0x00 0192 0000	0x00 0192 FFFF	64 KB
COMPUTE_CLUSTER0_GIC_REDISTRIBUTOR_SGI_PPI _1	0x00 0193 0000	0x00 0193 FFFF	64 KB
COMPUTE_CLUSTER0_GIC_REDISTRIBUTOR_CONTR OL_LPI_2	0x00 0194 0000	0x00 0194 FFFF	64 KB
COMPUTE_CLUSTER0_GIC_REDISTRIBUTOR_SGI_PPI _2	0x00 0195 0000	0x00 0195 FFFF	64 KB
COMPUTE_CLUSTER0_GIC_REDISTRIBUTOR_CONTR OL_LPI_3	0x00 0196 0000	0x00 0196 FFFF	64 KB
COMPUTE_CLUSTER0_GIC_REDISTRIBUTOR_SGI_PPI _3	0x00 0197 0000	0x00 0197 FFFF	64 KB
COMPUTE_CLUSTER0_GIC_REDISTRIBUTOR_CONTR OL_LPI_4	0x00 0198 0000	0x00 0198 FFFF	64 KB
COMPUTE_CLUSTER0_GIC_REDISTRIBUTOR_SGI_PPI _4	0x00 0199 0000	0x00 0199 FFFF	64 KB
COMPUTE_CLUSTER0_GIC_REDISTRIBUTOR_CONTR OL_LPI_5	0x00 019A 0000	0x00 019A FFFF	64 KB
COMPUTE_CLUSTER0_GIC_REDISTRIBUTOR_SGI_PPI _5	0x00 019B 0000	0x00 019B FFFF	64 KB
COMPUTE_CLUSTER0_GIC_REDISTRIBUTOR_CONTR OL_LPI_6	0x00 019C 0000	0x00 019C FFFF	64 KB
COMPUTE_CLUSTER0_GIC_REDISTRIBUTOR_SGI_PPI _6	0x00 019D 0000	0x00 019D FFFF	64 KB
COMPUTE_CLUSTER0_GIC_REDISTRIBUTOR_CONTR OL_LPI_7	0x00 019E 0000	0x00 019E FFFF	64 KB
COMPUTE_CLUSTER0_GIC_REDISTRIBUTOR_SGI_PPI _7	0x00 019F 0000	0x00 019F FFFF	64 KB



**Table 6-1. COMPUTE\_CLUSTER0 Memory Regions (continued)**

Region	Start Address	End Address	Region Size
COMPUTE_CLUSTER0_SS_CFG	0x00 0298 0000	0x00 0298 01FF	512 B
COMPUTE_CLUSTER0_CTL_CFG	0x00 0299 0000	0x00 0299 7FFF	32 KB
Region Registers of the A72SS0 Firewall	0x00 4504 0400	0x00 4504 07FF	1 KB
Region Registers of the C71SS0 L2 Pipe Firewall	0x00 4504 2000	0x00 4504 23FF	1 KB
Region Registers of the C71SS0 MDMA Firewall	0x00 4504 2400	0x00 4504 27FF	1 KB
Region Registers of the DRU Region Based Firewall	0x00 4504 7000	0x00 4504 73FF	1 KB
Region Registers of the DRU Channelized Firewall	0x00 4504 8000	0x00 4504 83FF	1 KB
Global Registers of the A72SS0 Firewall	0x00 45B1 0400	0x00 45B1 07FF	1 KB
Global Registers of the C71SS0 L2 Pipe Firewall	0x00 45B1 2000	0x00 45B1 23FF	1 KB
Global Registers of the C71SS0 MDMA Firewall	0x00 45B1 2400	0x00 45B1 27FF	1 KB
Global Registers of the DRU Region Based Firewall	0x00 45B1 7000	0x00 45B1 73FF	1 KB
Global Registers of the DRU Channelized Firewall	0x00 45B1 8000	0x00 45B1 83FF	1 KB
COMPUTE_CLUSTER0_DMSC_PRIVID	0x00 4583 0000	0x00 4580 3FFF	16 KB
Emulation and Debug Control of Each Arm and C71x Core	0x00 4590 0000	0x00 4590 3FFF	16 KB
Compute Cluster Configuration and Boot Vectors Control	0x00 45A0 0000	0x00 45A0 FFFF	64 KB
MSMC Coherent Interface for A72SS0	0x00 6000 0000	0x00 60FF FFFF	16 MB
MSMC Coherent Interface for C71SS0	0x00 6400 0000	0x00 64FF FFFF	16 MB
MSMC Coherent Interface for the NB0 Port of North Bridge in NAVSS0	0x00 6B00 0000	0x00 6BFF FFFF	16 MB
MSMC Coherent Interface for the NB1 Port of North Bridge in NAVSS0	0x00 6C00 0000	0x00 6CFF FFFF	16 MB
DRU Configuration Registers	0x00 6D00 0000	0x00 6DFF FFFF	16 MB
MSMC Configuration Registers	0x00 6E00 0000	0x00 6EFF FFFF	16 MB
MSMC SRAM	0x00 7000 0000	0x00 707F FFFF	8 MB
CLEC Registers	0x00 7800 0000	0x00 7FFF FFFF	128 MB
2 GB External SDRAM Space	0x00 8000 0000	0x00 FFFF FFFF	2 GB
8 GB External SDRAM Space <sup>(1)</sup>	0x08 0000 0000	0x09 FFFF FFFF	8 GB
DBG_CCROM (Compute Cluster Debug ROM Table)	0x4C 3000 0000	0x4C 3000 0FFF	4 KB
DBG_CTSET	0x4C 3010 0000	0x4C 3010 1FFF	8 KB
DBG_CTI0	0x4C 3010 2000	0x4C 3010 2FFF	4 KB
DBG_CTI1	0x4C 3010 3000	0x4C 3010 3FFF	4 KB
DBG_CTI2	0x4C 3010 4000	0x4C 3010 4FFF	4 KB
DBG_CTI3	0x4C 3010 5000	0x4C 3010 5FFF	4 KB
DBG_CTI4	0x4C 3010 6000	0x4C 3010 6FFF	4 KB
DBG_CTI5	0x4C 3010 7000	0x4C 3010 7FFF	4 KB
DBG_CTI7	0x4C 3010 9000	0x4C 3010 9FFF	4 KB
DBG_AGR0 (CPTracer 0)	0x4C 3014 0000	0x4C 3017 FFFF	256 KB
DBG_AGR1 (CPTracer 1)	0x4C 3018 0000	0x4C 301B FFFF	256 KB
DBG_CPAC0	0x4C 3040 0000	0x4C 307F FFFF	4 MB
DBG_CPAC4	0x4C 3140 0000	0x4C 317F FFFF	4 MB
COMPUTE_CLUSTER0_MSMC_ECC_AGGR0 Configuration Registers	0x4D 2000 0000	0x4D 2000 03FF	1 KB
COMPUTE_CLUSTER0_MSMC_ECC_AGGR1 Configuration Registers	0x4D 2000 0400	0x4D 2000 07FF	1 KB
COMPUTE_CLUSTER0_MSMC_ECC_AGGR2 Configuration Registers	0x4D 2000 0800	0x4D 2000 0BFF	1 KB

**Table 6-1. COMPUTE\_CLUSTER0 Memory Regions (continued)**

Region	Start Address	End Address	Region Size
Configuration Registers for the A72SS0 L2 ECC Aggregator (CC_ARM0_L2_ECC_AGGR0)	0x4D 2001 0000	0x4D 2001 03FF	1 KB
Configuration Registers for the A72SS0 Core 0 ECC Aggregator (CC_ARM0_CORE0_ECC_AGGR0)	0x4D 2001 0400	0x4D 2001 07FF	1 KB
Configuration Registers for the A72SS0 Core 1 ECC Aggregator (CC_ARM0_CORE1_ECC_AGGR0)	0x4D 2001 0800	0x4D 2001 0BFF	1 KB
Configuration Registers for the C71SS0 ECC Aggregator	0x4D 2005 0000	0x4D 2005 03FF	1 KB
DDRSS0_ECC_AGGR_CTL Registers	0x4D 200B 0000	0x4D 200B 03FF	1 KB
DDRSS0_ECC_AGGR_VBUS Registers	0x4D 200B 0400	0x4D 200B 07FF	1 KB
DDRSS0_ECC_AGGR_CFG Registers	0x4D 200B 0800	0x4D 200B 0BFF	1 KB
Configuration Registers for the GIC0 ECC Aggregator	0x4D 200C 0000	0x4D 200C 03FF	1 KB
Compute Cluster Registers	0x4D 2100 0000	0x4D 2100 FFFF	64 KB
Debug External APB View			
DBG_CCROM (Compute Cluster Debug ROM Table)	0x0000 0000	0x0000 0FFF	4 KB
DBG_CTSET	0x0010 0000	0x0010 1FFF	8 KB
DBG_CTI0	0x0010 2000	0x0010 2FFF	4 KB
DBG_CTI1	0x0010 3000	0x0010 3FFF	4 KB
DBG_CTI2	0x0010 4000	0x0010 4FFF	4 KB
DBG_CTI3	0x0010 5000	0x0010 5FFF	4 KB
DBG_CTI4	0x0010 6000	0x0010 6FFF	4 KB
DBG_CTI5	0x0010 7000	0x0010 7FFF	4 KB
DBG_CTI7	0x0010 9000	0x0010 9FFF	4 KB
DBG_AGR0 (CPTTracer 0)	0x0014 0000	0x0017 FFFF	256 KB
DBG_AGR1 (CPTTracer 1)	0x0018 0000	0x001B FFFF	256 KB
DBG_CPAC0	0x0040 0000	0x007F FFFF	4 MB
DBG_CPAC4	0x0140 0000	0x017F FFFF	4 MB

(1) The first 2 GB of this region are inaccessible.

### 6.1.2.2 Compute Cluster Firewalls

The following firewalls are present in COMPUTE\_CLUSTER0:

- Region based firewall for A72SS0 integrated in its AXI2VBUSMC bridge
- Two region based firewalls for C71SS0
- Region based firewall for DRU0
- Channelized firewall for DRU0

The A72SS0 firewall protects the traffic generated from the A72SS0 and is placed on the master port side unlike most device firewalls that are on the slave port side. This firewall has to be programed to enable the A72 master port access to the needed slaves.

The first C71SS0 region based firewall (C71SS0 MDMA) protects the traffic generated from the C71SS0 and same as A72SS0 is placed on the master port side. The second C71SS0 region based firewall protects the access to the C71SS0 L2 SRAM and is placed on the slave port side.

The DRU region based firewall is intended to protect the DRU configuration registers within a programmatically specified range. The DRU channelized firewall protects the following:

- The real time configuration registers for each channel
- The TR submission registers for each channel

- The data transfers for each channel

For more information about the DRU firewalls, see [Section 10.4.3.4 DRU Firewalls](#).

The A72SS0 and C71SS0 region based firewalls are same as the other device region based firewalls. For more information about their functionality, see [Section 3.3.4 Interconnect Firewalls](#) in [Chapter 3 System Interconnect](#).

### 6.1.2.3 Compute Cluster ECC Aggregators

As shown in [Figure 6-1](#) the COMPUTE\_CLUSTER0 consists of different modules, several interconnects and ECC aggregators. The memories (internal data buffers, command buffers, queue buffers, other FIFOs, configuration registers, etc.) associated with these blocks are ECC protected to increase functional reliability.

There are three ECC aggregators in the A72SS0, one in the C71SS0, one in GIC0, and three ECC aggregators in MSMC\_WRAP. [Table 6-2](#) through [Table 6-5](#) show the mapping between the end points, their memories protected by ECC and the RAM IDs. The mapping for the ECC aggregators in A72SS0 is shown in [Section 6.2.3.9 A72SS ECC Aggregators](#). The debug components are not ECC protected.

The RAM ID is written to the ECC\_VECTOR[10-0] ECC\_VECTOR field. For information about the ECC aggregator functionality, see [Section 12.11.4 ECC Aggregator](#).

**Table 6-2. COMPUTE\_CLUSTER0\_MSMC\_ECC\_AGGR0 RAM ID Mapping**

End Point	RAM ID <sup>(1)</sup>	ECC Protected RAM
MSMC_WRAP	0	edc_ctrl_eccaggr0
DRU	1	dru_cbass_dmsec_scr_edc
DRU	2	dru_cbass_dmsec_slv_brdg_edc
DRU	3	dru_cbass_edc_ctrl
DRU	4	dru_cbass_mmr_brdg_cfg_edc
DRU	5	dru_cbass_mmr_brdg_edc
DRU	7	dru_cbass_mmr_cfg_edc
DRU	8	dru_cbass_mmr_edc
DRU	9	dru_cbass_mmr_fw_ch_br_edc
DRU	10	dru_cbass_mmr_fw_ch_edc
DRU	11	dru_cbass_scr_edc
DRU	12	dru_cbass_slv_brdg_ecc_edc
DRU	13	dru_psi_edc
DRU	14	dru_0_edc
DRU	15	dru_1_edc
DRU	17	dru_eng_edc
DRU	18	dru_queue_edc
DRU	19	dru_rd_buf_edc
MSMC	20	msmc_mmr_busecc
MSMC	21	postarb_pipe_cfg_busecc
MSMC	22	dru0_slv_local_arb_busecc
MSMC	23	dru1_slv_local_arb_busecc
MSMC	24	dru0_mst_local_arb_busecc
MSMC	25	dru1_mst_local_arb_busecc
MSMC	26	emif0_slv_pipe_busecc
MSMC	27	emif0_mst_pipe_busecc
MSMC	28	cpu0_slv_local_arb_busecc
MSMC	29	cpu0_mst_local_arb_busecc
MSMC	36	cpu4_slv_local_arb_busecc

**Table 6-2. COMPUTE\_CLUSTER0\_MSMC\_ECC\_AGGR0 RAM ID Mapping (continued)**

End Point	RAM ID <sup>(1)</sup>	ECC Protected RAM
MSMC	37	cpu4_mst_local_arb_busecc
MSMC	46	cpu9_slv_local_arb_busecc
MSMC	47	cpu9_mst_local_arb_busecc
AXI2VBUSMC	48	en_msmc_p0_busecc_data
AXI2VBUSMC	49	en_msmc_p0_busecc
MSMC-Cache controller	68	rmw0_busecc
MSMC-Cache controller	69	rmw0_cache_tag_pipe_busecc
MSMC-Cache controller	70	rmw0_queue_busecc_0
MSMC-Cache controller	71	rmw0_queue_busecc_1
MSMC-Cache controller	72	rmw0_rmw_tag_update_busecc
MSMC-Cache controller	73	rmw0_sram_sf_pipe_busecc
MSMC-Cache controller	74	sram0_busecc
MSMC-Cache controller	75	dataram_bank0_busecc
MSMC-Cache controller	76	rmw1_busecc
MSMC-Cache controller	77	rmw1_cache_tag_pipe_busecc
MSMC-Cache controller	78	rmw1_queue_busecc_0
MSMC-Cache controller	79	rmw1_queue_busecc_1
MSMC-Cache controller	80	rmw1_rmw_tag_update_busecc
MSMC-Cache controller	81	rmw1_sram_sf_pipe_busecc
MSMC-Cache controller	82	sram1_busecc
MSMC-Cache controller	83	dataram_bank1_busecc
MSMC-Cache controller	84	rmw2_busecc
MSMC-Cache controller	85	rmw2_cache_tag_pipe_busecc
MSMC-Cache controller	86	rmw2_queue_busecc_0
MSMC-Cache controller	87	rmw2_queue_busecc_1
MSMC-Cache controller	88	rmw2_rmw_tag_update_busecc
MSMC-Cache controller	89	rmw2_sram_sf_pipe_busecc
MSMC-Cache controller	90	sram2_busecc
MSMC-Cache controller	91	dataram_bank2_busecc
MSMC-Cache controller	92	rmw3_busecc
MSMC-Cache controller	93	rmw3_cache_tag_pipe_busecc
MSMC-Cache controller	94	rmw3_queue_busecc_0
MSMC-Cache controller	95	rmw3_queue_busecc_1
MSMC-Cache controller	96	rmw3_rmw_tag_update_busecc
MSMC-Cache controller	97	rmw3_sram_sf_pipe_busecc
MSMC-Cache controller	98	sram3_busecc
MSMC—Data RAM	99	dataram_bank3_busecc
CLEC	100	clec_sram_ramecc
ECCAGGR0—P2P bridge	102	vbusp_cfg_ecc_aggr0_p2p_dst_busecc
CP4_DSP_CFG-P2P bridge	105	vbusp_cfg_dsp4_p2p_dst_busecc
DDRSS	106	emif_0_vsafe_si
CLEC	107	clec_clec_edc_ctrl_busecc

(1) Not present RAM IDs are reserved and not used.

**Table 6-3. COMPUTE\_CLUSTER0\_MSMC\_ECC\_AGGR1 RAM ID Mapping**

End Point	RAM ID <sup>(1)</sup>	ECC Protected RAM
MSMC_WRAP	0	edc_ctrl_eccaggr1
MSMC_WRAP-DMSC_WRAP	3	vbusp_dmesc_cbass_dru_mmr_fw_bridge_busecc
MSMC_WRAP-DMSC_WRAP	5	vbusp_dmesc_cbass_edc_ctrl_busecc_0
MSMC_WRAP-DMSC_WRAP	7	dmesc_mmr_boot_edc_ctrl_busecc_busecc
MSMC_WRAP-DMSC_WRAP	8	dmesc_mmr_emulation_edc_ctrl_busecc_busecc
MSMC_WRAP-DMSC_WRAP	9	dmesc_mmr_privid_edc_ctrl_busecc_busecc
MSMC_WRAP-CFG_WRAP	10	msmc_cfg_wrap_cbass_clk4_clk_edc_ctrl_cbass_int_clk4_busecc
MSMC_WRAP-CFG_WRAP	11	msmc_cfg_wrap_cbass_scr1_scr_edc_ctrl_busecc
MSMC_WRAP-CFG_WRAP	14	msmc_cfg_wrap_cbass_vbusp_msmc_ecc_aggr0_p2p_bridge_vbusp_msmc_ecc_aggr0_bridge_busecc
MSMC_WRAP-CFG_WRAP	15	msmc_cfg_wrap_cbass_vbusp_msmc_ecc_aggr1_p2p_bridge_vbusp_msmc_ecc_aggr1_bridge_busecc
MSMC_WRAP-DMSC_WRAP	16	vbusp_dmesc_cbass_edc_ctrl_cbass_int_busecc
AXI2VBUSMC-VDC	17	en_msmc_p0_vbusp_cfg_src_p2m_dst_busecc
AXI2VBUSMC-VDC	18	en_msmc_p0_vbusp_cfg_src_p2m_src_busecc
AXI2VBUSMC-VDC	19	en_msmc_p0_vbusp_cfg_src_m2m_src_busecc
ECCAGGR0-P2P bridge	29	vbusp_cfg_ecc_aggr0_p2p_src_busecc
AXI2VBUSMC-VDC	30	en_msmc_p0_vbusp_cfg_src_p2m_reassembly_busecc
MSMC_WRAP-CFG_WRAP	34	msmc_cfg_wrap_cbass_vbusp_gicss_p2m_bridge_vbusp_gicss_bridge_busecc
MSMC_WRAP-CFG_WRAP	35	msmc_cfg_wrap_cbass_vbusp_gicss_p2m_bridge_vbusp_gicss_bridge_reassembly_busecc
MSMC_WRAP-CFG_WRAP	37	msmc_cfg_wrap_cbass_vbusp_ddrss0_p2p_bridge_vbusp_ddrss0_bridge_busecc
DDRSS0	39	ddrss0_m2m_src_vbuss
DDRSS0	40	ddrss0_src_p2m_busecc
DDRSS0	41	ddrss0_src_p2m_reassembly_busecc
GIC-M2M bridge	42	msmc_gicss_m2m_bridge_src_edc_ctrl_busecc
GIC-M2M bridge	43	msmc_gicss_m2m_bridge_dst_edc_ctrl_busecc
CP4_DSP_CFG-P2P bridge	44	vbusp_cfg_dsp4_p2p_src_busecc
ECCAGGR2-P2P bridge	45	vbusp_cfg_ecc_aggr2_p2p_src_busecc
MSMC_WRAP-CFG_WRAP	46	msmc_cfg_wrap_cbass_vbusp_msmc_ecc_aggr2_p2p_bridge_vbusp_msmc_ecc_aggr2_bridge_busecc

(1) Not present RAM IDs are reserved and not used.

**Table 6-4. COMPUTE\_CLUSTER0\_MSMC\_ECC\_AGGR2 RAM ID Mapping**

End Point	RAM ID <sup>(1)</sup>	ECC Protected RAM
MSMC_WRAP	0	edc_ctrl_eccaggr2
DDRSS0	1	ddrss0_asafe_si
ECCAGGR2-P2P bridge	2	vbusp_cfg_ecc_aggr2_p2p_dst_busecc

**Table 6-5. C71SS0 ECC Aggregator RAM ID Mapping**

End Point	RAM ID <sup>(1)</sup>	ECC Protected RAM
C71SS0	0	edc_ctrl_eccaggr_corepac
CBASS P2P	1	c711_corepac_cfg_cbass_cfg_cbass_vbusp_eccaggr_cfg_p2p_bridge_vbusp_eccaggr_cfg_bridge_busecc
CBASS SCR	2	c711_corepac_cfg_cbass_cfg_cbass_scr_scr_c711_corepac_cfg_cbass_cfg_cbass_scr_scr_edc_ctrl_busecc
CBASS SCR	3	c711_corepac_cfg_cbass_cfg_cbass_clk1_clk_clk_edc_ctrl_cbass_int_clk1_clk_busecc

**Table 6-5. C71SS0 ECC Aggregator RAM ID Mapping (continued)**

End Point	RAM ID <sup>(1)</sup>	ECC Protected RAM
UMC Pipe0	6	busecc_pipe0_dp
UMC Pipe0	7	busecc_pipe0_p2
UMC Pipe1	8	busecc_pipe1_dp
UMC Pipe1	9	busecc_pipe1_p2
UMC Pipe2	10	busecc_pipe2_dp
UMC Pipe2	11	busecc_pipe2_p2
UMC Pipe3	12	busecc_pipe3_dp
UMC Pipe3	13	busecc_pipe3_p2
DMC	14	busecc_dmc
DMC Tag RAM	15	busecc_tagram_dmc
SE0	16	se_0_busecc
SE1	17	se_1_busecc
PMC	18	pmc_busecc

(1) Not present RAM IDs are reserved and not used.

## 6.2 Dual-A72 MPU Subsystem

This section describes the Dual-A72 Microprocessor Unit (MPU) Subsystem (A72SS) in the device.

### 6.2.1 A72SS Overview

#### 6.2.1.1 A72SS Introduction

The device implements one Dual-core Arm® Cortex®-A72 MPU, which is integrated inside the Compute Cluster, along with other modules. The Cortex-A72 cores are general-purpose processors that can be used for running customer applications.

The A72SS is built around the Arm Cortex-A72 MPCore (A72 cluster), which is provided by Arm and configured by TI. It is based on the symmetric multiprocessor (SMP) architecture, and thus it delivers high performance and optimal power management and debug capabilities.

The A72 processor is a multi-issue out-of-order superscalar execution engine with integrated L1 instruction and data caches, compatible with Armv8-A architecture. The Armv8-A architecture brings a number of new features. These include 64-bit data processing, extended virtual addressing and 64-bit general purpose registers.

The A72 processor features an in-order, 8-stage, dual-issue pipeline, and improved integer, Arm Neon™, Floating-Point Unit (FPU) and memory performance. It supports two execution states: AArch32 and AArch64. The AArch64 state gives the A72 CPU its ability to execute 64-bit applications, while the AArch32 state allows the processor to execute existing Armv7-A applications.

For more details on the Compute Cluster and its internal architecture, see *Compute Cluster*.

There is a single A72SS module in the device. [Table 6-6](#) shows A72SS allocation within device domains.

**Table 6-6. A72SS Modules Allocation within Device Domains**

Module Instance	Domain		
	WKUP	MCU	MAIN
A72SS0	–	–	✓

#### 6.2.1.2 A72SS Features

The A72SS supports the following key features:

- Arm Dual-A72 Cluster (Cortex-A72 MPCore) level features:
  - A72 CPU
    - Full Armv8-A architecture compliance
      - AArch32 and AArch64 execution states
        - AArch32 for full backward compatibility with Armv7
        - AArch64 for 64b support and new architectural features
      - All exception levels EL0-3
      - A32 instruction set (previously Arm instruction set)
      - T32 instruction set (previously Thumb instruction set)
      - A64 instruction set
    - Advanced SIMD and floating point extensions (Neon)
    - Armv8 cryptography extensions
    - Superscalar, variable length, out-of-order pipeline
    - Dynamic branch prediction with Branch Target Buffer (BTB) and Global History Buffer (GHB) RAMs, return stack, and indirect predictor
  - A72 L1/L2 cache memory and MMU
    - 48-entry, fully associative, L1 instruction TLB with native support for 4KB, 64KB, and 1MB page sizes
    - 32-entry, fully associative, L1 data TLB with support for 4KB, 64KB, and 1MB page sizes
    - 4-way, set associative, unified 1K entry L2 TLB per processor
    - 48KB L1 Instruction Cache per processor with parity protection
    - 32KB L1 Data Cache per processor with ECC protection

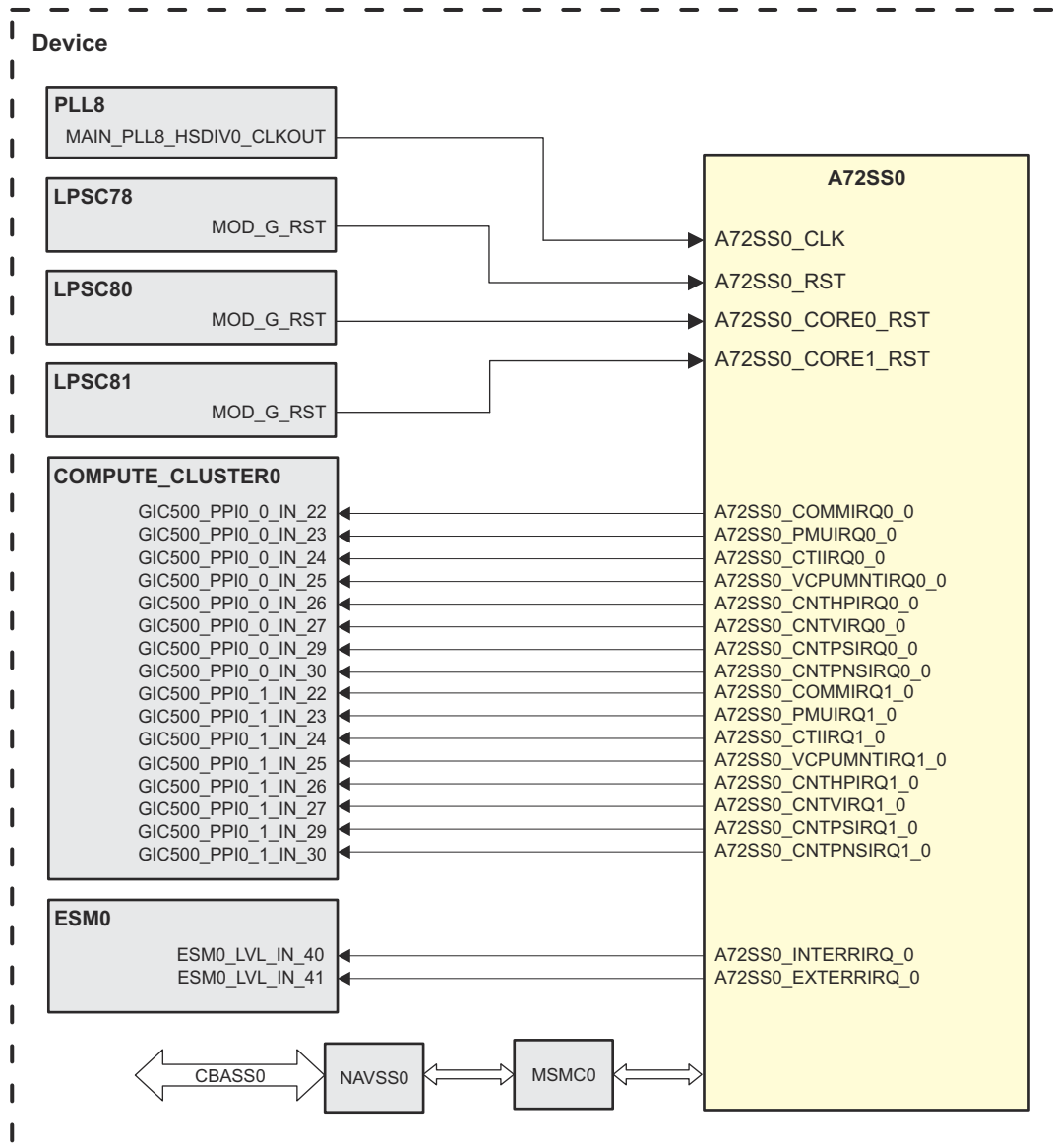
- 1MB Shared L2 Cache with ECC protection
  - Arm GICv3 architecture
  - Generic timers
  - Debug
    - Arm CoreSight™ architecture
    - Embedded Trace Macrocell (ETM)
    - Performance Monitor Unit (PMUv3 architecture)
- TI A72SS subsystem level features:
  - 512-bit wide, asynchronous VBUSM.C master interface
    - AXI2VBUSM\_MASTER bridge
    - Cache pre-warming via use of ACP port
  - Timebase input interfaces
    - 64-bit graycoded global time
    - 48-bit graycoded debug time
  - 32-bit VBUSP slave interface for debug (internally converted to APB)
  - 32-bit ATB output port for debug/trace
  - Interface with Arm GIC-500 interrupt controller
  - Support for ECC on internal RAMs via ECC aggregators
- SoC level features:
  - Supports the SoC multi-core cache coherency architecture
  - Dedicated A72SS clocking (Arm PLL) for full flexibility in performance trade-offs
  - Advanced power management with fine-grained control of individual A72 CPU power domains, coarse grained cluster-level power management, and low-power standby modes (WFI/WFE modes)
  - Dedicated RTI windowed watchdog timer per core



## 6.2.2 A72SS Integration

This section describes the A72SS integration in the device, including information about clocks, resets, and hardware requests.

Figure 6-2 shows the A72SS integration.



**Figure 6-2. A72SS Integration**

Table 6-7 through Table 6-9 summarize the A72SS integration.

**Table 6-7. A72SS Integration Attributes**

Module Instance		Attributes			
		Power Sleep Controller	Power Domain	Module Domain	Interconnect
A72SS0	A72 cluster	PSC0	PD14	LPSC78	CBASS0 <sup>(1)</sup>
	A72SS0_CORE0	PSC0	PD15	LPSC80	CBASS0 <sup>(1)</sup>

**Table 6-7. A72SS Integration Attributes (continued)**

A72SS0_CORE1	PSC0	PD16	LPSC81	CBASS0 <sup>(1)</sup>
--------------	------	------	--------	-----------------------

(1) Accessed through MSMC0

**Table 6-8. A72SS Clocks and Resets**

Clocks					
Module Instance		Module Clock Input	Source Clock Signal	Source	Description
A72SS0		A72SS0_CLK	MAIN_PLL8_HSDIV0_CL KOUT	PLL8	A72SS0 clock
Resets					
Module Instance		Module Reset Input	Source Reset Signal	Source	Description
A72SS0	A72 cluster	A72SS0_RST	MOD_G_RST	LPSC78	A72 cluster reset
	A72SS0_CORE0	A72SS0_CORE0_RST	MOD_G_RST	LPSC80	A72SS0_CORE0 reset
	A72SS0_CORE1	A72SS0_CORE1_RST	MOD_G_RST	LPSC81	A72SS0_CORE1 reset

**Table 6-9. A72SS Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
A72SS0	A72SS0_COMMIRQ0_0	GIC500_PPIO_0_I N_22	COMPUTE_CLUSTER0	A72SS0_CORE0 DCC comms channel interrupt	Level
	A72SS0_PMIIRQ0_0	GIC500_PPIO_0_I N_23	COMPUTE_CLUSTER0	A72SS0_CORE0 PMU counter overflow interrupt	Level
	A72SS0_CTIIRQ0_0	GIC500_PPIO_0_I N_24	COMPUTE_CLUSTER0	A72SS0_CORE0 cross trigger interface interrupt	Level
	A72SS0_VCPUMNTIRQ0_0	GIC500_PPIO_0_I N_25	COMPUTE_CLUSTER0	A72SS0_CORE0 virtual CPU maintenance interrupt	Level
	A72SS0_CNTHPIRQ0_0	GIC500_PPIO_0_I N_26	COMPUTE_CLUSTER0	A72SS0_CORE0 hypervisor timer interrupt	Level
	A72SS0_CNTVIRQ0_0	GIC500_PPIO_0_I N_27	COMPUTE_CLUSTER0	A72SS0_CORE0 virtual timer interrupt	Level
	A72SS0_CNTPSIRQ0_0	GIC500_PPIO_0_I N_29	COMPUTE_CLUSTER0	A72SS0_CORE0 secure physical timer interrupt	Level
	A72SS0_CNTPNSIRQ0_0	GIC500_PPIO_0_I N_30	COMPUTE_CLUSTER0	A72SS0_CORE0 non-secure physical timer interrupt	Level
	A72SS0_COMMIRQ1_0	GIC500_PPIO_1_I N_22	COMPUTE_CLUSTER0	A72SS0_CORE1 DCC comms channel interrupt	Level
	A72SS0_PMIIRQ1_0	GIC500_PPIO_1_I N_23	COMPUTE_CLUSTER0	A72SS0_CORE1 PMU counter overflow interrupt	Level
	A72SS0_CTIIRQ1_0	GIC500_PPIO_1_I N_24	COMPUTE_CLUSTER0	A72SS0_CORE1 cross trigger interface interrupt	Level
	A72SS0_VCPUMNTIRQ1_0	GIC500_PPIO_1_I N_25	COMPUTE_CLUSTER0	A72SS0_CORE1 virtual CPU maintenance interrupt	Level
	A72SS0_CNTHPIRQ1_0	GIC500_PPIO_1_I N_26	COMPUTE_CLUSTER0	A72SS0_CORE1 hypervisor timer interrupt	Level
	A72SS0_CNTVIRQ1_0	GIC500_PPIO_1_I N_27	COMPUTE_CLUSTER0	A72SS0_CORE1 virtual timer interrupt	Level
	A72SS0_CNTPSIRQ1_0	GIC500_PPIO_1_I N_29	COMPUTE_CLUSTER0	A72SS0_CORE1 secure physical timer interrupt	Level
	A72SS0_CNTPNSIRQ1_0	GIC500_PPIO_1_I N_30	COMPUTE_CLUSTER0	A72SS0_CORE1 non-secure physical timer interrupt	Level
	A72SS0_INTERRIRQ_0	ESM0_LVL_IN_40	ESM0	A72 cluster cache ECC error interrupt	Level
	A72SS0_EXTERRIRQ_0	ESM0_LVL_IN_41	ESM0	A72 cluster memory bus error interrupt	Level

## 6.2.3 A72SS Functional Description

### 6.2.3.1 A72SS Block Diagram

Figure 6-3 shows the block diagram of the A72SS subsystem.

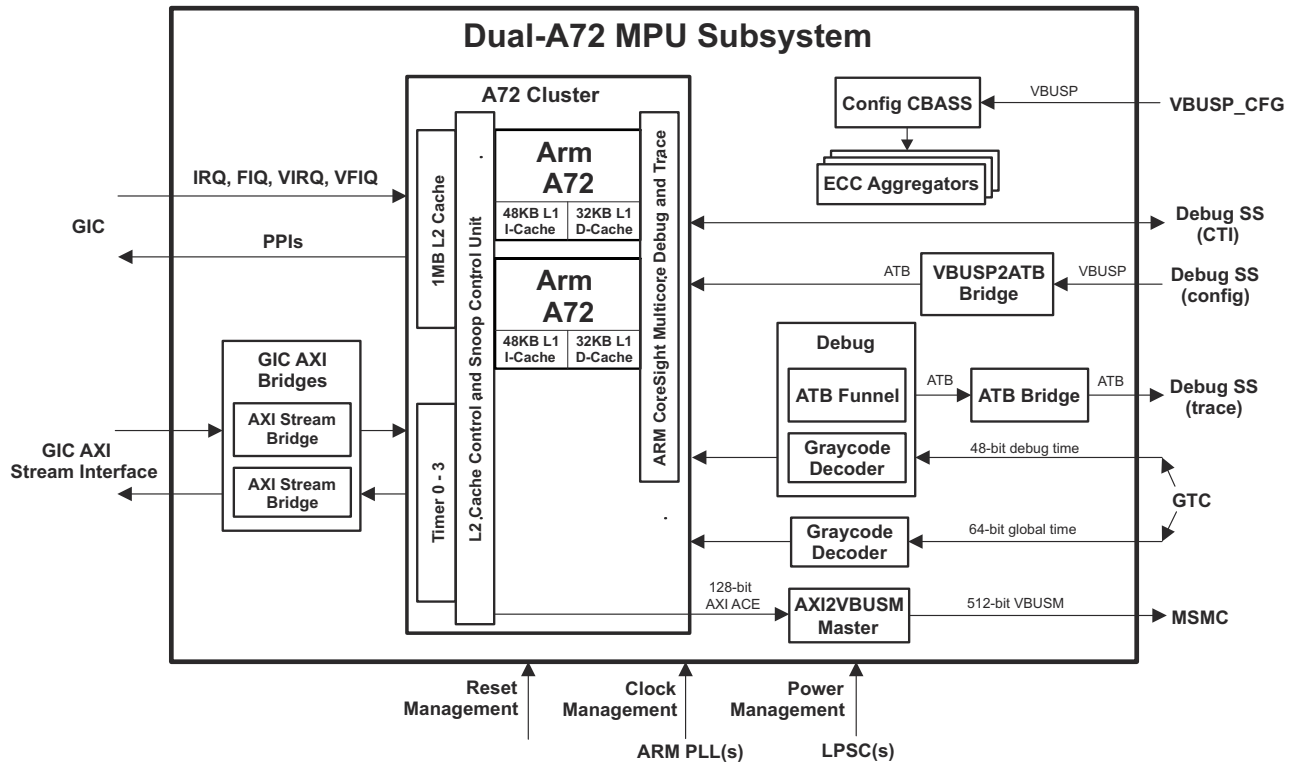


Figure 6-3. A72SS Block Diagram

### 6.2.3.2 A72SS A72 Cluster

The A72 cluster is provided by Arm and configured by TI. Table 6-10 summarizes the configuration of the A72 cluster for this device.

Table 6-10. A72 Cluster Configuration

Parameter	Value
Core type	A72 <sup>(1)</sup>
Number of cores	2
Bus width	512
L1 instruction cache size	48K
L1 data cache size	32K
L2 cache size	1M
ECC protection for L2 cache	Included
ECC/parity protection for CPU cache	Included
Advanced SIMD and Floating Point Extension	Included
Cryptography extension	Included
L2 FEQ size	28

(1) The A72 core revision used in this device is r1p0.

### 6.2.3.3 A72SS Interfaces and Async Bridges

The A72SS has the following main interfaces:

- 512-bit wide VBUSM.C master interface
  - Supported by AXI2VBUSM\_MASTER Bridge, which performs the following primary functions (among others):
    - Provides an asynchronous voltage domain crossing boundary for the AXI ACE master and ACP slave ports
    - Provides protocol conversion between AXI ACE (128-bit) and CBA VBUSM.C (512-bit)
    - Provides support for cache pre-warming
- 64-bit graycoded system input time
  - Graycode value provided by Global Timebase Counter (GTC)
  - Dedicated decoder (64-bit input) for graycode-to-binary conversion
- 48-bit graycoded debug input time
  - Graycode value provided by GTC
  - Dedicated decoder (48-bit input) for graycode-to-binary conversion
- 32-bit VBUSP slave interface for configuration of ECC aggregators
- 32-bit VBUSP slave interface for debug
  - Supported by VBUSP2APB Bridge, which performs VBUSP-to-APB conversion (for controlling the Arm A72 Cluster internal debug logic)
- 32-bit ATB output port for debug/trace
  - Supported by ATB Bridge, which performs clock and voltage level conversion on the combined ATB interface
  - Connected to the Debug Subsystem
- Cross Trigger Interface (CTI) for debug
  - Connected to the Debug Subsystem
- Interface(s) with Arm GIC-500 Interrupt Controller
  - Supported by GIC AXI Streaming Bridge, which performs clock and voltage level conversion on the AXI streaming protocol
  - Interrupts (PPIs) from Arm A72 Cluster to GIC-500
  - Interrupts (IRQ, FIQ, VIRQ, VFIQ) from GIC-500 to Arm A72 Cluster
- Power/clock interface(s)
  - Dedicated PLL for Arm A72 Cluster
  - Dedicated LPSC for Arm A72 Cluster, and also for each A72 core

### 6.2.3.4 A72SS Interrupts

#### 6.2.3.4.1 A72SS Interrupt Inputs

The A72 CPU receives interrupts at its inputs (IRQ, FIQ, VIRQ, VFIQ) via the dedicated Arm GIC-500 Interrupt Controller which is integrated inside the Compute Cluster and is tightly-coupled to the A72. The GIC-500 supports both A72 cores in the system. The GIC-500 is compliant to the Arm GICv3 specification and supports four types of interrupts:

- *Software Generated Interrupts (SGI)*
  - There are 16 SGIs (ID0-ID15)
  - These are inter-processor interrupts
- *Private Peripheral Interrupts (PPI)*
  - There are 16 PPIs (ID16-ID31)
  - These are wired interrupts dedicated to a specific CPU
  - Many are reserved to specific functions via convention
- *Shared Peripheral Interrupts (SPIs)*
  - There are 960 SPIs (ID32-ID991)
  - These are wired interrupts that can be routed to any core or cluster, based on the programming of that interrupt in the GIC-500

- *Locality-Specific Peripheral Interrupts (LPI)*
  - There are 57344 LPIs
  - These interrupts are used for message-based interrupts from a peripheral

The mapping of PPIs and SPIs to the GIC-500 interrupt inputs can be found in TBD.

#### Note

For a brief list of features supported by the GIC-500 module, see *Generic Interrupt Controller (GIC)*.  
For detailed description of the GIC-500 module, see the *Arm® CoreLink™ GIC-500 Generic Interrupt Controller Technical Reference Manual*.

#### 6.2.3.4.2 A72SS Interrupt Outputs

Table 6-11 lists the interrupts generated by the A72SS

**Table 6-11. A72SS Interrupt Outputs**

TI Interrupt Name <sup>(1)</sup>	Arm Interrupt Name	Interrupt Description
<b>A72 Core Interrupts (PPIs)<sup>(2)</sup></b>		
A72SS0_VCPUMNTIRQy_0	nVCPUMNTIRQ	This interrupt indicates that a virtual CPU interface on the corresponding core needs serviced. This interrupt is serviced by software switching to the virtual CPU and finding what specific service needs done.
A72SS0_CNTHPIRQy_0	nCNTHPIRQ	This interrupt indicates a physical timer event at the EL2 (hypervisor) exception level. Service of this interrupt is dependent on what the timer was set for. Software should take exception to EL2 and service accordingly.
A72SS0_CNTPNSIRQy_0	nCNTPNSIRQ	This interrupt indicates a physical timer event at the EL1 non-secure exception level. Service of this interrupt is dependent on what the timer was set for. Software should take exception to EL1 non-secure and service accordingly.
A72SS0_CNTPSIRQy_0	nCNTPSIRQ	This interrupt indicates a physical timer event at the EL1 secure exception level. Service of this interrupt is dependent on what the timer was set for. Software should take exception to EL1 secure and service accordingly.
A72SS0_CNTVIRQy_0	nCNTVIRQ	This interrupt indicates a virtual timer event. Service of this interrupt is dependent on what the timer was set for. Software should take exception and service accordingly.
A72SS0_PMIIRQy_0	nPMUIRQ	This interrupt is generated by the Performance Monitor Unit (PMU). The PMU can generate an interrupt based on several conditions, depending on programming. Software should take exception and query the PMU as to the cause of the interrupt, and act accordingly.
A72SS0_COMMIRQy_0	nCOMMIRQ	Communications channel receive or transmit interrupt
A72SS0_CTIIRQy_0	CTIIRQ	Cross Trigger Interface (CTI) interrupt
<b>A72 Cluster Interrupts</b>		
A72SS0_EXTERRIRQ_0	nEXTERRIRQ	This error indicates that an error has occurred on the memory bus. It is considered an external abort (or asynchronous error), because it cannot be attributed to a specific instruction. Depending on the system, software may try to recover, or reset the system.
A72SS0_INTERRIRQ_0	nINTERRIRQ	This error indicates an unrecoverable error in the cache system (L1 or L2 data RAM, L2 tag RAM, or SCU L1 duplicate tag RAM). It is considered an external abort (or asynchronous error), because it cannot be attributed to a specific instruction. Depending on the system, software can try to recover, or reset.

(1) In this column, y is the A72 core index (0 or 1)

(2) These are *per core* interrupts

---

### Note

The mapping of these interrupts in the system is summarized in *A72SS Integration*, and can also be found in *Interrupts*.

For more detailed description of these interrupts and their handling, see the *Arm® Cortex®-A72 MPCore Processor Technical Reference Manual*.

---

## 6.2.3.5 A72SS Power Management, Clocking and Reset

### 6.2.3.5.1 A72SS Power Management

The A72 cluster and each A72 CPU reside in a separate power domain, as follows:

- PD14 (PD\_A72\_CLUSTER\_0): Power domain of A72 cluster
- PD15 (PD\_A72\_0): Power domain of A72SS0\_CORE0
- PD16 (PD\_A72\_1): Power domain of A72SS0\_CORE1

There is a dedicated Local Power Sleep Controller (LPSC) for the A72 cluster and for each A72 core, as well. The LPSC assignment is as follows:

- LPSC78 (LPSC\_A72\_CLUSTER\_0): Dedicated for A72 cluster
- LPSC80 (LPSC\_A72\_0): Dedicated for A72SS0\_CORE0
- LPSC81 (LPSC\_A72\_1): Dedicated for A72SS0\_CORE1

For more details on these LPSCs, including power-up/down sequences, see *Power*.

### 6.2.3.5.2 A72SS Clocking

There is a dedicated PLL for the A72 cluster: PLL8 (ARM0 PLL). For more details on this PLL, see *Clocking*.

### 6.2.3.6 A72SS Debug Support

The A72SS supports the standard Arm CoreSight debug architecture. Details on Arm debug can be found *On-chip Debug*, and the relevant Arm specifications.

### 6.2.3.7 A72SS Timestamps

The A72SS has two timebase input interfaces:

- 64-bit global timestamp: Used for synchronizing the Arm internal timers. This is done via the Arm CNTVALUEB[63:0] bus
- 48-bit debug timestamp: Can be embedded in Arm trace streams at periodic and strategic locations, allowing the temporal relationship of different trace sources to be determined

Both of them are fed by the Global Timebase Counter (GTC), which provides a 64-bit graycode value. The MPU includes two graycode decoders (for global time and debug time, respectively), which take the asynchronous graycoded times, synchronize them to the appropriate clock, and convert them to binary time, as required by Arm.

---

### Note

Both graycode decoders are 64-bit but the one dedicated to debug time has a 48-bit input (the upper 16 bits are tied to 0).

---

For more details on the GTC, see *Global Timebase Counter (GTC)*.

### 6.2.3.8 A72SS Watchdog

The A72SS does not have an integrated watchdog timer. Instead, this feature is provided externally by the Real Time Interrupt (RTI) module, which implements a windowed watchdog timer capable of issuing warm reset to the SoC, when necessary.

There are two RTI modules in the MAIN domain that are assigned to the A72 cores:

- RTI0 used as windowed watchdog for A72SS0\_CORE0
- RTI1 used as windowed watchdog for A72SS0\_CORE1

For more details on the RTI windowed watchdog feature, see *Real Time Interrupt Module (RTI)*.

### 6.2.3.9 A72SS Internal Diagnostics

The A72SS integrates three ECC aggregators for internal diagnostics and to stimulate errors in associated RAMs for ECC testing purposes:

- A72SS0\_CORE0\_ECC\_AGGR: ECC aggregator for A72SS0\_CORE0 level
- A72SS0\_CORE1\_ECC\_AGGR: ECC aggregator for A72SS0\_CORE1 level
- A72SS0\_CLUSTER\_ECC\_AGGR: ECC aggregator for A72\_CLUSTER (L2) level

#### 6.2.3.9.1 A72SS ECC Aggregators During Low Power States

When a CPU core is in WFI/WFE, the corresponding CPU ECC aggregator MMR region is not accessible and will return error status. Similarly, when L2 is in WFI, the L2 ECC aggregator MMR region is not accessible and will return error status.

#### 6.2.3.9.2 A72SS CBASS Diagnostics

A72SS0\_CORE0\_ECC\_AGGR integrates all the SVBUS slave endpoints for A72SS0\_CORE0.

**Table 6-12. A72SS0\_CORE0\_ECC\_AGGR RAM ID Mapping for Interface Diagnostics**

Endpoint	Memory ID	Controller
A72SS0_CORE0	24	EDC controller for A72SS0_CORE0_ECC_AGGR

A72SS0\_CORE1\_ECC\_AGGR integrates all the SVBUS slave endpoints for A72SS0\_CORE1.

**Table 6-13. A72SS0\_CORE1\_ECC\_AGGR RAM ID Mapping for Interface Diagnostics**

Endpoint	Memory ID	Controller
A72SS0_CORE1	24	EDC controller for A72SS0_CORE1_ECC_AGGR

A72SS0\_CLUSTER\_ECC\_AGGR integrates all the SVBUS slave endpoints for CBASS and cluster.

**Table 6-14. A72SS0\_CLUSTER\_ECC\_AGGR RAM ID Mapping for Interface Diagnostics**

Endpoint	Memory ID	Controller
A72SS0_CLUSTER	32	EDC controller for A72SS0_CLUSTER_ECC_AGGR
	33-41	Internal bridges

#### 6.2.3.9.3 A72SS SRAM Diagnostics

The A72 cluster natively supports ECC/parity protection on some of its internal SRAMs. However, error injection for the memories is not natively supported by the A72 cluster. Instead, this capability is added at A72SS level via ECC aggregator. This is a valuable TI addition to the Arm native implementation.

[Table 6-15](#) shows the A72SS SRAM ECC support details.

**Table 6-15. A72SS SRAM ECC Support**

A72 RAM	ARM Native Support	TI Error Injection Support
L1 I-Cache Data RAM	Parity protection	Single error injection <sup>(1)</sup>
L1 I-Cache Tag RAM	Parity protection	Single error injection <sup>(1)</sup>
L1 I-Cache BTB RAM	No support	N/A
L1 I-Cache GHB RAM	No support	N/A
L1 I-Cache IP RAM	No support	N/A
L1 D-Cache Data RAM	ECC	Single and double error injection <sup>(1)</sup>
L1 D-Cache Tag RAM	ECC	Single and double error injection <sup>(1)</sup>
L1 PF PHT RAM	No support	N/A



**Table 6-15. A72SS SRAM ECC Support (continued)**

A72 RAM	ARM Native Support	TI Error Injection Support
L2 TLB RAM	Parity	Single error injection <sup>(1)</sup>
L2 Snoop Tag RAM	ECC	Single and double error injection <sup>(2)</sup>
L2 Tag RAM	ECC	Single and double error injection <sup>(2)</sup>
L2 Data RAM	ECC	Single and double error injection <sup>(2)</sup>
L2 Dirty RAM	ECC	Single and double error injection <sup>(2)</sup>
L2 Inclusion PLRU RAM	ECC	Single and double error injection <sup>(2)</sup>

(1) Supported by A72SS0\_CORE0/1\_ECC\_AGGR

(2) Supported by A72SS0\_CLUSTER\_ECC\_AGGR

**6.2.3.9.4 A72SS SRAM ECC Aggregator Configurations**

There are several schemas of memory protection employed inside of the processors. [Table 6-16](#) and [Table 6-17](#) describe the schema and arrangement. This describes what bits are protected, how they are protected, the behavior when an error is detected and what bits can be disturbed for ECC testing purposes. This also lists the RAM\_ID associated with each memory to the corresponding ECC aggregator.

The key for the protection schemes is as follows:

- Parity: Parity bit(s) to protect data
- ECC: Error Correction Code to protect data
- SECEDED: Single Error Correction, Double Error Detection

**Table 6-16. A72SS0\_CORE0/1\_ECC\_AGGR Mapping**

Memory	Protection	Parity/ECC Granularity	RAM Arrangement	RAM_ID	Notes
L1 I-Cache Data	Parity	1 parity bit / 16 bits of Instruction data	[71] – Parity of all odd bits of [63:32] and [67] [70] – Parity of all even bits of [63:32] and [66] [69] – Parity of all odd bits of [31:0] and [65] [68] – Parity of all even bits of [31:0] and [64]	0-5	Parity error invalidates the offending cache line, force a fetch from the L2 cache on the next access. No aborts are generated, location of error is reported in the A72 CPUMERR register.
L1 I-Cache Tag	Parity	2 parity bits / 36 bits tag entry	[107] – Parity of all odd bits of [104:72] [71] – Parity of all odd bits of [68:36] [35] – Parity of all odd bits of [32:0] [106] – Parity of all even bits of [104:72] [70] – Parity of all even bits of [68:36] [34] – Parity of all even bits of [32:0] [105], [69], [33] – Valid bit [104:72], [68:36], [32:0] – Tag	6-7	Parity error invalidates the offending cache line, force a fetch from the L2 cache on the next access. No aborts are generated, location of error is reported in the A72 CPUMERR register.
L1 D-cache Data	ECC SECEDED	32-bit word	[155:149] – ECC [148:117] – Data [116:110] – ECC [109:78] – Data [77:71] – ECC [70:39] – Data [38:32] – ECC [31:0] – Data	8-15	Single bit errors are corrected in the background and the line is removed from the cache as part of the correction process. No exception or interrupt is generated, but the A72 MERR register is updated to indicate a non-fatal error. Double bit errors are detected and an imprecise data abort is triggered and the line is evicted from the cache.



**Table 6-16. A72SS0\_CORE0/1\_ECC\_AGGR Mapping (continued)**

Memory	Protection	Parity/ECC Granularity	RAM Arrangement	RAM_ID	Notes
L1 D-Cache Tag	ECC SECEDED	Tag for a single cache line	[39:33] – ECC for [32:2] Tag and [1:0] State	16-19	Single bit errors are corrected in the background and the line is removed from the cache as part of the correction process. No exception or interrupt is generated, but the A72 MERR register is updated to indicate a nonfatal error. Double bit errors are detected and an imprecise data abort is triggered and the line is evicted from the cache.
L2 TLB	Parity		[129] – Parity Bit [128:0] – TLB data	20-23	Error detection results in entry invalidated, new pagewalk to refetch.

**Table 6-17. A72SS0\_CLUSTER\_ECC\_AGGR Mapping**

Memory	Protection	RAM Arrangement	RAM_ID	Notes
L2 Tag	ECC SECEDED	Odd way: [77:71] – ECC for [70:41] Tag and [40:39] State  Even Way: [38:32] – ECC for [31:2] Tag and [1:0] State	12-27	Accesses resulting in an L2 cache hit, where a single-bit error is detected on the data array, the L2 memory system supports in-line ECC correction. Uncorrected data is forwarded to the requesting unit, and in parallel, the ECC circuitry checks for accuracy. If a single-bit error is detected, any uncorrected data returned within two cycles before the error indicator must be discarded.
L2 Data	ECC SECEDED	[143:128] – ECC [127:0] - Data	4-11	Accesses resulting in an L2 cache hit, where a single-bit error is detected on the Data array, the L2 memory system supports in-line ECC correction. Uncorrected data is forwarded to the requesting unit, and in parallel, the ECC circuitry checks for accuracy. If a single-bit error is detected, any uncorrected data returned within two cycles before the error indicator must be discarded.
L2 Snoop Tag RAM	ECC SECEDED	[79:73] – ECC for [72:42] Tag and [41:40] State  [39:33] – ECC for [32:2] Tag and [1:0] State	0-3	For single-bit ECC errors detected, the request is flushed from the L2 pipeline and is forced to reissue. The tag bank where the single-bit error occurred, performs a read-modify-write sequence to correct the single-bit error in the array. The request is then reissued.
L2 Inclusion PLRU RAM	ECC SECEDED	[38:32] – ECC for [30:16] PLRU and [15:0] Inclusion	30-31	For single-bit ECC errors detected, the request is flushed from the L2 pipeline and is forced to reissue. The tag bank where the single-bit error occurred, performs a read-modify-write sequence to correct the single-bit error in the array. The request is then reissued.

**Table 6-17. A72SS0\_CLUSTER\_ECC\_AGGR Mapping (continued)**

Memory	Protection	RAM Arrangement	RAM_ID	Notes
L2 Dirty RAM	ECC SECDED	[47:44] – ECC	28-29	For single-bit ECC errors detected, the request is flushed from the L2 pipeline and is forced to reissue. The tag bank where the single-bit error occurred, performs a read-modify-write sequence to correct the single-bit error in the array. The request is then reissued.
		[43:40] – Page Attribute		
		[39:37] – ECC		
		[36] – Dirty bit		
		[35:32] – ECC		
		[31:28] – Page Attribute		
		[27:25] – ECC		
		[24] – Dirty bit		
		[23:20] – ECC		
		[19:16] – Page Attribute		
		[15:13] – ECC		
		[12] – Dirty bit		
		[11:8] – ECC		
		[7:4] – Page Attribute		
		[3:1] – ECC		
		[0] – Dirty bit		

### 6.2.3.10 A72SS Cache Pre-Warming

Cache pre-warming is a method by which data/instructions that will be needed in the near future can be loaded into the L2 cache before the MPU itself does a fetch. The goal is for the cache to have the data ready before the MPU actually needs it, so that software does not need to stall waiting for cache fetches. The mechanism is the MPU's ACP port combined with the Data Routing Unit (DRU).

The basic steps are as follows:

1. Software identifies some data or instruction it knows it will need in the near future
2. Software sets up a DRU pre-warm transfer for this data
3. DRU performs reads through the ACP port of the MPU (through AXI2VBUSM\_MASTER)
4. MPU initiates reads over the memory interface that load the data into L2 cache
5. MPU returns the read data through the ACP port
  - AXI2VBUSM\_MASTER discards this data (there is no need for the DRU to see it)

Now, when software tries to access this data, it will hit in the L2 cache.

For details on DRU programming to perform cache pre-warming, see *Multicore Shared Memory Controller (MSMC)*.

### 6.2.3.11 A72SS Boot

For Armv8 cores such as A72, two boot modes are supported: legacy 32-bit address boot mode, and 64-bit address boot mode. The selection is asserted before the A72 core is out of reset and is based on the configuration of the following Compute Cluster register:

- CC\_CP\_CONFIG\_0[3-0] AARCH bit field

When the A72 core is configured for a 32-bit address boot mode, the boot vector is fixed at address 0x0000\_0000, which corresponds to the start address of the 2KB boot RAM (PSRAM2KECC0\_RAM).

When the A72 core is configured for a 64-bit address boot mode, the boot vector is determined by the RVBARADDRx tie-off signal, which is configured via associated Compute Cluster registers:

- For A72SS0\_CORE0: CC\_RST\_VEC\_LO\_CP0\_0 and CC\_RST\_VEC\_HI\_CP0\_0
- For A72SS0\_CORE1: CC\_RST\_VEC\_LO\_CP1\_0 and CC\_RST\_VEC\_HI\_CP1\_0

The Compute Cluster RVBARADDRx setting is latched into the A72 RVBAR register, where reset is applied. Once reset is released, the A72 RVBAR register becomes read-only and the latched value is used as a boot vector location.

[Table 6-18](#) summarizes the A72 boot mode specifics.

**Table 6-18. A72 Boot Mode Specifics**

32-bit Address Boot Mode	64-bit Address Boot Mode
Boot vector is part of vector table	Boot vector is not part of vector table
Boot vector has a fixed address: 0x0000_0000. It corresponds to the 2KB boot RAM (PSRAM2KECC0_RAM)	Boot vector is determined by RVBARADDRx[39:2] tie-off setting per core. <sup>(1)</sup>
Location of the vector table is fixed by default (0x0000_0000)	No default location of vector table
No need to program A72 VBAR register if vector table stays at the default address	Each A72 core boot code must program VBAR_ELn core registers to initialize its location of the vector tables for the various Armv8 exception levels
Vector table is 32B	Vector table is 128B
Vector table location can be reprogrammed after boot	Vector table location is unknown until it is programmed after reset

(1) Although the associated Compute Cluster register provides support for a 50-bit wide reset base vector, the actual support is for a 40-bit reset base vector per SoC restriction.

#### 6.2.3.12 A72SS IPC with Other CPUs

The A72 cores can communicate with other device cores (R5F MCU, C66x DSP, C7x DSP, etc) by supporting interrupt generation to and from these cores. The interprocessor communication (IPC) interrupts are assigned in the corresponding Control Module memory-mapped registers (MMRs) called IPC\_SETx / IPC\_CLRx. For more information, see *Control Module (CTRL\_MMR)*.

## 6.3 Dual-R5F MCU Subsystem

This chapter describes the dual-R5F Microcontroller Unit (MCU) Subsystem (R5FSS) in the device.

### 6.3.1 R5FSS Overview

The R5FSS is a dual-core implementation of the Arm® Cortex®-R5F processor configured for split/lock operation. It also includes accompanying memories (L1 caches and tightly-coupled memories), standard Arm CoreSight™ debug and trace architecture, integrated vectored interrupt manager (VIM), ECC aggregators, and various other modules for protocol conversion and address translation for easy integration into the SoC.

#### Note

The Cortex-R5F processor is a Cortex-R5 processor that includes the optional floating point unit (FPU) extension. In this TRM, all references to the Cortex-R5 processor by default apply to the Cortex-R5F processor.

There are three R5FSS subsystems in the device. [Table 6-19](#) shows R5FSS allocation across device domains.

**Table 6-19. R5FSS Allocation Across Device Domains**

Module Instance	Domain		
	WKUP	MCU	MAIN
MCU_R5FSS0	–	✓	–
R5FSS0	–	–	✓
R5FSS1	–	–	✓

#### 6.3.1.1 R5FSS Features

Each R5FSS supports the following features:

- Dual-core Arm Cortex-R5F
  - Core revision: r1p3
  - Armv7-R profile
  - Split/lock operation
    - Split mode: Two independently operating cores (asymmetric multi processing, no coherence)
    - Lock (lockstep) mode: One main operating core with the other operating in lockstep
    - Boot-time configurable to be in split or lock mode
  - L1 memory system
    - 16KB instruction cache
      - 4x4KB ways
      - SECDED ECC protected per 64 bits
    - 16KB data cache
      - 4x4KB ways
      - SECDED ECC protected per 32 bits
    - 64KB tightly-coupled memory (TCM) per CPU
      - SECDED ECC protected per 32 bits
      - Readable/writable from system
      - Split into A and B banks (with B further splitting into B0 and B1 interleaved banks)
        - 32KB TCMA (ATCM)
        - 16KB TCMB0 (B0TCM)
        - 16KB TCMB1 (B1TCM)
  - Low interrupt latency with restartable instructions
  - Non-maskable interrupt (NMI)
  - Full-precision floating point (VFPv3)
  - 16 region memory protection unit (MPU)
  - 8 breakpoints

- 8 watchpoints
- Dynamic branch prediction with global history buffer and 4-entry return stack
- CoreSight debug access port (DAP)
- CoreSight embedded trace macrocell (ETM-R5) interface
- Performance monitoring unit (PMU)
- Interfaces
  - 64-bit VBUSM master pair (1 read, 1 write) for L3 memory accesses (per core)
  - 64-bit VBUSM slave for TCM access (per core)
    - Also allows access to cache for debug purposes
  - 32-bit VBUSM master pair (1 read, 1 write) for peripheral access
    - Note: This port is only supported for R5FSS0 and R5FSS1; it is not supported for MCU\_R5FSS0
  - 32-bit VBUSP master for peripheral access (per core)
  - 32-bit VBUSP slave configuration port (per core)
  - 32-bit VBUSP slave debug port
    - Allows access to all R5FSS internal debug logic

---

### Note

VBUSP is a pended protocol such that only a single transaction can be outstanding at any given time. Transactions are completed on VBUSP entirely before the next transaction is presented on the interface.

VBUSM allows multiple transactions to be outstanding which increases bus throughput.

- 
- Synchronous clock domain crossing on all interfaces
    - Interfaces can run at an integer division of the core frequency
  - Error detection logic (in lockstep mode only)
  - 32-bit to 48-bit region-based address translation (RAT) on memory access masters
    - 16 regions
      - Base address + size
      - Must be size aligned
  - Integrated vectored interrupt manager (VIM)
    - 512 interrupts per core
      - Only interrupts connected to R5F core 0 are available in lock mode
      - Each interrupt programmable as either IRQ or FIQ
      - Each interrupt has a programmable enable mask
      - Each interrupt has a programmable 4-bit priority
    - Priority interrupt supported
    - Vectored interrupt interface
      - Compatible with R5F VIC port
      - Programmable 32-bit vector address per interrupt
        - Address is SECEDED error protected
        - Default vector addresses provided on DED
      - Split or lockstep capable
      - Software interrupt generation
  - Integrated ECC aggregators
    - Support for error injection to all supported ECC memory blocks to test ECC functionality (add-on function from TI)
    - One ECC aggregator per core to cover all RAMs and caches associated with that core
  - Standard Arm CoreSight debug and trace architecture at the R5FSS level
    - Cross triggering: Supported by cross trigger interface (CTI) (per CPU) and cross trigger matrix (CTM) components
    - Processor trace: Supported by embedded trace macrocell (ETM) (per CPU) and advanced trace bus (ATB) funnel components
  - Boot

- From ROM or external memory
- From TCM

See [Section 6.3.3](#) for a functional block diagram and more details on the R5FSS.

#### 6.3.1.2 R5FSS Not Supported Features

The R5FSS does *not* support the following native R5F features in this device:

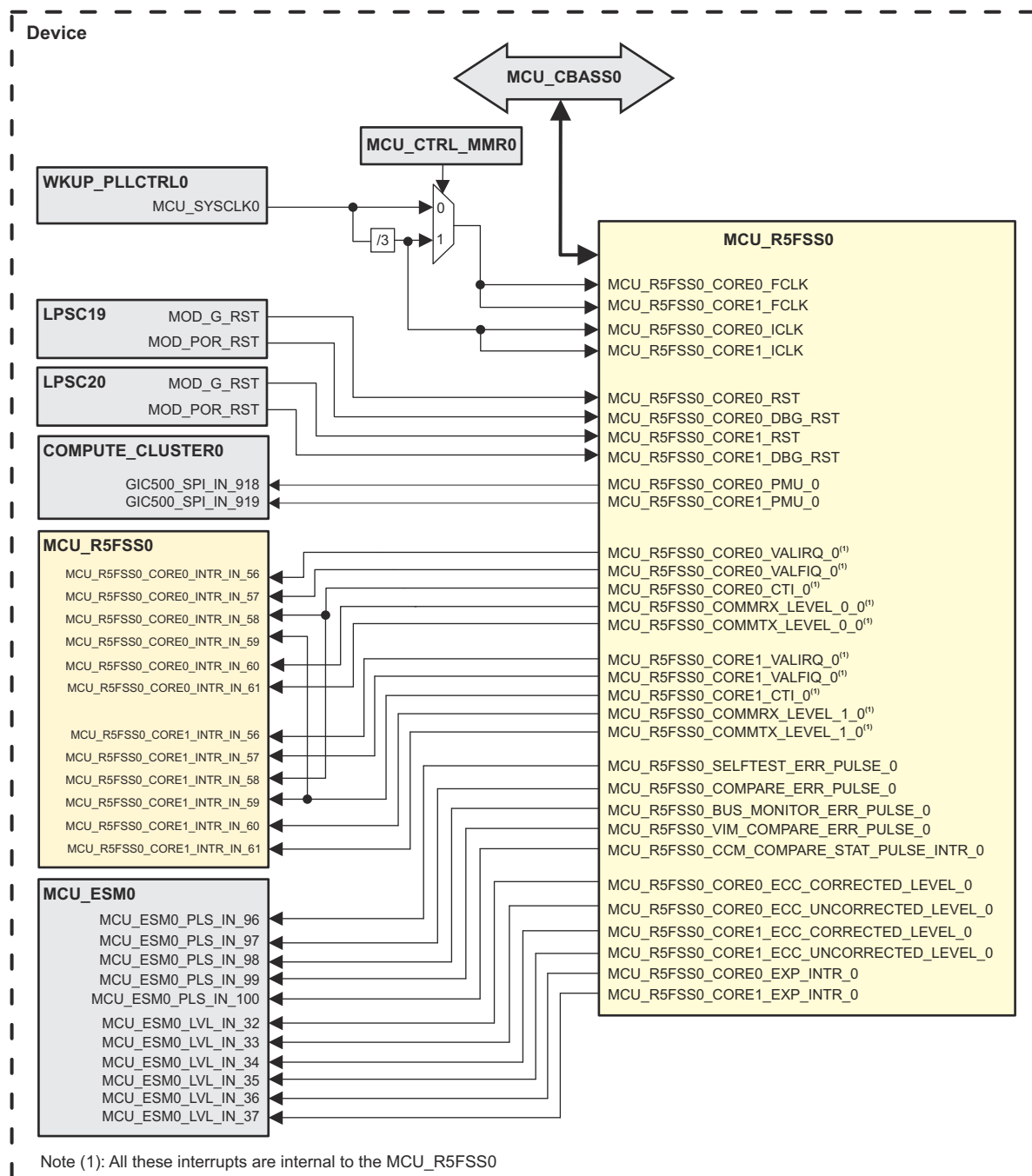
- ACP port (no coherence)
- Bus parity / ECC

## 6.3.2 R5FSS Integration

This section describes the R5FSS integration in the device, including information about clocks, resets, and hardware requests.

### 6.3.2.1 R5FSS Integration in MCU Domain

There is a single R5FSS subsystem integrated in the device MCU domain - MCU\_R5FSS0. Figure 6-4 shows the MCU\_R5FSS0 integration.



**Figure 6-4. MCU\_R5FSS0 Integration**

Table 6-20 through Table 6-22 summarize the MCU\_R5FSS0 integration.



**Table 6-20. MCU\_R5FSS0 Integration Attributes**

Module Instance		Attributes			
		Power Sleep Controller	Power Domain	Module Domain	Interconnect
MCU_R5FSS0	MCU_R5FSS0_CORE0	WKUP_PSC0	PD1	LPSC19	MCU_CBASS0
	MCU_R5FSS0_CORE1	WKUP_PSC0	PD1	LPSC20	MCU_CBASS0

**Table 6-21. MCU\_R5FSS0 Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
MCU_R5FSS0	MCU_R5FSS0_CORE0_FCLK	MCU_SYSCLK0 (default) or MCU_SYSCLK0/3 <sup>(1)</sup>	WKUP_PLLCT RL0	MCU_R5FSS0_CORE0 functional clock
	MCU_R5FSS0_CORE0_ICLK	MCU_SYSCLK0/3	WKUP_PLLCT RL0	MCU_R5FSS0_CORE0 interface clock
	MCU_R5FSS0_CORE1_FCLK	MCU_SYSCLK0 (default) or MCU_SYSCLK0/3 <sup>(1)</sup>	WKUP_PLLCT RL0	MCU_R5FSS0_CORE1 functional clock
	MCU_R5FSS0_CORE1_ICLK	MCU_SYSCLK0/3	WKUP_PLLCT RL0	MCU_R5FSS0_CORE1 interface clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
MCU_R5FSS0	MCU_R5FSS0_CORE0_RST	MOD_G_RST	LPSC19	MCU_R5FSS0_CORE0 main reset
	MCU_R5FSS0_CORE0_DBG_RST	MOD_POR_RST	LPSC19	MCU_R5FSS0_CORE0 debug reset (APB excluded)
	MCU_R5FSS0_CORE1_RST	MOD_G_RST	LPSC20	MCU_R5FSS0_CORE1 main reset
	MCU_R5FSS0_CORE1_DBG_RST	MOD_POR_RST	LPSC20	MCU_R5FSS0_CORE1 debug reset (APB excluded)

(1) Source clock selection is done via Control Module register (CTRLMMR\_MCU\_R5\_CORE0\_CLKSEL). MCU\_R5FSS0\_CORE1 always receives the same clock as MCU\_R5FSS0\_CORE0 based on this register.

**Table 6-22. MCU\_R5FSS0 Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCU_R5FSS0	MCU_R5FSS0_CORE0 interrupts				
	MCU_R5FSS0_CORE0_PMU_0	GIC500_SPI_IN_918	COMPUTE_CLUSTER0	MCU_R5FSS0_CORE0 performance monitor interrupt	Level
	MCU_R5FSS0_CORE0_VALIRQ_0	MCU_R5FSS0_CORE0_IN_TR_IN_56	MCU_R5FSS0_CORE0	MCU_R5FSS0_CORE0 validation IRQ interrupt	Level
	MCU_R5FSS0_CORE0_VALFIQ_0	MCU_R5FSS0_CORE0_IN_TR_IN_57	MCU_R5FSS0_CORE0	MCU_R5FSS0_CORE0 validation FIQ interrupt	Level
	MCU_R5FSS0_CORE0_CTI_0	MCU_R5FSS0_CORE0_IN_TR_IN_58	MCU_R5FSS0_CORE0	MCU_R5FSS0_CORE0 cross trigger interrupt	Level
	MCU_R5FSS0_COMMRX_LEVEL_0_0	MCU_R5FSS0_CORE0_IN_TR_IN_60	MCU_R5FSS0_CORE0	MCU_R5FSS0_CORE0 DTRRX full interrupt	Level

**Table 6-22. MCU\_R5FSS0 Hardware Requests (continued)**

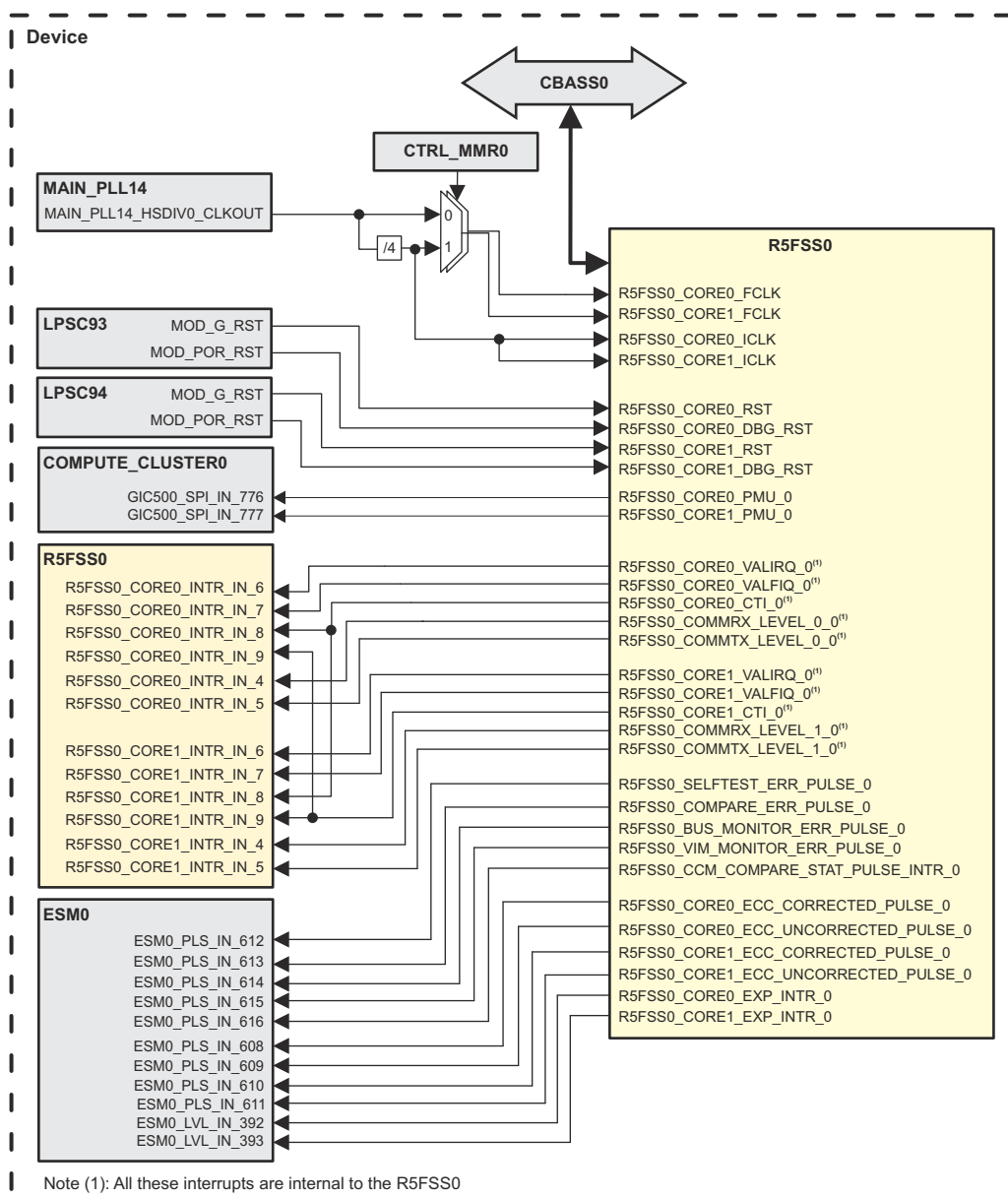
MCU_R5FSS0_COMMTX_LEVEL_0_0	MCU_R5FSS0_CORE0_IN TR_IN_61	MCU_R5FSS0_CO RE0	MCU_R5FSS0_CO RE0 DTRTX empty interrupt	Level
MCU_R5FSS0_CORE0_ECC_CORRECTED _LEVEL_0	MCU_ESM0_LVL_IN_32	MCU_ESM0	MCU_R5FSS0_CO RE0 SEC ECC interrupt	Level
MCU_R5FSS0_CORE0_ECC_UNCORRECT ED_LEVEL_0	MCU_ESM0_LVL_IN_33	MCU_ESM0	MCU_R5FSS0_CO RE0 DED ECC interrupt	Level
MCU_R5FSS0_CORE0_EXP_INTR_0	MCU_ESM0_LVL_IN_36	MCU_ESM0	MCU_R5FSS0_CO RE0 RAT exception interrupt	Level
<b>MCU_R5FSS0_CORE1 interrupts</b>				
MCU_R5FSS0_CORE1_PMU_0	GIC500_SPI_IN_919	COMPUTE_CLUST ER0	MCU_R5FSS0_CO RE1 performance monitor interrupt	Level
MCU_R5FSS0_CORE1_VALIRQ_0	MCU_R5FSS0_CORE1_IN TR_IN_56	MCU_R5FSS0_CO RE1	MCU_R5FSS0_CO RE1 validation IRQ interrupt	Level
MCU_R5FSS0_CORE1_VALFIQ_0	MCU_R5FSS0_CORE1_IN TR_IN_57	MCU_R5FSS0_CO RE1	MCU_R5FSS0_CO RE1 validation FIQ interrupt	Level
MCU_R5FSS0_CORE1_CTI_0	MCU_R5FSS0_CORE1_IN TR_IN_59	MCU_R5FSS0_CO RE1	MCU_R5FSS0_CO RE1 cross trigger interrupt	Level
MCU_R5FSS0_COMMRX_LEVEL_1_0	MCU_R5FSS0_CORE1_IN TR_IN_59	MCU_R5FSS0_CO RE0	MCU_R5FSS0_CO RE1 DTRRX full interrupt	Level
	MCU_R5FSS0_CORE1_IN TR_IN_60	MCU_R5FSS0_CO RE1		
MCU_R5FSS0_COMMTX_LEVEL_1_0	MCU_R5FSS0_CORE1_IN TR_IN_61	MCU_R5FSS0_CO RE1	MCU_R5FSS0_CO RE1 DTRTX empty interrupt	Level
MCU_R5FSS0_CORE1_ECC_CORRECTED _LEVEL_0	MCU_ESM0_LVL_IN_34	MCU_ESM0	MCU_R5FSS0_CO RE1 SEC ECC interrupt	Level
MCU_R5FSS0_CORE1_ECC_UNCORRECT ED_LEVEL_0	MCU_ESM0_LVL_IN_35	MCU_ESM0	MCU_R5FSS0_CO RE1 DED ECC interrupt	Level
MCU_R5FSS0_CORE1_EXP_INTR_0	MCU_ESM0_LVL_IN_37	MCU_ESM0	MCU_R5FSS0_CO RE1 RAT exception interrupt	Level
<b>MCU_R5FSS0_CCMR5 interrupts</b>				
MCU_R5FSS0_SELFTEST_ERR_PULSE_0	MCU_ESM0_PLS_IN_96	MCU_ESM0	MCU_R5FSS0 self- test failure interrupt	Pulse
MCU_R5FSS0_COMPARE_ERR_PULSE_0	MCU_ESM0_PLS_IN_97	MCU_ESM0	MCU_R5FSS0 CPU bus compare failure interrupt	Pulse
MCU_R5FSS0_BUS_MONITOR_ERR_PULS E_0	MCU_ESM0_PLS_IN_98	MCU_ESM0	MCU_R5FSS0 inactivity monitor failure interrupt	Pulse
MCU_R5FSS0_VIM_COMPARE_ERR_PULS E_0	MCU_ESM0_PLS_IN_99	MCU_ESM0	MCU_R5FSS0 VIM bus compare failure interrupt	Pulse

**Table 6-22. MCU\_R5FSS0 Hardware Requests (continued)**

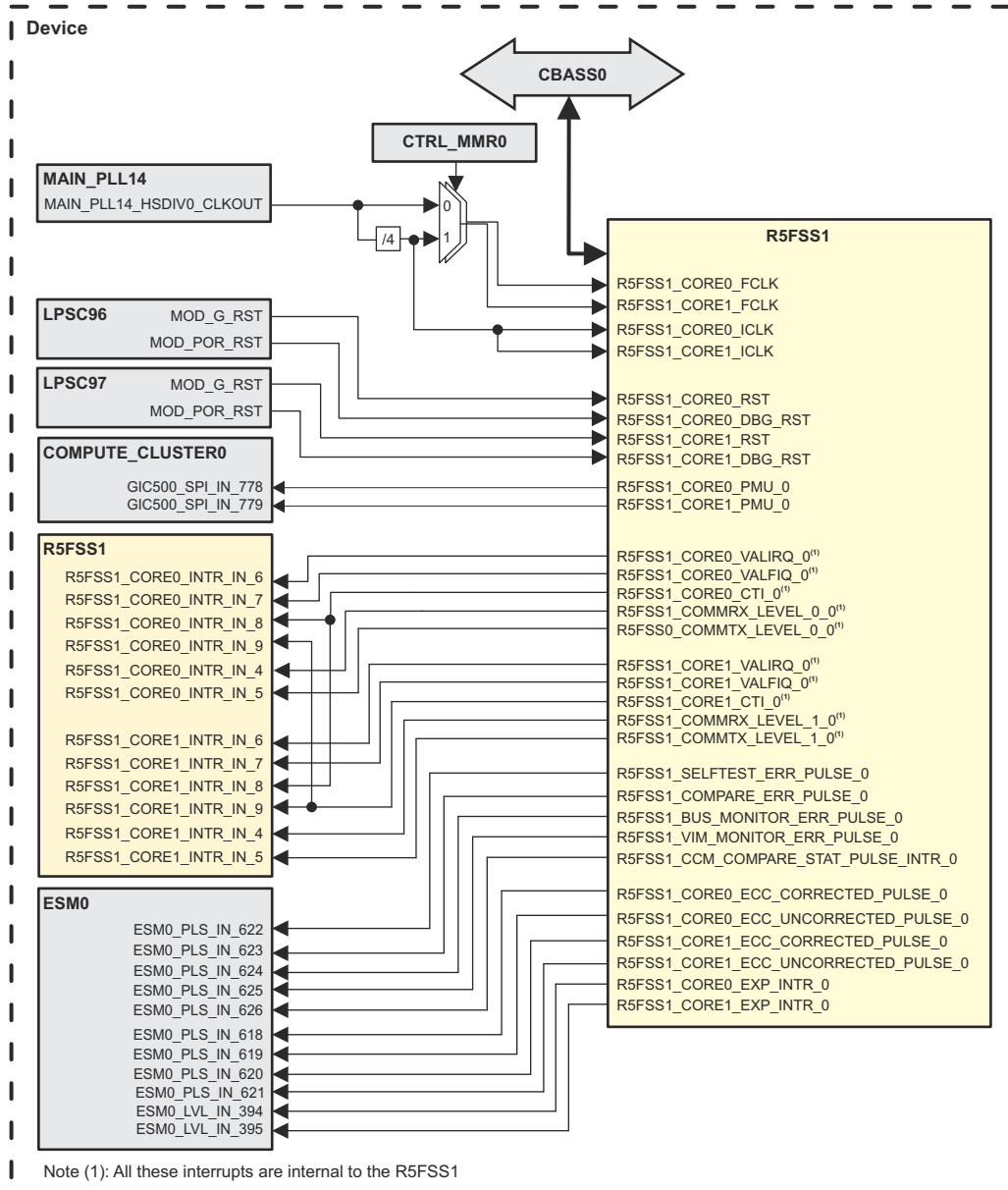
MCU_R5FSS0_CCM_COMPARE_STAT_PUL	MCU_ESM0_PLS_IN_100	MCU_ESM0	MCU_R5FSS0 CCMR5 in self-test or split mode interrupt	Pulse
SE_INTR_0				

### 6.3.2.2 R5FSS Integration in MAIN Domain

There are two R5FSS subsystems integrated in the device MAIN domain - R5FSS0 and R5FSS1. [Figure 6-5](#) and [Figure 6-6](#) show the integration of R5FSS0 and R5FSS1, respectively.



**Figure 6-5. R5FSS0 Integration**


**Figure 6-6. R5FSS1 Integration**
**Table 6-23. R5FSS0/1 Integration Attributes**

Module Instance		Attributes			
		Power Sleep Controller	Power Domain	Module Domain	Interconnect
R5FSS0	R5FSS0_CORE0	PSC0	PD24	LPSC93	MCU_CBASS0
	R5FSS0_CORE1	PSC0	PD24	LPSC94	MCU_CBASS0
R5FSS1	R5FSS1_CORE0	PSC0	PD25	LPSC96	MCU_CBASS0
	R5FSS1_CORE1	PSC0	PD25	LPSC97	MCU_CBASS0

**Table 6-24. R5FSS0/1 Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description

**Table 6-24. R5FSS0/1 Clocks and Resets (continued)**

R5FSS0	R5FSS0_CORE0_FCLK	MAIN_PLL14_HSDIV0_CLKOUT	MAIN_PLL14	R5FSS0_CORE0 functional clock
	R5FSS0_CORE0_ICLK	MAIN_PLL14_HSDIV0_CLKOUT/4	MAIN_PLL14	R5FSS0_CORE0 interface clock
	R5FSS0_CORE1_FCLK	MAIN_PLL14_HSDIV0_CLKOUT	MAIN_PLL14	R5FSS0_CORE1 functional clock
	R5FSS0_CORE1_ICLK	MAIN_PLL14_HSDIV0_CLKOUT/4	MAIN_PLL14	R5FSS0_CORE1 interface clock
R5FSS1	R5FSS1_CORE0_FCLK	MAIN_PLL14_HSDIV1_CLKOUT	MAIN_PLL14	R5FSS1_CORE0 functional clock
	R5FSS1_CORE0_ICLK	MAIN_PLL14_HSDIV1_CLKOUT/4	MAIN_PLL14	R5FSS1_CORE0 interface clock
	R5FSS1_CORE1_FCLK	MAIN_PLL14_HSDIV1_CLKOUT	MAIN_PLL14	R5FSS1_CORE1 functional clock
	R5FSS1_CORE1_ICLK	MAIN_PLL14_HSDIV1_CLKOUT/4	MAIN_PLL14	R5FSS1_CORE1 interface clock

Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
R5FSS0	R5FSS0_CORE0_RST	MOD_G_RST	LPSC93	R5FSS0_CORE0 main reset
	R5FSS0_CORE0_DBG_RST	MOD_DBG_POR_RST	LPSC93	R5FSS0_CORE0 debug reset (APB excluded)
	R5FSS0_CORE1_RST	MOD_G_RST	LPSC94	R5FSS0_CORE1 main reset
	R5FSS0_CORE1_DBG_RST	MOD_POR_RST	LPSC94	R5FSS0_CORE1 debug reset (APB excluded)
R5FSS1	R5FSS1_CORE0_RST	MOD_G_RST	LPSC96	R5FSS1_CORE0 main reset
	R5FSS1_CORE0_DBG_RST	MOD_POR_RST	LPSC96	R5FSS1_CORE0 debug reset (APB excluded)
	R5FSS1_CORE1_RST	MOD_G_RST	LPSC97	R5FSS1_CORE1 main reset
	R5FSS1_CORE1_DBG_RST	MOD_POR_RST	LPSC97	R5FSS1_CORE1 debug reset (APB excluded)

**Table 6-25. R5FSS0/1 Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
R5FSS0_CORE0 interrupts					
R5FSS0	R5FSS0_CORE0_PMU_0	GIC500_SPI_IN_776	COMPUTE_CLUSTER0	R5FSS0_CORE0 performance monitor interrupt	Level
	R5FSS0_CORE0_VALFIQ_0	R5FSS0_CORE0_INT_R_IN_6	R5FSS0_CORE0	R5FSS0_CORE0 validation IRQ interrupt	Level
	R5FSS0_CORE0_VALIRQ_0	R5FSS0_CORE0_INT_R_IN_7	R5FSS0_CORE0	R5FSS0_CORE0 validation FIQ interrupt	Level
	R5FSS0_CORE0_CTI_0	R5FSS0_CORE0_INT_R_IN_8	R5FSS0_CORE0	R5FSS0_CORE0 cross trigger interrupt	Level
		R5FSS0_CORE1_INT_R_IN_8	R5FSS0_CORE1		
	R5FSS0_COMMRX_LEVEL_0_0	R5FSS0_CORE0_INT_R_IN_4	R5FSS0_CORE0	R5FSS0_CORE0 DTRRX full interrupt	Level
	R5FSS0_COMMTX_LEVEL_0_0	R5FSS0_CORE0_INT_R_IN_5	R5FSS0_CORE0	R5FSS0_CORE0 DTRTX empty interrupt	Level
	R5FSS0_CORE0_ECC_CORRECTED_PULSE_0	ESM0_PLS_IN_608	ESM0	R5FSS0_CORE0 SEC ECC interrupt	Level
	R5FSS0_CORE0_ECC_UNCORRECTED_PULSE_0	ESM0_PLS_IN_609	ESM0	R5FSS0_CORE0 DED ECC interrupt	Level

**Table 6-25. R5FSS0/1 Hardware Requests (continued)**

R5FSS0_CORE0_EXP_INTR_0	R5FSS0_CORE0_INT R_IN_16	R5FSS0_CORE0	R5FSS0_CORE0 RAT exception interrupt	Level
	R5FSS0_CORE1_INT R_IN_16	R5FSS0_CORE1		
	ESM0_LVL_IN_392	ESM0		
<b>R5FSS0_CORE1 interrupts</b>				
R5FSS0_CORE1_PMU_0	GIC500_SPI_IN_777	COMPUTE_CLUS TER0	R5FSS0_CORE1 performance monitor interrupt	Level
R5FSS0_CORE1_VALFIQ_0	R5FSS0_CORE1_INT R_IN_6	R5FSS0_CORE1	R5FSS0_CORE1 validation IRQ interrupt	Level
R5FSS0_CORE1_VALIRQ_0	R5FSS0_CORE1_INT R_IN_7	R5FSS0_CORE1	R5FSS0_CORE1 validation FIQ interrupt	Level
R5FSS0_CORE1_CTI_0	R5FSS0_CORE1_INT R_IN_9	R5FSS0_CORE0	R5FSS0_CORE1 cross trigger interrupt	Level
	R5FSS0_CORE0_INT R_IN_9	R5FSS0_CORE1		
R5FSS0_COMMRX_LEVEL_1_0	R5FSS0_CORE1_INT R_IN_4	R5FSS0_CORE1	R5FSS0_CORE1 DTRRX full interrupt	Level
R5FSS0_COMMTX_LEVEL_1_0	R5FSS0_CORE1_INT R_IN_5	R5FSS0_CORE1	R5FSS0_CORE1 DTRTX empty interrupt	Level
R5FSS0_CORE1_ECC_CORRECTED_ PULSE_0	ESM0_PLS_IN_610	ESM0	R5FSS0_CORE1 SEC ECC interrupt	Level
R5FSS0_CORE1_ECC_UNCORRECTED_ PULSE_0	ESM0_PLS_IN_611	ESM0	R5FSS0_CORE1 DED ECC interrupt	Level
R5FSS0_CORE1_EXP_INTR_0	R5FSS0_CORE0_INT R_IN_17	R5FSS0_CORE0	R5FSS0_CORE1 RAT exception interrupt	Level
	R5FSS0_CORE0_INT R_IN_17	R5FSS0_CORE1		
	ESM0_LVL_IN_393	ESM0		
<b>R5FSS0_CCMR5 interrupts</b>				
R5FSS0_SELFTEST_ERR_PULSE_0	ESM0_PLS_IN_612	ESM0	R5FSS0 self-test failure interrupt	Pulse
R5FSS0_COMPARE_ERR_PULSE_0	ESM0_PLS_IN_613	ESM0	R5FSS0 CPU bus compare failure interrupt	Pulse
R5FSS0_BUS_MONITOR_ERR_PULSE_ _0	ESM0_PLS_IN_614	ESM0	R5FSS0 inactivity monitor failure interrupt	Pulse
R5FSS0_VIM_COMPARE_ERR_PULSE_ _0	ESM0_PLS_IN_615	ESM0	R5FSS0 VIM bus compare failure interrupt	Pulse
R5FSS0_CCM_COMPARE_STAT_PULS E_INTR_0	ESM0_PLS_IN_616	ESM0	R5FSS0 CCMR5 in self-test or split mode interrupt	Pulse
<b>R5FSS1_CORE0 interrupts</b>				
R5FSS1_CORE0_PMU_0	GIC500_SPI_IN_778	COMPUTE_CLUS TER0	R5FSS1_CORE0 performance monitor interrupt	Level
R5FSS1_CORE0_VALFIQ_0	R5FSS1_CORE0_INT R_IN_6	R5FSS1_CORE0	R5FSS1_CORE0 validation IRQ interrupt	Level
R5FSS1_CORE0_VALIRQ_0	R5FSS1_CORE0_INT R_IN_7	R5FSS1_CORE0	R5FSS1_CORE0 validation FIQ interrupt	Level
R5FSS1_CORE0_CTI_0	R5FSS1_CORE0_INT R_IN_8	R5FSS1_CORE0	R5FSS1_CORE0 cross trigger interrupt	Level
	R5FSS1_CORE1_INT R_IN_8	R5FSS1_CORE1		
R5FSS1_COMMRX_LEVEL_0_0	R5FSS1_CORE0_INT R_IN_4	R5FSS1_CORE0	R5FSS1_CORE0 DTRRX full interrupt	Level

R5FSS  
1

**Table 6-25. R5FSS0/1 Hardware Requests (continued)**

R5FSS1_COMMTX_LEVEL_0_0	R5FSS1_CORE0_INT R_IN_5	R5FSS1_CORE0	R5FSS1_CORE0 DTRTX empty interrupt	Level
R5FSS1_CORE0_ECC_CORRECTED_ PULSE_0	ESM0_PLS_IN_618	ESM0	R5FSS1_CORE0 SEC ECC interrupt	Level
R5FSS1_CORE0_ECC_UNCORRECTE D_PULSE_0	ESM0_PLS_IN_619	ESM0	R5FSS1_CORE0 DED ECC interrupt	Level
R5FSS1_CORE0_EXP_INTR_0	R5FSS1_CORE0_INT R_IN_16	R5FSS1_VIM0	R5FSS1_CORE0 RAT exception interrupt	Level
	R5FSS1_CORE0_INT R_IN_16	R5FSS1_VIM1		
	ESM0_LVL_IN_394	ESM0		
R5FSS1_CORE1 interrupts				
R5FSS1_CORE1_PMU_0	GIC500_SPI_IN_779	COMPUTE_CLUS TER0	R5FSS1_CORE1 performance monitor interrupt	Level
R5FSS1_CORE1_VALFIQ_0	R5FSS1_CORE1_INT R_IN_6	R5FSS0_CORE1	R5FSS1_CORE1 validation IRQ interrupt	Level
R5FSS1_CORE1_VALIRQ_0	R5FSS1_CORE1_INT R_IN_7	R5FSS0_CORE1	R5FSS1_CORE1 validation FIQ interrupt	Level
R5FSS1_CORE1_CTI_0	R5FSS1_CORE1_INT R_IN_9	R5FSS0_CORE0	R5FSS1_CORE1 cross trigger interrupt	Level
	R5FSS1_CORE0_INT R_IN_9	R5FSS0_CORE1		
R5FSS1_COMMRX_LEVEL_1_0	R5FSS1_CORE1_INT R_IN_4	R5FSS0_CORE1	R5FSS1_CORE1 DTRRX full interrupt	Level
R5FSS1_COMMTX_LEVEL_1_0	R5FSS1_CORE1_INT R_IN_5	R5FSS0_CORE1	R5FSS1_CORE1 DTRTX empty interrupt	Level
R5FSS1_CORE1_ECC_CORRECTED_ PULSE_0	ESM0_PLS_IN_620	ESM0	R5FSS1_CORE1 SEC ECC interrupt	Level
R5FSS1_CORE1_ECC_UNCORRECTE D_PULSE_0	ESM0_PLS_IN_621	ESM0	R5FSS1_CORE1 DED ECC interrupt	Level
R5FSS1_CORE1_EXP_INTR_0	R5FSS1_VIM0_IN_17	R5FSS1_VIM0	R5FSS1_CORE1 RAT exception interrupt	Level
	R5FSS1_VIM1_IN_17	R5FSS1_VIM1		
	ESM0_LVL_IN_395	ESM0		
R5FSS1_CCMR5 interrupts				
R5FSS1_SELFTEST_ERR_PULSE_0	ESM0_PLS_IN_622	ESM0	R5FSS1 self-test failure interrupt	Pulse
R5FSS1_COMPARE_ERR_PULSE_0	ESM0_PLS_IN_623	ESM0	R5FSS1 CPU bus compare failure interrupt	Pulse
R5FSS1_BUS_MONITOR_ERR_PULSE _0	ESM0_PLS_IN_624	ESM0	R5FSS1 inactivity monitor failure interrupt	Pulse
R5FSS1_VIM_COMPARE_ERR_PULSE _0	ESM0_PLS_IN_625	ESM0	R5FSS1 VIM bus compare failure interrupt	Pulse
R5FSS1_CCM_COMPARE_STAT_PULS E_INTR_0	ESM0_PLS_IN_626	ESM0	R5FSS1 CCMR5 in self-test or split mode interrupt	Pulse





restriction is that CPU0 must be in a higher power/reset state than CPU1. For instance, CPU1 cannot be out of reset if CPU0 is not.

In lockstep mode, the core logic from CPU1 is used as redundant logic to check for errors in CPU0. The CPU1 interfaces and RAMs are not used. Comparison logic automatically checks the redundant logic against the primary logic and flags any errors.

For a brief list of features supported by the R5F processor in this device, see [Section 6.3.1.1](#). For more detailed description of this processor, see the *Arm Cortex-R5 Technical Reference Manual*.

#### 6.3.3.2.1 L1 Caches

The R5F has a Harvard cache architecture, which means it has an independent L1 instruction cache (16KB) and L1 data cache (16KB). The instruction cache is protected by SECDED ECC per 64 bits. The data cache is protected by SECDED ECC per 32 bits.

#### 6.3.3.2.2 Tightly-Coupled Memories (TCMs)

The R5F has two tightly-coupled memories (TCMs), ATCM and BTCM. The BTCM is further broken down into two interleaved banks, B0TCM and B1TCM.

TCMs are low-latency, tightly integrated memories for the R5F to use. Either TCM can be used for any combination of instruction and/or data. TCM performance is equal to performance on instructions/data that are in cache. However, TCMs have some additional advantages over cache. TCMs can be loaded with instructions that do not cache well (such as ISRs) or preloaded with code by an external source, before that code is needed, to save cache miss time. TCMs are also a good place for blocks of data for intense processing. They can be loaded (or pre-loaded by an external source) before the data is needed, saving cache miss time. The data can then be directly accessed by an external source, instead of needing to do cache evicts.

As mentioned, TCMs can be accessed (either read or written) by an external source over the TCM VBUSM slave interface. This allows instructions or data to be preloaded, or for data to be read out after the R5F has processed it. The VBUSM slave has a lower priority to accessing TCMs than the R5F but care must be taken to keep an external source from reading or writing TCM data that the R5F is working on. This handshaking is external to any of the R5FSS hardware.

TCMs are protected by ECC per 32 bits. For this to work, ECC must be enabled before data is written in to the TCMs (either externally or from the R5F). ECC is enabled via the following R5F system control bits: ACTLR.ATCMPCEN, ACTLR.B0TCMPCEN, and ACTLR.B1TCMPCEN, respectively.

Whether or not the TCMs are enabled is controlled by the ENABLE bit in the corresponding ATCM/BTCM region register. The default (reset) value of this bit is determined by the CPU<sub>n</sub>\_INITRAMA and CPU<sub>n</sub>\_INITRAMB bootstraps, respectively. Both ATCM and BTCM are configured for a size of 32KB in this device. Note that the BTCM size is the total of both B0TCM and B1TCM (16KB each).

If a TCM is not enabled, then it does not appear in the R5F's memory view, but it can be accessed by an external source. If a TCM is enabled, then its place in the R5F memory map is determined by a combination of bootstrap signal and system register. If the CPU<sub>n</sub>\_LOCZRAMA bootstrap signal is high, then the initial base address of ATCM is 0x0000\_0000 and the initial address of BTCM is 20'h41010. If the CPU<sub>n</sub>\_LOCZRAMA bootstrap signal is low, then the initial base address of BTCM is 0x0000\_0000 and the initial base address of ATCM is 20'h41010.

#### Note

This base address of 0x41010 for ATCM/BTCM based on the CPU<sub>n</sub>\_LOCZRAMA bootstrap only affects the R5F's memory view. The SoC will see the ATCM/BTCM based on the TCM slave interface regions, as defined in [Section 6.3.3.2](#). The base address of either TCM may be overwritten via the ATCM or BTCM region register. Care must be taken not to move the base address of a TCM when it may be being accessed.

It is possible to preload a TCM with instructions and boot from it. See [Section 6.3.3.12](#) for details on TCM booting.

### 6.3.3.2.3 R5FSS Special Signals

[Table 6-26](#) through [Table 6-28](#) list some R5FSS features associated with special signals. Note that in lockstep mode only those for CPU0 apply.

**Table 6-26. MCU\_R5FSS0 Special Features**

Feature	Comment
Cluster affinity group ID	R5F cluster 0 (ID = 0x0)
Exception handling state at reset 0 = Arm 1 = Thumb	Controlled via MCU_SEC_MMR register setting. Defaults to Arm mode
Split or lockstep mode 0 = Split mode 1 = Lockstep mode	Controlled via MCU_SEC_MMR register setting. Defaults to a value defined by eFuse
CPU <sub>n</sub> execution halt when coming out of reset (CPU <sub>n</sub> _HALT)	Controlled via MCU_SEC_MMR register setting. Defaults to halted state
CPU <sub>n</sub> exception vectors base address	Controlled via MCU_SEC_MMR register setting. Defaults to ROM address 0x0000_4110_0000
CPU <sub>n</sub> VIM base address	0x40F8_0000
CPU <sub>n</sub> RAT base address	0x40F9_0000
CPU <sub>n</sub> RAT accesses ID	0x2 (CPU0); 0x3 (CPU1)
CPU <sub>n</sub> ATCM enable at reset (CPU <sub>n</sub> _INITRAMA)	Controlled via MCU_SEC_MMR register setting. Defaults to disabled state
CPU <sub>n</sub> BTCM enable at reset (CPU <sub>n</sub> _INITRAMB)	Controlled via MCU_SEC_MMR register setting. Defaults to enabled state
CPU <sub>n</sub> A/BTCM reset base address indicator (CPU <sub>n</sub> _LOCZRAMA) 0 = B at 0x0 1 = A at 0x0	Controlled via MCU_SEC_MMR register setting. Defaults to 1
CPU <sub>n</sub> non-maskable fast interrupts enable	Controlled via MCU_SEC_MMR register setting. Defaults to disabled state
CPU <sub>n</sub> VBUSM peripheral port enabled at reset	VBUSM port not used
CPU <sub>n</sub> VBUSP peripheral port enable at reset	Enabled
CPU <sub>n</sub> VBUSP peripheral port base address	Mapped to 0x0_4000_0000 MCU peripheral base address
CPU <sub>n</sub> VBUSP peripheral port size	16MB for MCUSP peripherals (0x0_4000_0000 to 0x0_40FF_FFFF)
CPU <sub>n</sub> VBUSM normal peripheral port base address	VBUSM port not used
CPU <sub>n</sub> VBUSM normal peripheral port size	VBUSM port not used
CPU <sub>n</sub> VBUSM virtual peripheral port base address	VBUSM port not used
CPU <sub>n</sub> VBUSM virtual peripheral port size	VBUSM port not used
CPU <sub>n</sub> clock stopped indication	Status logged into MCU_SEC_MMR register bit
CPU <sub>n</sub> WFI state	Status logged into MCU_SEC_MMR register bit
CPU <sub>n</sub> WFE state	Status logged into MCU_SEC_MMR register bit
CPU clockstop behavior 0: CPU clocks stopped in standby 1: CPU clocks not stopped in standby	Controlled via MCU_SEC_MMR register setting. Defaults to 0

**Table 6-27. R5FSS0 Special Features**

Feature	Comment
Cluster affinity group ID	R5F Cluster 1 (ID = 0x1)
Exception handling state at reset 0 = Arm 1 = Thumb	Controlled via MAIN_SEC_MMR register setting. Defaults to Arm mode

**Table 6-27. R5FSS0 Special Features (continued)**

Feature	Comment
Split or lockstep mode 0 = Split mode 1 = Lockstep mode	Controlled via MAIN_SEC_MMR register setting. Defaults to a value defined by eFuse
CPU <sub>n</sub> execution halt when coming out of reset (CPU <sub>n</sub> _HALT)	Controlled via MAIN_SEC_MMR register setting. Defaults to halted state
CPU <sub>n</sub> exception vectors base address	Controlled via MAIN_SEC_MMR register setting. Defaults to Bootvector RAM address 0x0000_0000_0200
CPU <sub>n</sub> VIM base address	0x0FF8_0000
CPU <sub>n</sub> RAT base address	0x0FF9_0000
CPU <sub>n</sub> RAT accesses ID	0x4 (CPU0); 0x5 (CPU1)
CPU <sub>n</sub> ATCM enable at reset (CPU <sub>n</sub> _INITRAMA)	Controlled via MAIN_SEC_MMR register setting. Defaults to disabled state
CPU <sub>n</sub> BTCM enable at reset (CPU <sub>n</sub> _INITRAMB)	Controlled via MAIN_SEC_MMR register setting. Defaults to enabled state
CPU <sub>n</sub> A/BTCM reset base address indicator (CPU <sub>n</sub> _LOCZRAMA) 0 = B at 0x0 1 = A at 0x0	Controlled via MAIN_SEC_MMR register setting. Defaults to 1
CPU <sub>n</sub> non-maskable fast interrupts enable	Controlled via MAIN_SEC_MMR register setting. Defaults to disabled state
CPU <sub>n</sub> VBUSM peripheral port enabled at reset	Enabled
CPU <sub>n</sub> VBUSP peripheral port enable at reset	Enabled
CPU <sub>n</sub> VBUSP peripheral port base address	Mapped to 0x0_0C00_0000 for low latency MAIN peripherals
CPU <sub>n</sub> VBUSP peripheral port size	64MB for MAIN peripherals (0x0_0C00_0000 to 0x0_0FFF_FFFF)
CPU <sub>n</sub> VBUSM normal peripheral port base address	Mapped to 0x0200_0000 for MAIN peripherals
CPU <sub>n</sub> VBUSM normal peripheral port size	16MB for MAIN peripherals (0x0_0200_0000 to 0x0_02FF_FFFF)
CPU <sub>n</sub> VBUSM virtual peripheral port base address	Mapped to 0x0200_0000 for MAIN peripherals
CPU <sub>n</sub> VBUSM virtual peripheral port size	16MB for MAIN peripherals (0x0_0200_0000 to 0x0_02FF_FFFF)
CPU <sub>n</sub> clock stopped indication	Status logged into MAIN_SEC_MMR register bit
CPU <sub>n</sub> WFI state	Status logged into MAIN_SEC_MMR register bit
CPU <sub>n</sub> WFE state	Status logged into MAIN_SEC_MMR register bit
CPU clockstop behavior 0: CPU clocks stopped in standby 1: CPU clocks not stopped in standby	Controlled via MAIN_SEC_MMR register setting. Defaults to 0

**Table 6-28. R5FSS1 Special Features**

Feature	Comment
Cluster affinity group ID	R5F Cluster 2 (ID = 0x2)
Exception handling state at reset 0 = Arm 1 = Thumb	Controlled via MAIN_SEC_MMR register setting. Defaults to Arm mode
Split or lockstep mode 0 = Split mode 1 = Lockstep mode	Controlled via MAIN_SEC_MMR register setting. Defaults to a value defined by eFuse
CPU <sub>n</sub> execution halt when coming out of reset (CPU <sub>n</sub> _HALT)	Controlled via MAIN_SEC_MMR register setting. Defaults to halted state
CPU <sub>n</sub> exception vectors base address	Controlled via MAIN_SEC_MMR register setting. Defaults to Bootvector RAM address 0x0000_0000_0200
CPU <sub>n</sub> VIM base address	0x0FF8_0000
CPU <sub>n</sub> RAT base address	0x0FF9_0000
CPU <sub>n</sub> RAT accesses ID	0x6 (CPU0); 0x7 (CPU1)

**Table 6-28. R5FSS1 Special Features (continued)**

Feature	Comment
CPU <sub>n</sub> ATCM enable at reset (CPU <sub>n</sub> _INITRAMA)	Controlled via MAIN_SEC_MMR register setting. Defaults to disabled state
CPU <sub>n</sub> BTCM enable at reset (CPU <sub>n</sub> _INITRAMB)	Controlled via MAIN_SEC_MMR register setting. Defaults to enabled state
CPU <sub>n</sub> A/BTCM reset base address indicator (CPU <sub>n</sub> _LOCZRAMA) 0 = B at 0x0 1 = A at 0x0	Controlled via MAIN_SEC_MMR register setting. Defaults to 1
CPU <sub>n</sub> non-maskable fast interrupts enable	Controlled via MAIN_SEC_MMR register setting. Defaults to disabled state
CPU <sub>n</sub> VBUSM peripheral port enabled at reset	Enabled
CPU <sub>n</sub> VBUSP peripheral port enable at reset	Enabled
CPU <sub>n</sub> VBUSP peripheral port base address	Mapped to 0x0_0C00_0000 for low latency MAIN peripherals
CPU <sub>n</sub> VBUSP peripheral port size	64MB for MAIN peripherals (0x0_0C00_0000 to 0x0_0FFF_FFFF)
CPU <sub>n</sub> VBUSM normal peripheral port base address	Mapped to 0x0200_0000 for MAIN peripherals
CPU <sub>n</sub> VBUSM normal peripheral port size	16MB for MAIN peripherals (0x0_0200_0000 to 0x0_02FF_FFFF)
CPU <sub>n</sub> VBUSM virtual peripheral port base address	Mapped to 0x0200_0000 for MAIN peripherals
CPU <sub>n</sub> VBUSM virtual peripheral port size	16MB for MAIN peripherals (0x0_0200_0000 to 0x0_02FF_FFFF)
CPU <sub>n</sub> clock stopped indication	Status logged into MAIN_SEC_MMR register bit
CPU <sub>n</sub> WFI state	Status logged into MAIN_SEC_MMR register bit
CPU <sub>n</sub> WFE state	Status logged into MAIN_SEC_MMR register bit
CPU clockstop behavior 0: CPU clocks stopped in standby 1: CPU clocks not stopped in standby	Controlled via MAIN_SEC_MMR register setting. Defaults to 0

### 6.3.3.3 R5FSS Interfaces

#### 6.3.3.3.1 Master Interfaces

The R5FSS has several master interfaces per core:

- 64-bit VBUSM master pair (1 read – “RMST”, 1 write – “WMST”) for L3 memory accesses; this is the main memory interface
  - Includes region-based address translation (RAT)
- 32-bit VBUSM master pair (1 read – “PRMST”, 1 write – “PWMST”) for peripheral access
  - Enabled at reset
  - Note: This port is only supported for R5FSS0 and R5FSS1; it is not supported for MCU\_R5FSS0
- 32-bit VBUSP master “PMST” for peripheral access
  - Includes logic that provides the R5F CPU with a private access to VIM and RAT
  - Enabled at reset

#### 6.3.3.3.2 Slave Interfaces

The R5FSS has several slave interfaces that define its internal memory space:

- 32-bit VBUSP configuration slave (per core)
  - Region [0]: ECC aggregator block
  - Region [1]: Lockstep comparator block (CCMR5); this region is only applicable to CORE0
- 64-bit TCM slave (per core)
  - Region [0]: ATCM
  - Region [1]: BTCM
  - Region [2]: Instruction cache RAMs
  - Region [3]: Data cache RAMs
- 32-bit VBUSP debug slave
  - Provides access to all R5FSS internal debug logic

Regions [0] and [1] of the TCM slave interface provide direct access to the TCM RAMs. Access to the RAMs is arbitrated with access from the R5F's L1 memory system. Excessive access while the R5F is also attempting access will degrade performance.

Regions [2] and [3] of the TCM slave interface provide access to the cache RAMs for testing purposes. Access to the cache RAMs can only be done while the caches are disabled and should only be done for test purposes.

In addition to the slave interfaces, there are peripherals (RAT and VIM) that are only accessible by the R5F. The R5F has an access to these modules via the VBUSP peripheral interface.

### 6.3.3.4 R5FSS Power, Clocking and Reset

#### 6.3.3.4.1 R5FSS Power

The following R5FSS power considerations should be noted:

- R5FSS has a single power domain for all its internal logic
- When operating in split mode, CPU0 must be in a higher power/reset state than CPU1.

For more details on R5FSS power management, including power-up and power-down sequences, see [Section 5.2, Power](#).

#### 6.3.3.4.2 R5FSS Clocking

The R5FSS has four clock inputs:

- CPU0\_CLK: This is the clock for CPU0 logic
- CPU1\_CLK: This is the clock for CPU1 logic
- CPU0\_ICLK: This is the clock for CPU0 interfaces
- CPU1\_ICLK: This is the clock for CPU1 interfaces

CPU0\_CLK and CPU1\_CLK are the clocks for all of the internal CPU logic. They are provided separately so that:

- In split mode, CPU1's clock may be turned off, while CPU0 is still in active mode
- In lock mode, they are sourced separately

CPU0\_ICLK and CPU1\_ICLK are the clocks for all of the interfaces for their associated CPU (for example: VBUSM and VBUSP bridges, exception generation, debug and trace logic). They are provided separately so that CPU1\_ICLK can be gated if CPU1 is in a lower power state, while CPU0 is ON, or when in lock mode.

The interface clock is an integer ratio of the CPU clock. The exact ratio for each R5F is provided in [Section 6.3.2](#).

#### 6.3.3.4.2.1 Changing MCU\_R5FSS0 CPU Clock Frequency

For each MCU\_R5FSS0 CPU, the interface clock is an integer ratio of the CPU0 clock. The exact core to interface clock ratio is configured via associated Control Module register:

- CTRLMMR\_MCU\_R5\_CORE0\_CLKSEL for both MCU\_R5FSS0 cores

The interface clock has a fixed frequency, so this register is essentially used to select the CPU clock frequency. Due to some design limitation, the core frequency switch for the MCU\_R5FSS0 may cause clock misalignment. To avoid this clock misalignment, the below software sequence should be implemented by a core other than MCU\_R5FSS0\_CORE0 or MCU\_R5FSS0\_CORE1 when changing the CPU clock frequency:

1. Power down MCU\_R5FSS0
2. Change the core to interface ratio by modifying the associated CTRL\_MMR register
3. Power up MCU\_R5FSS0 and follow the bring up sequence based on split or lockstep mode

Note that the sequence above is a static configuration change and any context in the MCU\_R5FSS0 will be not be preserved. There is no dynamic clock reconfiguration available.

#### 6.3.3.4.3 R5FSS Reset

The R5FSS has four reset inputs:

- CPU0\_RST: This is the reset for the non-debug logic of CPU0



- CPU0\_DBG\_RST: This resets the CPU0 debug logic, excluding the APB interface
- CPU1\_RST: This is the reset for the non-debug logic of CPU1
- CPU1\_DBG\_RST: This resets the CPU1 debug logic, excluding the APB interface

In addition to the reset signals, there are two halt signals:

- CPU0\_HALT
- CPU1\_HALT

These halt signals keep the CPUs from fetching instructions when they come out of reset. The main use is to have the CPUs halted until the TCMs are loaded (when booting from TCM), though halt could be used for any other purpose.

### 6.3.3.5 R5FSS Lockstep Error Detection Logic

When bootstrapped to lockstep mode, there is a logic that compares the outputs of the two cores. Whenever an error is detected, an interrupt is asserted. A logic is also included to test the comparison logic.

In lockstep mode, the logic from CPU1 is used to check CPU0. CPU1's RAMs and L1 memory system are not used. Instead, all inputs to CPU0 are copied, delayed by two cycles, and fed into CPU1's logic. All outputs from CPU0 are copied, delayed by two cycles, and compared to the outputs of CPU1.

The lockstep error detection logic in the R5FSS is implemented via the CPU compare module (CCMR5). The CCMR5 performs two main functions:

- CPU output compare: Compares the core compare bus outputs of the two R5F processor cores in lockstep mode
- Inactivity monitoring: This is to detect any transaction initiated by the diagnostic R5F core (CPU1) in lockstep mode. The diagnostic core is expected *not* to initiate any transaction in lockstep mode

#### 6.3.3.5.1 CPU Output Compare Block

To implement a lockstep CPU cluster, a diagnostic checker block is included. The diagnostic checker block compares the CPU compare outputs on every cycle. A difference in the CPU output signals is indicated by an error signal.

As the CPUs are working with a two-cycle phase difference, there should be no compare errors generated when the CPUs are reset two cycles apart. For the various CPU interfaces, the comparison will start on the first valid transaction on key control signals.

##### 6.3.3.5.1.1 Operating Modes

The CPU compare block can run in one out of four operating modes:

- Compare active block
- Self test
- Error forcing
- Self test error forcing

To select an operating mode for the CPU compare block, a dedicated key must be written to the key register R5FSS\_CCMKEYR1. R5FSS\_CCMSR1 is the status register associated with this block.

Table 6-29 provides details on the operating modes of the CPU compare block.

**Table 6-29. CPU Compare Block Operating Modes**

Mode	Key <sup>(1)</sup>	Self Test Error Signal	Compare Error Signal	CMPE1 <sup>(2)</sup>	STC1 <sup>(3)</sup>	STET1 <sup>(4)</sup>	STE1 <sup>(5)</sup>
Compare block active	0000	Enabled	Enabled	Enabled	Disabled	Disabled	Disabled
Self test	0110	Enabled	Disabled	Disabled	Enabled	Enabled	Enabled
Error forcing	1001	Error	Error	Disabled	Disabled	Disabled	Disabled
Self test error forcing	1111	Error	Enabled	Enabled	Disabled	Disabled	Disabled

(1) R5FSS\_CCMKEYR1[3:0] MKEY1 bit field

- (2) R5FSS\_CCMSR1[16] CMPE1: Compare error flag
- (3) R5FSS\_CCMSR1[8] STC1: Self test complete flag
- (4) R5FSS\_CCMSR1[1] STET1: Self test error type flag
- (5) R5FSS\_CCMSR1[0] STE1: Self test error flag

#### 6.3.3.5.1.2 Compare Block Active Mode

In compare block active mode, the output signals of both CPUs are compared, and any difference in the outputs is indicated by the compare error signal (CPU\_BUS\_CMP\_ERR\_INT). Additionally, as indicated in [Table 6-29](#), the self test error signal (SELFTEST\_ERR\_INT) is also asserted.

---

#### Note

The self test error signal is shared by both the CCMR5's CPU compare and inactivity monitor blocks.

---

#### Note

Not all flops inside the R5F CPU(s) are initialized at reset. To avoid an erroneous CCMR5 compare error, the application software needs to ensure that the CPU registers of both CPUs are initialized with the same values before the registers are used. This is to ensure that the programmer's model CPU registers of both CPU are initialized properly.

---

#### 6.3.3.5.1.3 Self Test Mode

In self test mode, the CCMR5 compare block is checked for faults by applying internally generated series of test patterns. During self test, the core compare disabled signal is deactivated. If a fault on the CCMR5 module gets detected, a self test error is generated.

When self test mode is entered, the CCMR5 module will generate predetermined test patterns to look for any hardware faults inside it. If a fault is detected, then a self test error flag (STE1) is set, a self test error signal is asserted, and the self test is terminated immediately after entering compare mismatch mode. If no fault is found during self test, the self test complete flag (STC1) will be set. The user needs to poll the R5FSS\_CCMSR1 status register to find out the self test status. In both cases – self test terminated and self test completed – the CCMR5 will remain in self test mode and will be idle, and therefore the R5FSS\_CCMKEYR1 key register will show the self test key until the mode is switched by writing another key to this register. During the self test operation, the compare error signal output is inactive, irrespective of the compare result. When switched out of self test mode, the core compare disabled signal is de-asserted.

There are two types of patterns generated by CCMR5 during self test mode: compare match test, and compare mismatch test. The CCMR5 first generates compare match test patterns followed by compare mismatch test patterns. During self test, each test pattern is applied on CPU output signal ports of the CCMR5's compare block and clocked for one cycle.

Self test of CPU output compare logic is done with respect to CPU clock. As mentioned, self test error is indicated by the SELFTEST\_ERR\_INT signal. Whether the self test failed during compare match test or compare mismatch test is indicated by the self test error type flag (STET1) in the R5FSS\_CCMSR1 status register. When the block's self test is completed, the corresponding self test complete flag (STC1) is set.

---

#### Note

During self test, both CPUs can execute normally, but the compare logic will not be checking any CPU signals. Also, during self test, only the compare unit logic is tested. The self test is not interruptible.

---

#### 6.3.3.5.1.4 Compare Match Test

During compare match test, there are four different test patterns generated to stimulate the hardware. An identical vector is applied to both input ports (CPU0 and CPU1 output signals, which are inputs to the CCMR5 block) at the same time expecting a compare match. These patterns cause the self test logic to exercise every CPU output signal. This test can be independently enabled by programming the CCMR5 registers. If the compare unit produces a compare mismatch then the self test error flag is set and the self test error signal is



generated. If an error is detected during compare match, then CCMR5 enters compare mismatch mode before terminating the test.

The four test patterns are:

- All 0's on both CPUs
- All 1's on both CPUs
- 0xA's on both CPUs
- 0x5's on both CPUs

These four test patterns will take four clock cycles to complete.

Table 6-30 illustrates the compare match test sequence. The core compare disabled signal is asserted on entry and de-asserted when complete.

**Table 6-30. Compare Match Test Sequence**

CPU0 Signal Position									CPU1 Signal Position									Cycle
n:8	7	6	5	4	3	2	1	0	n:8	7	6	5	4	3	2	1	0	
0x5	0	1	0	1	0	1	0	1	0x5	0	1	0	1	0	1	0	1	0
0xA	1	0	1	0	1	0	1	0	0xA	1	0	1	0	1	0	1	0	1
0's	0	0	0	0	0	0	0	0	0's	0	0	0	0	0	0	0	0	2
1's	1	1	1	1	1	1	1	1	1's	1	1	1	1	1	1	1	1	3

#### 6.3.3.5.1.5 Compare Mismatch Test

In compare mismatch test, the number of test patterns is equal to two times the number of CPU output signals to compare in compare mode active. The core compare disabled signal is asserted on entry and de-asserted when complete. An 'all-ones' vector is applied to the CCMR5's CPU0 input port and the same pattern is also applied to the CCMR5's CPU1 input port but with one bit flipped starting from signal position 0. The un-equal vector should cause the CCMR5 module to expect a compare mismatch at signal position 0. Note that a mismatch is an expected good result, while a match will indicate hardware fault. When a fault is detected, which means that a compare match is produced by the compare logic, the self test error flag is set and the self test error signal is asserted.

The above compare mismatch test algorithm repeats itself again in a domino fashion with next signal position flipped, while forcing all other signals to logic level '1'. This process is repeated until every single signal position is verified on both CPU signal ports.

The compare mismatch test is terminated if a compare match is detected and the module becomes idle. The compare mismatch test ensures that the compare unit is able to detect a mismatch on every CPU signal being compared.

Table 6-31 illustrates the compare mismatch test sequence.

**Table 6-31. Compare Mismatch Test Sequence**

CPU0 Signal Position										CPU1 Signal Position										Cycle
n	...	7	6	5	4	3	2	1	0	n	...	7	6	5	4	3	2	1	0	
1	1's	1	1	1	1	1	1	1	1	1	1's	1	1	1	1	1	1	1	0	0
1	1's	1	1	1	1	1	1	1	1	1	1's	1	1	1	1	1	1	0	1	1
1	1's	1	1	1	1	1	1	1	1	1	1's	1	1	1	1	1	0	1	1	2
1	1's	1	1	1	1	1	1	1	1	1	1's	1	1	1	1	0	1	1	1	3
...																				
1	1's	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	n-1
1	1's	1	1	1	1	1	1	1	1	0	1's	1	1	1	1	1	1	1	1	n
1	1's	1	1	1	1	1	1	1	0	1	1's	1	1	1	1	1	1	1	1	n+1
1	1's	1	1	1	1	1	1	0	1	1	1's	1	1	1	1	1	1	1	1	n+2

**Table 6-31. Compare Mismatch Test Sequence (continued)**

CPU0 Signal Position										CPU1 Signal Position										Cycle
n	...	7	6	5	4	3	2	1	0	n	...	7	6	5	4	3	2	1	0	
1	1's	1	1	1	1	1	0	1	1	1	1's	1	1	1	1	1	1	1	1	n+3
1	1's	1	1	1	1	0	1	1	1	1	1's	1	1	1	1	1	1	1	1	n+4
...																				
0	1's	1	1	1	1	1	1	1	1	1	1's	1	1	1	1	1	1	1	1	2n

#### 6.3.3.5.1.6 Error Forcing Mode

In error forcing mode, a test pattern is applied to the CPU related inputs of the compare logic to force an error at the compare error signal of the compare unit. The core compare disabled signal is asserted on entry and de-asserted when complete.

This mode is enabled by writing the dedicated key in the R5FSS\_CCMKEYR1 key register (see [Table 6-29](#)). The error signal is generated by asserting the CPU\_BUS\_CMP\_ERR\_INT signal.

Error forcing mode is similar to the compare mismatch test operation of self test mode, in which an un-equal vector is injected into the CCMR5 CPU signal ports. Instead of setting a self test error flag and asserting self test error signal, the error forcing mode forces the compare mismatch to set the compare error flag (CMPE1) and assert the compare error signal.

Only one hardcoded test pattern is applied into CCMR5 during error forcing mode. A repeated 0x5 pattern is applied to CPU0 signal port, while a repeated 0xA pattern is applied to the CPU1 signal port. The error forcing mode takes one cycle to complete. Hence, the failing signature is presented for one clock cycle and the mode is automatically switched to lockstep mode and the R5FSS\_CCMKEYR1 key register will show the lockstep key (0000). During this cycle the CPUs are not compared. The user should expect to receive the error signal from CCMR5 module once the error forcing mode is entered. If no error signal is set, then a hardware fault is present.

#### 6.3.3.5.1.7 Self Test Error Forcing Mode

In self test error forcing mode, an error is forced at the self test error signal. The compare unit is still running in lockstep mode and the key is switched to lockstep after one clock cycle. The core compare disabled signal is asserted on entry and de-asserted when complete.

#### 6.3.3.5.2 Inactivity Monitor Block

Another function of the CCMR5 module is to monitor the inputs of bus matrix coming from the diagnostic R5F core (CPU1) in lockstep mode, to detect any transaction initiated by diagnostic core. Output signals from the diagnostic core, which indicate a valid transaction on a master interface, are compared against their clamped values. If any signal value is different from its clamped value, an error signal is generated.

[Table 6-32](#) shows the R5F diagnostic core output signals, which are being monitored to detect any activity in the processor bus.

**Table 6-32. Inactivity Monitor Block Signals**

Signal Name	Interface	Description	Clamp Value
AWVALIDM1	L2 AXI Master	Indicates write address and control are valid	0
ARVALIDM1	L2 AXI Master	Indicates read address and control are valid	0
AWVALIDP1	AXI PP <sup>(1)</sup>	Indicates write address and control are valid	0
ARVALIDP1	AXI PP <sup>(1)</sup>	Indicates read address and control are valid	0
HTRANSP1[1:0]	AHB PP	Indicates the type of transfer – idle (00), busy (01), non-sequential (10), sequential (11)	0
BVALIDS1	AXI Slave	Indicates valid write response is available	0
RVALIDS1	AXI Slave	Indicates Read Data Channel Address and Control are valid	0
ATCEN01	ATCM	Enable for ATCM lower word	0
ATCEN11	ATCM	Enable for ATCM upper word	0

**Table 6-32. Inactivity Monitor Block Signals (continued)**

Signal Name	Interface	Description	Clamp Value
B0TCEN01	B0TCM	Enable for B0TCM lower word	0
B0TCEN11	B0TCM	Enable for B0TCM upper word	0
B1TCEN01	B1TCM	Enable for B1TCM lower word	0
B1TCEN11	B1TCM	Enable for B1TCM upper word	0

(1) Not supported for MCU\_R5FSS0

More details on these signals can be found in the *Arm Cortex-R5 Technical Reference Manual*.

The error response in case of a detected transaction is indicated by the INACTIVITY\_ERR\_INT signal.

#### 6.3.3.5.2.1 Operating Modes

The inactivity monitor block can run in one out of four operating modes:

- Compare active block
- Self test
- Error forcing
- Self test error forcing

To select an operating mode for the inactivity monitor block, a dedicated key must be written to the key register R5FSS\_CCMKEYR3. R5FSS\_CCMSR3 is the status register associated with this block.

Table 6-33 provides details on the operating modes of the inactivity monitor block.

**Table 6-33. Inactivity Monitor Block Operating Modes**

Mode	Key <sup>(1)</sup>	Self Test Error Signal	Compare Error Signal <sup>(2)</sup>	CMPE3 <sup>(3)</sup>	STC3 <sup>(4)</sup>	STET3 <sup>(5)</sup>	STE3 <sup>(6)</sup>
Compare block active	0000	Enabled	Enabled	Enabled	Disabled	Disabled	Disabled
Self test	0110	Enabled	Disabled	Disabled	Enabled	Enabled	Enabled
Error forcing	1001	Error	Error	Disabled	Disabled	Disabled	Disabled
Self test error forcing	1111	Error	Enabled	Enabled	Disabled	Disabled	Disabled

(1) CCMKEYR3[3:0] MKEY3 bit field

(2) Corresponds to bus monitor error signal for inactivity monitoring

(3) CCMSR3[16] CMPE3: Compare error (bus monitor error) flag

(4) CCMSR3[8] STC3: Self test complete flag

(5) CCMSR3[1] STET3: Self test error type flag

(6) CCMSR3[0] STE3: Self test error flag

#### 6.3.3.5.2.2 Compare Block Active Mode

In compare block active mode, the output signals of CPU1 (after clamping) are compared against their clamped values, and a mismatch is indicated by the bus monitor error signal. Additionally, as indicated in Table 6-33, the self test error signal is also asserted.

#### Note

The self test error signal is shared by both the CCMR5's CPU compare and inactivity monitor blocks.

#### 6.3.3.5.2.3 Self Test Mode

Inactivity monitor has a self test feature and self test is done by applying a predetermined set of internally generated test patterns on the Inactivity Monitor inputs. For the fault detection, the comparison is done against the clamped values. The generated patterns are same as output compare block test patterns, but they are applied on every HCLK cycles. Any fault detected is indicated by the self test error signal. If no fault is found during self test, the self test complete flag (STC3) will be set. The user needs to poll the R5FSS\_CCMSR3 status register to find out the self test status. In both cases – self test terminated and self test completed

– inactivity monitor will remain in self test mode and will be idle, and therefore the R5FSS\_CCMKEYR3 key register will show the self test key until the mode is switched by writing another key to this register.

### Note

Bus monitor error is disabled during self test mode.

Two types of test patterns are applied for inactivity monitor self test:

- Compare match pattern
- Compare mismatch pattern

Self test takes 16 cycles to complete. Self test error is indicated by the SELFTEST\_ERR\_INT signal. Whether the self test failed during compare match test or compare mismatch test is indicated by the self test error type flag (STET3) in the R5FSS\_CCMSR3 status register. When the block's self test is completed, the corresponding self test complete flag (STC3) is set.

#### 6.3.3.5.2.4 Compare Match Test

In compare match test, patterns are applied such that no compare error is generated. Because the comparison is done against the clamped values, and all compared signals are clamped to zero, only one test pattern is applicable for compare match test. A pattern of 'all-zeros' is applied for compare match test. If a compare mismatch is found, then self test error flag (STE3) is set and self test error is asserted. If an error is detected during compare match, then CCMR5 enters compare mismatch mode before terminating the test. If no mismatch is found, compare mismatch test is done next.

#### 6.3.3.5.2.5 Compare Mismatch Test

In compare mismatch test, a series of un-equal test patterns are applied on inactivity monitor inputs. Un-equal vectors are expected to generate a compare mismatch and a compare match indicates a fault. If a compare match is found, self test error flag is set and a self test error is indicated. Self test is terminated as well. If all patterns are generating mismatch, self test complete flag (STC3) is set.

Table 6-34 shows the patterns used for compare mismatch test.

**Table 6-34. Inactivity Monitor Compare Mismatch Patterns**

Inactivity Monitor Compare Mismatch Patterns												Cycle
11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	0	0	1	0	0	2
0	0	0	0	0	0	0	0	1	0	0	0	3
0	0	0	0	0	0	0	1	0	0	0	0	4
0	0	0	0	0	0	1	0	0	0	0	0	5
0	0	0	0	0	1	0	0	0	0	0	0	6
0	0	0	0	1	0	0	0	0	0	0	0	7
0	0	0	1	0	0	0	0	0	0	0	0	8
0	0	1	0	0	0	0	0	0	0	0	0	9
0	1	0	0	0	0	0	0	0	0	0	0	10
1	0	0	0	0	0	0	0	0	0	0	0	11

#### 6.3.3.5.2.6 Error Forcing Mode

In error forcing mode, a test pattern is applied to inactivity monitor inputs of the compare logic to force an error at the compare error signal of the compare unit. Only one hardcoded test pattern of 'all-ones' is applied into CCMR5 during error forcing mode. The error forcing mode takes one cycle to complete. Hence, the failing

signature is presented for one clock cycle and the mode is automatically switched to lockstep mode and the R5FSS\_CCMKEYR3 key register will show the lockstep key (0000). During this cycle the bus monitoring is disabled.

#### 6.3.3.5.2.7 Self Test Error Forcing Mode

In self test error forcing mode, an error is forced at the self test error signal. The compare unit is still running in lockstep mode and the key is switched to lockstep after one clock cycle.

#### 6.3.3.5.3 Polarity Inversion Logic

As already mentioned, there is logic (implemented via XOR) to test the comparison logic itself. The CCMR5 block can be configured to be in self test to check the integrity of the final XOR error generation logic, and can be switched back to functional compare mode after self test completion. To ensure that switching back to functional mode has happened, user can program CCMR5 polarity control register (R5FSS\_CCMPOLCNTRL) to invert the polarity of eight CPU output signals, which will cause an error generation (CCM\_STAT\_ERR\_INT). The eight signals that are chosen to be able to control the polarity are: AWWALIDMm, RVALIDSm, ATCEN0m, BOTCEN0m, BITCWEm, IRQACKm, AWWALIDPm, HWRITEP. See the *Arm Cortex-R5 Technical Reference Manual* for details on these signals.

The XOR inversion is placed between the two back-to-back delay flops of CPU1 outputs. Note that the XOR inversion logic is a diagnostic feature. If user does not observe CCM\_STAT\_ERR\_INT error generation after programming the CCMPOLCNTRL[7:0] POL\_INV inversion bits, this means that:

1. All eight signals of the inversion logics (XOR) are faulty and CCM\_STAT\_ERR\_INT is not asserted. This is a very difficult case to hit but theoretically it could happen if a hard fault gets accumulated over times on these polarity inversion XOR gates; OR
2. The CCMR5 block has not switched back to functional mode yet

User can perform self test again to identify whether the issue is #1 or #2. If the second self test passes that means the issue is #1.

### 6.3.3.6 R5FSS Vectored Interrupt Manager (VIM)

#### 6.3.3.6.1 VIM Overview

The VIM aggregates device interrupts and sends them to the R5F CPU(s). It can be used in either split or lockstep configuration. In split, it has two independent interrupt cores, one per CPU. In lockstep, CPU1 acts as a diagnostic on CPU0; only CPU0's outputs are used but all outputs are compared to CPU1 to provide diagnostic coverage.

The VIM module supports the following features:

- 512 interrupt inputs per R5F core
- Each interrupt has its own 4-bit programmable priority
  - Defined via the R5FSS\_VIM\_PRI\_INT\_j register
  - The VIM provides support for priority interruption of interrupts
- Each interrupt has its own enable mask
  - Interrupt enable is done via the R5FSS\_VIM\_INTR\_EN\_SET\_j register
  - Interrupt disable is done via the R5FSS\_VIM\_INTR\_EN\_CLR\_j register
- Each interrupt can be programmed as either an IRQ or FIQ
  - Defined via the R5FSS\_VIM\_INTMAP\_j register
- Each interrupt has its own programmable 32-bit vector address associated with it
  - Defined via the R5FSS\_VIM\_VEC\_INT\_j register
  - Protected with SECDDED
- One IRQn and one FIQn output per core
- Vectored interrupt interface
  - Compatible with R5F VIC port
- Default vector provided when a double-bit error is detected
- Split or lockstep capable

- In lockstep mode, only interrupts connected to VIM interrupt core 0 are available
- Software interrupt generation

### 6.3.3.6.2 VIM Interrupt Inputs

The VIM supports 512 interrupt inputs per core. Each interrupt can be either a level or a pulse (both active-high). The interrupt mapping for the two R5F cores can be found in [Section 9.4, Interrupt Sources](#).

### 6.3.3.6.3 VIM Interrupt Outputs

The VIM has two interrupt outputs per core:

- **CoreN\_IRQn**: This is a normal interrupt for core *N* (active-low level). It can be serviced via the VIC interface or through the MMR interface. Whenever an interrupt input goes high, if that interrupt is mapped as an IRQ (via the R5FSS\_VIM\_INTMAP\_j register) and is enabled (via the R5FSS\_VIM\_INTR\_EN\_SET\_j register), then it will cause an IRQ to assert
- **CoreN\_FIQn**: This is a fast (or non-maskable) interrupt for core *N* (active-low level). FIQs always have priority over IRQs. An FIQ can be serviced through the MMR interface. Whenever an interrupt input goes high, if that interrupt is mapped as an FIQ and is enabled, then it will cause an FIQ to assert

### 6.3.3.6.4 VIM Interrupt Vector Table (VIM RAM)

For each VIM interrupt core, there is an associated interrupt vector table (VIM RAM) that is used to store the address of ISRs. During register vectored interrupt and hardware vectored interrupt, VIM accesses the interrupt vector table using the vector value to fetch the address of the corresponding ISR. Note that both interrupt vector tables are identical in their memory organization.

The VIM RAM is basically comprised of a set of interrupt vector registers (R5FSS\_VIM\_VEC\_INT\_j). Hence, the interrupt vector table is organized in 512 words of 30 bits, with a base address corresponding to the physical address of the first register in the group.

#### Note

The lower two bits of the 32-bit interrupt vector are always 0s.

[Figure 6-8](#) shows the VIM RAM interrupt vector map.

VIM RAM Address Space	VIM RAM Entries
Base Address + 0h	Interrupt 0 Vector
Base Address + 4h	Interrupt 1 Vector
Base Address + 8h	Interrupt 2 Vector
Base Address + 7F8h	Interrupt 510 Vector
Base Address + 7FCh	Interrupt 511 Vector

**Figure 6-8. VIM RAM Interrupt Vector Map**

The interrupt vector table has protection by ECC to indicate corruption due to soft errors. The ECC logic inside VIM supports SECDED. See [Table 6-35](#) for the VIM RAM ID in the ECC aggregator map.

#### 6.3.3.6.5 VIM Interrupt Prioritization

The VIM supports the interruption of the currently active interrupt by one with a higher priority. FIQs and IRQs are completely separate but both use the same mechanism.

When an interrupt goes from pending to active (FIQ: reading the R5FSS\_VIM\_FIQVEC register; IRQ: reading the R5FSS\_VIM\_IRQVEC register, or the *coreN\_IRQACK* going high), then the interrupt is loaded into the corresponding active register (R5FSS\_VIM\_ACTFIQ / R5FSS\_VIM\_ACTIRQ), and all interrupts of an equal or lesser priority are masked (discarded). If prior to this interrupt being cleared (by writing to the R5FSS\_VIM\_FIQVEC register, or R5FSS\_VIM\_IRQVEC register) another interrupt of higher priority arrives, then the FIQn/IRQn will be asserted and that interrupt made pending as normal. If the CPU switches this interrupt to active (by reading the R5FSS\_VIM\_FIQVEC / R5FSS\_VIM\_IRQVEC register), then the currently active interrupt will be pushed onto a stack. When an interrupt is cleared by reading the R5FSS\_VIM\_FIQVEC / R5FSS\_VIM\_IRQVEC register, if there are any interrupts on the stack, the first entry is popped off and put back into the R5FSS\_VIM\_ACTFIQ / R5FSS\_VIM\_ACTIRQ register, so that software may continue where it left off.

#### 6.3.3.6.6 VIM ECC Support

The memory that holds the interrupt vector for each interrupt is protected by SECDED ECC. Single-bit errors are corrected and written back. Double-bit errors are not corrected. If a double-bit error occurs while trying to load a vector, then the R5FSS\_VIM\_DEDVEC register is used to provide the default vector for the *coreN\_IRQADDRV* signal, the R5FSS\_VIM\_IRQVEC register, and the R5FSS\_VIM\_FIQVEC register. The R5FSS\_VIM\_DEDVEC should point to an ISR that handles the fact that there was an uncorrectable error in the interrupt handling.

Some possible remediating actions would be to:

1. Reconstruct the vector table and re-start the application
  - a. Potentially switch to a completely software interrupt handler in the mean time
2. Restart the application from scratch
3. Reset the device
4. Sit in a loop (or WFI) while something external (for example, the ESM) responds to the DED interrupt that will be generated

It is up to the user and the application to determine the appropriate action.

---

#### Note

An interrupt that has an uncorrectable vector error (and thus uses the DED vector) will still have the priority of the original interrupt. This makes it possible for a higher priority interrupt to supersede the handling of the error.

Control and reporting are done by the R5FSS ECC aggregator.

---

#### 6.3.3.6.7 VIM Lockstep Mode

In lockstep mode, CPU1 is used as a diagnostic for CPU0. In this mode, only the interrupt inputs for CPU0 are used. Besides to CPU0, these interrupt inputs are also internally routed to CPU1 (through the level-sync / edge-detect logic dedicated to CPU1, and additionally through some delay circuits). The outputs from both VIM interrupt cores are then sent to the R5FSS CCMR5 module through dedicated compare buses (with CPU0's outputs delayed). The CCMR5 module is responsible for comparing the two sets of output signals and for reporting any mismatches by generating an interrupt (VIM\_BUS\_CMP\_ERR\_INT).

---

#### Note

In lockstep mode, only the RAM dedicated to CPU0 is used, so software *must not* do anything with the ECC interface on the RAM dedicated to CPU1.

---



### 6.3.3.6.8 VIM IDLE State

The VIM will indicate IDLE when there are no pending unmasked interrupts or MMR accesses. The VIM does not have a clock stop interface.

### 6.3.3.6.9 VIM Interrupt Handling

There are multiple ways to service an interrupt depending on how much of the hardware assistance offered by the VIM the software wants to take advantage of.

For IRQs, it is recommended to use the procedure in [Section 6.3.3.6.9.1](#), but the procedures in [Section 6.3.3.6.9.2](#) or [Section 6.3.3.6.9.3](#) (if a user wants to implement a fully software prioritization scheme) may be used as alternatives.

For FIQs, it is recommended to use the procedure in [Section 6.3.3.6.9.4](#), but the procedure in [Section 6.3.3.6.9.5](#) may be used as an alternative.

---

#### Note

These descriptions do not include steps such as stack pushes and state retention that software must take in order to return from the ISR. It is assumed that the programmer is aware of these steps.

---

#### 6.3.3.6.9.1 Servicing IRQ Through Vector Interface

If the associated CPU has the vector (VIC) interface enabled, then the following method is used for servicing IRQs:

1. Hardware handshake
  - a. CPU asserts *coreN\_IRQACK* high
  - b. VIM asserts *coreN\_IRQADDRV* to indicate that the *coreN\_IRQADDR* bus is stable with the correct vector address
  - c. CPU reads *coreN\_IRQADDR*, jumps to that address, and de-asserts *coreN\_IRQACK* low
  - d. VIM de-asserts *coreN\_IRQn* and *coreN\_IRQADDRV*, VIM masks (discards) all IRQs with the same or lower priority
  - e. VIM loads the value from the R5FSS\_VIM\_PRIIRQ[9:0] NUM bit field (which corresponds to the vector address) into the R5FSS\_VIM\_ACTIRQ[9:0] NUM bit field, which causes the R5FSS\_VIM\_ACTIRQ[31] VALID bit to be set
2. Service the interrupt
3. Depending on whether the original source of the interrupt was a pulse or a level (determined by reading the R5FSS\_VIM\_ACTIRQ[9:0] NUM bit field to determine number, and reading the appropriate bit in the R5FSS\_VIM\_INTTYPE\_j register to determine type)
  - a. Pulse
    - i. Clear the status by writing a '1' to the appropriate bit in the R5FSS\_VIM\_IRQSTS\_j register, or R5FSS\_VIM\_STS\_j register
    - ii. Clear the interrupt at the source. This way, the source can generate another pulse, if it needs to, and the VIM will process this as a new interrupt
  - b. Level
    - i. Clear the interrupt at the source
    - ii. Clear the status by writing a '1' to the appropriate bit in the R5FSS\_VIM\_IRQSTS\_j register, or R5FSS\_VIM\_STS\_j register. This way, the level should be gone at the input to the VIM, it will avoid falsely re-calling the interrupt. If the source maintains the level, then it means there is another interrupt
4. Write any value to the R5FSS\_VIM\_IRQVEC register
  - a. This will clear the priority mask and will cause all interrupts to be re-evaluated for the new highest priority interrupt
  - b. This will also clear the R5FSS\_VIM\_ACTIRQ[31] VALID bit



#### 6.3.3.6.9.2 Servicing IRQ Through MMR Interface

When an IRQ interrupt is received, the CPU should follow these steps if not using the vector interface:

1. Read the R5FSS\_VIM\_IRQVEC register and jump to that address to service the ISR
  - a. Reading this register will mask (discard) all interrupts of an equal or lower priority and de-assert the *coreN\_IRQn* output. If another interrupt of a higher priority becomes available, the *coreN\_IRQn* will re-assert, allowing priority interruption of an interrupt
  - b. Reading this register will cause the value from the R5FSS\_VIM\_PRIIRQ[9:0] NUM bit field to be loaded into the R5FSS\_VIM\_ACTIRQ[9:0] NUM bit field, and the R5FSS\_VIM\_ACTIRQ[31] VALID bit to be set
2. Service the interrupt
3. Depending on whether the original source of the interrupt was a pulse or a level
  - a. Pulse
    - i. Clear the status by writing a '1' to the appropriate bit in the R5FSS\_VIM\_STS\_j register, or R5FSS\_VIM\_IRQSTS\_j register
    - ii. Clear the interrupt at the source
  - b. Level
    - i. Clear the interrupt at the source
    - ii. Clear the status by writing a '1' to the appropriate bit in the R5FSS\_VIM\_STS\_j register, or R5FSS\_VIM\_IRQSTS\_j register
4. Write any value to the R5FSS\_VIM\_IRQVEC register
  - a. This will clear the priority mask and will cause all interrupts to be re-evaluated for the new highest priority interrupt
  - b. This will also clear the R5FSS\_VIM\_ACTIRQ[31] VALID bit

#### 6.3.3.6.9.3 Servicing IRQ Through MMR Interface (Alternative)

If a user does not want to use the R5FSS\_VIM\_IRQVEC register, the VIM may be used as a more traditional interrupt controller. Note that in this mode, there is no hardware priority masking (because the R5FSS\_VIM\_IRQVEC register is never read). Software would be responsible for doing all priority operations.

1. Determine which interrupt to service
  - a. Read the R5FSS\_VIM\_PRIIRQ register to determine which interrupt is the highest priority IRQ currently asserted, OR
  - b. Optionally read the R5FSS\_VIM\_IRQGSTS register to determine which groups have IRQs pending, then read the R5FSS\_VIM\_IRQSTS\_j register and use a software prioritization scheme to determine which IRQ to service
2. Service the interrupt
3. Depending on whether the original source of the interrupt was a pulse or a level
  - a. Pulse
    - i. Clear the status by writing a '1' to the appropriate bit in the R5FSS\_VIM\_STS\_j register, or R5FSS\_VIM\_IRQSTS\_j register
    - ii. Clear the interrupt at the source.
  - b. Level
    - i. Clear the interrupt at the source
    - ii. Clear the status by writing a '1' to the appropriate bit in the R5FSS\_VIM\_STS\_j register, or R5FSS\_VIM\_IRQSTS\_j register

#### 6.3.3.6.9.4 Servicing FIQ

When an FIQ interrupt is received, the CPU should follow these steps:

1. Read the R5FSS\_VIM\_FIQVEC register and jump to that address to service the ISR
  - a. Reading this register will mask (discard) all interrupts of an equal or lower priority and de-assert the *coreN\_FIQn* output. If another interrupt of a higher priority becomes available, the *coreN\_FIQn* will re-assert, allowing priority interruption of an interrupt.

- b. Reading this register will cause the value from the R5FSS\_VIM\_PRIFIQ[9:0] NUM bit field to be loaded into the R5FSS\_VIM\_ACTFIQ[9:0] NUM bit field, and the R5FSS\_VIM\_ACTFIQ[31] VALID bit to be set
  2. Service the interrupt
  3. Depending on whether the original source of the interrupt was a pulse or a level (determined by reading the R5FSS\_VIM\_ACTFIQ[9:0] NUM bit field to determine number, and reading the appropriate bit in the R5FSS\_VIM\_INTTYPE\_j register to determine type)
    - a. Pulse
      - i. Clear the status by writing a '1' to the appropriate bit in the R5FSS\_VIM\_STS\_j register, or R5FSS\_VIM\_FIQSTS\_j register
      - ii. Clear the interrupt at the source. This way, the source can generate another pulse, if it needs to, and the VIM will process this as a new interrupt
    - b. Level
      - i. Clear the interrupt at the source
      - ii. Clear the status by writing a '1' to the appropriate bit in the R5FSS\_VIM\_STS\_j register, or R5FSS\_VIM\_FIQSTS\_j register. This way, the level should be gone at the input to the VIM, it will avoid falsely re-calling the interrupt. If the source maintains the level, then it means there is another interrupt
  4. Write any value to the R5FSS\_VIM\_FIQVEC register
    - a. This will clear the priority mask and will cause all interrupts to be re-evaluated for the new highest priority interrupt
    - b. This will also clear the R5FSS\_VIM\_ACTFIQ[31] VALID bit

#### 6.3.3.6.9.5 Servicing FIQ (Alternative)

If a user does not want to use the R5FSS\_VIM\_FIQVEC register, the VIM may be used as a more traditional interrupt controller. Note that in this mode, there is no hardware priority masking (because the R5FSS\_VIM\_FIQVEC register is never read). Software would be responsible for doing all priority operations.

1. Determine which interrupt to service
  - a. Read the R5FSS\_VIM\_PRIFIQ register to determine which interrupt is the highest priority FIQ currently asserted, OR
  - b. Optionally read the R5FSS\_VIM\_FIQGSTS register to determine which groups have IRQs pending, then read the R5FSS\_VIM\_FIQSTS\_j register and use a software prioritization scheme to determine which FIQ to service
2. Service the interrupt
3. Depending on whether the original source of the interrupt was a pulse or a level
  - a. Pulse
    - i. Clear the status by writing a '1' to the appropriate bit in the R5FSS\_VIM\_STS\_j register, or R5FSS\_VIM\_FIQSTS\_j register
    - ii. Clear the interrupt at the source.
  - b. Level
    - i. Clear the interrupt at the source
    - ii. Clear the status by writing a '1' to the appropriate bit in the R5FSS\_VIM\_STS\_j register, or R5FSS\_VIM\_FIQSTS\_j register.

#### 6.3.3.7 R5FSS Region Address Translation (RAT)

The R5F is a 32-bit processor, which means it can only access 4GB of directly addressable memory. The R5FSS includes a region-based address translation (RAT) unit per core, which allows the R5F to access higher address ranges. The R5FSS RAT module translates a 32-bit input address into a 48-bit output address. It supports 16 regions. For more details on RAT functionality, refer to [Section 8.4, Region-based Address Translation \(RAT\) Module](#).

### 6.3.3.8 R5FSS ECC Support

The R5F provides native ECC and parity support on all related memories, generating and checking the redundancy automatically. The methods for checking and reporting errors are available in the *Arm Cortex-R5 Technical Reference Manual*.

The R5FSS adds the capability of testing this logic by allowing errors (single and double bit) to be injected into memories (for testing purposes) via an ECC aggregator (per core). Note that because the R5FSS ECC aggregator is only used in error-injection mode, it only supports a subset of the generic ECC aggregator functionality in the device.

For a detailed description of the generic ECC aggregator functionality, see *ECC Aggregator*. For register descriptions of R5FSS CPU0 and CPU1 ECC aggregators, see *R5FSS\_CPU0\_ECC\_AGGR\_CFG\_REGS Registers* and *R5FSS\_CPU1\_ECC\_AGGR\_CFG\_REGS Registers*, respectively.

[Table 6-35](#) provides the RAM ID for each core. This is needed for bit field [10-0] ECC\_VECTOR in the corresponding R5FSS\_CPU0\_VECTOR / R5FSS\_CPU1\_VECTOR register (part of the ECC aggregator register space).

**Table 6-35. RAM ID Map for ECC Aggregator (Per Core)**

RAM ID	Memory Name
0	CPU0/1 ITAG RAM0
1	CPU0/1 ITAG RAM1
2	CPU0/1 ITAG RAM2
3	CPU0/1 ITAG RAM3
4	CPU0/1 IDATA BANK0
5	CPU0/1 IDATA BANK1
6	CPU0/1 IDATA BANK2
7	CPU0/1 IDATA BANK3
8	CPU0/1 DTAG RAM0
9	CPU0/1 DTAG RAM1
10	CPU0/1 DTAG RAM2
11	CPU0/1 DTAG RAM3
12	CPU0/1 DDIRTY RAM
13	CPU0/1 DDATA RAM0
14	CPU0/1 DDATA RAM1
15	CPU0/1 DDATA RAM2
16	CPU0/1 DDATA RAM3
17	CPU0/1 DDATA RAM4
18	CPU0/1 DDATA RAM5
19	CPU0/1 DDATA RAM6
20	CPU0/1 DDATA RAM7
21	CPU0/1 ATCM BANK0
22	CPU0/1 ATCM BANK1
23	CPU0/1 B0TCM BANK0
24	CPU0/1 B0TCM BANK1
25	CPU0/1 B1TCM BANK0
26	CPU0/1 B1TCM BANK1
27	CPU0/1 VIM RAM

### 6.3.3.9 R5FSS Memory View

The memory view of each R5F (that is, the memory map as seen by each R5F) is a function of several things:

- Exception vector bootstrap: The R5F exception table (including boot vector) is always 32 bytes at address 0x00000000 as seen by the R5F. If not booting from a TCM, then boot is done over the main memory interface. The exception vector bootstrap is under software control, which allows these 32 bytes at address 0x00000000 to be remapped somewhere else in the SoC memory map.
- TCM locations: TCMs can be enabled or disabled and located at different places in the memory map, depending on bootstrap configuration. For more details, see [Table 6-26](#), and [Section 6.3.3.2.2](#).
- Peripheral interface locations: The R5F natively supports three interfaces for peripheral access. Each can be enabled/disabled and located based on bootstrap configuration. Note that the VBUSP peripheral interface must be enabled in order to use RAT and VIM.
- RAT base address: This is determined by a bootstrap. This address is located within the VBUSP peripheral interface address space.
- VIM base address: This is determined by a bootstrap. This address is located within the VBUSP peripheral interface address space.
- RAT programming: The RAT can take regions of memory accessible by the main memory interface and map them to different addresses.

The combination of the above determines what the R5F sees where in the memory map, and over what interface different transactions come out. Every transaction that does not directly address a TCM or a peripheral interface comes over the main memory interface. Transactions on the main memory interface can be further remapped with the RAT.

See [Chapter 2, Memory Map](#), for the complete R5F memory view for this device.

### 6.3.3.10 R5FSS Interrupts

Interrupt outputs: FPIXcm, FPUFCm, FPOFCm, FPDZCm, FPIDCm, and FPIOCm from R5F are not connected. Consequently, the only way to check the status of the cumulative exception status flags is by reading the FPSCR register.

### 6.3.3.11 R5FSS Debug and Trace

The R5FSS supports standard Arm CoreSight debug and trace architecture. For more details, see the *On-chip Debug* chapter.

### 6.3.3.12 R5FSS Boot Options

There are two methods of booting the R5F, or rather, two methods of placing the exception vectors (of which the boot vector is one).

The first method is to have the exception vectors external to the R5F. The user can place the exception vectors at the address indicated by the exception vector bootstrap and then program the boot vector there. When the processor exits reset, it will fetch the boot vector from this location.

The second method is to boot from a TCM. To do this, software should take the following steps:

1. Assert the correct bootstraps
  - a. To boot from ATCM, set CPUUn\_INITRAMA (or CPUUn\_INITRAMB to boot from BTCM)
  - b. Assert CPUUn\_LOCZRAMA properly for the desired TCM
2. Assert CPUUn\_HALT
3. Release the CPU from reset
4. Load the desired code into the TCM via the TCM slave port
  - a. Exception vectors should be located at address 0x00000000 of the TCM
5. De-assert CPUUn\_HALT

## 6.4 C66x DSP Subsystem

This section describes the C66x Digital Signal Processor Subsystem (C66SS) in the device.

### Note

The purpose of this chapter is to provide a basic overview of the C66x DSP and to specify its SoC integration. For full functional description of C66x DSP, refer to the documents listed in [Section 6.4.4](#).

### 6.4.1 C66SS Overview

The C66x subsystem is based on the TI's standard TMS320C66x DSP CorePac module. It includes subsystem logic to ease the C66x CorePac integration into the SoC, while maximizing software reuse from previous devices.

The C66x DSP extends the performance of the C64x+ and C674x DSPs through enhancements and new features. Many of the new features target increased performance for vector processing. The C64x+ and C674x DSPs support 2-way SIMD operations for 16-bit data and 4-way SIMD operations for 8-bit data. On C66x DSP, the vector processing capability is improved by extending the width of the SIMD instructions.

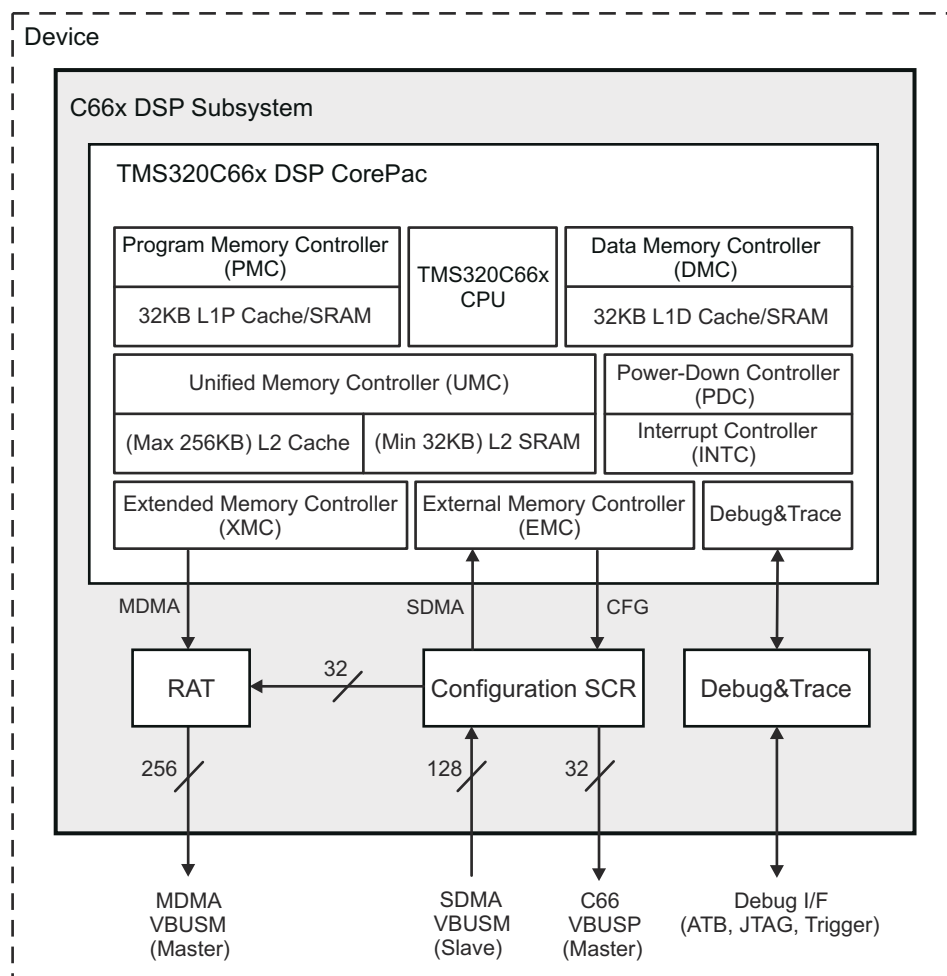
The C66x DSP can execute instructions that operate on 128-bit vectors. For example, the QMPY32 instruction is able to perform the element-to-element multiplication between two vectors of four 32-bit data each. The C66x DSP also supports SIMD for floating-point operations. Improved vector processing capability (each instruction can process multiple data in parallel) combined with the natural instruction level parallelism of C6000 architecture (for example, execution of up to eight instructions per cycle) results in a very high level of parallelism that can be exploited by DSP programmers through the use of TI's optimized C/C++ compiler.

There are two functionally identical C66SS modules in the device. [Table 6-36](#) shows C66SS modules allocation within device domains.

**Table 6-36. C66SS Modules Allocation within Device Domains**

Module Instance	Domain		
	WKUP	MCU	MAIN
C66SS0	–	–	✓
C66SS1	–	–	✓

[Figure 6-9](#) shows an overview of the C66SS.



**Figure 6-9. C66SS Overview**

## 6.4.2 C66SS Features

The C66SS supports the following features:

- Fixed/Floating-point C66x CPU based on a superset of the C64x+ and C67x+ ISA
- Program memory controller (PMC):
  - 32KB L1 program memory (L1P), configurable as cache and/or SRAM
  - When configured as a cache, L1P is a direct-mapped cache with a 32-byte cache line
  - L1P page size is 2KB
  - L1P memory controller provides bandwidth management, memory protection, and power-down functions

### Note

The C66x L1P SRAM option is disabled (not supported) in this device. Hence, C66x L1P is always configured as full cache.

- Data memory controller (DMC):
  - 32KB L1 data memory (L1D), configurable as cache and/or SRAM
  - When configured as a cache, L1D is a 2-way set-associative cache with a 64-byte cache line
  - L1D page size is 2KB
  - L1D memory controller provides bandwidth management, memory protection, and power-down functions
- Unified memory controller (UMC):
  - 288KB L2 memory, only part of which is cacheable

- Maximum 256KB can be configured as cache or SRAM
- 32KB is always mapped as SRAM
- When configured as a cache, L2 is a 4-way set associative cache with a 128-byte cache line
- L2 page size is 16KB
- L2 memory controller supports hardware prefetching and also provides bandwidth management, memory protection, and power-down functions
- Internal DMA (IDMA) engine
- External memory controller (EMC):
  - One 128-bit VBUSM slave port from main SCR
  - One 32-bit VBUSP master port to configuration SCR
- Extended memory controller (XMC):
  - One 256-bit VBUSM master port to main SCR
- Multi-stream prefetch buffer
- Address extension/translation (32-bit to 48-bit) on MDMA port using region address translation (RAT)
  - 16 regions
- Memory protection for multiple segments
- Memory protection for all internal L1/L2 RAMs
- Error detection for L1P data and tag RAMs
- Error detection and correction for L1D data and tag RAMs
- Error detection and correction for all L2 data and tag RAMs
- Hardware management for L2 SRAM and L1D cache coherency
- Integrated C66x CorePac interrupt controller (INTC) that works in conjunction with dedicated chip-level interrupt router for distribution of system interrupts to the C66x core
- Integrated leakage and dynamic power management
- Debug/emulation capabilities:
  - Support for stop mode, real-time and monitor mode debug capabilities
  - Support for processor instruction trace and system trace (printf-style debug)

### 6.4.3 C66SS Unsupported Features

The C66SS does *not* support the following:

- XMC memory protection and address extension (MPAX) 36-bit addressing. Note that:
  - RAT module is used for address translation instead. Translation is 32-bit to 48-bit
  - MPAX is supported for memory protection within the C66x CorePac
- "Full logic/RAM retention" mode featuring wake-up on both interrupt and DMA event (logic in "always on" domain). Only OFF mode is supported by C66SS, requiring full boot
- Cache coherency with MSMC (connection is through CBASS and not directly)

### 6.4.4 C66x DSP Reference Documents

The following TI documents provide more detailed description of C66x DSP:

- *TMS320C66x DSP CorePac User Guide* ([SPRUGW0](#))
- *TMS320C66x DSP CPU and Instruction Set Reference Guide* ([SPRUGH7](#)).
- *TMS320C66x DSP Cache User Guide* ([SPRUGY8](#)).



## 6.4.5 C66SS Integration

This section describes the C66SS integration in the device, including information about clocks, resets, and hardware requests.

Figure 6-10 and Figure 6-11 show the C66SS0 and C66SS1 integration, respectively.

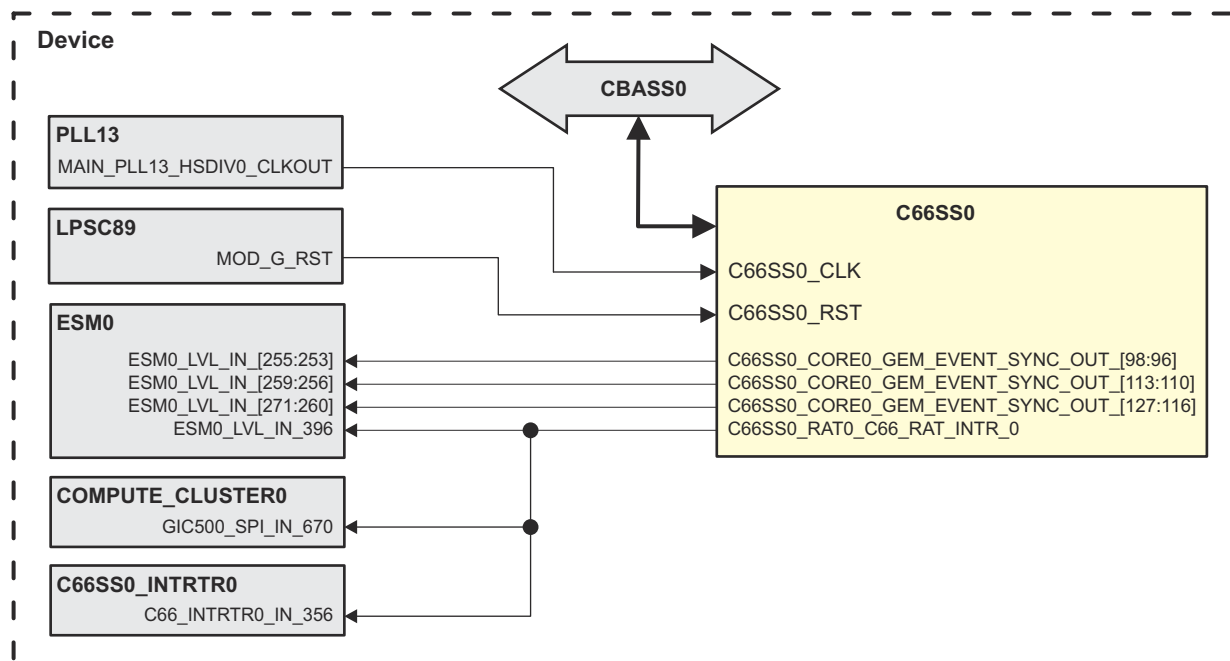


Figure 6-10. C66SS0 Integration

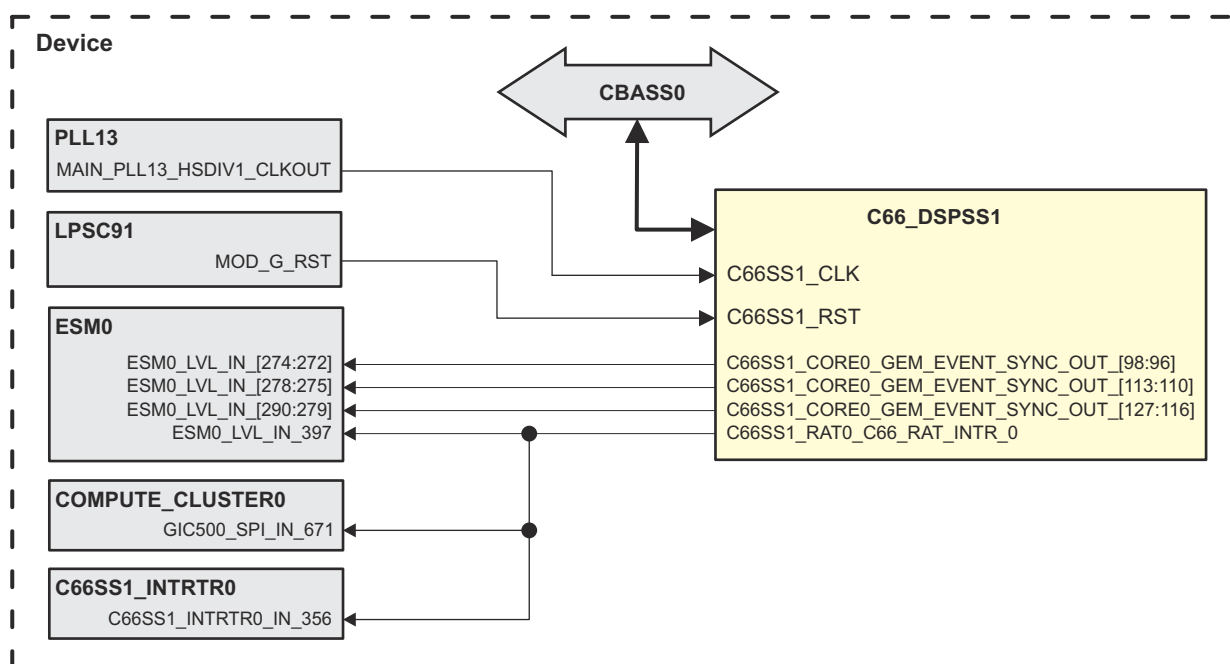


Figure 6-11. C66SS1 Integration

Table 6-37 through Table 6-39 summarize the C66SS integration.



**Table 6-37. C66SS Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
C66SS0	PSC0	PD22	LPSC89	CBASS0
C66SS1	PSC0	PD23	LPSC91	CBASS0

**Table 6-38. C66SS Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
C66SS0	C66SS0_CLK	MAIN_PLL13_HSDIV0_CLKO UT	PLL13	C66SS0 main functional clock
C66SS1	C66SS1_CLK	MAIN_PLL13_HSDIV1_CLKO UT	PLL13	C66SS1 main functional clock

Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
C66SS0	C66SS0_RST	MOD_G_RST	LPSC89	C66SS0 hardware reset
C66SS1	C66SS1_RST	MOD_G_RST	LPSC91	C66SS1 hardware reset

**Table 6-39. C66SS Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
C66SS0	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_96	ESM0_LVL_IN_253	ESM0	C66SS0 event output 96	Level
	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_97	ESM0_LVL_IN_254	ESM0	C66SS0 event output 97	Level
	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_98	ESM0_LVL_IN_255	ESM0	C66SS0 event output 98	Level
	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_110	ESM0_LVL_IN_256	ESM0	C66SS0 event output 110	Level
	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_111	ESM0_LVL_IN_257	ESM0	C66SS0 event output 111	Level
	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_112	ESM0_LVL_IN_258	ESM0	C66SS0 event output 112	Level
	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_113	ESM0_LVL_IN_259	ESM0	C66SS0 event output 113	Level
	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_116	ESM0_LVL_IN_260	ESM0	C66SS0 event output 116	Level
	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_117	ESM0_LVL_IN_261	ESM0	C66SS0 event output 117	Level
	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_118	ESM0_LVL_IN_262	ESM0	C66SS0 event output 118	Level
	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_119	ESM0_LVL_IN_263	ESM0	C66SS0 event output 119	Level
	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_120	ESM0_LVL_IN_264	ESM0	C66SS0 event output 120	Level
	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_121	ESM0_LVL_IN_265	ESM0	C66SS0 event output 121	Level
	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_122	ESM0_LVL_IN_266	ESM0	C66SS0 event output 122	Level
	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_123	ESM0_LVL_IN_267	ESM0	C66SS0 event output 123	Level
	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_124	ESM0_LVL_IN_268	ESM0	C66SS0 event output 124	Level

**Table 6-39. C66SS Hardware Requests (continued)**

	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_125	ESM0_LVL_IN_269	ESM0	C66SS0 event output 125	Level
	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_126	ESM0_LVL_IN_270	ESM0	C66SS0 event output 126	Level
	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_127	ESM0_LVL_IN_271	ESM0	C66SS0 event output 127	Level
	C66SS0_RAT0_C66_RAT_INTR_0	ESM0_LVL_IN_396 GIC500_SPI_IN_670 C66SS0_INTRTR0_IN_356	ESM0 COMPUTE_CLUSTER0 C66SS0_INT_RTR0	C66SS0 RAT exception interrupt	Level
C66SS1	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_96	ESM0_LVL_IN_272	ESM0	C66SS1 event output 96	Level
	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_97	ESM0_LVL_IN_273	ESM0	C66SS1 event output 97	Level
	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_98	ESM0_LVL_IN_274	ESM0	C66SS1 event output 98	Level
	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_110	ESM0_LVL_IN_275	ESM0	C66SS1 event output 110	Level
	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_111	ESM0_LVL_IN_276	ESM0	C66SS1 event output 111	Level
	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_112	ESM0_LVL_IN_277	ESM0	C66SS1 event output 112	Level
	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_113	ESM0_LVL_IN_278	ESM0	C66SS1 event output 113	Level
	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_116	ESM0_LVL_IN_279	ESM0	C66SS1 event output 116	Level
	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_117	ESM0_LVL_IN_280	ESM0	C66SS1 event output 117	Level
	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_118	ESM0_LVL_IN_281	ESM0	C66SS1 event output 118	Level
	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_119	ESM0_LVL_IN_282	ESM0	C66SS1 event output 119	Level
	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_120	ESM0_LVL_IN_283	ESM0	C66SS1 event output 120	Level
	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_121	ESM0_LVL_IN_284	ESM0	C66SS1 event output 121	Level
	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_122	ESM0_LVL_IN_285	ESM0	C66SS1 event output 122	Level
	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_123	ESM0_LVL_IN_286	ESM0	C66SS1 event output 123	Level
	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_124	ESM0_LVL_IN_287	ESM0	C66SS1 event output 124	Level
	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_125	ESM0_LVL_IN_288	ESM0	C66SS1 event output 125	Level
	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_126	ESM0_LVL_IN_289	ESM0	C66SS1 event output 126	Level
	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_127	ESM0_LVL_IN_290	ESM0	C66SS1 event output 127	Level
	C66SS1_RAT0_C66_RAT_INTR_0	ESM0_LVL_IN_397 GIC500_SPI_IN_671 C66SS1_INTRTR0_IN_356	ESM0 COMPUTE_CLUSTER0 C66SS1_INT_RTR0	C66SS1 RAT exception interrupt	Level

## 6.4.6 C66SS Functional Description

## 6.4.7 C66SS Cache Configuration

### Note

The purpose of this section is to provide an overview of the C66x cache memory architecture and to specify its configuration in this device. Details on the C66x cache functionality can be found in the *TMS320C66x DSP Cache User Guide* ([SPRUGY8](#)).

The device contains a 288KB level-2 memory (L2), a 32KB level-1 program memory (L1P), and a 32KB level-1 data memory (L1D). Each memory has a unique location in the memory map (see [Chapter 2, Memory Map](#)).

After device reset, L1P and L1D cache are configured as all cache, by default. The L1P and L1D cache can be reconfigured via software through the L1PMODE field of the L1P Configuration Register (L1PMODE) and the L1DMODE field of the L1D Configuration Register (L1DCFG) of the C66x CorePac. L1D is a 2-way set-associative cache, while L1P is a direct-mapped cache.

## 6.4.8 L1P Memory Configuration

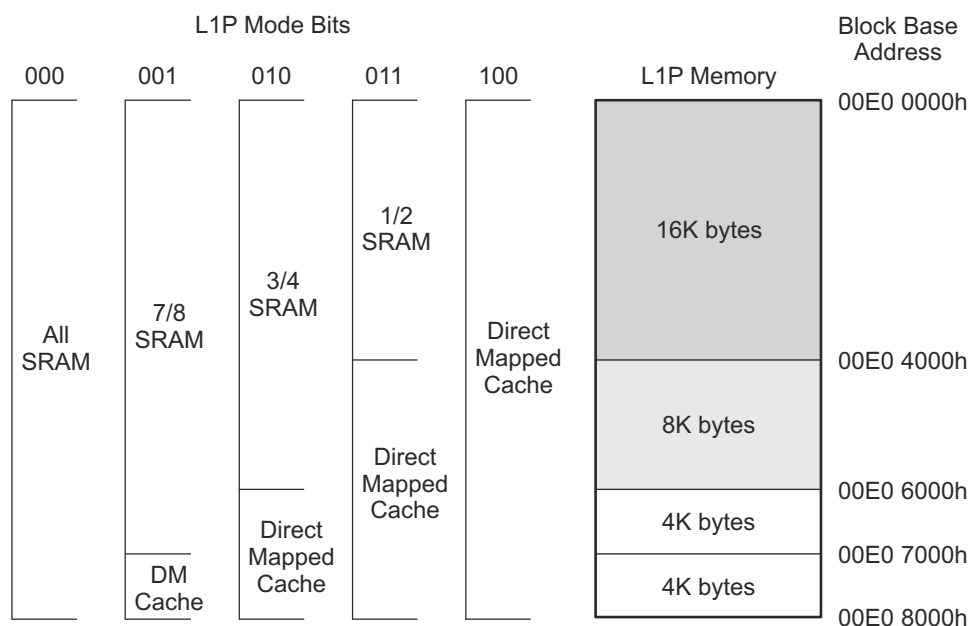
### Note

The C66x L1P SRAM option is disabled (not supported) in this device. Hence, C66x L1P is always configured as full cache.

The L1P memory configuration for this device is as follows:

- Region 0 size is 0KB (disabled)
- Region 1 size is 32KB with no wait states

[Figure 6-12](#) shows the available SRAM/cache configurations for L1P.



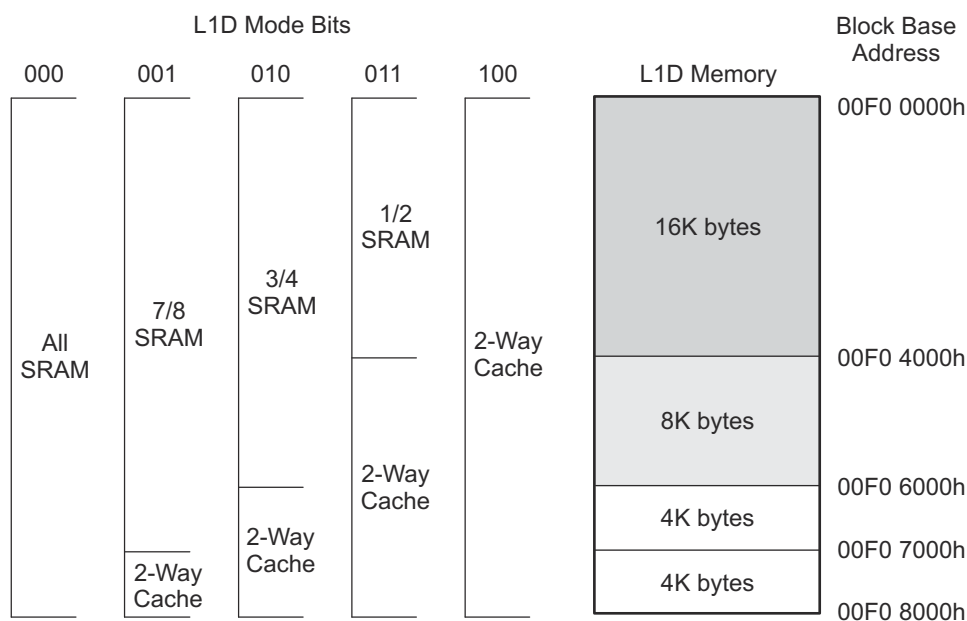
**Figure 6-12. L1P Memory Configurations**

## 6.4.9 L1D Memory Configuration

The L1D memory configuration for this device is as follows:

- Region 0 size is 0KB (disabled)
- Region 1 size is 32KB with no wait states.

Figure 6-13 shows the available SRAM/cache configurations for L1D.



**Figure 6-13. L1D Memory Configurations**

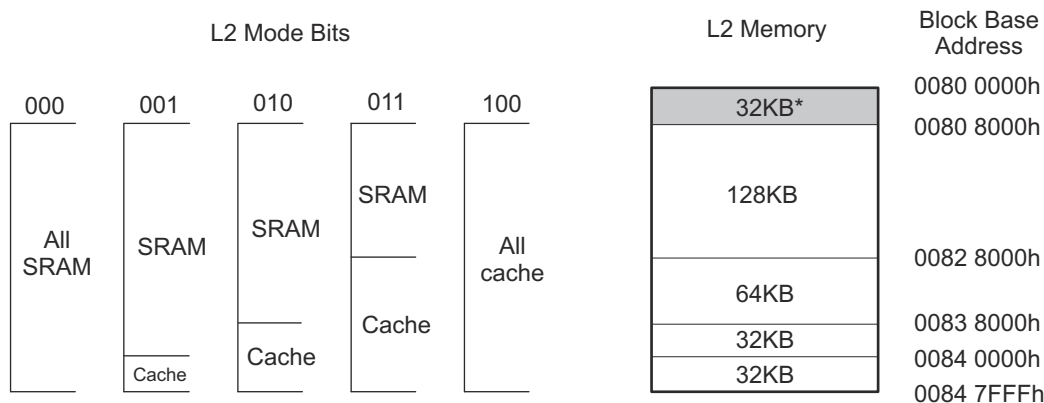
#### 6.4.10 L2 Memory Configuration

The L2 memory configuration for this device is as follows:

- 288KB of memory
- Local starting address is 0080 0000h

256KB of L2 memory can be configured as all SRAM, all 4-way set-associative cache, or a mix of the two. The other 32KB are always SRAM. The amount of L2 memory that is configured as cache is controlled through the L2MODE field of the L2 Configuration Register (L2CFG) of the C66x CorePac.

Figure 6-14 shows the available SRAM/cache configurations for L2. By default, L2 is configured as all SRAM after device reset.



\*Note: The 32KB gray-shaded block is all SRAM, non-cacheable, non-configurable by L2MODE

**Figure 6-14. L2 Memory Configurations**

### 6.4.11 C66SS ECC/Parity Support

The device supports ECC/parity on all C66x CorePac internal SRAMs. All the ECC/parity features are enabled by default (that is, after device reset). Software can disable the features per SRAM type if not required.

Table 6-40 summarizes the ECC/parity support in C66x CorePac.

**Table 6-40. C66SS ECC/Parity Support**

C66SS Memory	ECC/Parity Support
L1P Data RAM	Parity
L1P Tag RAM	Parity
L1D Data RAM	ECC-SECDED
L1D Tag RAM	ECC-SECDED
L2 Data RAM	ECC-SECDED
L2 Tag RAM	ECC-SECDED

### 6.4.12 C66SS Region Address Translation

The C66SS implements a region-based address translation (RAT) in order to extend the addressing capability of the C66x CPU from 32-bit to 48-bit. The RAT supports 16 regions, each having dedicated MMRs that define its attributes: base address (32-bit), size (up to 4GB), and translated address (48-bit). Any input transaction that starts inside of a programmed region will have its address translated, if the region is enabled. Any disabled region is ignored from the translation lookup.

For more details on RAT operation, see the *Region Address Translation* chapter.

### 6.4.13 C66SS Boot Configuration

In this device, the C66x boots from system memory instead of booting from ROM. The boot vector that defines the C66x fetch address is controlled by the following MAIN\_SEC\_MMR registers:

- For C66SS0: CLSTR2\_CORE0\_BOOTVECT\_LO[21:0] GEM\_ISTP\_RST\_VAL
- For C66SS1: CLSTR3\_CORE0\_BOOTVECT\_LO[21:0] GEM\_ISTP\_RST\_VAL

DMSC is the only master that can change the boot vector value, if needed. By default, the boot vector points to the C66x L2 SRAM at 0080 0000h.

In general, a device CPU host (typically R5F) loads a boot image to a given system memory location, sets the GEM\_ISTP\_RST\_VAL bit field to that address location, and then releases the C66x from reset. At that point, the C66x will begin fetching code from that location.

#### Note

If the GEM\_ISTP\_RST\_VAL value is modified, the change will be taken into account by C66x upon the next reset.

C66x is by default booted as non-secure mode. Due to address alignment requirement of boot vector value for C66x, a 2KB boot RAM (PSRAM2KECC) is added at SoC level. It is shared by the following CPU cores in the device: C66x, Cortex-A72 and Cortex-R5F.

### 6.4.14 C66SS Memory View

The C66x view of the address space is provided in the *Memory Map* chapter. Note that although most of the C66SS internal space is only accessible by the C66x, some modules (such as L1D and L2 memories) are also accessible by other SoC masters.

### 6.4.15 C66SS Interrupt Conditions

The C66SS supports the following features for interrupts:

- 1 NMI input event
- 128 input events
- 128 output events

### 6.4.16 C66SS Interrupt Inputs

The INTC integrated within the C66x CorePac provides flexible management of system events. The 128 input events include both internally-generated events (within the C66x CorePac) and chip-level events. In addition to these 128 events, the INTC also receives the non-maskable event and routes it straight through to the C66x DSP.

The 128 system events are either event inputs or event combinations generated by the event combiner. The event combiner logic has the capability of grouping multiple event inputs to four possible event outputs. These outputs are then provided to the interrupt selector and treated as additional system events.

The C66SS interrupt map can be found in the *Interrupts* chapter.

### 6.4.17 C66SS Interrupt Outputs

The C66x CorePac provides a 128-bit event output bus that can be used to either generates events under software control, or can convey the state of the corresponding functional interrupt/event input. [Table 6-39](#) shows which event outputs are used in this device and what is their mapping.

### 6.4.18 C66SS Clocking Considerations

The main functional clock for the C66SS is derived from a chip-level PLL. The clock rate (DIV2, DIV3, or DIV4) for the subsystem logic and the bus interfaces is set via the following MAIN\_SEC\_MMR registers:

- For C66SS0: CLSTR2\_CFG[1:0] SSCLK\_MODE
- For C66SS1: CLSTR3\_CFG[1:0] SSCLK\_MODE

The SSCLK\_MODE configuration is also propagated to the C66x CorePac and it affects the clock rates for XMC\_MDMA, EMC\_SDMA, and EMC\_CFG.

### 6.4.19 C66SS Power-Up/Down Sequences

The power-up and power-down sequences for the C66SS can be found in the *Power* chapter.

### 6.4.20 C66SS Endianness Support

The following MAIN\_SEC\_MMR registers provide the capability to select the endianness of the C66x core:

- For C66SS0: CLSTR2\_CORE0\_CFG[16] BIG\_ENDIAN
- For C66SS1: CLSTR3\_CORE0\_CFG[16] BIG\_ENDIAN

Note that while the C66x CPU core can operate in big-endian mode, the final output at the C66SS boundary is in little-endian format, as the rest of the chip is always little-endian. A logic internal to the C66SS is used to swizzle the data when the C66x is set to big-endian mode.

### 6.4.21 C66SS Debug Support

The C66SS supports the following debug features:

- Invasive debug
  - Halt mode debug
  - Real-time debug

- Monitor mode debug
- Non-invasive debug
  - Trace
- Advanced event triggering (AET)

For more details, see the *On-Chip Debug* chapter.

## 6.4.22 C66SS Registers

## 6.4.23 C66x CorePac Registers

[Table 6-42](#) lists the memory-mapped registers for the C66x CorePac. All register offset addresses not listed in [Table 6-42](#) should be considered as reserved locations and the register contents should not be modified.

**Table 6-41. C66x CorePac Instances**

Instance	Base Address
C66_COREPAC_ICFG	0100 0000h <sup>(1)</sup> <sup>(2)</sup>

- (1) C66SS0/1 private memory-mapped register space. This region is only accessible by its associated C66x core; it is not accessible by any other SoC master. The base address is the same for each C66x memory view.
- (2) Although the base address for this region is 0100 0000h, the first register is at address 0180 0000h. The [0100 0000h - 0180 0000h] space is reserved.

### Note

This section provides only a register summary for the C66x CorePac. The registers listed in [Table 6-42](#) are described in detail in *TMS320C66x DSP CorePac User Guide* ([SPRUGW0](#)).

**Table 6-42. C66x CorePac Registers**

Offset <sup>(1)</sup>	Acronym	Register Name	C66_COREPAC_ICFG Physical Address
0h to Ch	EVTFLAG0 to EVTFLAG3	Event Flag Register 0-3	0180 0000h to 0180 000Ch
20h to 2Ch	EVTSET0 to EVTSET3	Event Set Register 0-3	0180 0020h to 0180 002Ch
40h to 4Ch	EVTCLR0 to EVTCLR3	Event Clear Register 0-3	0180 0040h to 0180 004Ch
80h to 8Ch	EVTMASK0 to EVTMASK3	Event Mask Register 0-3	0180 0080h to 0180 008Ch
A0h to ACh	MEVTFLAG0 to MEVTFLAG3	Masked Event Flag Register 0-3	0180 00A0h to 0180 00ACh
C0h to CCh	EXPMASK0 to EXPMASK3	Exception Mask Register 0-3	0180 00C0h to 0180 00CCh
E0h to ECh	MEXPFLAG0 to MEXPFLAG3	Masked Exception Flag Register 0-3	0180 00E0h to 0180 00ECh
104h to 10Ch	INTMUX1 to INTMUX3	Interrupt Mux Register 1-3	0180 0104h to 0180 010Ch
140h	AEGMUX0	Advanced Event Generator Mux Register 0	0180 0140h
144h	AEGMUX1	Advanced Event Generator Mux Register 1	0180 0144h
180h	INTXSTAT	Interrupt Exception Status Register	0180 0180h
184h	INTXCLR	Interrupt Exception Clear Register	0180 0184h
188h	INTDMASK	Dropped Interrupt Mask Register	0180 0188h
1C0h	EVTASRT	Event Assert Register	0180 01C0h
10000h	PDCCMD	Power-Down Controller Command Register	0181 0000h
11100h	EDCINTMASK	Error Detect and Correct Interrupt Mask Register	0181 1100h
12000h	MM_REVID	C66x CorePac Revision ID Register	0181 2000h
20000h	IDMA0_STAT	IDMA Channel 0 Status Register	0182 0000h
20004h	IDMA0_MASK	IDMA Channel 0 Mask Register	0182 0004h
20008h	IDMA0_SOURCE	IDMA Channel 0 Source Address Register	0182 0008h



**Table 6-42. C66x CorePac Registers (continued)**

Offset <sup>(1)</sup>	Acronym	Register Name	C66_COREPAC_ICFG Physical Address
2000Ch	IDMA0_DEST	IDMA Channel 0 Destination Address Register	0182 000Ch
20010h	IDMA0_COUNT	IDMA Channel 0 Count Register	0182 0010h
20100h	IDMA1_STAT	IDMA Channel 1 Status Register	0182 0100h
20108h	IDMA1_SOURCE	IDMA Channel 1 Source Address Register	0182 0108h
2010Ch	IDMA1_DEST	IDMA Channel 1 Destination Address Register	0182 010Ch
20110h	IDMA1_COUNT	IDMA Channel 1 Count Register	0182 0110h
20200h	CPUARBE	EMC DSP Arbitration Control Register	0182 0200h
20204h	IDMAARBE	EMC IDMA Arbitration Control Register	0182 0204h
20208h	SDMAARBE	EMC Slave DMA Arbitration Control Register	0182 0208h
20210h	ECFGARBE	EMC CFG Arbitration Control Register	0182 0210h
20300h	ICFGMPFAR	CFG Memory Protection Fault Address Register	0182 0300h
20304h	ICFGMPFSR	CFG Memory Protection Fault Status Register	0182 0304h
20308h	ICFGMPFCR	CFG Memory Protection Fault Command Register	0182 0308h
20408h	ECFGERR	CFG Bus Error Register	0182 0408h
2040Ch	ECFGERRCLR	CFG Bus Error Clear Register	0182 040Ch
20500h to 2053Ch	PAMAP0 to PAMAP15	PAMAP Register 0-15	0182 0500h to 0182 053Ch
21104h	EDCINTFLG	Error Detect and Correct Interrupt Flag Register	0182 1104h
21108h	L1DEDCMD	L1D Error Detect Command Register	0182 1108h
2110Ch	L1DDCSTAT	L1D Error Detect DATA Correctable Status Register	0182 110Ch
21110h	L1DDNCSTAT	L1D Error Detect DATA Non-Correctable Status Register	0182 1110h
21114h	L1DTCSTAT	L1D Error Detect TAG Correctable Status Register	0182 1114h
21118h	L1DTNCSTAT	L1D Error Detect TAG Non-Correctable Status Register	0182 1118h
2111Ch	L1DDEDADDR	L1D Error Detect Correctable and Non-Correctable DATA Address Register	0182 111Ch
21120h	L1DTEDADDR	L1D Error Detect Correctable and Non-Correctable TAG Address Register	0182 1120h
21124h	L1DEDCNT	L1D EDC Count Register	0182 1124h
21128h	L2TEDCMD	L2 Error Detect Command Register	0182 1128h
2112Ch	L2TCSTAT	L2 Error Detect TAG Correctable Status Register	0182 112Ch
21130h	L2TNCSTAT	L2 Error Detect TAG Non-Correctable Status Register	0182 1130h
21134h	L2TEDADDR	L2 Error Detect TAG Correctable and Non-Correctable Address Register	0182 1134h
21138h	L2MCSTAT	L2 Error Detect MPPA Correctable Status Register	0182 1138h
2113Ch	L2MNCSTAT	L2 Error Detect MPPA Non-Correctable Status Register	0182 113Ch
21140h	L2MEDADDR	L2 Error Detect MPPA Correctable and Non-Correctable Address Register	0182 1140h
21144h	L2SCSTAT	L2 Error Detect SNOP Correctable Status Register	0182 1144h
21148h	L2SNCSTAT	L2 Error Detect SNOP Non-Correctable Status Register	0182 1148h
2114Ch	L2SEDADDR	L2 Error Detect SNOP Correctable and Non-Correctable Address Register	0182 114Ch
21150h	L2LCSTAT	L2 Error Detect LRU Correctable Status Register	0182 1150h
21154h	L2LNCSTAT	L2 Error Detect LRU Non-Correctable Status Register	0182 1154h
21158h	L2LEDADDR	L2 Error Detect LRU Correctable and Non-Correctable Address Register	0182 1158h
2115Ch	L2TEDCNT	L2 Error Detect Parity Error Count Register	0182 115Ch
21160h	L1PTEDCMD	L1P Error Detect TAG Command Register	0182 1160h
21164h	L1PTEDSTAT	L1P Error Detect TAG Status Register	0182 1164h
21168h	L1PTEDADDR	L1P Error Detect TAG Lower Address Register	0182 1168h
2116Ch	L1DTEDCNT	L1P Error Detect TAG Parity Error Count Register	0182 116Ch
40000h	L2CFG	L2 Configuration Register	0184 0000h

**Table 6-42. C66x CorePac Registers (continued)**

Offset <sup>(1)</sup>	Acronym	Register Name	C66_COREPAC_ICFG Physical Address
40020h	L1PCFG	L1P Configuration Register	0184 0020h
40024h	L1PCC	L1P Cache Control Register	0184 0024h
40040h	L1DCFG	L1D Cache Configuration Register	0184 0040h
40044h	L1DCC	L1D Cache Control Register	0184 0044h
41000h	CPUARBU	L2 DSP Arbitration Control Register	0184 1000h
41004h	IDMAARBU	L2 IDMA Arbitration Control Register	0184 1004h
41008h	SDMAARBU	L2 Slave DMA Arbitration Control Register	0184 1008h
4100Ch	UCARBU	L2 User Coherence Arbitration Control Register	0184 100Ch
41010h	MDMAARBU	L2 Master DMA Arbitration Control Register	0184 1010h
41040h	CPUARBD	L1 DSP Arbitration Control Register	0184 1040h
41044h	IDMAARBD	L1 IDMA Arbitration Control Register	0184 1044h
41048h	SDMAARBD	L1 Slave DMA Arbitration Control Register	0184 1048h
4104Ch	UCARBD	L1 User Coherence Arbitration Control Register	0184 104Ch
44000h	L2WBAR	L2 Writeback Base Address Register	0184 4000h
44004h	L2WWC	L2 Writeback Word Count Register	0184 4004h
44010h	L2WIBAR	L2 Writeback-Invalidate Base Address Register	0184 4010h
44014h	L2WIWC	L2 Writeback-Invalidate Word Count Register	0184 4014h
44018h	L2IBAR	L2 Invalidate Base Address Register	0184 4018h
4401Ch	L2IWC	L2 Invalidate Word Count Register	0184 401Ch
44020h	L1PIBAR	L1 Program Invalidate Base Address Register	0184 4020h
44024h	L1PIWC	L1 Program Invalidate Word Count Register	0184 4024h
44030h	L1DWIBAR	L1D Writeback-Invalidate Base Address Register	0184 4030h
44034h	L1DWIWC	L1D Writeback-Invalidate Word Count Register	0184 4034h
44040h	L1DWBAR	L1D Writeback Base Address Register	0184 4040h
44044h	L1DWWC	L1D Writeback Word Count Register	0184 4044h
44048h	L1DIBAR	L1D Invalidate Base Address Register	0184 4048h
4404Ch	L1DIWC	L1D Invalidate Word Count Register	0184 404Ch
45000h	L2WB	L2 Writeback Register	0184 5000h
45004h	L2WBINV	L2 Writeback-Invalidate Register	0184 5004h
45008h	L2INV	L2 Invalidate Register	0184 5008h
45028h	L1PINV	L1 Program Invalidate Register	0184 5028h
45040h	L1DWB	L1D Writeback Register	0184 5040h
45044h	L1DWBINV	L1D Writeback-Invalidate Register	0184 5044h
45048h	L1DINV	L1D Invalidate Register	0184 5048h
46004h	L2EDSTAT	L2 Error Detection Status Register	0184 6004h
46008h	L2EDCMD	L2 Error Detection Command Register	0184 6008h
4600Ch	L2EDADDR	L2 Error Detection Address Register	0184 600Ch
46018h	L2EDCPEC	L2 Error Detection Correctable Parity Error Counter Register	0184 6018h
4601Ch	L2EDCNEC	L2 Error Detection Non-Correctable Parity Error Counter Register	0184 601Ch
46020h	MDMAERR	MDMA Bus Error Register	0184 6020h
46024h	MDMAERRCLR	MDMA Bus Error Clear Register	0184 6024h
46030h	L2EDCEN	L2 Error Detection and Correction Enable Register	0184 6030h
46404h	L1PEDSTAT	L1P Error Detection Status Register	0184 6404h
46408h	L1PEDCMD	L1P Error Detection Command Register	0184 6408h

**Table 6-42. C66x CorePac Registers (continued)**

Offset <sup>(1)</sup>	Acronym	Register Name	C66_COREPAC_ICFG Physical Address
4640Ch	L1PEDADDR	L1P Error Detection Address Register	0184 640Ch
48000h to 483FCh	MAR0 to MAR255	Memory Attribute Registers	0184 8000h to 0184 83FCh
4A000h	L2MPFAR	L2 Memory Protection Fault Address Register	0184 A000h
4A004h	L2MPFSR	L2 Memory Protection Fault Set Register	0184 A004h
4A008h	L2MPFCR	L2 Memory Protection Fault Clear Register	0184 A008h
4A200h to 4A27Ch	L2MPPA0 to L2MPPA31	L2 Memory Protection Page Attribute Registers	0184 A200h to 0184 A27Ch
4A400h	L1PMPFAR	L1P Memory Protection Fault Address Register	0184 A400h
4A404h	L1PMPFSR	L1P Memory Protection Fault Set Register	0184 A404h
4A408h	L1PMPFCR	L1P Memory Protection Fault Clear Register	0184 A408h
4A640h to 4A67Ch	L1PMPPA0 to L1PMPPA15	L1P Memory Protection Page Attribute Registers	0184 A640h to 0184 A67Ch
4AC00h	L1DMPFAR	L1D Memory Protection Fault Address Register	0184 AC00h
84AC04h	L1DMPFSR	L1D Memory Protection Fault Set Register	0184 AC04h
4AC08h	L1DMPFCR	L1D Memory Protection Fault Clear Register	0184 AC08h
4AD00h to 4AD0Ch	MPLK0 to MPLK3	Memory Protection Lock Registers	0184 AD00h to 0184 AD0Ch
4AD10h	MPLKCMD	Memory Protection Lock Command Register	0184 AD10h
4AD14h	MPLKSTAT	Memory Protection Lock Status Register	0184 AD14h
4AE40h to 4AE7Ch	L1DMPPA0 to L1DMPPA15	L1D Memory Page Protection Attribute Registers	0184 AE40h to 0184 AE7Ch

(1) All offsets in this table are based on the address of the first register in the ICFG region (0180 0000h).

## 6.4.24 C66x RAT Registers

Table 6-44 lists the memory-mapped registers for the C66x RAT registers. All register offset addresses not listed in Table 6-44 should be considered as reserved locations and the register contents should not be modified.

**Table 6-43. C66x RAT Instances**

Instance	Base Address
C66_COREPAC_C66_RATCFG	07FF 0000h <sup>(1)</sup>

- (1) C66SS0/1 private memory-mapped register space. This region is only accessible by its associated C66x core; it is not accessible by any other SoC master. The base address is the same for each C66x memory view.

**Table 6-44. C66x RAT Registers**

Offset	Acronym	Register Name	C66_COREPAC_C66_RATCFG Physical Address
0h	<a href="#">C66_RAT_PID</a>	Revision Register	07FF 0000h
4h	<a href="#">C66_RAT_CONFIG</a>	Config Register	07FF 0004h
20h + formula	<a href="#">C66_RAT_CTRL_J</a>	Region Control Register	07FF 0020h + formula
24h + formula	<a href="#">C66_RAT_BASE_J</a>	Region Base Register	07FF 0024h + formula
28h + formula	<a href="#">C66_RAT_TRANS_L_J</a>	Region Translated Lower Address	07FF 0028h + formula
2Ch + formula	<a href="#">C66_RAT_TRANS_U_J</a>	Region Translated Upper Address	07FF 002Ch + formula
804h	<a href="#">C66_RAT_DESTINATION_ID</a>	Destination ID Register	07FF 0804h
820h	<a href="#">C66_RAT_EXCEPTION_LOGGING_CONTROL</a>	Exception Logging Control Register	07FF 0820h
824h	<a href="#">C66_RAT_EXCEPTION_LOGGING_HEADER0</a>	Exception Logging Header 0 Register	07FF 0824h
828h	<a href="#">C66_RAT_EXCEPTION_LOGGING_HEADER1</a>	Exception Logging Header 1 Register	07FF 0828h
82Ch	<a href="#">C66_RAT_EXCEPTION_LOGGING_DATA0</a>	Exception Logging Data 0 Register	07FF 082Ch
830h	<a href="#">C66_RAT_EXCEPTION_LOGGING_DATA1</a>	Exception Logging Data 1 Register	07FF 0830h
834h	<a href="#">C66_RAT_EXCEPTION_LOGGING_DATA2</a>	Exception Logging Data 2 Register	07FF 0834h
838h	<a href="#">C66_RAT_EXCEPTION_LOGGING_DATA3</a>	Exception Logging Data 3 Register	07FF 0838h
840h	<a href="#">C66_RAT_EXCEPTION_PEND_SET</a>	Exception Logging Interrupt Pending Set Register	07FF 0840h
844h	<a href="#">C66_RAT_EXCEPTION_PEND_CLEAR</a>	Exception Logging Interrupt Pending Clear Register	07FF 0844h
848h	<a href="#">C66_RAT_EXCEPTION_ENABLE_SET</a>	Exception Logging Interrupt Enable Set Register	07FF 0848h
84Ch	<a href="#">C66_RAT_EXCEPTION_ENABLE_CLEAR</a>	Exception Logging Interrupt Enable Clear Register	07FF 084Ch
850h	<a href="#">C66_RAT_EOI_REG</a>	EOI Register	07FF 0850h

#### 6.4.25 C66\_RAT\_PID Register (Offset = 0h) [reset = 66803100h]

C66\_RAT\_PID is shown in [Figure 6-15](#) and described in [Table 6-46](#).

Return to [Summary Table](#).

This register contains the major and minor revisions for the module.

**Table 6-45. C66\_RAT\_PID Instances**

Instance	Physical Address
C66_COREPAC_C66_RATCFG	07FF 0000h

**Figure 6-15. C66\_RAT\_PID Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REV																															
R-66803100h																															

LEGEND: R = Read Only; -n = value after reset

**Table 6-46. C66\_RAT\_PID Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	REV	R	66803100h	TI internal data. Identifies revision of peripheral.

### 6.4.26 C66\_RAT\_CONFIG Register (Offset = 4h) [reset = 300210h]

C66\_RAT\_CONFIG is shown in [Figure 6-16](#) and described in [Table 6-48](#).

Return to [Summary Table](#).

This register contains the configuration values for the module.

**Table 6-47. C66\_RAT\_CONFIG Instances**

Instance	Physical Address
C66_COREPAC_C66_RATCFG	07FF 0004h

**Figure 6-16. C66\_RAT\_CONFIG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED								ADDR_WIDTH							
R-0h								R-30h							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRS								REGIONS							
R-1h								R-10h							

LEGEND: R = Read Only; -n = value after reset

**Table 6-48. C66\_RAT\_CONFIG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved.
23-16	ADDR_WIDTH	R	30h	Number of address bits.
15-8	ADDRS	R	2h	Number of addresses.
7-0	REGIONS	R	10h	Number of regions.

### 6.4.27 C66\_RAT\_CTRL\_j Register (Offset = 20h) [reset = 0h]

C66\_RAT\_CTRL\_j is shown in [Figure 6-17](#) and described in [Table 6-50](#).

Return to [Summary Table](#).

This region controls the size and the enable for a region.

Offset = 20h + (j \* 10h); where j = 0h to Fh.

**Table 6-49. C66\_RAT\_CTRL\_j Instances**

Instance	Physical Address
C66_COREPAC_C66_RATCFG	07FF 0020h

**Figure 6-17. C66\_RAT\_CTRL\_j Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN	RESERVED														
R/W-0h								R-0h							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED										SIZE					
R-0h										R/W-0h					

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 6-50. C66\_RAT\_CTRL\_j Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	EN	R/W	0h	Enable for the region.
30-6	RESERVED	R	0h	Reserved.
5-0	SIZE	R/W	0h	Size of the region in address bits. 0h = 1B 1h = 2B 2h = 4B ... 20h = 4GB

#### 6.4.28 C66\_RAT\_BASE\_j Register (Offset = 24h) [reset = 0h]

C66\_RAT\_BASE\_j is shown in [Figure 6-18](#) and described in [Table 6-52](#).

Return to [Summary Table](#).

This register is used for the base address for a region. This is the source address for matching to a region.

Offset = 24h + (j \* 10h); where j = 0h to Fh.

**Table 6-51. C66\_RAT\_BASE\_j Instances**

Instance	Physical Address
C66_COREPAC_C66_RATCFG	07FF 0024h

**Figure 6-18. C66\_RAT\_BASE\_j Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BASE																															
R/W-0h																															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 6-52. C66\_RAT\_BASE\_j Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	BASE	R/W	0h	Base address for the region. It must be aligned to the programmed size.



#### 6.4.29 C66\_RAT\_TRANS\_L\_j Register (Offset = 28h) [reset = 0h]

C66\_RAT\_TRANS\_L\_j is shown in [Figure 6-19](#) and described in [Table 6-54](#).

Return to [Summary Table](#).

This register contains the translated lower address bits for a region.

Offset = 28h + (j \* 10h); where j = 0h to Fh.

**Table 6-53. C66\_RAT\_TRANS\_L\_j Instances**

Instance	Physical Address
C66_COREPAC_C66_RATCFG	07FF 0028h

**Figure 6-19. C66\_RAT\_TRANS\_L\_j Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOWER																															
R/W-0h																															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 6-54. C66\_RAT\_TRANS\_L\_j Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	LOWER	R/W	0h	Translated lower address bits for the region. It must be aligned to the programmed size.

### 6.4.30 C66\_RAT\_TRANS\_U\_j Register (Offset = 2Ch) [reset = 0h]

C66\_RAT\_TRANS\_U\_j is shown in [Figure 6-20](#) and described in [Table 6-56](#).

Return to [Summary Table](#).

This register contains the translated upper address bits for a region.

Offset = 2Ch + (j \* 10h); where j = 0h to Fh

**Table 6-55. C66\_RAT\_TRANS\_U\_j Instances**

Instance	Physical Address
C66_COREPAC_C66_RATCFG	07FF 002Ch

**Figure 6-20. C66\_RAT\_TRANS\_U\_j Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																UPPER															
R-0h																R/W-0h															

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 6-56. C66\_RAT\_TRANS\_U\_j Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved.
15-0	UPPER	R/W	0h	Translated upper address bits for the region.

### 6.4.31 C66\_RAT\_DESTINATION\_ID Register (Offset = 804h) [reset = 0h]

C66\_RAT\_DESTINATION\_ID is shown in [Figure 6-21](#) and described in [Table 6-58](#).

Return to [Summary Table](#).

This register defines the destination ID value for error messages.

**Table 6-57. C66\_RAT\_DESTINATION\_ID Instances**

Instance	Physical Address
C66_COREPAC_C66_RATCFG	07FF 0804h

**Figure 6-21. C66\_RAT\_DESTINATION\_ID Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																DEST_ID															
R-0h																R/W-0h															

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 6-58. C66\_RAT\_DESTINATION\_ID Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved.
7-0	DEST_ID	R/W	0h	Destination ID.

### 6.4.32 C66\_RAT\_EXCEPTION\_LOGGING\_CONTROL Register (Offset = 820h) [reset = 0h]

C66\_RAT\_EXCEPTION\_LOGGING\_CONTROL is shown in [Figure 6-22](#) and described in [Table 6-60](#).

Return to [Summary Table](#).

This register controls the exception logging.

**Table 6-59.**  
**C66\_RAT\_EXCEPTION\_LOGGING\_CONTROL**  
**Instances**

Instance	Physical Address
C66_COREPAC_C66_RATCFG	07FF 0820h

**Figure 6-22. C66\_RAT\_EXCEPTION\_LOGGING\_CONTROL Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED						DISABLE_INTR	DISABLE_F
R-0h						R/W-0h	R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 6-60. C66\_RAT\_EXCEPTION\_LOGGING\_CONTROL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	Reserved.
1	DISABLE_INTR	R/W	0h	Disables logging interrupt when set.
0	DISABLE_F	R/W	0h	Disables logging when set.

### 6.4.33 C66\_RAT\_EXCEPTION\_LOGGING\_HEADER0 Register (Offset = 824h) [reset = 0h]

C66\_RAT\_EXCEPTION\_LOGGING\_HEADER0 is shown in [Figure 6-23](#) and described in [Table 6-62](#).

Return to [Summary Table](#).

This register contains the first word of the header.

**Table 6-61.**  
**C66\_RAT\_EXCEPTION\_LOGGING\_HEADER0**  
**Instances**

Instance	Physical Address
C66_COREPAC_C66_RATCFG	07FF 0824h

**Figure 6-23. C66\_RAT\_EXCEPTION\_LOGGING\_HEADER0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPE_F								SRC_ID								DEST_ID															
R-0h								R-0h								R-0h															

LEGEND: R = Read Only; -n = value after reset

**Table 6-62. C66\_RAT\_EXCEPTION\_LOGGING\_HEADER0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	TYPE_F	R	0h	Type. 4h = RAT.
23-8	SRC_ID	R	0h	Source ID.
7-0	DEST_ID	R	0h	Destination ID.

#### 6.4.34 C66\_RAT\_EXCEPTION\_LOGGING\_HEADER1 Register (Offset = 828h) [reset = 0h]

C66\_RAT\_EXCEPTION\_LOGGING\_HEADER1 is shown in [Figure 6-24](#) and described in [Table 6-64](#).

Return to [Summary Table](#).

This register contains the second word of the header.

**Table 6-63.**  
**C66\_RAT\_EXCEPTION\_LOGGING\_HEADER1**  
**Instances**

Instance	Physical Address
C66_COREPAC_C66_RATCFG	07FF 0828h

**Figure 6-24. C66\_RAT\_EXCEPTION\_LOGGING\_HEADER1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GROUP								CODE								RESERVED															
R-0h								R-0h								R-0h															

LEGEND: R = Read Only; -n = value after reset

**Table 6-64. C66\_RAT\_EXCEPTION\_LOGGING\_HEADER1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	GROUP	R	0h	Group.
23-16	CODE	R	0h	Code. 1h = Boundary crossing error.
15-0	RESERVED	R	0h	Reserved.

#### 6.4.35 C66\_RAT\_EXCEPTION\_LOGGING\_DATA0 Register (Offset = 82Ch) [reset = 0h]

C66\_RAT\_EXCEPTION\_LOGGING\_DATA0 is shown in [Figure 6-25](#) and described in [Table 6-66](#).

Return to [Summary Table](#).

This register contains the first word of the data.

**Table 6-65.**  
**C66\_RAT\_EXCEPTION\_LOGGING\_DATA0 Instances**

Instance	Physical Address
C66_COREPAC_C66_RATCFG	07FF 082Ch

**Figure 6-25. C66\_RAT\_EXCEPTION\_LOGGING\_DATA0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_L																															
R-0h																															

LEGEND: R = Read Only; -n = value after reset

**Table 6-66. C66\_RAT\_EXCEPTION\_LOGGING\_DATA0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	ADDR_L	R	0h	Address lower 32 bits.

### 6.4.36 C66\_RAT\_EXCEPTION\_LOGGING\_DATA1 Register (Offset = 830h) [reset = 0h]

C66\_RAT\_EXCEPTION\_LOGGING\_DATA1 is shown in [Figure 6-26](#) and described in [Table 6-68](#).

Return to [Summary Table](#).

This register contains the second word of the data.

**Table 6-67.**

**C66\_RAT\_EXCEPTION\_LOGGING\_DATA1 Instances**

Instance	Physical Address
C66_COREPAC_C66_RATCFG	07FF 0830h

**Figure 6-26. C66\_RAT\_EXCEPTION\_LOGGING\_DATA1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																ADDR_H															
R-0h																R-0h															

LEGEND: R = Read Only; -n = value after reset

**Table 6-68. C66\_RAT\_EXCEPTION\_LOGGING\_DATA1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved.
15-0	ADDR_H	R	0h	Address upper 12 bits.



#### 6.4.37 C66\_RAT\_EXCEPTION\_LOGGING\_DATA2 Register (Offset = 834h) [reset = 0h]

C66\_RAT\_EXCEPTION\_LOGGING\_DATA2 is shown in [Figure 6-27](#) and described in [Table 6-70](#).

Return to [Summary Table](#).

This register contains the third word of the data.

**Table 6-69.**  
**C66\_RAT\_EXCEPTION\_LOGGING\_DATA2 Instances**

Instance	Physical Address
C66_COREPAC_C66_RATCFG	07FF 0834h

**Figure 6-27. C66\_RAT\_EXCEPTION\_LOGGING\_DATA2 Register**

31	30	29	28	27	26	25	24
RESERVED				ROUTEID			
R-0h				R-0h			
23	22	21	20	19	18	17	16
ROUTEID							
R-0h							
15	14	13	12	11	10	9	8
RESERVED		WRITE	READ	DEBUG	CACHEABLE	PRIV	SECURE
R-0h		R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
PRIV_ID							
R-0h							

LEGEND: R = Read Only; -n = value after reset

**Table 6-70. C66\_RAT\_EXCEPTION\_LOGGING\_DATA2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0h	Reserved.
27-16	ROUTEID	R	0h	Route ID.
15-14	RESERVED	R	0h	Reserved.
13	WRITE	R	0h	Write.
12	READ	R	0h	Read.
11	DEBUG	R	0h	Debug.
10	CACHEABLE	R	0h	Cacheable.
9	PRIV	R	0h	Priv.
8	SECURE	R	0h	Secure.
7-0	PRIV_ID	R	0h	Priv ID.

### 6.4.38 C66\_RAT\_EXCEPTION\_LOGGING\_DATA3 Register (Offset = 838h) [reset = 0h]

C66\_RAT\_EXCEPTION\_LOGGING\_DATA3 is shown in [Figure 6-28](#) and described in [Table 6-72](#).

Return to [Summary Table](#).

This register contains the fourth word of the data. Reading this register will clear the error pending bit.

**Table 6-71.**

**C66\_RAT\_EXCEPTION\_LOGGING\_DATA3 Instances**

Instance	Physical Address
C66_COREPAC_C66_RATCFG	07FF 0838h

**Figure 6-28. C66\_RAT\_EXCEPTION\_LOGGING\_DATA3 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																						BYTECNT									
R-0h																						R-0h									

LEGEND: R = Read Only; -n = value after reset

**Table 6-72. C66\_RAT\_EXCEPTION\_LOGGING\_DATA3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0h	Reserved.
9-0	BYTECNT	R	0h	Byte count.

### 6.4.39 C66\_RAT\_EXCEPTION\_PEND\_SET Register (Offset = 840h) [reset = 0h]

C66\_RAT\_EXCEPTION\_PEND\_SET is shown in [Figure 6-29](#) and described in [Table 6-74](#).

Return to [Summary Table](#).

This register allows to set the exception pending signal.

**Table 6-73. C66\_RAT\_EXCEPTION\_PEND\_SET  
Instances**

Instance	Physical Address
C66_COREPAC_C66_RATCFG	07FF 0840h

**Figure 6-29. C66\_RAT\_EXCEPTION\_PEND\_SET Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							PEND_SET
R-0h							R/W1S-0h

LEGEND: R = Read Only; R/W = Read/Write; R/W1S = Read/Write 1 to Set Bit; -n = value after reset

**Table 6-74. C66\_RAT\_EXCEPTION\_PEND\_SET Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved.
0	PEND_SET	R/W1S	0h	Write a 1 to set the exception pending signal.

#### 6.4.40 C66\_RAT\_EXCEPTION\_PEND\_CLEAR Register (Offset = 844h) [reset = 0h]

C66\_RAT\_EXCEPTION\_PEND\_CLEAR is shown in [Figure 6-30](#) and described in [Table 6-76](#).

Return to [Summary Table](#).

This register allows to clear the pend signal.

**Table 6-75. C66\_RAT\_EXCEPTION\_PEND\_CLEAR  
Instances**

Instance	Physical Address
C66_COREPAC_C66_RATCFG	07FF 0844h

**Figure 6-30. C66\_RAT\_EXCEPTION\_PEND\_CLEAR Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							PEND_CLR
R-0h							R/W1C-0h

LEGEND: R = Read Only; R/W = Read/Write; R/W1C = Read/Write 1 to Clear Bit; -n = value after reset

**Table 6-76. C66\_RAT\_EXCEPTION\_PEND\_CLEAR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved.
0	PEND_CLR	R/W1C	0h	Write a 1 to clear the exception pending signal.

#### 6.4.41 C66\_RAT\_EXCEPTION\_ENABLE\_SET Register (Offset = 848h) [reset = 0h]

C66\_RAT\_EXCEPTION\_ENABLE\_SET is shown in [Figure 6-31](#) and described in [Table 6-78](#).

Return to [Summary Table](#).

This register allows to set the interrupt enable signal.

**Table 6-77. C66\_RAT\_EXCEPTION\_ENABLE\_SET Instances**

Instance	Physical Address
C66_COREPAC_C66_RATCFG	07FF 0848h

**Figure 6-31. C66\_RAT\_EXCEPTION\_ENABLE\_SET Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							ENABLE_SET
R-0h							R/W1S-0h

LEGEND: R = Read Only; R/W = Read/Write; R/W1S = Read/Write 1 to Set Bit; -n = value after reset

**Table 6-78. C66\_RAT\_EXCEPTION\_ENABLE\_SET Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved.
0	ENABLE_SET	R/W1S	0h	Write a 1 to set the exception interrupt enable signal.

#### 6.4.42 C66\_RAT\_EXCEPTION\_ENABLE\_CLEAR Register (Offset = 84Ch) [reset = 0h]

C66\_RAT\_EXCEPTION\_ENABLE\_CLEAR is shown in [Figure 6-32](#) and described in [Table 6-80](#).

Return to [Summary Table](#).

This register allows to clear the interrupt enable signal.

**Table 6-79. C66\_RAT\_EXCEPTION\_ENABLE\_CLEAR Instances**

Instance	Physical Address
C66_COREPAC_C66_RATCFG	07FF 084Ch

**Figure 6-32. C66\_RAT\_EXCEPTION\_ENABLE\_CLEAR Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							ENABLE_CLR
R-0h							R/W1C-0h

LEGEND: R = Read Only; R/W = Read/Write; R/W1C = Read/Write 1 to Clear Bit; -n = value after reset

**Table 6-80. C66\_RAT\_EXCEPTION\_ENABLE\_CLEAR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved.
0	ENABLE_CLR	R/W1C	0h	Write a 1 to clear the exception interrupt enable signal.

#### 6.4.43 C66\_RAT\_EOI\_REG Register (Offset = 850h) [reset = 0h]

C66\_RAT\_EOI\_REG is shown in [Figure 6-33](#) and described in [Table 6-82](#).

Return to [Summary Table](#).

##### EOI Register

The EOI register is used to re-trigger the pulse interrupt signal to ensure that any nested interrupt events are serviced. The software interrupt handler must write to the EOI register at the end of the current interrupt processing routine, so that new events can re-trigger the pulse interrupt signal again. For level interrupt signals the EOI register is not functional and must not be used.

**Table 6-81. C66\_RAT\_EOI\_REG Instances**

Instance	Physical Address
C66_COREPAC_C66_RATCFG	07FF 0850h

**Figure 6-33. C66\_RAT\_EOI\_REG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																EOI_WR															
R-0h																R/W-0h															

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 6-82. C66\_RAT\_EOI\_REG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved.
15-0	EOI_WR	R/W	0h	EOI value.

## 6.5 C71x DSP Subsystem

This section describes the C71x Digital Signal Processor Subsystem (C71SS) in the device.

### 6.5.1 C71SS Overview

The TMS320C71x is the next-generation fixed and floating-point DSP platform. The C71x DSP is a new core in the Texas Instruments' DSP family. The C71x DSP supports vector signal processing, providing significant lift in DSP processing power over a broad range of general signal processing tasks in comparison to the C6x DSP family. In addition, the C71x provides several specialized functions which accelerate targeted functions by more than 30 times. Besides expanding vector processing capabilities, the new C71x core also incorporates advanced techniques to improve control code efficiency and ease of programming such as branch prediction, protected pipeline, precise exception and virtual memory management.

The C71x builds on the C6x's legacy but it is not binary compatible with any previous C6x DSPs. Many of the C71x architecture features are new and unique. The C71x is designed as a platform with these characteristics:

- True 64-bit machine with 64-bit memory addressing (64-bit virtual memory addressing, 40-bit physical addressing) and single-cycle 64-bit base arithmetic operations
- Achieves 4 to 32 times DSP processing capacity compared to C66x
- Preserves out-of-the-box performance parity for C code which has been optimized for C66x with a few exceptions
- Provides a direct software migration path from C66x
- Provides direct architecture support for accelerating OpenCL

There is a single C71SS module in the device. It is tightly coupled with a matrix multiply accelerator (MMA).

[Table 6-83](#) shows C71SS allocation within device domains.

**Table 6-83. C71SS Allocation within Device Domains**

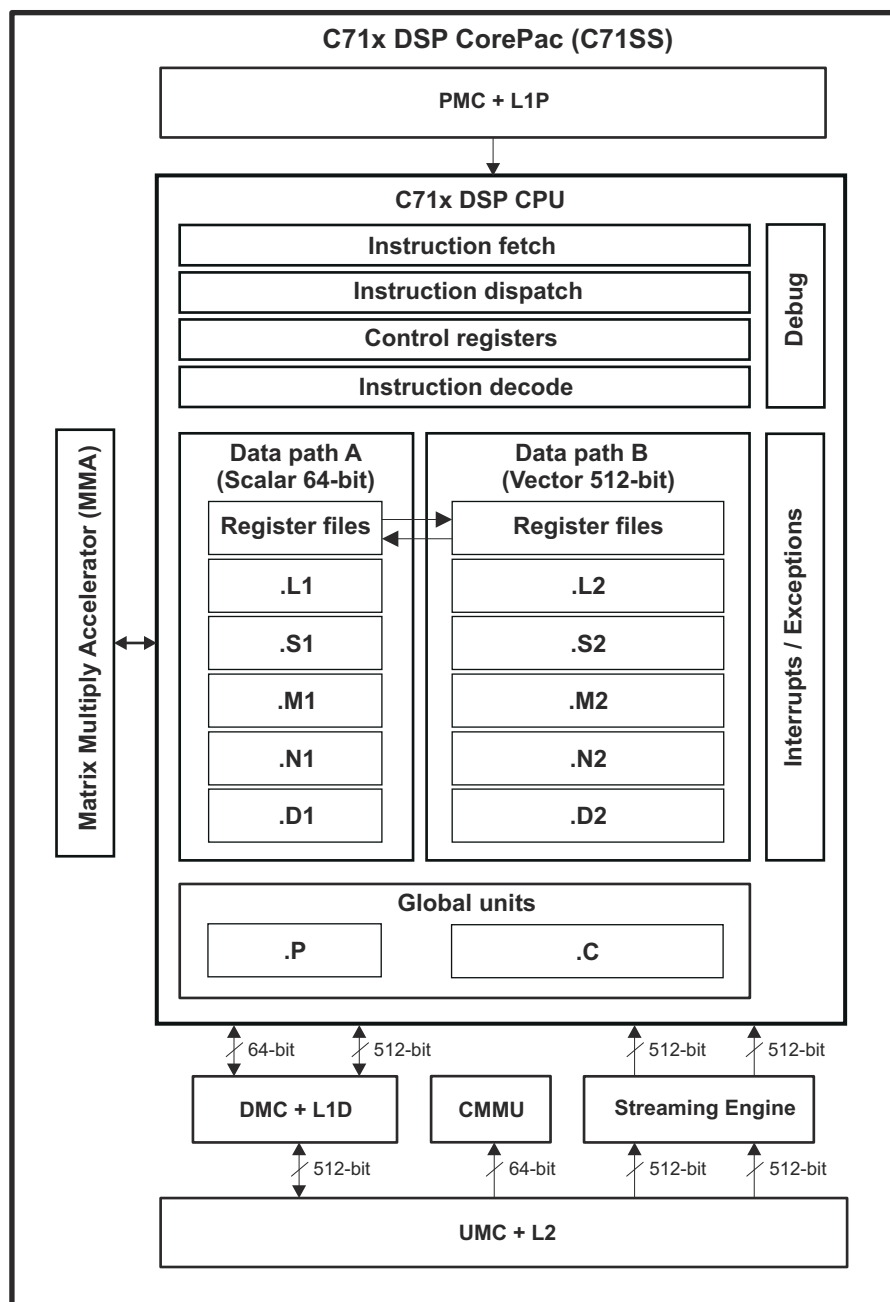
Module Instance	Domain		
	WKUP	MCU	MAIN
C71SS0 (+ MMA)	–	–	✓

#### Note

The C71SS is also referred to as C71x CorePac.

[Figure 6-34](#) shows an overview of the C71SS.





**Figure 6-34. C71SS Overview**

#### 6.5.1.1 C71SS Features

The C71SS (C71x CorePac) supports the following key features:

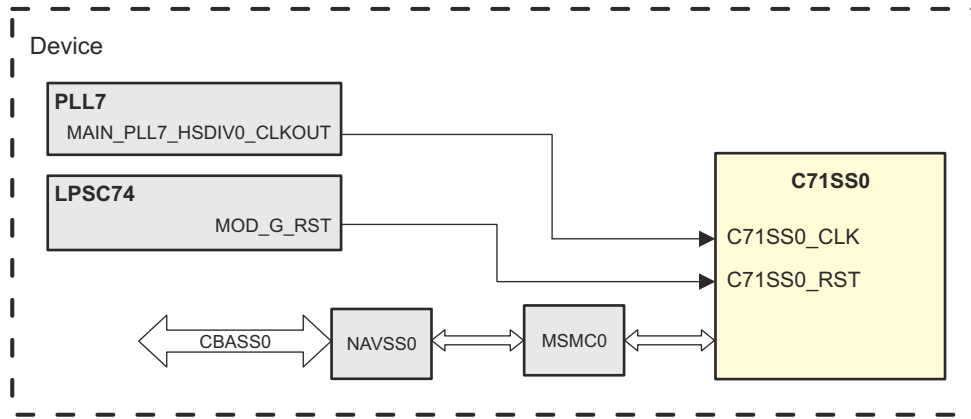
- C71x CPU
- Matrix multiplication accelerator (MMA)
- Hierarchical cache memory system
  - Level 1 (L1):
    - L1 program memory controller (PMC) with associated L1 program memory (L1P)
    - L1 data memory controller (DMC) with associated L1 data memory (L1D), either L1 data cache or L1 scratch memory
  - Level 2 (L2):

- L2 unified memory controller (UMC) with associated L2 memory, either L2 cache or L2 scratch memory
- Streaming engine (SE)
- CorePac memory management unit (CMMU), fully compliant with Armv8-A memory architecture
- Power/clock/reset management unit
- ECC/parity support
  - ECC protection of L2 memory and L1D memory
  - Parity protection of L1P memory
- PBIST and LBIST support
- Cluster Level Event Controller (CLEC) CPU interface
- C71x debug subsystem
- Extended configuration register (ECR) interface for register access (instead of MMR)

### 6.5.2 C71SS Integration

This section describes the C71SS integration in the device, including information about clocks, resets, and hardware requests.

Figure 6-35 shows the C71SS0 integration.



**Figure 6-35. C71SS0 Integration**

Table 6-84 through Table 6-85 summarize the C71SS integration.

**Table 6-84. C71SS Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
C71SS0	PSC0	PD12	LPSC74	CBASS0 <sup>(1)</sup>

(1) Accessed through MSMC0.

**Table 6-85. C71SS Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
C71SS0	C71SS0_CLK	MAIN_PLL7_HSDIV0_CLKOUT	PLL7	C71SS0 main functional clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
C71SS0	C71SS0_RST	MOD_G_RST	LPSC74	C71SS0 hardware reset

### 6.5.3 C71SS Functional Description

#### Note

This chapter provides only a brief description of C71SS supported features. Full description of C71x DSP functionality is provided in standalone TI documentation.

#### 6.5.3.1 C71x DSP CPU

The C71x CPU is a true 64-bit core which provides the following key features:

- Instruction fetch unit
- Instruction dispatch unit, advanced instruction packing
- Instruction decode unit
- CPU dual datapath
  - One 64-bit scalar side (side A) and one 512-bit vector side (side B)
  - Side A includes the following functional units
    - Four main scalar processing units (.L1, .S1, .M1, .N1) capable of operating on up to 64-bit wide data
    - Two units (.D1, .D2) for address calculations, enabling parallel load/store operations
    - The following operations can be executed at the same time in a single clock cycle
      - One non-aligned 64-bit load or store operation
      - Two 64-bit arithmetic/logical operations (non-multiply arithmetic instructions)
      - One 128-bit multiply operation
  - Side B includes the following functional units
    - Five main vector processing units (.L2, .S2, .M2, .N2, .C) capable of operating on up to 512-bit wide vector data
    - A predication processing unit (.P) for vector predication
    - The following vector operations can be executed at the same time in a single clock cycle
      - One non-aligned 512-bit load or store operation
      - Two 512-bit arithmetic/logical operations (non-multiply arithmetic instructions)
      - One 1024-bit multiply operation
      - One 512-bit correlation operation or regular arithmetic operation
      - One vector predicate manipulation operation
  - Vector side B can perform up to 128×16-bit fixed-point multiply-accumulate (MAC), 4.0 times the MAC capacity compared to C66x
  - Vector side B can perform up to 80 single precision FLOPs/cycle or 32 double precision FLOPs/cycle, 5.0 times the floating point compared to C66x
  - The C71x CPU can load or store in parallel 64 bits and 512 bits of data per clock cycle, more than four times the bandwidth compared to C66x. In addition, a novel streaming data interface can read an additional 1024 bits of data per clock cycle, providing a total of 12 times more bandwidth compared to the C66x
  - Large set of machine registers:
    - Up to 16×512-bit global vector registers
    - Up to 16×64-bit global scalar registers
    - Local registers
  - Register file cross paths
    - Two cross paths, 64-bit each
    - Allow functional units from one data path to access a 64-bit operand from the opposite side global register file
    - The cross paths can not access the local register files directly
- CPU control logic
  - Support for two security states
    - Secure state: CPU can access both secure and non-secure space
    - Non-secure state: CPU can only access non-secure space, and cannot access secure space

- Support for six privilege and execution levels for security and virtualization support with banked control registers for full isolation and protection
  - Secure root supervisor
  - Secure root user
  - Non-secure root supervisor
  - Non-secure root user
  - Non-secure guest supervisor
  - Non-secure guest user
- Pipeline can operate in both unprotected and protected modes
  - Unprotected mode: Traditional C6x VLIW DSP operating mode with exposed instruction delay slots
  - Protected mode: Control code execution mode where instructions delay slots are not exposed
- Scoreboarding mechanism to help hide memory system stalls and to allow the compiler to generate more efficient control codes
- Supports recoverable interrupts and internal precise exception
  - No need to disable interrupts during multiple assignment code. In-flight results in multiple-assignment code are recorded by pipeline capture queues. These queues allow all programs to be interrupted at any arbitrary cycle, even during software pipelined loops, and then to be restarted correctly upon return from the interrupt or exception handler
  - Pipeline capture queues may be unloaded and reloaded, allowing the OS to swap out tasks regardless of whether the task was operating in protected or unprotected pipeline mode
  - Programmable event priority, hardware automatic event mask based on priority levels
  - Hardware performs automatic stacking at even handler entry and automatic unstacking at event exiting
- Nested loop counters (NLC): Hardware mechanism to facilitate low-overhead loop collapsing by providing the computations associated with nested loop counters and predicates, avoid the needs of using explicit instructions
- Test, debug, and interrupt logic
- Enhanced instruction set architecture (ISA)
  - Single-cycle 64-bit arithmetic, logical, and shift instructions
  - Improvements to load/store instructions
    - Endian aware vector load instructions
    - Unpacking load / packing store instructions
    - Vector load/store with element reversal
    - Vector load with data duplications
  - Specialized instructions to speed up key benchmarks
    - Horizontal MAX/MIN search acceleration instructions
    - Horizontal ADD, SUB instructions
    - Dedicated FIR instructions
    - Sliding window sum of absolute differences (SAD)
    - Maskable complex dot products
  - Circular comparison instructions to accelerate Viterbi
  - New and improved atomic instructions to replace LL/SC/CL combination
  - Increases orthogonality between 32-bit, 64-bit, and vector operations compared to C66x
  - Supports binlog instruction for vision algorithms
  - Specialized instructions only allowed to execute at required privilege level and secure state
- C71x OpenCL features
  - Full compliance with IEEE 754 floating point standard
    - Support subnormal numbers
  - DVM support
    - Support fencing for uTLB transactions

The matrix multiply accelerator (MMA) is included as a special functional unit in the C71x CorePac and is tightly coupled to CPU operation.

### 6.5.3.2 C71x DSP Matrix Multiply Accelerator

The matrix multiply accelerator (MMA) provides the following key features:

- Support for fully connected layer using matrix multiply with arbitrary dimension
- Support for convolution layer using 2D convolution with matrix multiply with read panel
- Support for ReLU non-linearity layer OTF
- Support for high utilization (>85%) for typical convolutional neural network (CNN), such as AlexNet, ResNet, and others
- Ability to support any CNN network topologies limited only by memory size and bandwidth
- Coupled with C71x CPU using DSP chassis for data formatting
- MMA and C71x CPU are on a shared power domain
  - MMA cannot be independently powered off or clock gated at LPSC level (although it has extensive clock gating within the IP)

### 6.5.3.3 C71x DSP Cache Memory System

The C71x CorePac implements a hierarchical cache memory system with the following levels:

- Level 1 (L1):
  - L1 program memory controller (PMC) with associated L1 program memory (L1P)
  - L1 data memory controller (DMC) with associated L1 data memory (L1D)
- Level 2 (L2):
  - L2 unified memory controller (UMC) with associated L2 memory

#### 6.5.3.3.1 C71x DSP L1 Program Memory

The L1P cache controller supports a fixed cache size of 32KB. The purpose of the L1P cache is to maximize performance of the code execution. L1P cache is necessary to facilitate fetching program code at a fast clock rate in order to maintain a large system memory. The cache is responsible for hiding the latency associated with executing code from the slower system memory.

The L1P memory system provides the following key features:

- L1 program memory controller (PMC) with 32KB L1P memory, all cache (no support for L1P SRAM)
- L1P cache
  - 4-way set associative
  - 64-byte line size
  - Virtually indexed, virtually tagged (48-bit virtual address)
  - Auto-prefetching on L1P misses from L2
- Prefetch and branch prediction
- ECC SECDED support
- Software initiated coherence operations
  - Single cycle invalidate with support for three modes: all cache lines, only user cache lines, only supervisor cache lines
- Virtual memory
  - Virtual-to-physical addressing on misses
  - Micro-TLB (uTLB) to handle address translation and for code protection
- Extended control register (ECR) access
  - L1P ECR registers are not memory mapped and instead are mapped to a MOVC CPU instruction

#### 6.5.3.3.2 C71x DSP L1 Data Memory

The L1D includes a non-blocking cache controller, which has a main cache and a small fully associative victim cache. Main cache is read-allocate and supports write-back and write-through. Victim cache services read and write hit directly with no performance overhead as main cache.

The purpose of the L1D memory/cache is to maximize performance of the data processing. The cache is necessary to facilitate reading and writing data at the full CPU clock rate, while still having a large system

memory. It is the cache's responsibility to hide much of the latency associated with reading from and writing to the slower system memory. The L1D memory includes two logical sections:

- L1D partition 0: Can support 32KB (max) / 8KB (min) of cache. Cache may only reside in partition 0. A cache size of 0KB is not supported, which means that disabling L1D cache functionality is not possible
- L1D partition 1: Can support 40KB (max) / 16KB (min) of SRAM (may include LUT and histogram)

---

#### Note

The minimum cache size (8KB) can only be achieved via L1DMODE register programming.

---

The L1D memory system provides the following key features:

- L1 data memory controller (DMC) with 48KB L1D memory, configurable as cache and/or SRAM
- L1D partition 0 (cache/SRAM) + L1D partition 1 (SRAM only)
- L1D cache
  - 32KB cache (max), configurable down to 8KB cache (min)
  - Dual datapath (DP0 + DP1) support
  - Direct-mapped (1-way set-associative) main cache
  - 128-byte cache line size
  - Read-allocate cache
  - Support for both write-back and write-through modes
  - Support for victim cache
    - 16 cache lines, fully associative
  - Physically indexed, physically tagged (40-bit physical address)
  - Support for speculative loads
  - Hit under miss
  - Posted write miss support
  - Write merging on all outstanding write transactions inside L1D
- L1D SRAM
  - 16KB (min), configurable up to 40KB (max)
  - Accessible from CPU or DMA
- Lookup table (LUT) and histogram
  - LUT
    - Up to 4 sets of look up tables can be specified simultaneously
    - Supports interpolation mode
    - Indices are unsigned values at 32-bit lanes of the source register
    - Look up table elements can be signed or unsigned, bytes, half-words or words
  - Histogram
    - Up to 4 sets of histograms can be specified simultaneously
    - Supports regular and weighted histogram operations
    - Indices are unsigned values at 32-bit lanes of the source register
    - Histogram weights are at 32-bit lanes of the source register
    - Histogram weights can only be signed bytes or signed half-words
    - Histogram bin data saturates to the minimum or maximum values of its data bin type
- Bandwidth
  - 1024-bit data throughput
  - 16×64-bit banks
- ECC SECDED support
- Coherence
  - Full MESI support
  - Support for global cache coherence operations
  - Snoops and cache maintenance operation support from L2
  - Snoops for L2 SRAM, MSMC SRAM and external (DDR) addresses

- Virtual memory support
  - Support for wider (40-bit) physical address
- ECR access
  - L1D ECR registers are not memory mapped and instead are mapped to a MOVC CPU instruction

### 6.5.3.3.3 C71x DSP L2 Memory

The L2 memory system provides the following key features:

- L2 unified memory controller (UMC) with 512KB L2 memory, configurable as cache and/or SRAM
  - L2 cache
    - 8-way set-associative
    - 128-byte cache line size
    - Write-allocate cache
    - Support for both write-back and write-through modes
    - Physically indexed, physically tagged (40-bit physical address)
    - Supports 2×64-byte streams from one streaming engine
    - External MMR and MDMA accesses on an unified interface to MSMC
    - Caches MMU page tables
    - Cache pre-warming from SE and/or MSMC
  - L2 SRAM
    - Security firewall on L2 SRAM accesses
    - DMA access to L2 SRAM on merged MSMC I/F
  - Bandwidth
    - 2048-bit data throughput (see [Section 6.5.3.4, C71x DSP Streaming Engine](#))
    - 4×512-bit banks, with 2 virtual banks each
  - ECC SECDED support
  - Coherence
    - Full MESI support
    - Support for global coherence operations
    - Snoops for L2 SRAM, MSMC SRAM and external (DDR) addresses
    - User coherence commands from SE
    - Full coherence between L1D cache, SE, L2 SRAM, MSMC SRAM and DDR
  - ECR access
    - L2 ECR registers are not memory mapped and instead are mapped to a MOVC CPU instruction

### 6.5.3.4 C71x DSP Streaming Engine

The streaming engine (SE) supports the following key features:

- Provides a flexible, high bandwidth mechanism for reading large quantities of data into C71x CPU
- Supplies two 512-bit data streams (S0, S1) directly from L2 to the vector units
- Provides two stream address generators, supporting the two data streams
  - Flexible multi-dimensional address calculators
  - Provide offset addresses for load and store instructions
- Supports element promotion, decimation, duplication, transpose loads, predication
- LEZR feature can be enabled to return N number of zero vectors to CPU after selected dimension ends. LEZR and transposed mode are not supported together. LEZR is still supported when transposed mode is disabled..
- Provides 6D addressing
  - Access patterns up to 6D can be programmed ahead
  - 6D data is presented as 512-bit vector per cycle
- Communicates with L2 memory controller for requests beyond L2 (MSMC, DDR)
- Coherent with L1D data at stream open/close boundaries
- ECC SECDED support



The SE can sustain 1024-bits/cycle total bandwidth (512 bits/cycle on each of two streams) to the DSP CPU. When combined with its 512-bit/cycle vector load unit and 512-bit/cycle store unit, the DSP CPU can access 2048 bits/cycle of data.

#### 6.5.3.5 C71x DSP CorePac Memory Management Unit

The CorePac Memory Management Unit (CMMU) extends the C71x architecture with support for address translation, access permission and protections and memory attributes determination and checking. It is implemented per C71x cluster as a two-level TLB structure. The CMMU works with the C71x L1 caches, stream buffers in each processor and CorePac memory system of CorePac cluster to translate virtual addresses to physical addresses, controls tablewalk hardware that accesses translation tables in main memory. The CMMU enables fine-grained memory system control through a set of virtual-to-physical address mappings and memory attributes held in the L1 level micro translation look-aside buffer (uTLB) and CorePac cluster level translation look-aside buffers (TLBs).

The CMMU provides the following key features:

- Supports AArch32 LPAE and AArch64 page table format
  - Programmable levels of page table translation
  - 4KB, 16KB, 64KB translation granules
- Supports multiple page sizes: 4KB, 16KB, 64KB, 2MB, 32MB, 512MB, 1GB, 16GB
- Supports both hardware and software managed table walks
- Hierarchical Table Lookaside Buffer (TLB) caching page entries, intermediate page walk pointers for optimum address translation performance
  - Micro-TLB (uTLB) as first level TLB
    - 16-entry fully associative dual interface L1 data uTLB
    - 8-entry fully associative L1 instruction uTLB
  - CorePac (L2) TLB as second level TLB
    - 512-entry 4-way associative TLB cache, caching both instruction and data page translations with virtual-to-physical address mapping
    - Intermediate table walk caches to reduce memory access during multi-level page walk
    - Allows caching and sharing of intermediate table walks and page tables in the system cache hierarchy
- Security extensions that facilitate the development of secure applications
- Soft-error protection on all RAM and data storage structures

#### 6.5.3.6 C71x DSP ECC Support

[Table 6-86](#) provides the RAM ID for the C71x DSP ECC aggregator.

**Table 6-86. RAM ID Map for C71x DSP ECC Aggregator**

RAM ID	Endpoint
0	C71x Corepac
1	CBASS P2P
2	CBASS SCR
3	CBASS SCR
4	Reserved
5	Reserved
6	UMC pipe 0
7	UMC pipe 0
8	UMC pipe 1
9	UMC pipe 1
10	UMC pipe 2
11	UMC pipe 2
12	UMC pipe 3

**Table 6-86. RAM ID Map for C71x DSP ECC Aggregator (continued)**

RAM ID	Endpoint
13	UMC pipe 3
14	DMC
15	DMC tag RAM
16	SE, stream 0
17	SE, stream 1
18	PMC

#### 6.5.3.7 C71x DSP Boot Configuration

In this device, the C71x boots from system memory instead of booting from ROM. A 1KB boot RAM (PSRAM1KECC) is used to store boot vectors for the C71x CPU.

#### 6.5.3.8 C71x DSP Power-Up/Down Sequences

The power-up and power-down sequences for the C71SS can be found in *Power*.

#### 6.5.3.9 C71x DSP Interrupt Control

The cluster level event controller (CLEC) serves the role of an interrupt controller for the C71x DSP. For more details, see *Interrupts*.

## 6.6 Graphics Accelerator (GPU)

This chapter describes the integration of the Graphics Processing Unit (GPU) subsystem in the device.

### 6.6.1 GPU Overview

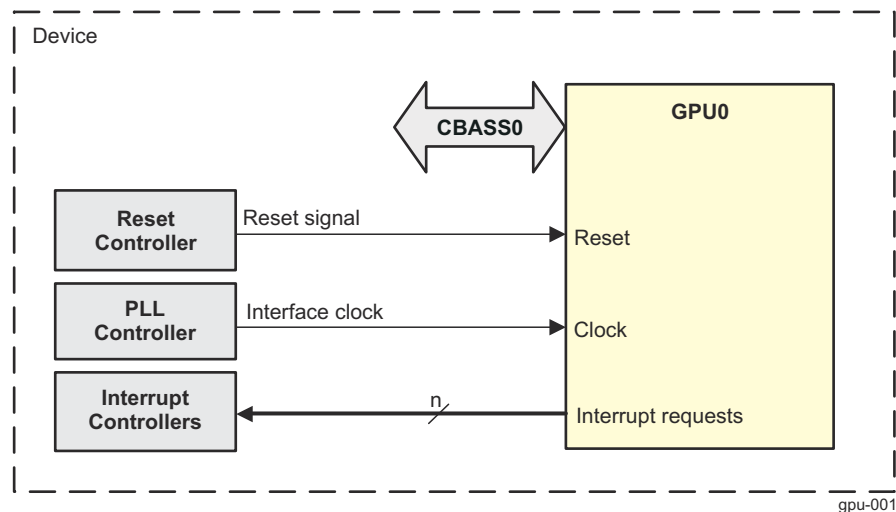
The Graphics Processing Unit (GPU) accelerates 3-dimensional (3D) and 2-dimensional (2D) graphics and compute applications.

The GPU is based on the PowerVR® Series8XE GE8430 core from Imagination Technologies.

The GPU module is a scalable architecture which efficiently processes a number of different workload concurrently:

- 3D Graphic Workload, which involves vertex data and pixel data processing for rendering of 3D scenes.
- 2D Graphic Workload, which involves pixel data processing for rendering 2D objects.
- Compute Applications Workload, which involves general purpose data processing.

Figure 6-36 shows the GPU module overview.



**Figure 6-36. GPU Module Overview**

#### 6.6.1.1 GPU Features Overview

- APIs support:
  - OpenGL® - ES 3.2
  - OpenCL™ - 1.2 EP
  - Vulkan 1.2
- Single-core GPU architecture is consisting of:
  - 1 × PowerVR® Series8XE GE8430 core
- Unified Shading Clusters (USCs):
  - Multithreaded engine incorporating vertex shader, pixel shader and compute shader functionality
  - Incorporates an ALU architecture with high SIMD efficiency
- Tile-based deferred rendering architecture with concurrent processing of multiple tiles
- 96GFLOPS and 6GPix/s at 750MHz clock
- Hardware virtualization
- Hardware security
- FBC and FBDC (Frame Buffer Compression/Decompression)
- Dedicated MIPS processor for firmware execution
- Advanced DMA driven operation for minimum host CPU interaction
- Programmable high quality image anti-aliasing
- Fully virtualized memory addressing in a unified memory architecture:

- Up to 64 GB physical address space
- Up to 1 TB virtual address space

**6.6.1.2 GPU Not Supported Features**

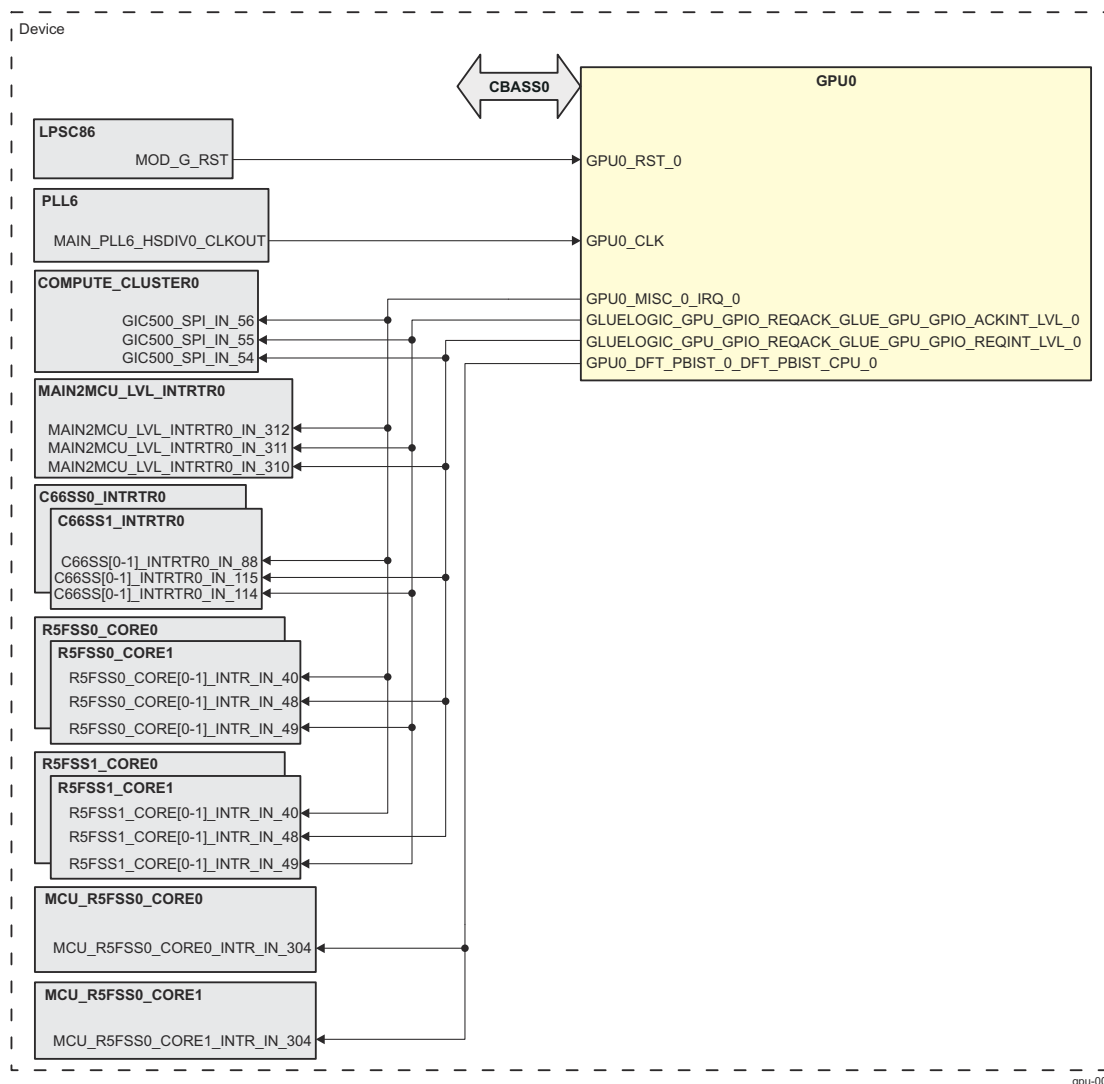
- Memory coherency
- ECC support on memories
- ECC support on bus interfaces

## 6.6.2 GPU Integration

This section describes GPU0 integration in the device, including information about clocks, resets, and hardware requests.

### 6.6.2.1 GPU Integration in MAIN Domain

There is one GPU0 module integrated in the device MAIN domain. [Figure 6-37](#) shows the integration of GPU0.



**Figure 6-37. GPU0 Integration**

[Table 6-87](#) through [Table 6-89](#) summarize the integration of GPU0 in the device MAIN domain.

**Table 6-87. GPU0 Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
GPU0	MAIN_PSC	PD20 (GPU_COMMON)	LPSC86	CBASS0
		PD21 (GPU_CORE)	LPSC88	

**Table 6-88. GPU0 Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description

**Table 6-88. GPU0 Clocks and Resets (continued)**

GPU0	GPU0_CLK	MAIN_PLL6_HSDIV0_CLKO UT	PLL6	GPU0 configuration clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
GPU0	GPU0_RST_0	MOD_G_RST	LPSC86	GPU0 reset

**Table 6-89. GPU0 Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
GPU0	GPU0_MISC_0_IRQ_0	GIC500_SPI_IN_56	COMPUTE_CLUSTER0	GPU0 interrupt request	Level
		C66SS0_INTRTR0_IN_88	C66SS0_INTRTR0		Level
		C66SS1_INTRTR0_IN_88	C66SS1_INTRTR0		Level
		MAIN2MCU_LVL_INTRTR0_IN_312	MAIN2MCU_LVL_INTRTR0		Level
		R5FSS1_CORE0_INTR_IN_40	R5FSS1_CORE0		Level
		R5FSS1_CORE1_INTR_IN_40	R5FSS1_CORE1		Level
		R5FSS0_CORE0_INTR_IN_40	R5FSS0_CORE0		Level
		R5FSS0_CORE1_INTR_IN_40	R5FSS0_CORE1		Level
	GLUELOGIC_GPU_GPIO_REQACK_GLUE_GPU_GPIO_ACKINT_LVL_0	GIC500_SPI_IN_55	COMPUTE_CLUSTER0	GPU0 GPIO acknowledged interrupt request	Level
		C66SS0_INTRTR0_IN_115	C66SS0_INTRTR0		Level
		C66SS1_INTRTR0_IN_115	C66SS1_INTRTR0		Level
		MAIN2MCU_LVL_INTRTR0_IN_311	MAIN2MCU_LVL_INTRTR0		Level
		R5FSS1_CORE0_INTR_IN_49	R5FSS1_CORE0		Level
		R5FSS1_CORE1_INTR_IN_49	R5FSS1_CORE1		Level
		R5FSS0_CORE0_INTR_IN_49	R5FSS0_CORE0		Level
		R5FSS0_CORE1_INTR_IN_49	R5FSS0_CORE1		Level
	GLUELOGIC_GPU_GPIO_REQACK_GLUE_GPU_GPIO_REQINT_LVL_0	GIC500_SPI_IN_54	COMPUTE_CLUSTER0	GPU0 GPIO interrupt request	Level
		C66SS0_INTRTR0_IN_114	C66SS0_INTRTR0		Level
		C66SS1_INTRTR0_IN_114	C66SS1_INTRTR0		Level
		MAIN2MCU_LVL_INTRTR0_IN_310	MAIN2MCU_LVL_INTRTR0		Level
		R5FSS1_CORE0_INTR_IN_48	R5FSS1_CORE0		Level
		R5FSS1_CORE1_INTR_IN_48	R5FSS1_CORE1		Level
		R5FSS0_CORE0_INTR_IN_48	R5FSS0_CORE0		Level
		R5FSS0_CORE1_INTR_IN_48	R5FSS0_CORE1		Level

**Table 6-89. GPU0 Hardware Requests (continued)**

GPU0_DFT_PBIIST_0_DFT _PBIIST_CPU_0	MCU_R5FSS0_CORE0_INT R_IN_304	MCU_R5FSS0_CORE0	GPU0 DFT PBIIST request	Level
	MCU_R5FSS0_CORE1_INT R_IN_304	MCU_R5FSS0_CORE1		Level

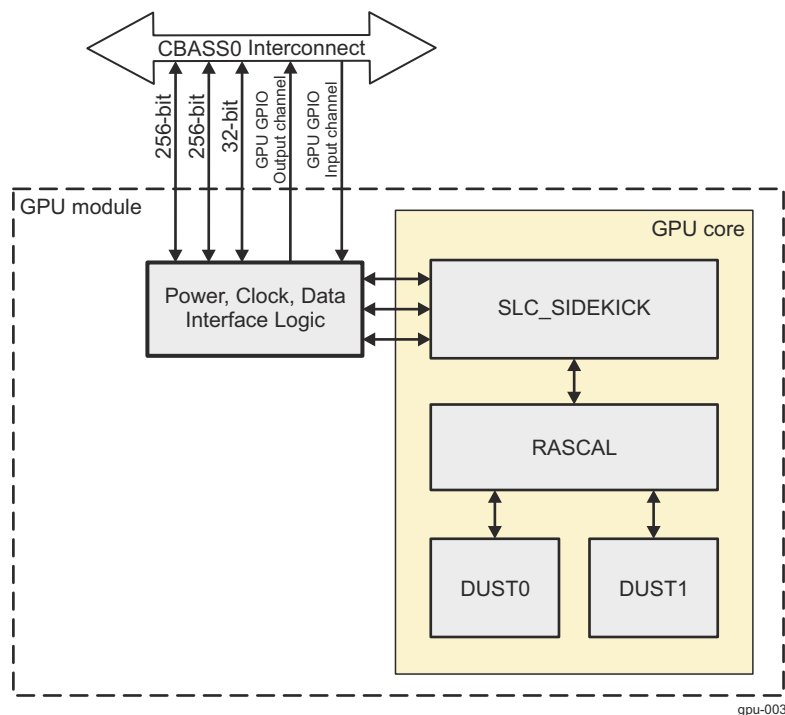
### 6.6.3 GPU Functional Description

#### 6.6.3.1 GPU Block Diagram

The GPU architecture comprises the following elements:

- A power, clock, and data interface logic
- A GPU core, partitioned into the following sub-blocks:
  - SLC\_SIDEKICK
  - RASCAL
  - DUST0 and DUST1

Figure 6-38 shows the GPU top-level block diagram.



**Figure 6-38. GPU Block Diagram**

The GPU module has 1 × GPU core. The wrapper around GPU core enables it integrates into device.

The GPU module is connected to the CBASS0 interconnect by:

- Two 256-bit data master interfaces for data transfer (read and write separate).
- One 32-bit data slave interface for GPU core configuration.

GPIO offers an additional communication channel between CPU and GPU. For more information on the GPIO registers, see [Section 5.1, Control Module \(CTRL\\_MMR\)](#).

The GPU module is based on tile-based deferred rendering (TBDR) and processes data in two phases:

1. The first phase is the *Vertex Processing Phase*, which involves vertex operations such as transformation and vertex lighting, as well as dividing a 3D scene into tiles.
2. The second phase is the *Pixel Processing Phase*, which involves pixel operations such as rasterization, texturing and shading of pixels.

Sub-block SLC\_SIDEKICK incorporates System Level Cache (SLC), MIPS processor, FBC, and FBDC. The SLC\_SIDEKICK handle the reset and clock control of the RASCAL, DUST0, and DUST1 blocks, including during the power transition state.

Sub-block RASCAL incorporates rasterization step.



Sub-blocks DUST0 and DUST1 incorporates Unified Shading Cluster (USC) and Texture Processing Unit (TPU).

#### 6.6.3.2 GPU Clock Configuration

The GPU subsystem operates from one clock from dedicated PLL for core operations (GPU0\_CLK).

When no longer needed by the GPU subsystem, GPU0\_CLK can be cut by software.

For additional information, see [Table 6-88](#), *GPU0 Clocks and Resets*.

#### 6.6.3.3 GPU Reset

The GPU subsystem has its own reset domain. Reset of the GPU is performed by activating the GPU0\_RST\_0 signal.

Reset domains refer to:

- GPU0\_RST\_0 resets RASCAL, DUST0, DUST1 blocks, SLC\_SIDEKICK sub-block and power, clock, and data interface logic from GPU core.

For additional information, see [Table 6-88](#), *GPU0 Clocks and Resets*.

#### 6.6.3.4 GPU Power Management

The GPU subsystem has its own power domains (GPU\_COMMON and GPU\_CORE).

Power domains refer to:

- SLC\_SIDEKICK and Wrapper blocks are part of GPU\_COMMON power domain.
- RASCAL, DUST0 and DUST1 blocks are part of GPU\_CORE power domain.

For additional information, see [Table 6-87](#), *GPU0 Integration Attributes*.

#### 6.6.3.5 GPU Interrupt Requests

The GPU subsystem can generate four interrupt signals -

GPU0\_MISC\_0\_IRQ\_0, GLUELOGIC\_GPU\_GPIO\_REQACK\_GLUE\_GPU\_GPIO\_ACKINT\_LVL\_0,  
GLUELOGIC\_GPU\_GPIO\_REQACK\_GLUE\_GPU\_GPIO\_REQINT\_LVL\_0,  
GPU0\_DFT\_PBI0\_DFT\_PBI0\_CPU\_0.

For additional information, see [Table 6-89](#), *GPU0 Hardware Requests*.

## 6.7 Multi-Standard HD Video Decoder (DECODER)

This section describes the dual-core multi-standard HD video decoder DECODER module in the device.

The DECODER module is a D5520MP2 dual-core PowerVR® VPU (video processor unit) from Imagination Technologies.

### 6.7.1 DECODER Overview

The DECODER is capable of supporting:

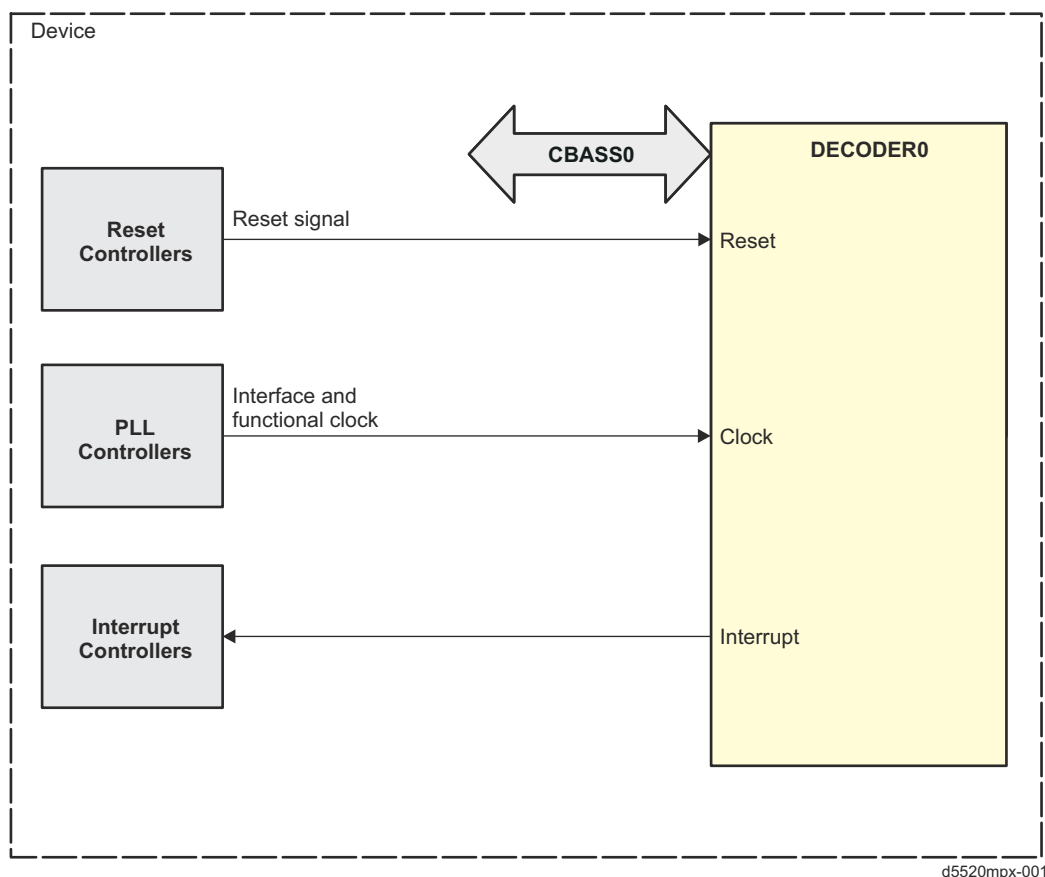
- 1x 4kp60 decode or
- 2x 4kp30 decodes or
- 4x 1080p60 decodes or
- 8x 1080p30 decodes

The device has one instance of the DECODER module. [Table 6-90](#) shows DECODER module allocation within device domains.

**Table 6-90. DECODER Modules Allocation within Device Domains**

Module Instance	Domain		
	WKUP	MCU	MAIN
DECODER0	-	-	✓

[Figure 6-39](#) shows the DECODER module overview.



**Figure 6-39. DECODER Module Overview**

#### 6.7.1.1 DECODER Features

The DECODER decoder supports the following decoding schemes:

- HEVC
- H.264
- MPEG-4
- WMV-9
- VC-1
- MPEG-2
- H.263
- DivX
- AVS
- RealVideo
- VP8
- VP6
- Sorenson
- JPEG

The DECODER also features:

- Hardware scaling
- 90°, 180°, 270° rotation support
  - Software driver runs on main host processor (no dedicated middle-ware processor is required as in previous decoder generation)
- Support secure playback (differentiated access to secure data space secure streams), enforced via device level firewalls controlled by DMSC.

#### 6.7.1.2 DECODER Not Supported Features

The DECODER does not support the following features:

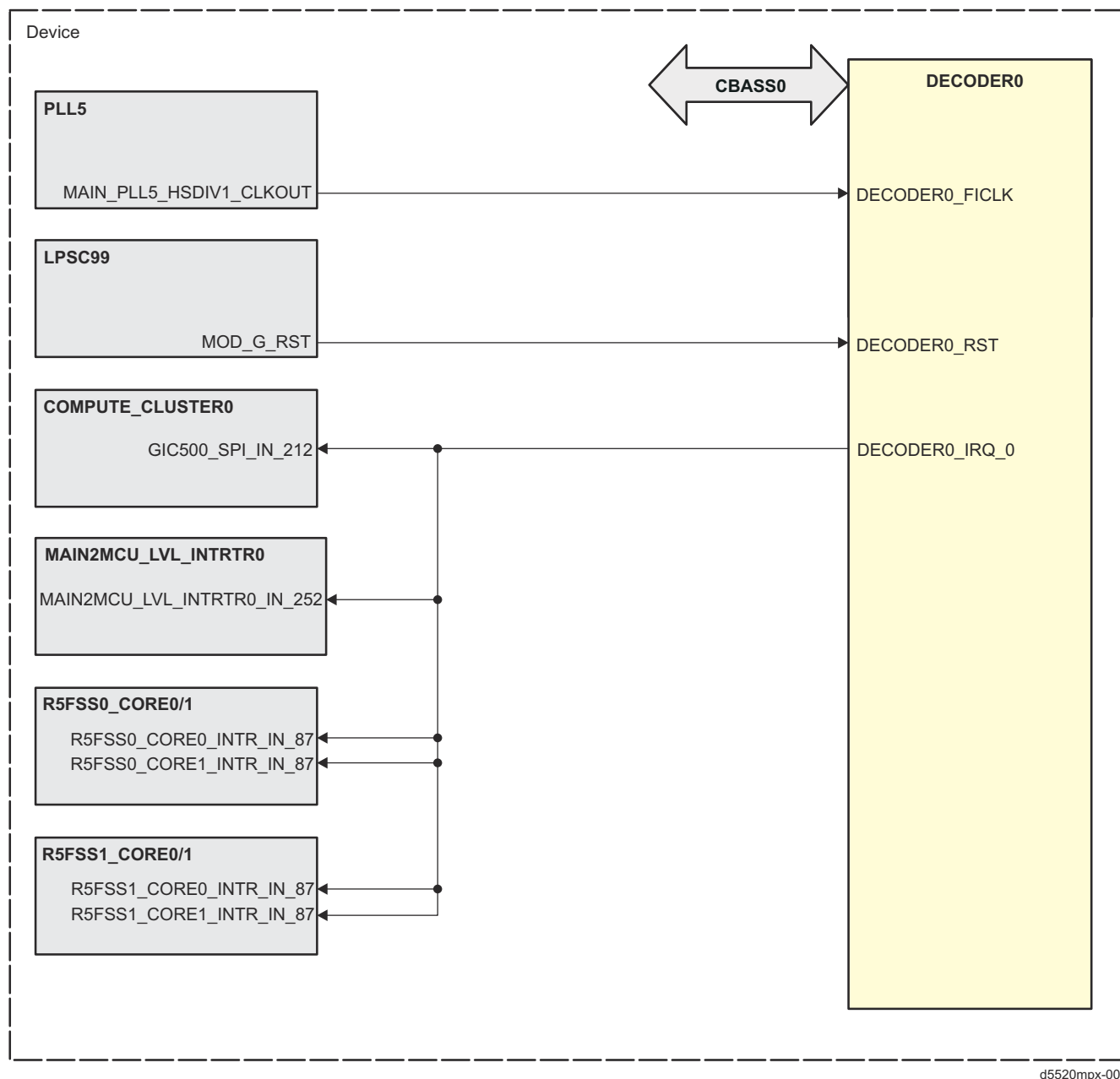
- ECC of memories
- Cache support

## 6.7.2 DECODER Integration

This section describes the DECODER integration in the device, including information about clocks, resets, and hardware requests.

### 6.7.2.1 DECODER Integration in MAIN Domain

There is one DECODER module integrated in the device MAIN domain. [Figure 6-40](#) shows the integration of DECODER module.



**Figure 6-40. DECODER Integration**

[Table 6-91](#) through [Table 6-93](#) summarize the integration of DECODER in the device MAIN domain.

**Table 6-91. DECODER Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
DECODER0	PSC0	PD26	LPSC99	CBASS0

**Table 6-92. DECODER Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
DECODER0	DECODER0_FICLK	MAIN_PLL5_HSDIV1_CLKOUT	PLL5	DECODER0 interface and functional clock

Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
DECODER0	DECODER0_RST	MOD_G_RST	LPSC99	DECODER0 asynchronous module reset

**Table 6-93. DECODER Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DECODER0	DECODER0_IRQ_0	GIC500_SPI_IN_212	COMPUTE_CLUSTER0	DECODER0 signal for various interrupt conditions. The source of the interrupt can be found by polling the interrupt status registers.	Level
		MAIN2MCU_LVL_INTRTR0_IN_252	MAIN2MCU_LVL_INTRTR0	DECODER0 signal for various interrupt conditions. The source of the interrupt can be found by polling the interrupt status registers.	Level
		R5FSS0_CORE0_INTR_IN_87	R5FSS0_CORE0	DECODER0 signal for various interrupt conditions. The source of the interrupt can be found by polling the interrupt status registers.	Level
		R5FSS0_CORE1_INTR_IN_87	R5FSS0_CORE1	DECODER0 signal for various interrupt conditions. The source of the interrupt can be found by polling the interrupt status registers.	Level
		R5FSS1_CORE0_INTR_IN_87	R5FSS1_CORE0	DECODER0 signal for various interrupt conditions. The source of the interrupt can be found by polling the interrupt status registers.	Level
		R5FSS1_CORE1_INTR_IN_87	R5FSS1_CORE1	DECODER0 signal for various interrupt conditions. The source of the interrupt can be found by polling the interrupt status registers.	Level

### Note

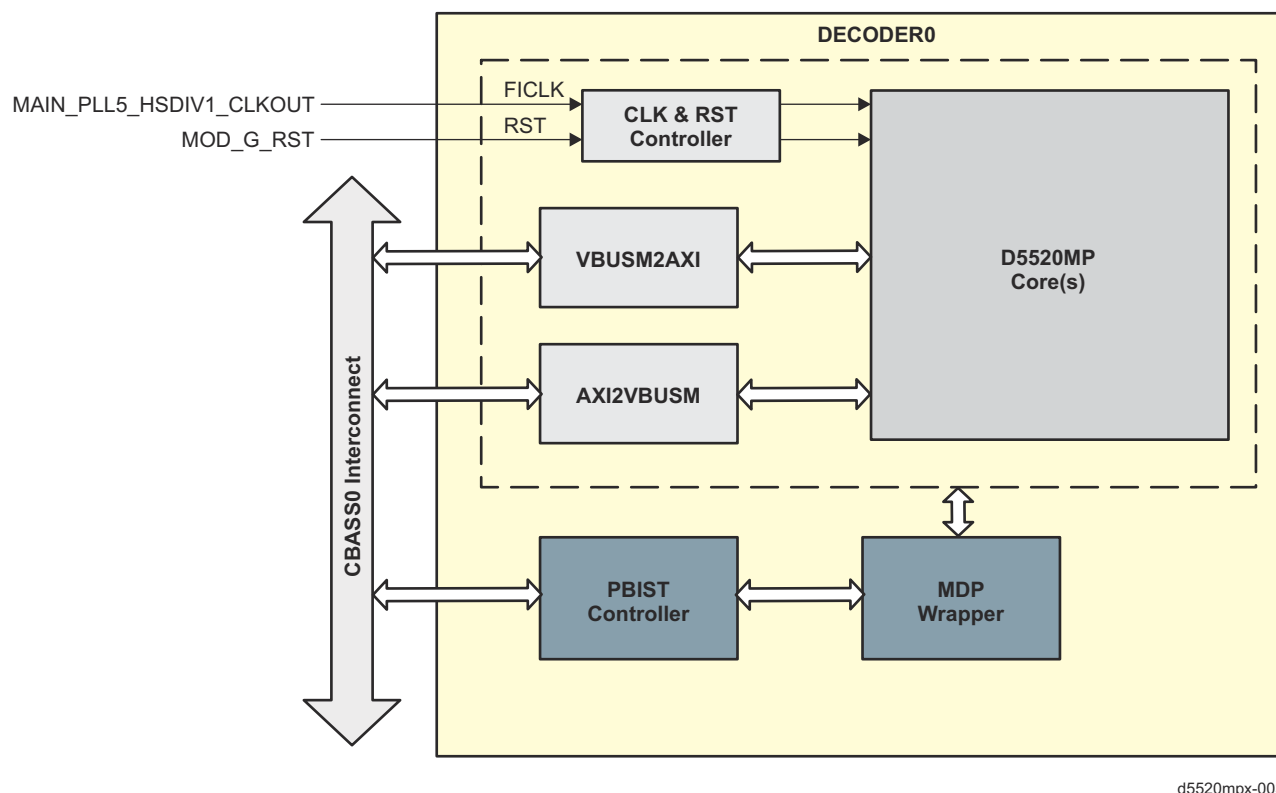
For more information on the interconnects, see [Chapter 3, System Interconnect](#).

For more information on the power, reset and clock management, see the corresponding sections within [Chapter 5, Device Configuration](#).

For more information on the device interrupt controllers, see [Section 9.2, Interrupt Controllers](#).

### 6.7.3 DECODER Functional Description

The DECODER is a multi-format video decoder which is based on D5520MP video decoder IP. The DECODER has AXI3 interfaces for both slave configuration and master data transfer. The DECODER has embedded processor which parses the bit-stream headers and does the hardware management to reduce the load requirements on the main host processor. Figure 6-41 shows the DECODER high level functional block architecture diagram.



d5520mpx-003

**Figure 6-41. DECODER Functional Block Diagram**

#### 6.7.3.1 DECODER Clock Configuration

The DECODER has single clock DECODER0\_FICLK. All the logic processing, data master interface and slave configuration interface will work on same clock.

#### 6.7.3.2 DECODER Reset

The DECODER has its own reset domain. Global reset of the DECODER is performed by activating the DECODER0\_RST signal in the LPSC99 domain.

#### 6.7.3.3 DECODER Interrupts

The DECODER has one interrupt signal DECODER0\_IRQ\_0. The decoder interrupt is generated by the D5520 core to signal various interrupt conditions.

## 6.8 Multi-Standard HD Video Encoder (ENCODER)

This section describes the dual-core multi-standard HD video encoder ENCODER module in the device.

The ENCODER module is a VXE384MP2 core PowerVR® VPU (video processor unit) from Imagination Technologies.

### 6.8.1 ENCODER Overview

The ENCODER is capable of supporting:

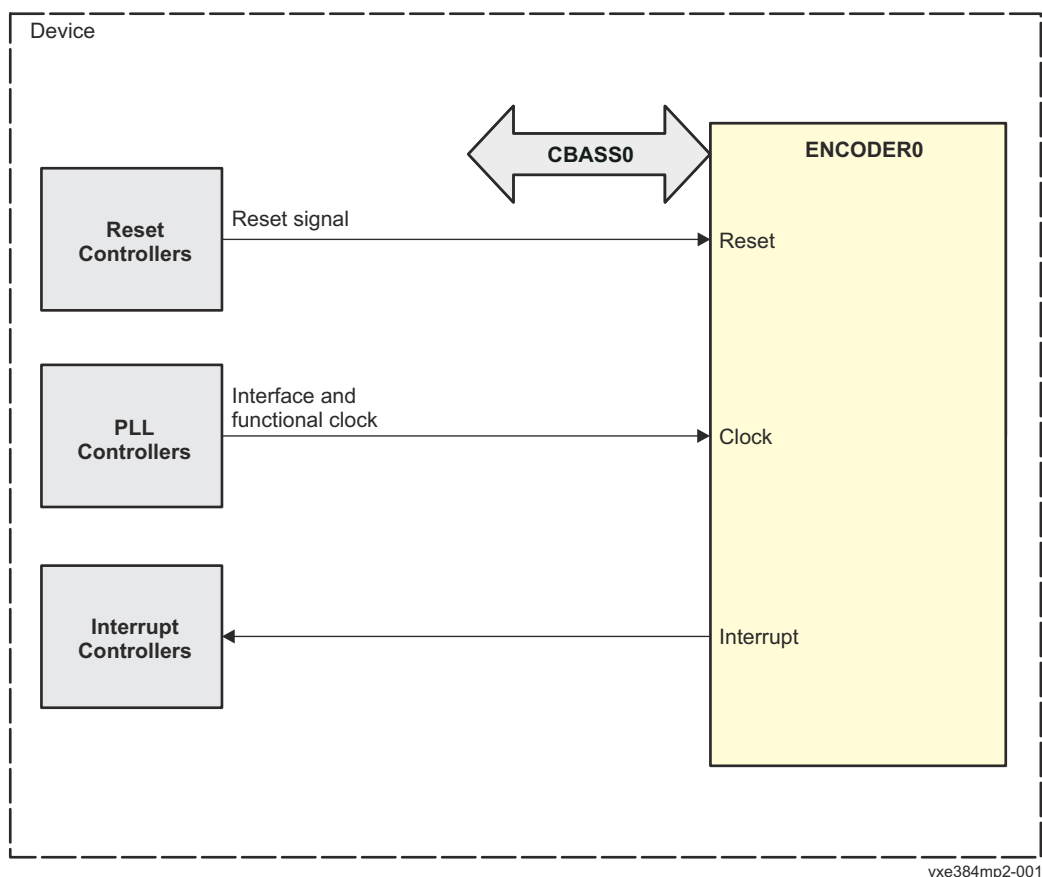
- 1x 1080p60 video stream encoding or
- 2x or 3x 1080p30 video stream encodings

The device has one instance of the ENCODER module. [Table 6-94](#) shows ENCODER module allocation within device domains.

**Table 6-94. ENCODER Modules Allocation within Device Domains**

Module Instance	Domain		
	WKUP	MCU	MAIN
ENCODER0	-	-	✓

[Figure 6-42](#) shows the ENCODER module overview.



**Figure 6-42. ENCODER Module Overview**

#### 6.8.1.1 ENCODER Features

The ENCODER encoder supports the following encoding schemes:

- H.264 MVC

- H.264 HP
- H.264 MP
- H.264 BP
- MPEG-4 SP
- H.263 BP

It also encodes still images in these formats:

- MPEG2 MP
- JPEG

The ENCODER also features:

- Support of multiple slices per frame (H.264) under software control
- Search range of +/-128 horizontally and +/-104 vertically
- Integrated scaler
- Dedicated hardware support for:
  - Integer motion estimation
  - Sup-pel motion estimation
  - Intra search
  - Transform and inverse transform
  - Quantization and inverse quantization
  - Motion vector prediction and estimation
  - Zigzag scan, run length and variable length encoding

#### **6.8.1.2 ENCODER Not Supported Features**

The ENCODER does not support the following features:

- ECC of memories
- Image rotation

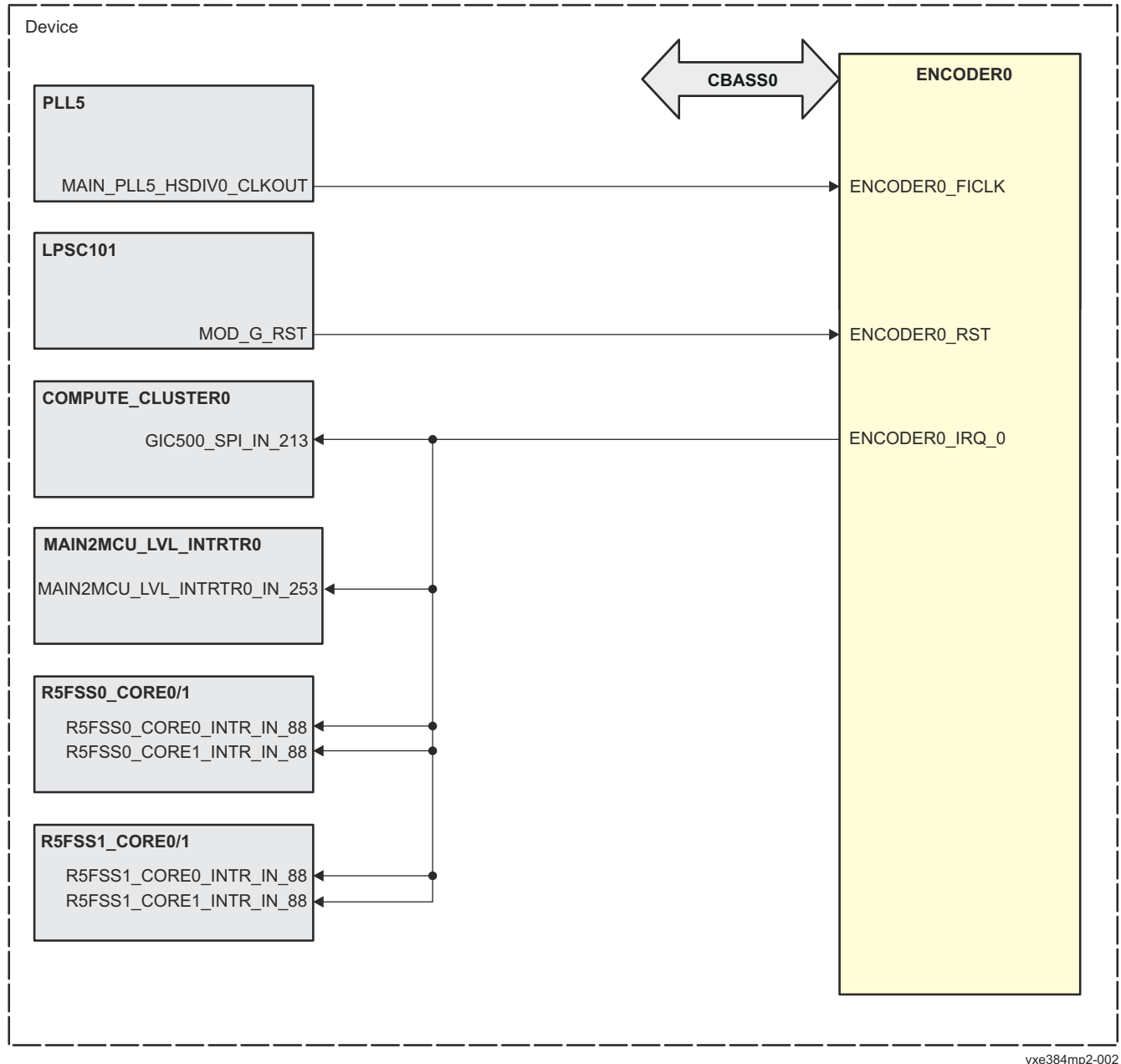


## 6.8.2 ENCODER Integration

This section describes the ENCODER integration in the device, including information about clocks, resets, and hardware requests.

### 6.8.2.1 ENCODER Integration in MAIN Domain

There is one ENCODER module integrated in the device MAIN domain. [Figure 6-43](#) shows the integration of ENCODER module.



**Figure 6-43. ENCODER Integration**

[Table 6-95](#) through [Table 6-97](#) summarize the integration of ENCODER in the device MAIN domain.

**Table 6-95. ENCODER Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
ENCODER0	PSC0	PD27	LPSC101	CBASS0

**Table 6-96. ENCODER Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
ENCODER0	ENCODER0_FICLK	MAIN_PLL5_HSDIV0_CLKOUT	PLL5	ENCODER0 interface and functional clock

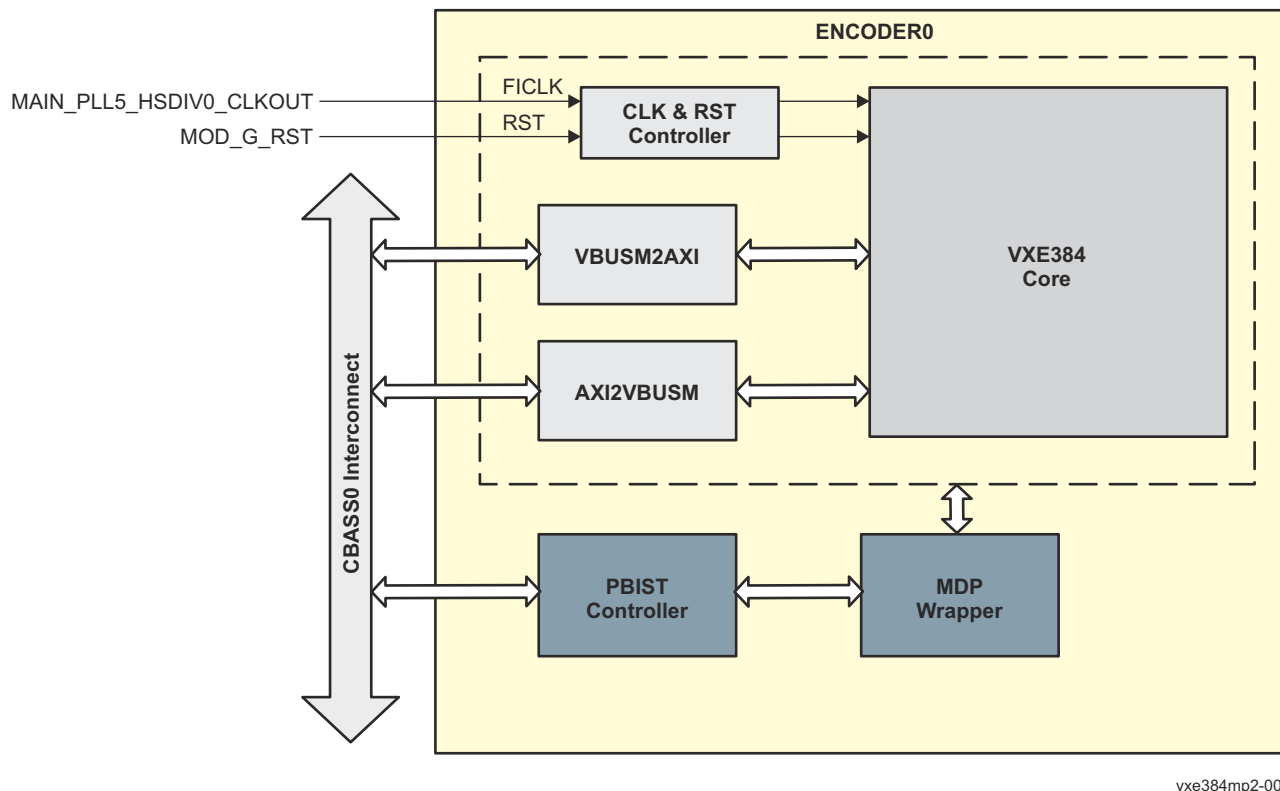
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
ENCODER0	ENCODER0_RST	MOD_G_RST	LPSC101	ENCODER0 asynchronous module reset

**Table 6-97. ENCODER Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
ENCODER0	ENCODER0_IR_Q_0	GIC500_SPI_IN_213	COMPUTE_CLUSTER0	ENCODER0 signal for various interrupt conditions. The source of the interrupt can be found by polling the interrupt status registers.	Level
		MAIN2MCU_LVL_INTRTR0_IN_253	MAIN2MCU_LVL_INTRTR0	ENCODER0 signal for various interrupt conditions. The source of the interrupt can be found by polling the interrupt status registers.	Level
		R5FSS0_CORE0_INTR_IN_88	R5FSS0_CORE0	ENCODER0 signal for various interrupt conditions. The source of the interrupt can be found by polling the interrupt status registers.	Level
		R5FSS0_CORE1_INTR_IN_88	R5FSS0_CORE1	ENCODER0 signal for various interrupt conditions. The source of the interrupt can be found by polling the interrupt status registers.	Level
		R5FSS1_CORE0_INTR_IN_88	R5FSS1_CORE0	ENCODER0 signal for various interrupt conditions. The source of the interrupt can be found by polling the interrupt status registers.	Level
		R5FSS1_CORE1_INTR_IN_88	R5FSS1_CORE1	ENCODER0 signal for various interrupt conditions. The source of the interrupt can be found by polling the interrupt status registers.	Level

### 6.8.3 ENCODER Functional Description

The ENCODER is a multi-format video encoder which is based on VXE384MP2 video encoder core IP. The ENCODER has AXI3 interfaces for both slave configuration and master data transfer. Figure 6-44 shows the ENCODER high level functional block architecture diagram.



**Figure 6-44. ENCODER Functional Block Diagram**

#### 6.8.3.1 ENCODER Clock Configuration

The ENCODER has single clock ENCODER0\_FICLK. All the logic processing, data master interface and slave configuration interface will work on same clock.

#### 6.8.3.2 ENCODER Reset

The ENCODER has its own reset domain. Global reset of the ENCODER is performed by activating the ENCODER0\_RST signal.

#### 6.8.3.3 ENCODER Interrupts

The ENCODER has one interrupt signal ENCODER0\_IRQ\_0. The encoder interrupt is generated by the VXE384 core to signal various interrupt conditions.

## 6.9 Vision Pre-processing Accelerator (VPAC)

This chapter describes the Vision Pre-processing Accelerator (VPAC) in the device.

### 6.9.1 VPAC Overview

The Vision Pre-processing Accelerator (VPAC) subsystem is a set of common vision primitive functions, performing pixel data processing tasks, such as: color processing and enhancement, noise filtering, wide dynamic range (WDR) processing, lens distortion correction, pixel remap for de-warping, on-the-fly scale generation, on-the-fly pyramid generation. The VPAC offloads these common tasks from the main SoC processors (ARM, DSP, etc.), so these CPUs can be utilized for differentiated high-level algorithms. The VPAC is designed to support multiple cameras by working in time-multiplexing mode. The VPAC also includes an imaging pipe, which can be integrated on-the-fly with external camera sensor, as well as does memory-to-memory (M2M) processing on pixel data.

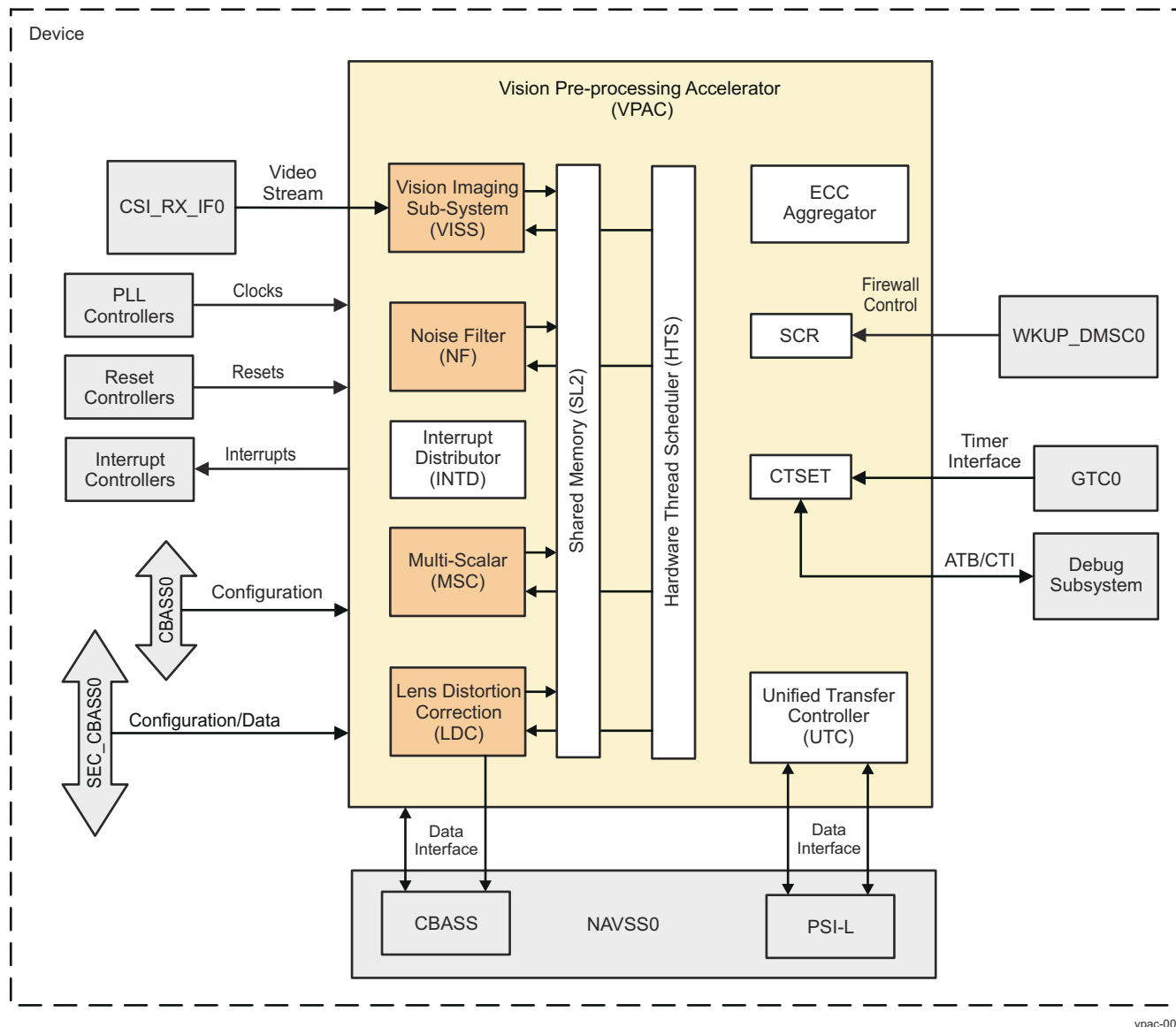
The VPAC subsystem provides 4 processing blocks: Vision Imaging Sub-System (VISS), Lens Distortion Correction (LDC), Multi-Scalar (MSC) and Noise Filter (NF), along with Hardware Thread Scheduler (HTS), Load Store Engin (LSE), and 512 KB of internal L2 memory

The device includes a single instantiation of the VPAC subsystem. [Table 6-98](#) shows the VPAC allocation across device domains.

**Table 6-98. VPAC Allocation Across Device Domains**

Instance	Domain		
	WKUP	MCU	MAIN
VPAC0	-	-	✓

[Figure 6-45](#) provides an overview of the VPAC subsystem.



vpac-001

**Figure 6-45. VPAC Overview**

### 6.9.1.1 VPAC Features

The VPAC includes the following processing and infrastructure sub-modules:

- **Vision Imaging Sub-System (VISS).** The vision imaging pipe does image processing on raw data which includes Wide Dynamic Range (WDR) merge, Defect Pixel Correction (DPC), Lens Shading Correction (LSC), Global/Local Brightness and Contrast Enhancement (GLBCE), Spatial Noise Filtering, demosaicing, color conversion, and Edge Enhancement (EE). It can operate on sensor data either on-the-fly or in memory-to-memory mode. The VISS provides the following main features:
  - Performance is up to 1 pixel/cycle
  - Up to 16-bit input RAW formats (8b, 12b, 14b, 16b) (see [Section 6.9.3.6](#))
  - Processing up to 4096 pixels/line (8MP frame)
  - Support for concurrent 12-bit and 8-bit YUV output
  - Support for generic 2x2 RAW format (Bayer, RCCB, RCCC etc.)
  - Support for other color spaces:
    - RGB output
    - Support for color saturation and gray scale for HSV/HSL, etc.

- Support for multiple WDR/HDR formats:
  - Combined Companded format (up to 16-bit companded)
  - Separated Exposures – up to 3 frame Motion Adaptive HDR Merge
  - Staggered HDR
  - Digital Lateral Overlap (DLO)
- Support of statistics for Auto-Exposure/Auto-White-Balance or Auto-Focus (configurable option to select either AE/AWB or AF)
- Support for Flexible CFA for generating multi-plane output for any 2x2 RAW format sensor configuration.
  - Supports RCBC, RCCC, Bayer, RGBC as well as other formats
  - Can generate up to 4 independent color planes
  - Support for up to 3 directions per color plane
  - Adaptive threshold based on intensity measure
- Support for multiple color format output generation
  - Supports traditional 12-bit YUV output
  - Support of 8-bit YUV output (see [Section 6.9.3.6](#))
  - Supports RGB output (8-bit only)
  - Supports saturation outputs (8 bits only)
  - Supports Grayscale/Luma/IR/Clear output
- Global/Local Brightness and Contrast Enhancement module (16-bit)
- Spatial Noise Filter (NSF4V)
  - Supports raw input image in generic 2x2 color pattern format including Bayer pattern
  - Supports 16-bit raw input image
  - Spatially adaptive noise level threshold
  - Configurable soft or hard thresholding
  - Supports elliptical lens shading gain adjustment of adaptive noise threshold with 2 sets of independent settings
  - White Balance gain
  - Signal level detection with flexible cross-color-channel or independent averaging
- Option to bypass visual enhancement functions (NSF4V and GLBCE)
- Edge Enhancement (EE)
  - Enhance the visual quality of the image (Luma only)
  - Edge enhancer filter and the edge sharpener filter can be combined by configuration
  - Option to bypass visual enhancement functions (CAC, NSF4V, and GLBCE)
- Lens Distortion Correction (LDC) block. The LDC engine is YUV domain processor designed to perform perspective and geometric transforms. This can be used for creating several effects, including but not limited to:
  - Lens Distortion Correction, including Fish Eye Lenses
  - Epipolar rectification for Stereo
  - Generic perspective transforms for rotation scaling or Augmented Reality like effects

The output of the LDC block can be sent to external memory (DDR) or sent to other VPAC sub-module (Scalar, Noise filter) for further pre-processing via local shared memory (SL2). The LDC block provides the following main features:

- Performance up to 1 cycle/pixel (bilinear) and 2 cycles/pixel (bicubic)
- Autonomous memory-to-memory operation
- Tile based processing
- Generic mesh based distortion model to correct multiple distortion types
- Perspective warp transformation for perspective correction; affine transform is a subset of perspective warp and supports scaling and rotation
- Supported formats (see [Section 6.9.3.6](#)):
  - YUV420 12-bit packet input and YUV420 12-bit packed output
  - YUV420 12-bit unpacked input and YUV420 12-bit unpacked output
  - YUV420 8-bit packet input and YUV420 8-bit packed output
  - YUV422 8-bit packed input and YUV420 8-bit packed output

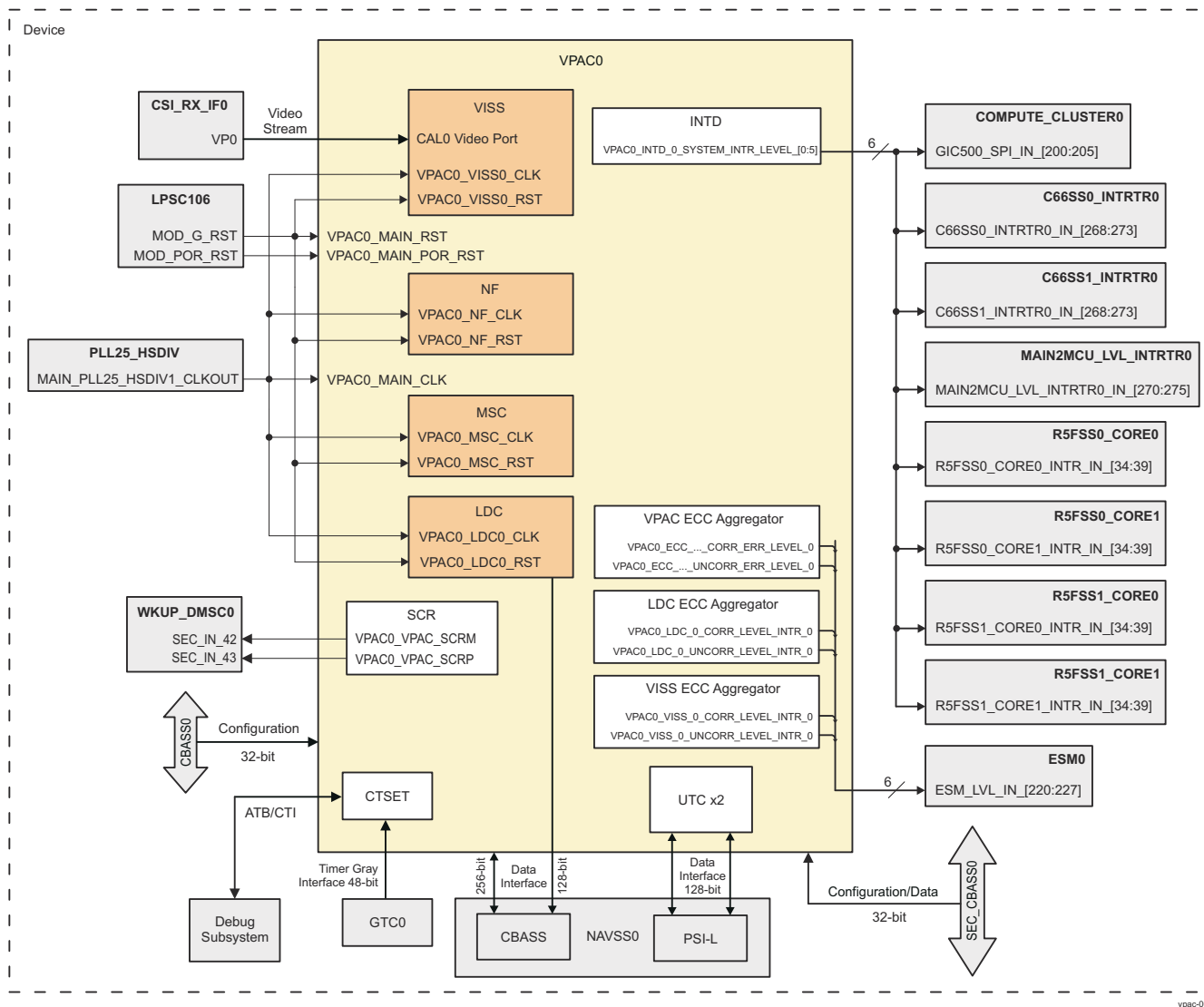
- YUV422 8-bit packed input and YUV422 8-bit packed output
  - Supports up to 8192 x 8192 image dimension
  - Pixel Interpolation
    - Bicubic interpolation for Y and bilinear interpolation for Cb/Cr
    - Bilinear interpolation mode to offer double throughput
  - Support of Luma Only or Chroma Only Mode
  - Supports variable block size frame processing (9 regions)
- Multi-Scalar (MSC) block. The MSC block reads data from memory (DDR or on-chip) to shared memory (SL2) and generates up to 10 scaled outputs from a given input with various scaling ratios (between 1x and x0.25). The output of the MSC block can be sent to external memory (DDR) from the local shared memory (SL2), or can be further noise filtered using the VPAC Noise Filter (NF) block. The MSC supports two independent threads (scalars), which combined can generate up to max 10 scales. The MSC block provides the following main features:
  - Multi-scaling capable: 10 simultaneous scaled outputs from 1 or 2 input planes; up to 2 simultaneous processing thread with total of 10 scaled output with 1 to N and 1 to M configurations (where N+M <=10)
  - Each scaling engine can be configured to perform Pyramid or inter-octave scale generation
  - Scaling ratio supported: 1x ~ x/4 (with 1x scaling, MSC can perform a 3~5-tap generic separable convolution filtering)
  - 5-tap separable filter kernel implementation
  - Programmable kernel sizes for vertical/horizontal filter (3, 4, or 5)
  - Poly phase implementation with support of 64 or 32 phases poly phase support
  - 4 sets of 5-tap x 32 phase coefficients (two can be combined to support 5-tap x 64 phase coefficients and one of set can hold additional 5-tap customer filter coefficients for non-resizing filter)
  - 2 additional sets of 5-tap Gaussian filter coefficients (single-phase) dedicated for Pyramid (Octave) generation
  - Coefficients in 10-bit (S10Q8 format - signed value of 10 bits with 8-bit fraction)
  - Separate initial horizontal and vertical phase programming option
  - Support for two planes of data processing
    - Data in a plane can be uniform (for example, Y plane) or interleaved (Cb/Cr interleaved)
  - 8-bit or 12-bit component data
  - Performance up to 1 input pixel/cycle
  - Output data rate is dependent on downscaling ratio
  - Programmable input and output ROI (or clipping) for each scaler
  - On-the-fly (OTF) or memory-to-memory (M2M) operation with line buffers in SL2 (frame resolution support is not constrained by MSC line buffer)
- Noise Filter (NF). The NF block reads data from memory (DDR or on-chip) to shared memory (SL2) and does Bilateral filtering to remove noise. The output of the NF block can be sent to external memory (DDR) from shared memory (SL2) or can be further re-sized using the MSC block. The NF block provides the following main features:
  - Support for bilateral and general filtering
  - Supports filter size up to 5x5
  - Supports true 2D bilateral filtering
  - Supports filter size up to 5x5 of programmable static weights
  - LUT based bilateral weights generation (8-bit for weight)
  - Supported formats: one plane (interleaved plane and non-interleaved) YUV 420, 12-bit or 8-bit
  - Performance up to 1 pixel/cycle
- Hardware Thread Scheduler (HTS). The HTS block used for inter-processor communication among various VPAC sub-modules (VISS, LDC, Scalar, and NF) and with the local DMA Engine (UTC). The HTS block provides the following main features:
  - Scheduling the HWA threads and associated tasks
  - Enables rate-controlled task execution
  - Support running several independent threads asynchronously in the sub-system
  - Debug functions:
    - Halting HWA threads at logical task boundary

- Handling pipeline suspend and subsequent abort by software intervention
- Shared Level 2 (SL2) memory. The SL2 is used as intermediate storage for Hardware Accelerator (HWA) to exchange data across VPAC sub-modules (that is, VISS, LDC, MSC, and NF) and from DDR/System Memory. VPAC supports Realtime and Non-Realtime processing threads simultaneously.

### 6.9.2 VPAC Integration

This section describes the VPAC integration in the device MAIN domain, including information about clocks, resets, and hardware requests.

There is one VPAC subsystem integrated in the device MAIN domain - VPAC0. [Figure 6-46](#) shows the integration of VPAC.



**Figure 6-46. VPAC Integration**

[Table 6-99](#) through [Table 6-101](#) summarize the integration of VPAC in the device MAIN domain.

**Table 6-99. VPAC Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect



**Table 6-99. VPAC Integration Attributes (continued)**

VPAC0	PSC0	PD29	LPSC106	CBASS0 NAVSS0_CBASS NAVSS0_PSI_L SEC_CBASS0
-------	------	------	---------	--

**Table 6-100. VPAC Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
VPAC0	VPAC0_MAIN_CLK	MAIN_PLL25_HSDIV1_CLKOUT	PLL25_HSDIV 1	VPAC0 main functional clock.
	VPAC0_VISS0_CLK			VPAC0 VISS0 functional clock.
	VPAC0_LDC0_CLK			VPAC0 LDC0 functional clock.
	VPAC0_MSC_CLK			VPAC0 MSC functional clock.
	VPAC0_NF_CLK			VPAC0 NF functional clock.
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
VPAC0	VPAC0_MAIN_POR_RST	MOD_POR_RST	LPSC106	VPAC0 POR reset
	VPAC0_MAIN_RST	MOD_G_RST		VPAC0 main reset
	VPAC0_VISS0_RST			VPAC0 VISS0 reset
	VPAC0_LDC0_RST			VPAC0 LDC0 reset
	VPAC0_MSC_RST			VPAC0 MSC reset
	VPAC0_NF_RST			VPAC0 NF reset

### Note

The VPAC and DMPAC clocks are derived from the same PLL and as a result, the following maximum clock frequency restrictions apply:

**Table 6-101. VPAC Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
VPAC0	VPAC0_INTD_0_SYSTEM_INTERRUPT_LEVEL[0:5]	GIC500_SPI_IN[200:205]	COMPUTE_CLUSTER0	VPAC0 interrupts	Level
		C66SS0_INTRTR0_IN[268:273]	C66SS0_INTRTR0		Level
		C66SS1_INTRTR0_IN[268:273]	C66SS1_INTRTR0		Level
		MAIN2MCU_LVL_INTRTR0_IN[270:275]	MAIN2MCU_LVL_INTRTR0		Level
		R5FSS0_CORE0_INTR_IN[34:39]	R5FSS0_CORE0		Level
		R5FSS0_CORE1_INTR_IN[34:39]	R5FSS0_CORE1		Level
		R5FSS1_CORE0_INTR_IN[34:39]	R5FSS1_CORE0		Level
		R5FSS1_CORE1_INTR_IN[34:39]	R5FSS1_CORE1		Level

**Table 6-101. VPAC Hardware Requests (continued)**

VPAC0_ECC_AGGR_0_ECC_ CORRECTED_ERR_LEVEL_0	ESM_LVL_IN_220	ESM0	VPAC0 subsystem (UTC memory) ECC Aggregator correctable interrupt	Level
VPAC0_ECC_AGGR_0_ECC_ UNCORRECTED_ERR_LEVEL _0	ESM_LVL_IN_221		VPAC0 subsystem (UTC memory) ECC Aggregator uncorrectable interrupt	Level
VPAC0_VISS_0_CORR_LEVEL _INTR_0	ESM_LVL_IN_222		VPAC0 VISS0 ECC Aggregator correctable interrupt	Level
VPAC0_VISS_0_UNCORR_LE VEL_INTR_0	ESM_LVL_IN_223		VPAC0 VISS0 ECC Aggregator uncorrectable interrupt	Level
VPAC0_LDC_0_CORR_LEVEL _INTR_0	ESM_LVL_IN_226		VPAC0 LDC0 ECC Aggregator correctable interrupt	Level
VPAC0_LDC_0_UNCORR_LEV EL_INTR_0	ESM_LVL_IN_227		VPAC0 LDC0 ECC Aggregator uncorrectable interrupt	Level
VPAC0_VPAC_SCRM_INTR	SEC_IN_42	WKUP_DMSC0	VPAC0 SCRM (SL2) firewall exception pending interrupt	-
VPAC0_VPAC_SCRP_INTR	SEC_IN_43		VPAC0 SCRP firewall exception pending interrupt	-

### 6.9.3 VPAC Subsystem Level

This section covers VPAC subsystem-level details.

#### 6.9.3.1 VPAC Subsystem Clocks

Each block within the VPAC subsystem receives its own clocking signal:

- MAIN\_CLK is the operation clock for all infrastructure modules, such as UTC/DRU, HTS, ECC Aggregator, INTD, etc.
- VISS0\_CLK is the operation clock for the VISS and its sub-modules
- LDC0\_CLK is the operation clock for the LDC module
- MSC\_CLK is the operation clock for the MSC module
- NF\_CLK is the operation clock for the NF module

For more details on the source of each clock at SoC level, see *VPAC Integration*.

#### 6.9.3.2 VPAC Subsystem Resets

Each block within the VPAC subsystem can be reset through its own HW reset signal:

- MAIN\_RST is the reset for all infrastructure modules, such as UTC/DRU, HTS, ECC Aggregator, etc.
- VISS0\_RST is the reset for the VISS and its sub-modules
- LDC0\_RST is the reset for the LDC module
- MSC\_RST is the reset for the MSC module
- NF\_RST is the reset for the NF module

For more details on the source of each reset signal at SoC level, see *VPAC Integration*.

#### 6.9.3.3 VPAC Subsystem Interrupts

The VPAC subsystem outputs six physical system interrupt lines: VPAC\_LEVEL\_0\_INTR through VPAC\_LEVEL\_5\_INTR. For more information on how the interrupt lines are connected at SoC level, see *VPAC Integration*. The interrupt aggregation within the VPAC subsystem is done using the internal INTD module. Each interrupt line can be mapped to the same list of source interrupt events generated by the modules within the VPAC subsystem.

[Table 6-102](#) shows the INTD interrupt control registers for each VPAC interrupt line. Each interrupt line has a separate set of registers for level and pulse input event types. Each bit in a register corresponds to a particular interrupt event. For more details on the mapping of interrupt events to INTD register bits, see [Table-XXX](#) further below.

- The ENABLE registers enable the mapping of the internal VPAC interrupt events, generated by a module within the VPAC subsystem, to the VPAC output system interrupt lines. Writing a '1' in a register bit position allows the corresponding module interrupt event to trigger the respective system interrupt. Writing a '0' in the same bit position blocks the module interrupt event from triggering the system interrupt.
- The ENABLE\_CLR registers disables the mapping of the internal VPAC interrupt events to the VPAC output system interrupt lines.
- The STATUS registers are used to capture which VPAC sub-module interrupt events, that have been mapped to a system interrupt, are active and pending. This status is used when multiple module interrupts are grouped into a single system interrupt, so the software can read the status register to determine exactly which module interrupt has caused the system interrupt.
  - For level interrupts, the status is based on the active level of the module interrupt input, and when the input is inactive then the status is inactive as well.
  - For pulsed interrupts, the status is set upon the pulse and reflects a pending status and software must clear the pending status since the module does not indicate the inactive status (unlike a level interrupt, which does).
- The STATUS\_CLR registers allows software to clear the pending status of interrupt events. This register is only valid and functional for pulsed interrupts and has no effect for level interrupts.

**Table 6-102. VPAC Subsystem Interrupt Control Registers**

Output System Interrupt Line	Input Interrupt Event Type	Interrupt Enable Registers	Interrupt Clear Registers	Interrupt Status Registers	Interrupt Status Clear Registers
VPAC_LEV EL_0_INTR	Level	VPAC_INTD_ENABLE_R EG_LEVEL_VPAC_OUT_ 0_0 through VPAC_INTD_ENABLE_R EG_LEVEL_VPAC_OUT_ 0_7	VPAC_INTD_ENABLE_CL R_REG_LEVEL_VPAC_O UT_0_0 through VPAC_INTD_ENABLE_CL R_REG_LEVEL_VPAC_O UT_0_7	VPAC_INTD_STATUS_RE G_LEVEL_VPAC_OUT_0 _0 through VPAC_INTD_STATUS_RE G_LEVEL_VPAC_OUT_0 _7	VPAC_INTD_STATUS_CL R_REG_LEVEL_VPAC_O UT_0_0 through VPAC_INTD_STATUS_CL R_REG_LEVEL_VPAC_O UT_0_7
	Pulse	VPAC_INTD_ENABLE_R EG_PULSE_VPAC_OUT_ 0_0 through VPAC_INTD_ENABLE_R EG_PULSE_VPAC_OUT_ 0_7	VPAC_INTD_ENABLE_CL R_REG_PULSE_VPAC_O UT_0_0 through VPAC_INTD_ENABLE_CL R_REG_PULSE_VPAC_O UT_0_7	VPAC_INTD_STATUS_RE G_PULSE_VPAC_OUT_0 _0 through VPAC_INTD_STATUS_RE G_PULSE_VPAC_OUT_0 _7	VPAC_INTD_STATUS_CL R_REG_PULSE_VPAC_O UT_0_0 through VPAC_INTD_STATUS_CL R_REG_PULSE_VPAC_O UT_0_7
VPAC_LEV EL_1_INTR	Level	VPAC_INTD_ENABLE_R EG_LEVEL_VPAC_OUT_ 1_0 through VPAC_INTD_ENABLE_R EG_LEVEL_VPAC_OUT_ 1_7	VPAC_INTD_ENABLE_CL R_REG_LEVEL_VPAC_O UT_1_0 through VPAC_INTD_ENABLE_CL R_REG_LEVEL_VPAC_O UT_1_7	VPAC_INTD_STATUS_RE G_LEVEL_VPAC_OUT_1 _0 through VPAC_INTD_STATUS_RE G_LEVEL_VPAC_OUT_1 _7	VPAC_INTD_STATUS_CL R_REG_LEVEL_VPAC_O UT_1_0 through VPAC_INTD_STATUS_CL R_REG_LEVEL_VPAC_O UT_1_7
	Pulse	VPAC_INTD_ENABLE_R EG_PULSE_VPAC_OUT_ 1_0 through VPAC_INTD_ENABLE_R EG_PULSE_VPAC_OUT_ 1_7	VPAC_INTD_ENABLE_CL R_REG_PULSE_VPAC_O UT_1_0 through VPAC_INTD_ENABLE_CL R_REG_PULSE_VPAC_O UT_1_7	VPAC_INTD_STATUS_RE G_PULSE_VPAC_OUT_1 _0 through VPAC_INTD_STATUS_RE G_PULSE_VPAC_OUT_1 _7	VPAC_INTD_STATUS_CL R_REG_PULSE_VPAC_O UT_1_0 through VPAC_INTD_STATUS_CL R_REG_PULSE_VPAC_O UT_1_7
VPAC_LEV EL_2_INTR	Level	VPAC_INTD_ENABLE_R EG_LEVEL_VPAC_OUT_ 2_0 through VPAC_INTD_ENABLE_R EG_LEVEL_VPAC_OUT_ 2_7	VPAC_INTD_ENABLE_CL R_REG_LEVEL_VPAC_O UT_2_0 through VPAC_INTD_ENABLE_CL R_REG_LEVEL_VPAC_O UT_2_7	VPAC_INTD_STATUS_RE G_LEVEL_VPAC_OUT_2 _0 through VPAC_INTD_STATUS_RE G_LEVEL_VPAC_OUT_2 _7	VPAC_INTD_STATUS_CL R_REG_LEVEL_VPAC_O UT_2_0 through VPAC_INTD_STATUS_CL R_REG_LEVEL_VPAC_O UT_2_7
	Pulse	VPAC_INTD_ENABLE_R EG_PULSE_VPAC_OUT_ 2_0 through VPAC_INTD_ENABLE_R EG_PULSE_VPAC_OUT_ 2_7	VPAC_INTD_ENABLE_CL R_REG_PULSE_VPAC_O UT_2_0 through VPAC_INTD_ENABLE_CL R_REG_PULSE_VPAC_O UT_2_7	VPAC_INTD_STATUS_RE G_PULSE_VPAC_OUT_2 _0 through VPAC_INTD_STATUS_RE G_PULSE_VPAC_OUT_2 _7	VPAC_INTD_STATUS_CL R_REG_PULSE_VPAC_O UT_2_0 through VPAC_INTD_STATUS_CL R_REG_PULSE_VPAC_O UT_2_7
VPAC_LEV EL_3_INTR	Level	VPAC_INTD_ENABLE_R EG_LEVEL_VPAC_OUT_ 3_0 through VPAC_INTD_ENABLE_R EG_LEVEL_VPAC_OUT_ 3_7	VPAC_INTD_ENABLE_CL R_REG_LEVEL_VPAC_O UT_3_0 through VPAC_INTD_ENABLE_CL R_REG_LEVEL_VPAC_O UT_3_7	VPAC_INTD_STATUS_RE G_LEVEL_VPAC_OUT_3 _0 through VPAC_INTD_STATUS_RE G_LEVEL_VPAC_OUT_3 _7	VPAC_INTD_STATUS_CL R_REG_LEVEL_VPAC_O UT_3_0 through VPAC_INTD_STATUS_CL R_REG_LEVEL_VPAC_O UT_3_7
	Pulse	VPAC_INTD_ENABLE_R EG_PULSE_VPAC_OUT_ 3_0 through VPAC_INTD_ENABLE_R EG_PULSE_VPAC_OUT_ 3_7	PAC_INTD_ENABLE_CL R_REG_PULSE_VPAC_O UT_3_0 through VPAC_INTD_ENABLE_CL R_REG_PULSE_VPAC_O UT_3_7	VPAC_INTD_STATUS_RE G_PULSE_VPAC_OUT_3 _0 through VPAC_INTD_STATUS_RE G_PULSE_VPAC_OUT_3 _7	VPAC_INTD_STATUS_CL R_REG_PULSE_VPAC_O UT_3_0 through VPAC_INTD_STATUS_CL R_REG_PULSE_VPAC_O UT_3_7

**Table 6-102. VPAC Subsystem Interrupt Control Registers (continued)**

Output System Interrupt Line	Input Interrupt Event Type	Interrupt Enable Registers	Interrupt Clear Registers	Interrupt Status Registers	Interrupt Status Clear Registers
VPAC_LEV_EL_4_INTR	Level	VPAC_INTD_ENABLE_REG_LEVEL_VPAC_OUT_4_0 through VPAC_INTD_ENABLE_REG_LEVEL_VPAC_OUT_4_7	VPAC_INTD_ENABLE_CLR_REG_LEVEL_VPAC_OUT_4_0 through VPAC_INTD_ENABLE_CLR_REG_LEVEL_VPAC_OUT_4_7	VPAC_INTD_STATUS_REG_LEVEL_VPAC_OUT_4_0 through VPAC_INTD_STATUS_REG_LEVEL_VPAC_OUT_4_7	VPAC_INTD_STATUS_CLR_REG_LEVEL_VPAC_OUT_4_0 through VPAC_INTD_STATUS_CLR_REG_LEVEL_VPAC_OUT_4_7
	Pulse	VPAC_INTD_ENABLE_REG_PULSE_VPAC_OUT_4_0 through VPAC_INTD_ENABLE_REG_PULSE_VPAC_OUT_4_7	VPAC_INTD_ENABLE_CLR_REG_PULSE_VPAC_OUT_4_0 through VPAC_INTD_ENABLE_CLR_REG_PULSE_VPAC_OUT_4_7	VPAC_INTD_STATUS_REG_PULSE_VPAC_OUT_4_0 through VPAC_INTD_STATUS_REG_PULSE_VPAC_OUT_4_7	VPAC_INTD_STATUS_CLR_REG_PULSE_VPAC_OUT_4_0 through VPAC_INTD_STATUS_CLR_REG_PULSE_VPAC_OUT_4_7
VPAC_LEV_EL_5_INTR	Level	VPAC_INTD_ENABLE_REG_LEVEL_VPAC_OUT_5_0 through VPAC_INTD_ENABLE_REG_LEVEL_VPAC_OUT_5_7	VPAC_INTD_ENABLE_CLR_REG_LEVEL_VPAC_OUT_5_0 through VPAC_INTD_ENABLE_CLR_REG_LEVEL_VPAC_OUT_5_7	VPAC_INTD_STATUS_REG_LEVEL_VPAC_OUT_5_0 through VPAC_INTD_STATUS_REG_LEVEL_VPAC_OUT_5_7	VPAC_INTD_STATUS_CLR_REG_LEVEL_VPAC_OUT_5_0 through VPAC_INTD_STATUS_CLR_REG_LEVEL_VPAC_OUT_5_7
	Pulse	VPAC_INTD_ENABLE_REG_PULSE_VPAC_OUT_5_0 through VPAC_INTD_ENABLE_REG_PULSE_VPAC_OUT_5_7	VPAC_INTD_ENABLE_CLR_REG_PULSE_VPAC_OUT_5_0 through VPAC_INTD_ENABLE_CLR_REG_PULSE_VPAC_OUT_5_7	VPAC_INTD_STATUS_REG_PULSE_VPAC_OUT_5_0 through VPAC_INTD_STATUS_REG_PULSE_VPAC_OUT_5_7	VPAC_INTD_STATUS_CLR_REG_PULSE_VPAC_OUT_5_0 through VPAC_INTD_STATUS_CLR_REG_PULSE_VPAC_OUT_5_7

The VISS interrupt events are mapped to register bits within the VPAC\_INTD\_xxx\_VPAC\_OUT\_0\_0 through VPAC\_INTD\_xxx\_VPAC\_OUT\_5\_0 registers. [Table 6-103](#) provides more details on the VISS interrupt events. All VISS interrupts are single pulse event signals.

**Table 6-103. VPAC Subsystem VISS Interrupt Events**

Register Bit	Interrupt Event Name	Description
0	RAWFE_CFG_ERR_INTR	Config read or write memory access occurred during functional operation and likely corrupted functional operation. VISS merges all config error sources from RawFE and refer to RawFE spec for the entire error source list.
1	RAWFE_AEW_PULSE_INTR	H3A AEW interrupt.
2	RAWFE_AF_PULSE_INTR	H3A AF interrupt.
3	RAWFE_H3A_PULSE_INTR	H3A interrupt.
4	RAWFE_H3A_BUF_OVRFLOW_PULSE_INTR	H3A output buffer overflow.
5	NSF4V_LINEMEM_CFG_ERR_INTR	VBUSP diagnostic read access of RAM, while NSF data using RAM for functional purpose.
6	NSF4V_HBLANK_ERR_INTR	Horizontal Blanking too short between lines.
7	NSF4V_VBLANK_ERR_INTR	Vertical Blanking too short between frames.
8	GLBCE_CFG_ERR_INTR	Either non-shadowed registers written or static memories are accessed during active window.
9	GLBCE_FILT_START_INTR	GLBCE started filtering. This interrupt is issued at the rising edge of filtering signal.
10	GLBCE_FILT_DONE_INTR	GLBCE ended filtering. This interrupt is issued at the falling edge of filtering signal.
11	GLBCE_HSYNC_ERR_INTR	Generated when delayed HS/HE signals doesn't match with derived signals from GLBCE core.

**Table 6-103. VPAC Subsystem VISS Interrupt Events (continued)**

Register Bit	Interrupt Event Name	Description
12	GLBCE_VSYNC_ERR_INTR	Generated when delayed VS/VE signals doesn't match with derived signals from GLBCE core.
13	GLBCE_VP_ERR_INTR	This interrupt is issued, if there is a data input while filtering is high.
14	FCFA_CFG_ERR_INTR	Either non-shadowed registers written or line memories are accessed during active window.
15	FCC_CFG_ERR_INTR	Configuration access to registers/memories has corrupted functional operation. Configuration read or write memory access occurred during functional operation. Merged independent error sources at VISS. Refer to Section <i>VISS Flexible Color Processing (FCP)</i> for all sources.
16	FCC_OUTIF_OVF_ERR_INTR	FIFO overflow on FIFO for Y12 LSE I/F. Merged all FCC output overflow error sources at VISS. Refer to Section <i>VISS Flexible Color Processing (FCP)</i> for all sources.
17	FCC_HIST_READ_ERR_INTR	Host was not able to read the entire histogram mem between VS-VE window (triggered when the first access to histogram has been performed but the last has not been performed).
18	EE_CFG_ERR	Configuration happened to EE regions causing corruption during frame processing.
19	EE_SYNCOVF_ERR	EE horizontal synchronization FIFO overflow interrupt.
20	LSE_FR_DONE_EVT_INTR	LSE frame done interrupt.
21	LSE_SL2_RD_ERR_INTR	Set whenever there is an error response on VBUSM read command for any input channel.
22	LSE_SL2_WR_ERR_INTR	Set whenever there is an error response on VBUSM write command for any output channel.
23	LSE_CAL_VP_ERR_INTR	Set whenever one of the following input frame errors is detected at VPORT_INPUT.
24	LSE_OUT_FR_START_EVT_INTR	Output Frame Start (from VISS top level).

The LDC interrupt events are mapped to register bits within the VPAC\_INTD\_xxx\_VPAC\_OUT\_0\_1 through VPAC\_INTD\_xxx\_VPAC\_OUT\_5\_1 registers. [Table 6-104](#) provides more details on the LDC interrupt events. All LDC interrupts are single pulse event signals.

**Table 6-104. VPAC Subsystem LDC Interrupt Events**

Register Bit	Interrupt Event Name	Description
0	LDC_PIX_IBLK_OUTOFBOUND_INTR	Back mapped pixel co-ordinate goes out of the pre-computed input pixel bounding box.
1	LDC_MESH_IBLK_OUTOFBOUND_INTR	Block mesh co-ordinate goes out of the pre-computed mesh bounding box.
2	LDC_PIX_IBLK_MEMOVF_INTR	Input Pixel block memory overflow.
3	LDC_MESH_IBLK_MEMOVF_INTR	Mesh block memory overflow.
4	LDC_IFR_OUTOFBOUND_INTR	Back mapped input co-ordinate goes out of input frame range.
5	LDC_INT_SZOVF_INTR	Affine and perspective transform precision overflow error.
6	LDC_FR_DONE_EVT_INTR	Frame done LDC.
7	LDC_SL2_WR_ERR_INTR	Error on SL2 VBSUM Write interface.
8	LDC_VBUSM_RD_ERR_INTR	Error on Input VBUSM Read interface.

The MSC interrupt events are mapped to register bits within the VPAC\_INTD\_xxx\_VPAC\_OUT\_0\_2 through VPAC\_INTD\_xxx\_VPAC\_OUT\_5\_2 registers. [Table 6-105](#) provides more details on the MSC interrupt events. All MSC interrupts are single pulse event signals.

**Table 6-105. VPAC Subsystem MSC Interrupt Events**

Register Bit	Interrupt Event Name	Description
0	MSC_LSE_FR_DONE_EVT_0_INTR	Frame done MSC processing thread 0.
1	MSC_LSE_FR_DONE_EVT_1_INTR	Frame done MSC processing thread 1.
2	MSC_LSE_SL2_RD_ERR_INTR	Error on SL2 VBSUM Read interface.
3	MSC_LSE_SL2_WR_ERR_INTR	Error on SL2 VBSUM Write interface.

The NF interrupt events are mapped to register bits within the VPAC\_INTD\_xxx\_VPAC\_OUT\_0\_2 through VPAC\_INTD\_xxx\_VPAC\_OUT\_5\_2 registers. [Table 6-106](#) provides more details on the NF interrupt events. All NF interrupts are single pulse event signals.

**Table 6-106. VPAC Subsystem NF Interrupt Events**

Register Bit	Interrupt Event Name	Description
8	NF_FR_DONE_INTR	Frame done NF.
9	NF_SL2_WR_ERR_INTR	Error on SL2 VBSUM Write interface.
10	NF_SL2_RD_ERR_INTR	Error on SL2 VBSUM Read interface.

The HTS interrupt events are mapped to register bits within the VPAC\_INTD\_xxx\_VPAC\_OUT\_0\_3 through VPAC\_INTD\_xxx\_VPAC\_OUT\_5\_3 registers. [Table 6-107](#) provides more details on the HTS interrupt events. All HTS interrupts are single pulse event signals, save for the SPARE\_PEND\_xxx interrupts, which can be both single pulse events and level events.

**Table 6-107. VPAC Subsystem HTS Interrupt Events**

Register Bit	Interrupt Event Name	Description
0	PIPE_DONE_0	Pipeline 0 done.
1	PIPE_DONE_1	Pipeline 1 done.
2	PIPE_DONE_2	Pipeline 2 done.
3	PIPE_DONE_3	Pipeline 3 done.
4	PIPE_DONE_4	Pipeline 4 done.
5	PIPE_DONE_5	Pipeline 5 done.
6	PIPE_DONE_6	Pipeline 6 done.
7	TDONE	HWA0 thread done.
8	RESERVED	Reserved.
9	TDONE	HWA2 thread done.
10	RESERVED	Reserved.
11	TDONE	HWA4 thread done.
12	TDONE	HWA5 thread done.
13	TDONE	HWA6 thread done.
14	SPARE_DEC_0	Spare 0 scheduler decrement pulse (ehost mode).
15	SPARE_DEC_1	Spare 1 scheduler decrement pulse (ehost mode).
16	SPARE_PEND_0_PULSE	Spare 0 scheduler pend assertion (ehost mode).
17	SPARE_PEND_0_LEVEL	Spare 0 scheduler pend assertion (ehost mode).
18	SPARE_PEND_1_PULSE	Spare 1 scheduler pend assertion (ehost mode).
19	SPARE_PEND_1_LEVEL	Spare 1 scheduler pend assertion (ehost mode).
20	WATCHDOGTIMER_ERR_0	Watchdog Timeout Error for HWA0.
21	RESERVED	Reserved.
22	WATCHDOGTIMER_ERR_2	Watchdog Timeout Error for HWA2.
23	RESERVED	Reserved.
24	WATCHDOGTIMER_ERR_4	Watchdog Timeout Error for HWA4.



**Table 6-107. VPAC Subsystem HTS Interrupt Events (continued)**

Register Bit	Interrupt Event Name	Description
25	WATCHDOGTIMER_ERR_5	Watchdog Timeout Error for HWA5.
26	WATCHDOGTIMER_ERR_6	Watchdog Timeout Error for HWA6.

The UTC interrupt events indicating TR complete are mapped to register bits within the VPAC\_INTD\_xxx\_VPAC\_OUT\_0\_4 through VPAC\_INTD\_xxx\_VPAC\_OUT\_5\_4 registers for UTC0, and VPAC\_INTD\_xxx\_VPAC\_OUT\_0\_5 through VPAC\_INTD\_xxx\_VPAC\_OUT\_5\_5 plus VPAC\_INTD\_xxx\_VPAC\_OUT\_0\_6 through VPAC\_INTD\_xxx\_VPAC\_OUT\_5\_6 registers for UTC1. [Table 6-108](#) provides more details on the UTC TR complete interrupt events. All UTC TR complete interrupts are single pulse event signals.

**Table 6-108. VPAC Subsystem UTC TR Complete Interrupt Events**

Register Bit	Interrupt Event Name	Description
31-0	UTC0_COMPLETE_INTR[31:0]	TR complete interrupt.
31-0 plus 31-0	UTC1_COMPLETE_INTR[63:0]	TR complete interrupt.

The UTC interrupt events indicating error are mapped to register bits within the VPAC\_INTD\_xxx\_VPAC\_OUT\_0\_7 through VPAC\_INTD\_xxx\_VPAC\_OUT\_5\_7 registers for both UTC0 and UTC1. [Table 6-109](#) provides more details on the UTC error interrupt events. All UTC error interrupts are single pulse event signals.

**Table 6-109. VPAC Subsystem UTC Error Interrupt Events**

Register Bit	Interrupt Event Name	Description
0	UTC0_ERR	UTC0 error.
1	UTC1_ERR	UTC1 error.
2	UTC0_PROT_ERR_INTR	UTC0 protocol violation.
3	UTC1_PROT_ERR_INTR	UTC1 protocol violation.

The CTSET interrupt events are mapped to register bits within the VPAC\_INTD\_xxx\_VPAC\_OUT\_0\_7 through VPAC\_INTD\_xxx\_VPAC\_OUT\_5\_7 registers. [Table 6-110](#) provides more details on the CTSET events. All CTSET error interrupts are single pulse event signals.

**Table 6-110. VPAC Subsystem CTSET Interrupt Events**

Register Bit	Interrupt Event Name	Description
4	CTM_PULSE_INTR	Counter Timer Module (CTM) pulse interrupt.

#### 6.9.3.4 VPAC Subsystem SL2 Memory Infrastructure

The SL2 memory subsystem implements an interconnect between DMA and HWAs initiators. It contains 512KB Shared L2 memory to acts temporary local memory to transfer data between external memory and HWAs.

The SL2 memory does not implement ECC, as it contains mostly pixel data.

#### 6.9.3.5 VPAC Subsystem DMA Infrastructure

Two UTC instances are used inside the VPAC subsystem. UTC0 deals with real time (RT) transfer , and UTC1 is used for non-realtime (NRT) transfer. Both UTC0 and UTC1 data transfer channels are used for DMA transfers in and out of the VPAC SL2 memory. All 32 channels of the UTC0 are mapped on *channel\_number* = [95:64] and are recommended to be used for real time data transfer only. All 64 channels of UTC1 are mapped on *channel\_number* = [63:0] and are recommended to be used for all other DMA transfer needs of the VPAC subsystem.

All data movement transactions at VPAC subsystem boundary are multiples of aligned 128-byte transfers.



### 6.9.3.6 VPAC Subsystem Data Formats Support

Table 6-111 summarizes the data formats supported in VPAC Subsystem.

**Table 6-111. VPAC Subsystem Data Formats Support**

No	Module	Input			Output		
		Bit-depth	Chroma (NV12)	Packing	Bit-depth	Chroma (NV12)	Packing
1	VISS	12	RAW	no	12 & 8	420	yes
2		14	RAW	no	12 & 8	420	yes
3		16	RAW	no	12 & 8	420	yes
4		8	RAW	yes	8	420	yes
5		12	RAW	yes	12 & 8	420	yes
5		All of the above			8	422	yes
6	LDC	8	422	yes	8	420	yes
7		8	422	yes	8	422	yes
8		8	420	yes	8	420	yes
9		12	420	yes/no	12	420	yes
10		12	420	yes/no	12 & 8	420	no
11	MSC	12	Planar	yes	12	planar	yes
12		8	Planar	yes	8	planar	yes
13	NF	12	Y/UV (NV12)	yes	12	UV (NV12)	yes

### 6.9.3.7 VPAC Subsystem Debug Features

The VPAC subsystem is a combination of multiple asynchronous pipelines running concurrently. The Counter, Timer and System Event Trace (CTSET) module within the VPAC subsystem provides the following main features:

- Monitoring up to 255 VPAC events.
- ATB interface compliant trace packets (synchronous to VPAC clock domain).
- Counter (count=8) and Timer (count=4).
- All debug events can be configured to be either selected for trace packet, timer or counter features.
- In one of the combination, debug events can be selected to for timer event out to be used to generate external CTI compliant trigger.

For more information on the CTSET module operation, see Chapter *On-Chip Debug*.

Table 6-112 lists the VPAC subsystem events that are mapped on the CTSET module.

**Table 6-112. VPAC Subsystem CTSET Event List**

Index	Event Name	HWA	Event Type	Event Description
0	viss_err	VISS0	pulse	VISS Related Error message (config, ovf, sync, blank, access, vp) <sup>(1)</sup>
1	rawfe_h3a_pulse_intr	VISS0	pulse	H3A interrupt
2	rawfe_dbg_ctl_vs_event	VISS0	pulse	Verticle start in the beginning of the RAWFE pipeline
3	rawfe_dbg_ctl_ve_event	VISS0	pulse	Verticle end in the beginning of the RAWFE pipeline
4	rawfe_dbg_ctl_hs_event	VISS0	pulse	Horizontal start in the beginning of the RAWFE pipeline
5	rawfe_dbg_ctl_he_event	VISS0	pulse	Horizaontal end in the beginning of the RAWFE pipeline
6	rawfe_dbg_ctl_lse_slv_stall_event	VISS0	pulse	RAWFE is stalling lse slave interface due to FCP or MTC stall
7	rawfe_dbg_ctl_lse_mst_stall_event	VISS0	pulse	H3A master interface to LSE is stalled

**Table 6-112. VPAC Subsystem CTSET Event List (continued)**

Index	Event Name	HWA	Event Type	Event Description
8	rawfe_dbg_ctl_lse_intf_stall_event	VISS0	pulse	Within a frame (VS to VE at RFE i/p) LSE is not sending data to RFE
9	rawfe_dbg_ctl_x_y_match_event	VISS0	pulse	X,Y pixel position has reached the start of RAWFE pipeline
10	rawfe_dbg_ctl_dpc_otf_corr_event	VISS0	pulse	DPC OTF corrected a pixel position
11	rawfe_dbg_ctl_pipe_adv_event	VISS0	pulse	Active pipeline advancement
12	nsf4v_in_hs_event	VISS0	pulse	Start of Active Line at NSF4V input
13	nsf4v_in_vs_event	VISS0	pulse	Start of Active Frame at NSF4V input
14	nsf4v_in_he_event	VISS0	pulse	End of Active Line at NSF4V output
15	nsf4v_in_ve_event	VISS0	pulse	End of Active Frame at NSF4V output
16	glbce_filt_start_intr	VISS0	pulse	GLBCE filtering start ( generated at the rising edge of filtering signal)
17	glbce_filt_done_intr	VISS0	pulse	GLBCE filtering done (generated at falling edge of filtering signal)
18	glbce_sol_event	VISS0	pulse	Start of Active Line at GLBCE input
19	glbce_sof_event	VISS0	pulse	Start of Active Frame at GLBCE input
20	glbce_eol_event	VISS0	pulse	End of Active Line at GLBCE output
21	glbce_eof_event	VISS0	pulse	End of Active Frame at GLBCE output
22	fcfa_sol_event	VISS0	pulse	Start of line processing for CFA input
23	fcfa_sof_event	VISS0	pulse	Start of frame processing for CFA input
24	fcc_stall_event	VISS0	pulse	Stall has occurred on any one of the output LSE interfaces
25	fcc_eol_if_y12_event	VISS0	pulse	End of line on Y12 LSE I/f (only generated for valid lines)
26	fcc_eof_if_y12_event	VISS0	pulse	End of frame on Y12 LSE I/f
27	fcc_eol_if_uv12_event	VISS0	pulse	End of line on UV12 LSE I/f (only generated for valid lines)
28	fcc_eof_if_uv12_event	VISS0	pulse	End of frame on UV12 LSE I/f
29	fcc_eol_if_y8r8_event	VISS0	pulse	End of line on Y8R8 LSE I/f (only generated for valid lines)
30	fcc_eof_if_y8r8_event	VISS0	pulse	End of frame on Y8R8 LSE I/f
31	fcc_eol_if_c8g8_event	VISS0	pulse	End of line on C8G8 LSE I/f (only generated for valid lines)
32	fcc_eof_if_c8g8_event	VISS0	pulse	End of frame on C8G8 LSE I/f
33	fcc_eol_if_s8b8_event	VISS0	pulse	End of line on S8B8 LSE I/f (only generated for valid lines)
34	fcc_eof_if_s8b8_event	VISS0	pulse	End of frame on S8B8 LSE I/f
35	fcc_flexcc_eop_event	VISS0	pulse	End of processing (EOP) for FlexCC
36	lse_fr_done_evt_intr	VISS0	pulse	Frame processing done event
37	lse_out_fr_start_evt_intr	VISS0	pulse	Travelled Frame Start from LSE Core
39	ldc_err	LCD0	pulse	LDC processing error <sup>(2)</sup>
40	mesh_iblk_fetch_start_event	LCD0	pulse	Input mesh block fetch start event
41	mesh_iblk_fetch_done_event	LCD0	pulse	Input mesh block fetch completion event
42	pix_iblk_fetch_start_event	LCD0	pulse	Input pixel block fetch start event (Active for both Y and C)
43	pix_iblk_fetch_done_event	LCD0	pulse	Input pixel block fetch done event (Active for both Y and C)
44	frame_start_event	LCD0	pulse	Frame processing start event

**Table 6-112. VPAC Subsystem CTSET Event List (continued)**

Index	Event Name	HWA	Event Type	Event Description
45	lse_stall_event	LCD0	pulse	LSE write stall event
46	coreblk_proc_done_event	LCD0	pulse	LDC Core Block processing done
47	coreblk_wr_done_event	LCD0	pulse	LDC Core block write complete on C12 channel
48	coreyblk_wr_done_event	LCD0	pulse	LDC Core block write complete on Y12 channel
49	vpac_ldc_fr_done_evt_intr	LCD0	pulse	Frame processing done event
51	msc_err	MSC	pulse	Error Response for VBUSM read/Write SL2 access (msc_lse_sl2_rd_err_intr    msc_lse_sl2_wr_err_intr)
52	msc_lse_fr_done_evt_0_intr	MSC	pulse	Frame processing done event for MSC Thread0
53	msc_lse_fr_done_evt_1_intr	MSC	pulse	Frame processing done event for MSC Thread1
55	nf_err	NF	pulse	Error Response for VBUSM read/Write SL2 access (vpac_nf_sl2_wr_err_intr    vpac_nf_sl2_rd_err_intr)
56	vpac_nf_fr_done_evt_intr	NF	pulse	Frame processing start event
58	cal_vp_stall	CAL	pulse	CAL VP stall
60	start	HTS	pulse	HWA-0 (task start)
61	done	HTS	pulse	HWA-0 (task done)
62	init	HTS	pulse	HWA-0 (init)
63	eop	HTS	pulse	HWA-0 (eop)
64	out_ch0_done	HTS	pulse	HWA-0-ch0-done
65	out_ch1_done	HTS	pulse	HWA-0-ch1-done
66	out_ch2_done	HTS	pulse	HWA-0-ch2-done
67	out_ch3_done	HTS	pulse	HWA-0-ch3-done
68	out_ch4_done	HTS	pulse	HWA-0-ch4-done
69	out_ch5_done	HTS	pulse	HWA-0-ch5-done
70	start	HTS	pulse	HWA-2 (task start)
71	done	HTS	pulse	HWA-2 (task done)
72	init	HTS	pulse	HWA-2 (init)
73	eop	HTS	pulse	HWA-2 (eop)
74	out_ch0_done	HTS	pulse	HWA-2-ch0-done
75	out_ch1_done	HTS	pulse	HWA-2-ch1-done
76	out_ch2_done	HTS	pulse	HWA-2-ch2-done
77	out_ch3_done	HTS	pulse	HWA-2-ch3-done
78	start	HTS	pulse	HWA-4 (task start)
79	done	HTS	pulse	HWA-4 (task done)
80	init	HTS	pulse	HWA-4 (init)
81	eop	HTS	pulse	HWA-4 (eop)
82	start	HTS	pulse	HWA-5 (task start)
83	done	HTS	pulse	HWA-5 (task done)
84	init	HTS	pulse	HWA-5 (init)
85	eop	HTS	pulse	HWA-5 (eop)
86	out_ch0_done	HTS	pulse	HWA-4-5-ch0-done
87	out_ch1_done	HTS	pulse	HWA-4-5-ch1-done

**Table 6-112. VPAC Subsystem CTSET Event List (continued)**

Index	Event Name	HWA	Event Type	Event Description
88	out_ch2_done	HTS	pulse	HWA-4-5-ch2-done
89	out_ch3_done	HTS	pulse	HWA-4-5-ch3-done
90	out_ch4_done	HTS	pulse	HWA-4-5-ch4-done
91	out_ch5_done	HTS	pulse	HWA-4-5-ch5-done
92	out_ch6_done	HTS	pulse	HWA-4-5-ch6-done
93	out_ch7_done	HTS	pulse	HWA-4-5-ch7-done
94	out_ch8_done	HTS	pulse	HWA-4-5-ch8-done
95	out_ch9_done	HTS	pulse	HWA-4-5-ch9-done
96	start	HTS	pulse	HWA-6 (task start)
97	done	HTS	pulse	HWA-6 (task done)
98	init	HTS	pulse	HWA-6 (init)
99	eop	HTS	pulse	HWA-6 (eop)
100	pipe_done	HTS	pulse	Pipeline-0 Done
101	pipe_done	HTS	pulse	Pipeline-1 Done
102	pipe_done	HTS	pulse	Pipeline-2 Done
103	pipe_done	HTS	pulse	Pipeline-3 Done
104	pipe_done	HTS	pulse	Pipeline-4 Done
105	pipe_done	HTS	pulse	Pipeline-5 Done
106	pipe_done	HTS	pulse	Pipeline-6 Done
107	spare_dec_0	HTS	pulse	Spare 0 sch decrement pulse (ehost mode)
108	spare_dec_1	HTS	pulse	Spare 1 sch decrement pulse (ehost mode)
109	spare_pend_0	HTS	level	Spare 0 sch pend assertion (ehost mode)
110	spare_pend_1	HTS	level	Spare 1 sch pend assertion (ehost mode)
142:111	utc1_channel_start[31:0]	HTS	pulse	MMR config to select HWA SL2 master ports access control signals
173:143	utc1_channel_start[63:32] if CTSET_RT_UTC_IN='0' else utc0_channel_start[31:0]	UTC	pulse	Channel start
206:175	utc1_ctset_intr[31:0]	HTS	pulse	MMR config to select external master port access control signals
238:207	utc1_ctset_intr[63:32] if CTSET_RT_UTC_OUT='0' else utc0_ctset_intr[31:0]	UTC	pulse	CTSET event output (for HWA channel_done)
254:239	[utc1,utc0] {wrrreq,1'b0,wrrreq stall,wrrreq valid creq, rd rreq stall, rd valid rreq, rd creq stall, rd valid creq}	HTS	pulse	MMR config to select external ctset events or UTC SL2 access control signals

- (1) The signals combined to generate the viss\_err event are: rawfe\_cfg\_err\_intr || glbce\_cfg\_err\_intr || fcfa\_cfg\_err\_intr || fcc\_cfg\_err\_intr || ee\_cfg\_err || nsfv4\_line\_cfg\_err || rawfe\_h3a\_buf\_overflow\_pulse\_intr || nsf4v\_hblank\_err\_intr || nsf4v\_vblank\_err\_intr || glbce\_vp\_err\_intr || glbce\_hsync\_err\_intr || glbce\_vsync\_err\_intr || fcc\_outif\_ovf\_err\_intr || fcc\_hist\_read\_err\_intr || ee\_syncovf\_err\_intr || lse\_sl2\_rd\_err\_intr || lse\_sl2\_wr\_err\_intr || lse\_cal\_vp\_err\_intr.
- (2) The signals combined to generate the ldc\_err event: pix\_iblk\_outofbound\_intr || mesh\_iblk\_outofbound\_intr || pix\_iblk\_memovf\_intr || mesh\_iblk\_memovf\_intr || ifr\_outofbound\_intr || int\_szovf\_intr || vpac\_ldc\_sl2\_wr\_err\_intr || vpac\_ldc\_vbusm\_rd\_err\_intr.

The VPAC subsystem also supports halting an execution pipeline using external or user halt request. Halting a pipeline is achieved by not granting thread start to HWA. Once halted, VPAC waits for an external sync trigger or config resume to restart execution. VISS, MSC and NF modules halt at line boundary, and LDC halts at block boundary.

The VPAC subsystem supports generic debug capability/features and ARM cross trigger interface for debug triggers:

- Halting the HWA threads on debug request input at logical trigger boundary.
- Continue/restart VPAC from halted state (step/continue) based on external sync trigger and HTS\_DBG\_CNTL register.
- Debug compliant dbg\_attn output after processing external halt request and when VPAC halts debug enabled pipelines.
- CTI (Cross Trigger Interface) compliant out halted trigger after processing external halt request and reaching halted state of VPAC.
- Stalling all pipelines of VPAC (with configurable option to disable impact of debug features on a pipeline).
- Indicate to Host through readable debug\_rdy bit in HTS to indicate readiness of HWA resource in case of halted (dbgattn asserted)

The VPAC subsystem debug capability supported is as captured in the HTS\_DBG\_CAP register. Ext\_halt, ext\_sync, hwa\_halted are CTI (Cross Trigger Interface) mapped async trigger interface (refer to the async protocol of trigger interface in the ARM® CoreSight™ Architecture Specification).

The VISS module, when used in OTF mode, must not be included as debuggable module within VPAC. VISS must be disabled (HTS: debug pipeline disable) to respond to halt request, otherwise behavior is undefined. The VISS module, when used in memory to memory mode, can be halted. But VISS LUTs must not be read during halted state, as it can corrupt the design pipeline.

#### 6.9.3.8 VPAC Subsystem Security Features

The VPAC subsystem implements region basead firewall on slave endpoints within config CBASS and SL2 CBASS. Both CBASS is driven by independent DMSC FW interface from SoC level. All internal masters on SL2 SCR are configured with ISC. [Figure 6-47](#) provides FW/ISC summary.

Module	Interconnet	Security component	Parameter	Endpoints	FW_ID( dec)	FW/ISC/QoS config address offset(hex)	privID reset value( dec)
VPAC	SCRP	Region FW	2 region	ECC AGG	6048	0	
			1 region	VPAC_TOP	6049	400	
			8 region	INTD	6050	800	
			8 region	HTS	6051	C00	
			1 region	CTSET	6052	1000	
			4 region	VISS0	6053	1400	
			4 region	LDC0	6055	1C00	
			2 region	MSC	6057	2400	
			1 region	NF	6058	2800	
			8 region	UTC0	6059	2C00	
			8 region	UTC1	6060	3000	
	SCRM (SL2)	ISC	1 port x 1 initiator	VISS0		0	213
			1 port x 1 initiator	LDC0		800	215
			1 port x 2 initiator	MSC		1000	217
			1 port x 1 initiator	NF		1400	218
		Region FW	8 region	SL2 Bank0	6080	0	
			8 region	SL2 Bank1	6081	400	
			8 region	SL2 Bank2	6082	800	
			8 region	SL2 Bank3	6083	C00	

vpac-005

**Figure 6-47. VPAC Subsystem FW/ISC Configuration**

## 6.9.4 VPAC Vision Imaging Subsystem (VISS)

This chapter describes the Vision Imaging Subsystem (VISS).

### 6.9.4.1 VISS Top Level

This section describes the Vision ISS (VISS) top level details.

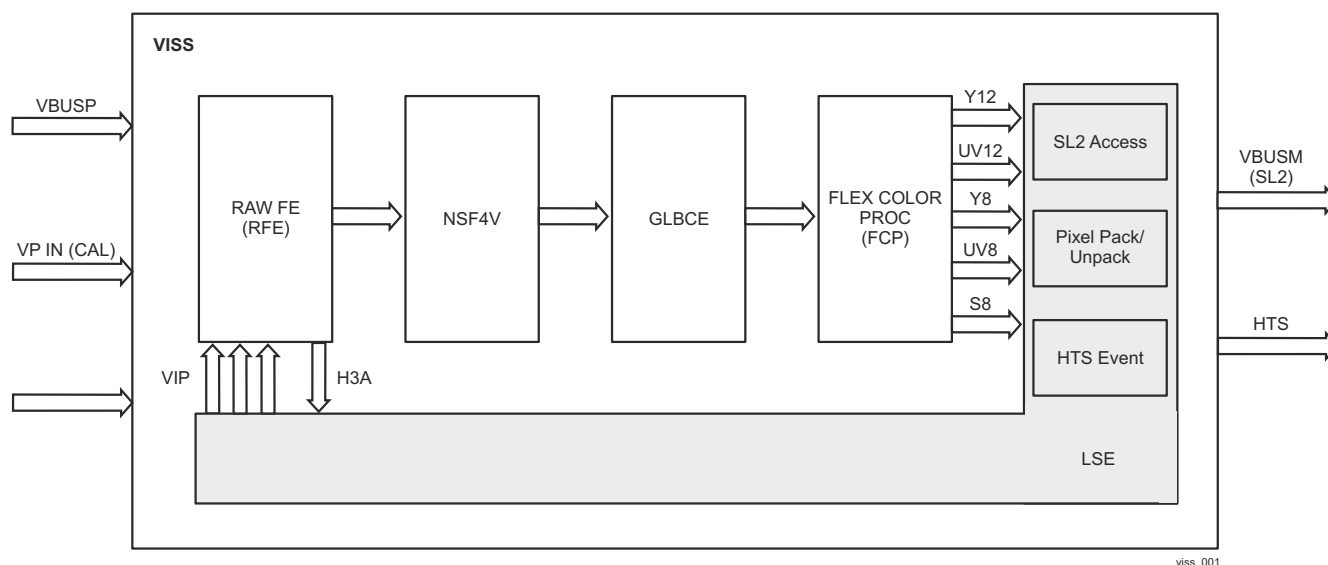
#### 6.9.4.1.1 VISS Features

The VPAC includes the following processing and infrastructure sub-modules:

- Vision Imaging Sub-System (VISS). The vision imaging pipe does image processing on raw data which includes Wide Dynamic Range (WDR) merge, Defect Pixel Correction (DPC), Lens Shading Correction (LSC), Global/Local Brightness and Contrast Enhancement (GLBCE), demosaicing, color conversion, and Edge Enhancement (EE). It can operate on sensor data either on-the-fly or in memory-to-memory mode. The VISS provides the following main features:
  - Performance is 1 pixel per cycle
  - Up to 16-bit input RAW formats (8b, 12b, 14b, 16b)
  - Support for concurrent 12-bit and 8-bit YUV output
  - Support for generic 2x2 RAW format (Bayer, RCCB, RCCC etc.)
  - Support for other color spaces:
    - RGB output
    - Support for color saturation and gray scale for HSV/HSL, etc. (Hue is not supported)

#### 6.9.4.1.2 VISS Block Diagram

The VISS module does on-the-fly processing on raw pixels captured from camera sensor (via CSI RX module). It also does memory to memory processing (via DDR memory) for data captured from other sources at SoC level. [Figure 6-48](#) is simplified block diagram of VISS top level.



**Figure 6-48. VISS Block Diagram**

The VISS consists of the following Hardware Accelerators (HWAs):

- RFE (RAW-FE): The RAW Frond End HW block does RAW pixel (that is, Bayer, RCCC, RGBW, etc.) processing on captured image data from sensor and pass-on to NSF4V, GLBCE block, and then to Flexible Color Processing (FCP) HW block for demosaicing and color conversion.
- NSF4V: Spatial Noise Filter supporting generic 2x2 pixel format with 16-bit pixel size. The NSF4V module can be bypassed in the applications where visual enhancement is not desired.

- GLBCE: The GLBCE module is used for dynamic range control within image for visual quality. If contrast enhancement on input image is required for visual quality, the RFE output is processed by the NSF4v and GLBCE blocks. Otherwise, the RFE output is bypassed to FCP.
- FCP: The Flexible Color Processing HW receives data from the GLBCE and does demosaicing and color conversion. The output of FCP is sent to VPAC shared memory to be written into DDR for the rest of the vision processing by programmable processors (for example, DSP or Arm), or other Vision HW blocks (for example, DMPAC).
- LSE: The Load and Store Engine is an infrastructure block, which performs the following functions:
  - CAL video port: The CSI RX module at SoC level receives CSI-2 sensor data, extracts pixels and drives the result data on its video port interface. The video port is mapped onto one of LSE input interfaces for on-the-fly image processing to reduce DDR bandwidth and latency. The LSE also provides horizontal and vertical blanking cycles to allow core data path to settle at proper boundary of line and frame.
  - SL2 memory access: This module supports load/store data from/to SL2 using VBUSM interface. Loaded data from SL2 are passed on to unpacker function of LSE for RFE. Packed data after FCP/H3A processing is written into SL2.
  - Pixel pack/unpack: Source data (512-bit) for RFE processing is loaded from SL2 and passed onto unpacker function for pixel extraction. Extracted pixels are driven on to the video port of RFE. Similarly, the FCP produced pixels are passed through packer function for eventual write into SL2. The H3A generated data is pseudo mapped as pixels of 32-bit for packing purpose and directly driven by the RFE.
  - Event control: The HTS events are generated at line level. These events are routed to LSE to start the processing. For each consumed line, the HTS needs a *task done* indication. For some initial lines of image, there would not be any valid data output. Similarly, the H3A generated data will be at paxel/window height number of lines. For initial lines not producing any valid data, the HTS needs to be indicated with separate mask bits for each output streams. For these initial lines, when there is no valid output due to lines delay inside VISS, the LSE still generates mask output to HTS indicating lack of proper output data.

#### 6.9.4.1.3 VISS Data Flow within VPAC

VISS receives data from sensor through:

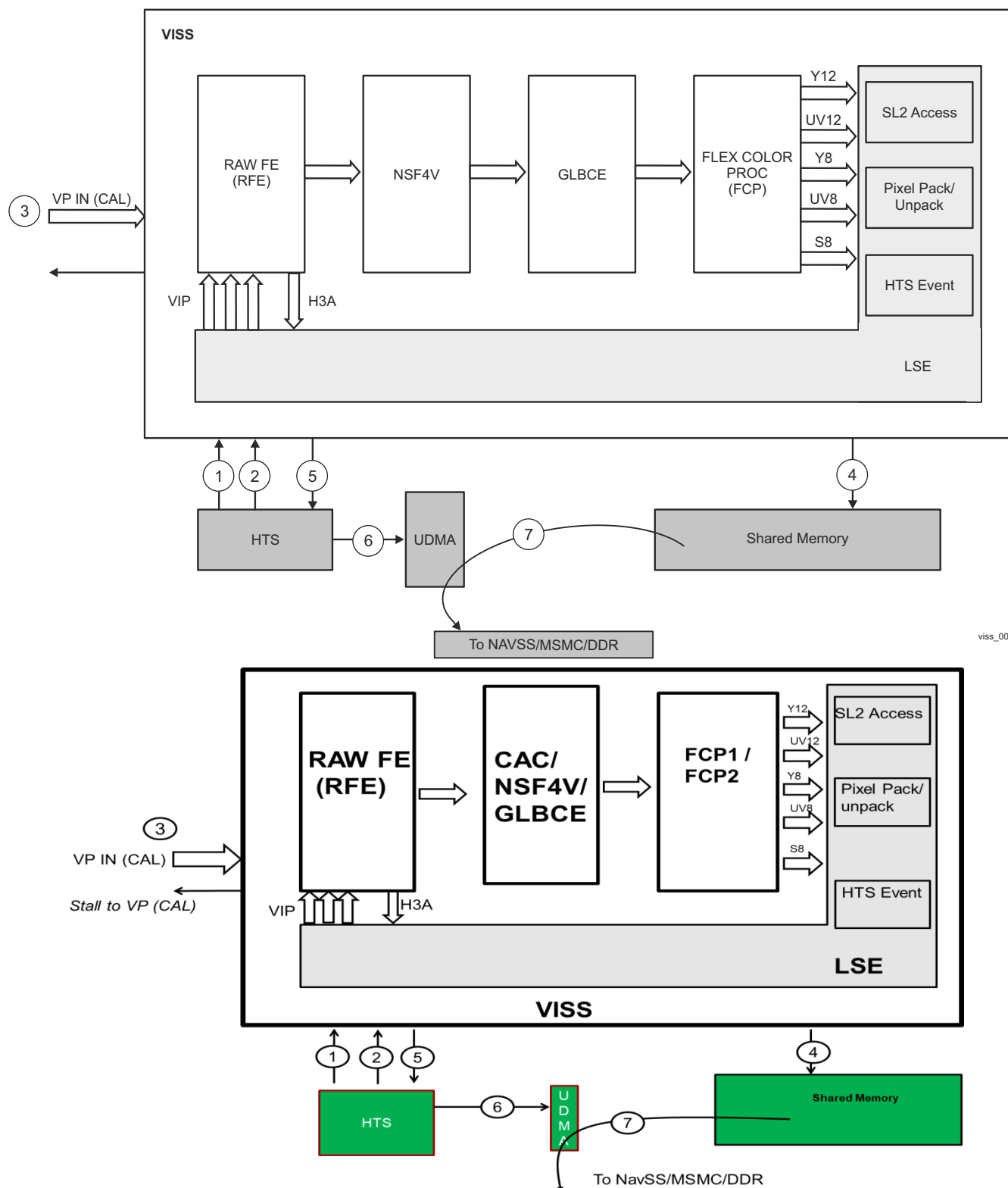
- CSI RX video port for OTF processing (Sensor -> CSI RX -> LSE -> RFE -> FCP)
- DMA of VPAC (Sensor -> CAL -> DDR -> VPAC\_SL2 -> LSE -> RFE -> FCP)

##### 6.9.4.1.3.1 VISS On-the-fly Processing

The VISS operation is controlled through a single HTS thread. For details of the HTS operation refer to Section *VPAC Hardware Thread Scheduler (HTS)*. The VISS configuration is valid for a single or multiple frame. External-host manages VISS configuration at end of each frame, if required. The HTS controls image processing between lines and handles managing shared memory for VISS inputs/outputs. Sub-frame processing is not natively supported. SW must configure VISS and HTS to start initialization process before enabling sensor capture. After the initialization sequence, the HTS will generate a start trigger to VISS. The generation of the start trigger depends on the availability of H3A data out buffer and FCP data out buffer inside the SL2 memory.

At this stage, the LSE waits for CSI RX video port (VP) to start sending valid pixels. As and when VP data starts streaming in, it is routed to the RFE video port. As RFE and FCP are running in streaming mode, the FCP indicates to LSE about completion of one line of operation. After receiving valid data out, the HTS will trigger UDMA for data store into DDR. Each 'thread done' accompanies output mask for all output buffers. Mask bits indicate to HTS about validity of output buffer as for each thread done all output buffers do not need to be produced. The HTS will issue next start after checking all output buffers availability. The streaming data from CSI RX is storable upto 2KB storage inside VP memory of CSI RX. Any temporary stall can be absorbed, but prolonged stall is buffer overflow inside CSI RX. In VPAC, VISS produced data is transferred using Real Time fabric of NAVSS/MSMC and low predictable latency needs to be guaranteed.





**Figure 6-49. VISS On-the-fly Operation**

The following OTF operation steps are illustrated in [Figure 6-49](#):

1. *Init* from HTS to initialize VISS.



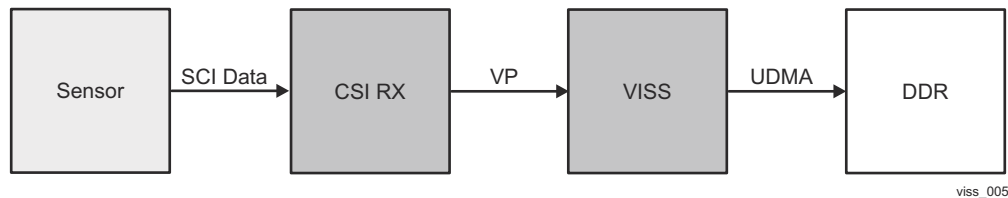
2. Thread start. Step 1+2 prior to enabling CSI RX + PHY + sensor.
3. Video port data streaming. Unit of operation inside VISS at line level.
4. Output data produced by FCP and H3A at line level. H3A will produce data only at those lines which aligns with paxel\_height/window\_height.
5. Production of 1 line depends on consumption of 1 line on VP interface. 'Thread done' triggered after completion of 1 processed line write into SL2. In case there is no valid data to be written into SL2, 'thread done' is generated after consumption of 1 line of data from VP. Note: *Horizontal blanking will be configured inside LSE to facilitate writing complete line out of FCP.*
6. HTS triggers DMA to store output buffer from SL2 to DDR.
7. DMA loads data from SL2 and writes into DDR.

The following assumptions are made for the OTF operation described above:

- UDMA write is mapped onto Real Time Thread for guaranteed QoS. Channel done by UDMA must be generated after completion of write into DDR. Next thread start depend on availability of minimum one buffer for each output line in SL2. More buffers in SL2 can take care of instantaneous drop in UDMA transfer capacity.
- Cycle gap between HTS done and next start must be minimal. CSI RX VP interface will be stalled whenever LSE input buffer on VP interface reaches near full condition.
- Configurable vertical latency times 'Tstart' generated by HTS to let RFE + NSF4V + GLBCE + FCP flush its internal pipe, even though last line of current frame is already fed into RFE.
- Buffer dependencies:
  - H3A output buffer (AE or AF)
  - FCP output buffer (Y12, YV12, Y8, UV8, S8)

#### 6.9.4.1.3.1.1 Non-WDR or Companded WDR Sensors

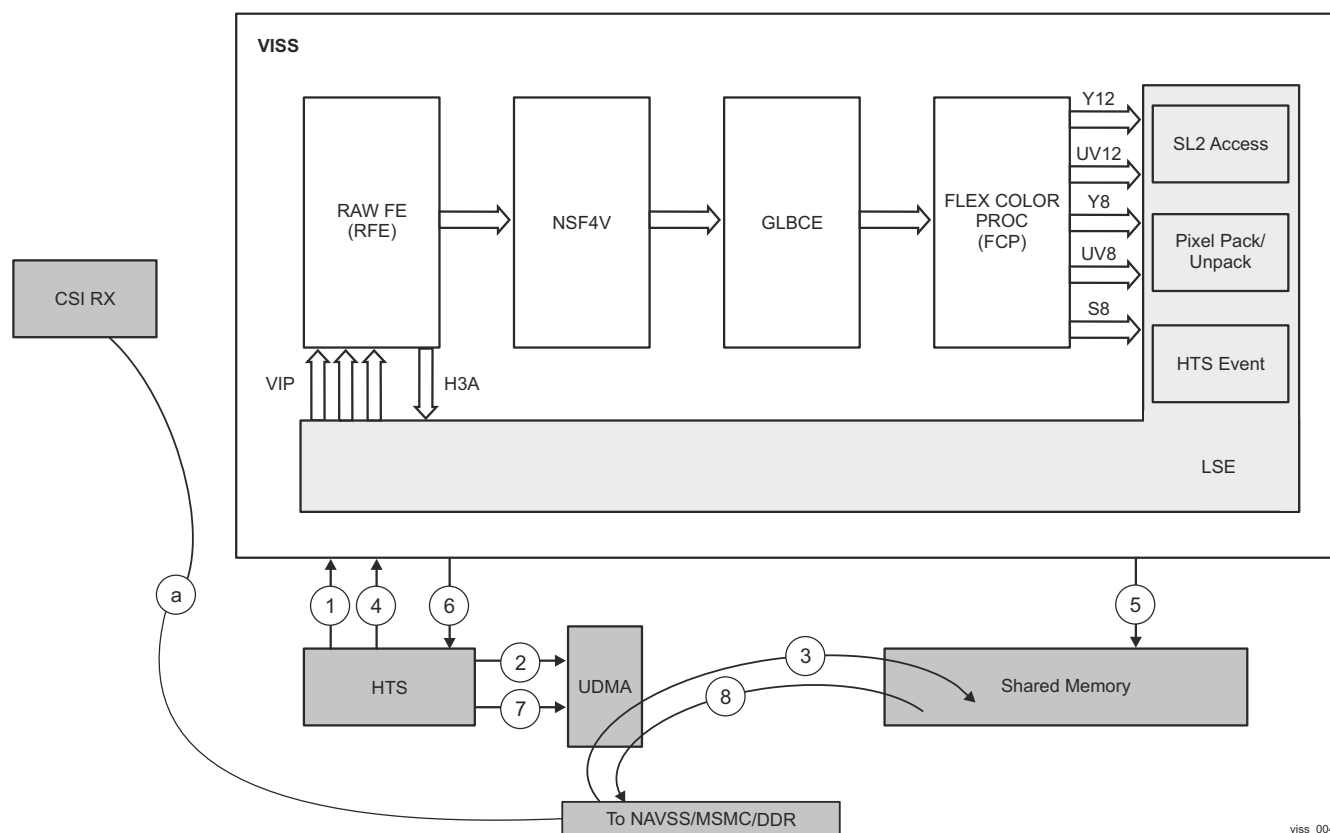
VISS is enabled for single input data when either sensor does not support WDR or sensor does on-chip WDR merge and pass on merged data after companding to 12 bit / 14 bits as shown in [Figure 6-50](#).



**Figure 6-50. VISS Non-WDR/Companded: High Level**

#### 6.9.4.1.3.2 VISS Memory to Memory Image Processing

CSI RX writes CSI-2 received pixel data into DDR. Once a frame is written, host is triggered to initiate VISS processing. HTS will go through 'init' sequence to initialize VISS and its SL2 pointers. As UDMA is head of pipeline for memory to memory mode of image processing, its scheduler inside HTS needs to be configured for generating "frame height" number of triggers to fetch image lines one by one. This is essential to bring determinism in pipeline as there is no producer for UDMA load thread.



**Figure 6-51. VISS Memory to Memory Operation**

The following memory to memory operation steps are illustrated in [Figure 6-51](#):

1. CSI-2 stream captured via CSI RX into DDR. This traffic must be routed through Real Time fabric of NAVSS/MSMC. CSI RX capture happens independent of VISS processing.
1. *Init* from HTS to initialize VISS.
2. HTS triggers UDMA to fetch data for processing. There could be max 3 independent input data.
3. UDMA loads the data and stores into SL2.
4. Upon completion of all input buffer loading (at line level) UDMA channel 'done' event triggers VISS thread start (still needs to ensure there are buffer available for writing VISS output data).
5. VISS produces line #N corresponding to loading of line #N + vertical\_latency.
6. 'Tdone' triggered to HTS.
7. HTS triggers UDMA for output buffer transfer into DDR.
8. UDMA loads data from SL2 and write into DDR.

The following assumptions are made for the memory to memory operation described above:

- CSI RX write is mapped onto hard real time fabric of NavSS/MSMC.
- Configurable vertical blanking times 'Tstart' generated by HTS to let RFE + NSF4V + GLBCE + FCP flush its internal pipe, eventhough last line of current frame is already fed into RFE.
- Buffer dependencies:
  - RFE input buffer (exp0, exp1, exp2)
  - H3A output Buffer (AE or AF)
  - FCP output Buffer (Y12, UV12, Y8, UV8, S8)

#### 6.9.4.1.4 VISS Data Formats Support

[Table 6-113](#) lists the data formats supported by VISS. VISS does not support YUV Input. In case of external YUV Sensor, CSI RX directly writes into DDR and the data is subsequently processed by the rest of VPAC.

**Table 6-113. VISS Data Formats**

Direction	Pixel Width	Pixel Container Width (M2M)	Packing	Video port (OTF)	Comments
VISS Input	RAW12/14/16	16b	No	16b / pixel	1. Companded data/External ISP 2. MSB bits are padded with zeros within fixed width pipeline
	RAW8	8b	Yes	16b / pixel	
	RAW12	12b	Yes	16b / pixel	
VISS Output	YUV420 (12b) NV12	12b	Yes	N/A	
	YUV420 (12b) NV12	16b	No	N/A	
	YUV420 (8b) NV12	8b	Yes	N/A	
	YUV422 (8b) UYVY/YUY2/NV16	8b	Yes	N/A	
	RGB (planar)	8b	Yes	N/A	With RGB output, limited YUV data outputs are available (only 12-bit YUV data is available)
	S8	8bit	Yes	N/A	

#### 6.9.4.1.5 VISS VPORT Interface

The video port (VPORT) interface is used to stream the pixel data from up stream to down stream module. In VISS, the VPORT interface is used to receive data from CSI RX (in streaming mode) and in between VISS sub-modules (RFE -> NSF4V -> GLBCE -> FCP).

The VPORT supports stall signal (VP\_STALL) and also either one pixel or 2 pixel data per cycle.

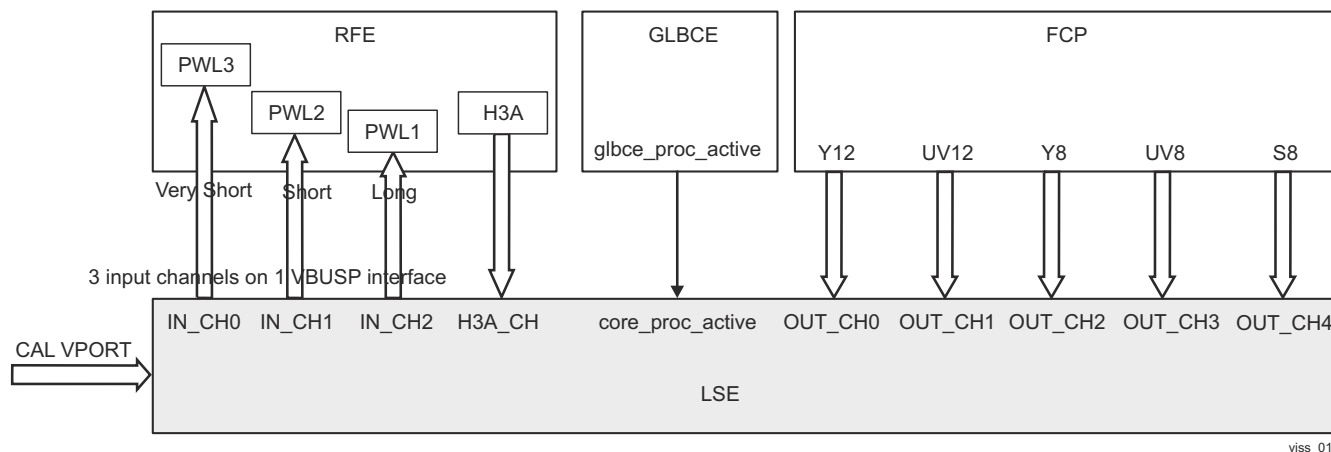
The pixel data (VP\_DATA) received on the VPORT is organized in 32 bits as follows:

- VP\_DATA[15:0] = pixel (n)
- VP\_DATA[31:16] = pixel (n+1). This is ignored, when VPORT\_TWO\_PIXEL = 0
- MSBs are padded with 0's when less than 16 bits are used.

#### 6.9.4.1.6 VISS Submodule Integration Specifics

##### 6.9.4.1.6.1 LSE Integration

Figure 6-52 captures the channel integration between LSE and rest of VISS submodules. Though digram shows three input channels between LSE to RFE, 3 channels data is transferred using single VBUSP interface.



**Figure 6-52. VISS LSE Channel Integration**

LSE supports YUV422 interleaving only on 8-bit data.

The LSE configuration for VISS can be read in VISS\_LSE\_STATUS\_PARAM register fields.

For more information on LSE module operation, see Chapter *Load and Store Engine (LSE)*.

#### **6.9.4.1.6.2 GLBCE Integration**

GLBCE is Global/Local Brightness and Contrast Enhancement module.

The GLBCE module supports the following main features:

- One pixel/cycle operation
- Up to 4096 pixels/line (line size)
- Bayer image
- When GLBCE is off, the RFE/NSF4V output is bypassed to FCP

For more information on GLBCE operation, see [Section 6.9.4.5, VISS Global/Local Brightness and Contrast Enhancement \(GLBCE\) Module](#).

##### **6.9.4.1.6.2.1 GLBCE Startup**

GLBCE requires approximately 5400 clock pulses after startup before it could receive data on its video port. Therefore, before RFE -> GLBCE -> FCP path could be used after a GLBCE reset, the SW shall use the following sequence to provide those required clock pulses:

- Prior to enabling rest of VISS pipe, configure VISS clock control to allow free running clock to GLBCE.
- Wait for the 'glbce\_filtering\_done' event. It indicates that GLBCE has received enough PCLK pulses for internal initialization and that it is now ready to receive pixels.
- Configure GLBCE clock to normal mode. Enable the rest of VISS pipeline.

GLBCE has a minimum input restriction of 480x240 pixels. The input frame must be at least this size.

GLBCE LUTs must not be changed during the active frames. They can only be updated in vertical blanking. GLBCE\_LUT\_FI may be changed frame by frame depending on the GLBCE tuning method. Other LUTs (GLBCE\_REV\_PERCEPT\_LUT\_xx, GLBCE\_FWD\_PERCEPT\_LUT\_xx, and GLBCE\_WDR\_GAMMA\_LUT\_xx) are not expected to be updated dynamically except in very limited case.

##### **6.9.4.1.6.2.2 GLBCE Bypass**

When the GLBCE is disabled (by VISS\_CNTL[0] GLBCE\_EN = 0), any access targeted to GLBCE registers or GLBCE statistics memory will respond with error status, and 'glbce\_cfg\_err' interrupt will be generated.

##### **6.9.4.1.7 VISS Stall Handling**

VISS can handle momentarily stall condition created due to lack of enough band width. Streaming mode is described in the following sections.

##### **6.9.4.1.7.1 Stall Handling for Streaming Mode**

Streaming stall handling is similar to memory to memory mode, except that final effect will be reflected in generating stall to the VPORT interface. Upon stall assertion, VISS expects no more than 2 VPORT pixel clock cycles.

##### **6.9.4.1.8 VISS Interrupts**

The VISS submodules generate the interrupt events described in *Section VPAC Subsystem Interrupts*.

##### **6.9.4.1.8.1 Interrupts Merging**

The configuration error interrupts are merged at VISS level. The following list describe the configuration error eventss and the corresponding merged interrupts.

- RAWFE\_CFG\_ERR\_INTR: Generated when configuration interface access the RFE end points, while frame processing is active. Merged interrupts from RFE:
  - lut1\_cfg\_err\_intr
  - lut2\_cfg\_err\_intr

- lut3\_cfg\_err\_intr
- wdr\_lut\_cfg\_err\_intr
- dpc\_line\_cfg\_err\_intr
- dpc\_lut\_cfg\_err\_intr
- h3a\_lut\_cfg\_err\_intr
- h3a\_line\_cfg\_err\_intr
- GLBCE\_CFG\_ERR\_INTR: Generated when the configuration interface access the GLBCE register region or statistics memory, while frame processing is active. Merged interrupts from GLBCE:
  - glbce\_statmem\_cfg\_err\_intr
  - glbce\_mmr\_cfg\_err\_intr
- FCFA\_CFG\_ERR\_INTR: Generated when the configuration interface access the CFA register region, or LUT or line memory, while frame processing is active. Merged interrupts from CFA:
  - lut\_cfg\_err\_intr
  - cfa\_mmr\_err\_intr
  - cfa\_pix\_err\_intr
- FCC\_OUTIF\_OVF\_ERR\_INTR: Generated when any of the FCC output interfaces overflows. Merged interrupts from CC:
  - overflow\_if\_y12\_intr
  - overflow\_if\_uv12\_intr
  - overflow\_if\_y8r8\_intr
  - overflow\_if\_c8g8\_intr
  - overflow\_if\_s8b8\_intr
- FCC\_CFG\_ERR\_INTR: Generated when the configuration interface access FCC register region or its LUTs, while frame processing is active. Merged interrupts from FCC:
  - contrast0\_cfg\_err\_intr
  - contrast1\_cfg\_err\_intr
  - contrast2\_cfg\_err\_intr
  - lut\_12to80\_cfg\_err\_intr
  - lut\_12to81\_cfg\_err\_intr
  - lut\_12to82\_cfg\_err\_intr
- EE\_CFG\_ERR\_INTR: Configuration happened to EE regions causing corruption during frame processing. Merged EE sources are:
  - eelut\_cfg\_err\_intr
  - ee\_pix\_err\_intr
- EE\_SYNCOVF\_ERR\_INTR:
  - ee\_hz\_align12\_intr
  - ee\_hz\_align8\_intr

#### 6.9.4.1.8.2 Handling of Configuration Error Interrupts

Table 6-114 give hints on handling configuration error interrupts.

**Table 6-114. VISS Configuration Error Interrupts Handling**

Memory	Error Generation Window	HWA-level Register Status Clearing Mechanism
	RAWFE	

**Table 6-114. VISS Configuration Error Interrupts Handling (continued)**

Memory	Error Generation Window	HWA-level Register Status Clearing Mechanism
PWL1_LUT, PWL2_LUT, PWL3_LUT	Config read access during active line OR write access during active frame.	Write '1' to corresponding bit in RAWFE_INT_STAT register.
WDR_LUT		
H3A_LUT		
H3A_ACCM		
H3A_LINE		
DPC_LUT	Config read/write access during active frame at H3A boundary.	
DPC_LINE	Config read/write access during active frame.	
LSC Table	Config read access during active line OR write access during active frame.	
	Config read/write access during active frame. Active frame in lsc case is VS at pwl to VE at lsc input delayed by 1 cycle.	
NSF4V		
NSF4V_LINE	Config read/write access when datapath is accessing the corresponding memory on the same cycle.	No register status in NSF4V.
GLBCE		
GLBCE non-shadowed registers	Config write access during active frame. Active frame window is from VS_IN to Filter done.	Write '1' to GLBCE_INT_STAT[0] MMR_CFG_ERR register bit.
GLBCE_STAT	Config read/write access during active frame. Active frame window is from VS_IN to Filter done.	Write '1' into GLBCE_INT_STAT[1] STATMEM_CFG_ERR register bit.
GLBCE_LINE	Not mapped to config.	N/A
CFA		
CFA FIR Filter registers	Config write access during active frame.	Write '1' to CFA_INT_STATUS[2] CFA_MMR_ERR register bit.
CFA_LUT	Config read/write access during active frame.	Write '1' into CFA_INT_STATUS[0] LUT_CFG_ERR register bit.
CFA_LINE	Config read/write access during active frame.	Write '1' to CFA_INT_STATUS[1] CFA_PIX_ERR register bit.
FCC		
LUT_CONTRAST	Config access (read/write) occurs during active line when the LUT is enabled.	Write '1' to corresponding bit in FCC_FLEXCC_INT_STATUS register.
HISTOGRAM	Config access has occurred to the first location but not to the last location till the start of next frame implying that full histogram was not read.	
LUT_COLOR	Config access (read/write) occurs during active line when the LUT is enabled.	
CC_LINE	No error generation logic.	N/A
EE		
EE_LINE	Config accesses during active frame.	Write '1' to EE_INT_STATUS[1] EE_PIX_ERR register bit.
LUT	Config accesses during active frame.	Write '1' to EE_INT_STATUS[0] EELUT_CFG_ERR register bit.

#### 6.9.4.1.9 VISS Error Correcting Code (ECC) Support

ECC is a mechanism for providing increased system reliability (via reduction of memory soft errors) by allowing single bit errors to be detected and corrected and double bit errors to be detected.

One or more memories within VISS are ECC protected using an ECC Memory Wrapper which implements Single Error Correction and Double Error Detection (SECDDED).

The ECC wrapper provides Single Error Detection and Correction (SED/SEC). This logic detects and corrects a single bit error (1 bit error per ECC word or per ECC data segment). For memories that contain critical and/or persistent data, automatic (immediate or delayed) write-back of the corrected data to the corresponding memory address is supported. In addition, the ECC wrapper also supports multiple options for partial word writes, such as read-modify-write or multiple ECC code segments per word.

The ECC wrapper also provides Double Error Detection (DED). This logic only detects (does not correct) double errors (2 bit errors per ECC word or per ECC data segment).

An ECC Aggregator at the VISS level consolidates the ECC configuration and status bits for all the ECC supported memories in the subsystem. It provides a single EOI-handshake based interrupt to the host (for both single and double error detections) and a standard 32-bit VBUSP interface for configuring and querying the various ECC wrappers via their ECC register set.

The following VISS memories are ECC protected:

- GLBCE statistics memory.
- RAWFE non pixel line buffer memories. See Section *VISS RAW Frond-End (RAWFE)*, for more information.
- FCP non pixel line buffer memories

For more information on the ECC Aggregator operation, see Section *ECC Aggregator* in Chapter *Safety Modules*.

#### 6.9.4.1.10 VISS Programmer's Guide

##### 6.9.4.1.10.1 VISS Initialization Sequence

VISS initialization is part of any other HWA initialization on VPAC. Prior to starting HWA initialization through HTS, VISS must be configured in below order.

1. Configure RFE registers, GLBCE registers, NSF4V registers and FCP registers:
  - The VISS\_CNTL[0] GLBCE\_EN register bit needs to be set, in order to configure the GLBCE registers.
2. Select input / output stream properties inside LSE.
3. After reset and prior to enabling LSE, apply the following configuration:
  - a. Make sure VISS\_CNTL[0] GLBCE\_EN bit is '1'.
  - b. Enable IRQ for interrupt event 'glbce\_filtering\_done'.
  - c. Set VISS\_GLBCECONFIG[0] GLBCE\_PCLKFREE register bit to '1'.
  - d. Wait for interrupt 'glbce\_filtering\_done'.
  - e. Clear VISS\_GLBCECONFIG[0] GLBCE\_PCLKFREE register bit.
4. Enable LSE input and output buffer lines.
5. Enable pipeline and schedulers associated with VISS HTS configuration (pipeline enable must happen after enabling all attached schedulers).
6. Enabling HTS will trigger init sequence which will initialize LSE, RFE and FCP. This will also trigger head of pipe for NAVSS UDMA fetch (in case of memory to memory operation). Otherwise, VISS waits for streaming data from CSI RX.
7. After initializing VISS/VPAC, configure and enable CSI RX, CSI RX PHY and camera sensor in order.
8. Once streaming data starts, it will continue till the end of frame. Blanking (set in VISS\_LSE\_CFG[31-22] VP\_HBLNK\_CNT register field) must be configured more than VISS blanking needs (see *VISS Blanking Requirements*).
9. Once a complete frame is fully written into DDR, the HTS module comes back to init state and restart from step #6. For some usecase, LSE input/out enables after each frame might be considered as well (step #5).
10. After each 'frame\_done', if the configuration needs to be changed, it must be done within vertical blanking period for not shadowed registers (refer to Sections *VISS RAW Frond-End (RAWFE)* and *VISS Flexible Color Processing (FCP) Module*, for more details).
11. At any stage, if there is a need to stop the streaming interface, the HTS must first be paused and then aborted, and then reset and re-configure the entire VISS as part or recovery from abort.

**6.9.4.1.10.2 VISS Configuration Restrictions**

Frame size restriction:

- If GLBCE is enabled in the VISS pipeline, the minimum frame required is 480x240 pixels.
- If GLBCE is disabled in the VISS pipeline, the minimum frame required is 64x16 pixels.
- Frame width and height has to be even.
- Histogram can only be enabled when the frame width is 256 pixels minimum.

LSE H3A channel enabling restriction:

- When the H3A output channel is enabled in LSE, at least one other output channel needs to be enabled.

**6.9.4.1.10.3 VISS Real-time Operating Requirements**

SW needs to configure UTC/UDMA channel registers for write out buffers.

SW needs to reconfigure RAWFE LSE LUT, if needed, and any other non-shadowed registers during vertical blanking period only. Some control bits would be shadowed, which can be configured during a complete frame.



#### 6.9.4.2 VISS Load Store Engine (LSE)

For more information on the VISS LSE module operation, see [Section 6.9.4.1.6.1](#), *LSE Integration*, and *Load Store Engine (LSE)*.

### 6.9.4.3 VISS RAW Frond-End (RAWFE)

This section describes the Vision Imaging Sub-system (VISS) RAW Frond-End (RAWFE) module in the device.

#### 6.9.4.3.1 RAWFE Overview

The RAWFE is a set of raw pixel processing functions to enhance the RAW image signal received from the sensor. The key application of the RAWFE is in the ADAS space where in it is used for both analytics as well as visual use cases.

##### 6.9.4.3.1.1 RAWFE Supported Features

Each RAWFE module implements the following features:

- Support for any 2x2 RAW data pattern.
- Support for merge up to 3 separate exposures.
- PWL (piecewise linear) and LUT based de-companding at the front end of each exposure channel
- Support for WDR merge, including 2 exposure merge, 3 exposure merge and 2 exposure hybrid (12 bit and 16 bit) merge.
- Support for companding LUT from 24 bits down to 12 bits.
  - LUTs support higher precision in lower end.
- Support for LUT as well as adaptive Defect Pixel Correction.
- Support for table based lens shading correction.
- Support for 3A statistics
  - Statistics can be generated on independent exposures or merged data after tone map.

#### 6.9.4.3.2 RAWFE Functional Description

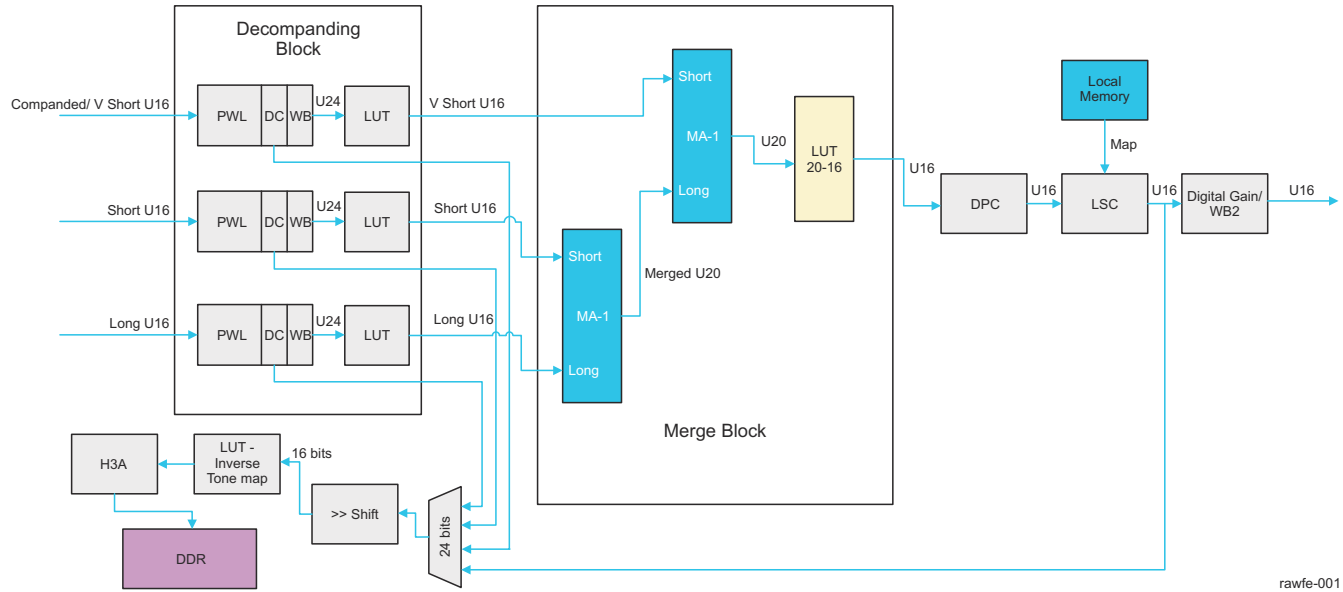
##### 6.9.4.3.2.1 RAWFE Functional Operation

The RAWFE processes captured image data from sensor and passes the data to the Flexible Color processing module (FCP) for demosaicing and color conversion.

[Figure 6-53](#) shows a high level conceptual block diagram of the RAWFE module.

The RAWFE consists of the following main components

- Decompaning block: This is used to decompand the sensor compressed raw to native bit depth. The same block can be used for tone mapping as well. The decompanding block can take sensor compressed RAW data and can decompand to up to 24 bits.
- WDR Merge: Each instance of the WDR merge block can take in 2 independent exposures (up to 16 bits) and generate up to 20 bit data.
- LSC Block: The block performs lens shading correction by applying gains stored in a look up memory.
- DPC: The block performs defect pixel correction using either LUT based memory or adaptive defect correction.
- H3A: The H3A block generates 2 different set of statistics, one for AWB and AE and the other for auto focus.



**Figure 6-53. RAWFE Block Diagram**

#### 6.9.4.3.2 RAWFE ECC for RAMs

Table 6-115 lists the RAWFE ECC RAMs

**Table 6-115. Table 3: RAM ECC capabilities**

Module	# Rams	ECC
PWL LUT1	2	ECC, with single bit correction, including writeback.
PWL LUT2	2	ECC, with single bit correction, including writeback.
PWL LUT3	2	ECC, with single bit correction, including writeback.
WDR LUT	2	ECC, with single bit correction, including writeback.
H3A LUT	2	ECC, with single bit correction, including writeback.
DPC (line buffer)	4	No ecc
LSC LUT	1	ECC, with single bit correction, including writeback.
LSC GAIN	1	ECC, with single bit correction, including writeback.
H3A accum	1	No ecc
H3A line	1	No ecc

#### 6.9.4.3.3 RAWFE Interrupts

The following section lists the RAWFE interrupts

##### 6.9.4.3.3.1 RAWFE CPU Interrupts

Table 6-116 lists the interrupts generated by the RAWFE sub-modules.

**Table 6-116. RAWFE Interrupts**

Interrupt	Description
LUT1_CFG_ERR	Config access to LUT1 ram has corrupted functional operation, config read or write memory access occurred during functional operation. Only asserted if any (read/write) cfg access occurs during active line OR write occurs in horizontal blanking within a frame.
LUT2_CFG_ERR	Config access to LUT2 ram has corrupted functional operation, config read or write memory access occurred during functional operation. Only asserted if any (read/write) cfg access occurs during active line OR write occurs in horizontal blanking within a frame.

**Table 6-116. RAWFE Interrupts (continued)**

Interrupt	Description
LUT3_CFG_ERR	Config access to LUT3 ram has corrupted functional operation, config read or write memory access occurred during functional operation. Only asserted if cfg access occurs during active line. Only asserted if any (read/write) cfg access occurs during active line OR write occurs in horizontal blanking within a frame.
WDR_LUT_CFG_ERR	Config access to WDR LUT ram has corrupted functional operation, config read or write memory access occurred during functional operation. Only asserted if any (read/write) cfg access occurs during active line OR write occurs in horizontal blanking within a frame.
H3A_LUT_CFG_ERR	Config access to H3A LUT ram has corrupted functional operation, config read or write memory access occurred during functional operation. Only asserted if any (read/write) cfg access occurs during active line OR write occurs in horizontal blanking within a frame.
H3A_ACCM_CFG_ERR	Config access to H3A accum ram has corrupted functional operation, config read or write memory access occurred during functional operation. Asserted if any cfg access occurs during active frame.
H3A_LINE_CFG_ERR	Config access to H3A line ram has corrupted functional operation, config read or write memory access occurred during functional operation. Asserted if any cfg access occurs during active frame.
DPC_LUT_CFG_ERR	Config access to DPC LUT ram has corrupted functional operation, config read or write memory access occurred during functional operation. Asserted if any cfg access occurs during active frame.
DPC_LINE_CFG_ERR	Config access to DPC line ram has corrupted functional operation, config read or write memory access occurred during functional operation. Only asserted if any cfg access occurs during active line OR write occurs in horizontal blanking within a frame.
LSC_CFG_ERR	Config access to LSC ram has corrupted functional operation, config read or write memory access occurred during functional operation. Asserted if any cfg access occurs during active frame. Active frame in lsc case is VS at pwl to ve at lsc input delayed by 1 cycle.
H3A_AEW	h3a aew interrupt.
H3A_AF	h3a af interrupt
H3A	h3a interrupt
H3A_BUF_OVRFLOW_PULSE_INTR	Generated when an overflow occurs in the H3a output buffer

#### 6.9.4.3.3.2 RAWFE Debug Events

**Table 6-117** lists the RAWFE debug events. Debug events are used for hardware debug as well as performance measurement. They are not intended to be used as interrupts where the cpu needs to process them.

**Table 6-117. RAWFE Debug events**

Event	Description
VS_EVENT	ventricle start in the beginning of the rawfe pipeline
VE_EVENT	ventricle end at the end of the rawfe pipeline
HS_EVENT	horizontal start in the beginning of the rawfe pipeline
HE_EVENT	horizontal end at the end of the rawfe pipeline
LSE_SLAVE_STALL	rawfe is stalling lse slave interface due to ext or mtc stall
LSE_MASTER_STALL	lse master interface is stalled
LSE_INTERFACE_IDLE	within a frame lse is not sending data
X_Y_EVENT_MATCH	x-y pixel position has reached the start of rawfe pipeline
DPC_OTF_CORR_EVENT	dpc of corrected a pixel position
PIPE_ADV_EVENT	active pipeline advancement

#### 6.9.4.3.4 RAWFE Sub-Modules Details

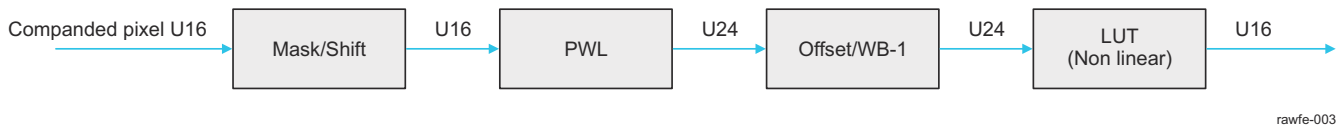
The following sections describe in detail the sub-modules present within the RAWFE module.

##### 6.9.4.3.4.1 RAWFE Decompaning Block

The decompaning block is responsible for decompaning pwl (or LUT) compressed data back to its linear bit width. The decompaning block can generate up to 24 bits of data, however the data needs to be tone mapped down to 16 bits before additional processing can take place. There are 3 instances of the Decompaning block each implementing the same functionality.

The input interface of the decompaning block is 16 bits.

Figure 6-54 shows a high level block diagram of the decompaning block.



**Figure 6-54. RAWFE Decompaning Block Diagram**

##### 6.9.4.3.4.1.1 RAWFE Mask & Shift

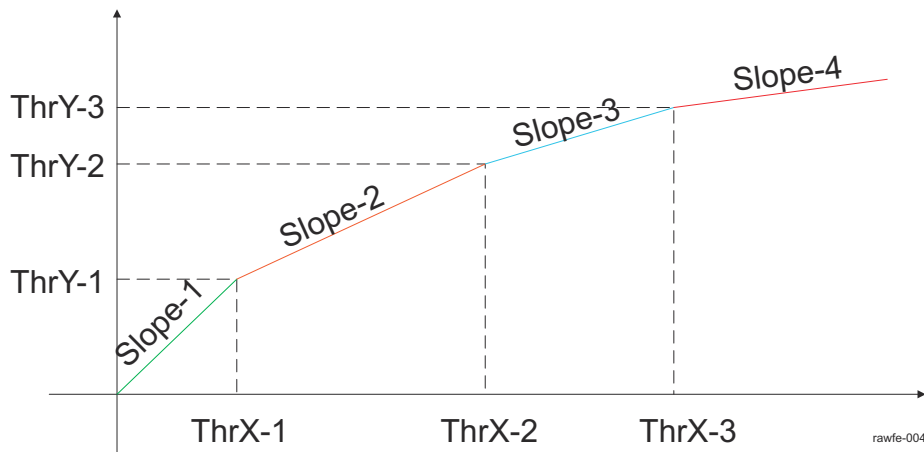
The mask and shift block is used to mask out any control bits present in the data. The 'Mask' is a 16 bit configuration register which will perform a bit wise 'and' operation on the incoming data. (For example for sensor data with data in 12 bits and 4 bits of control information, the mask value would be programmed to 0xFFF). The Mask operation is followed by a shift operation. The 'Shift' field is a 3 bit configuration register which can perform a right shift (>>) from 0 to 7 bits.

##### 6.9.4.3.4.1.2 RAWFE Piece Wise Linear Operation

The PWL block is used to apply a piece-wise-linear gain on the incoming pixel data and is generally used to uncompress sensor compressed data. The knee points of the PWL curve should be programmed to correspond with the compression knee points used by the sensor configuration.

The PWL block supports an input size from 8 – 16 bits and can support an output up to 24 bits.

Figure 6-55 depicts a PWL curve applied on incoming data

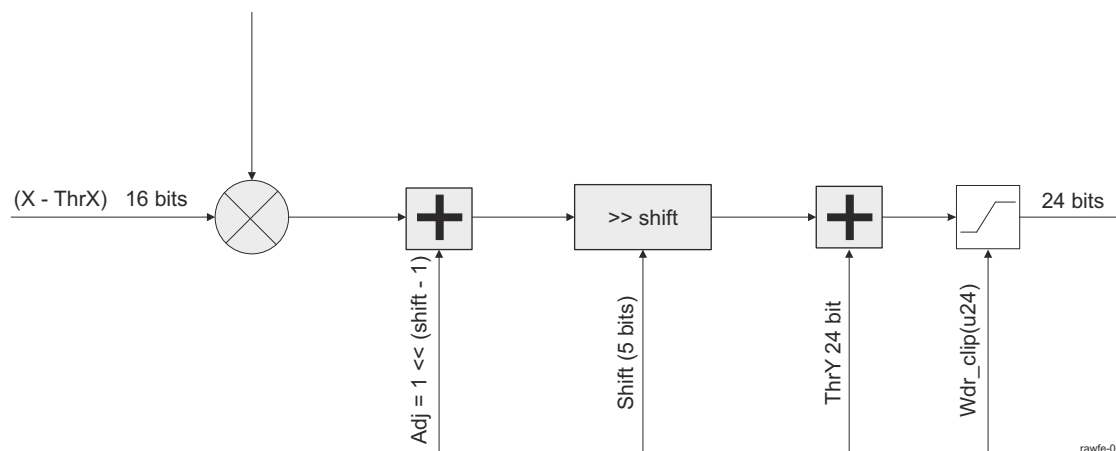


**Figure 6-55. RAWFE PWL Curve implementation**

The PWL block implements the following equation assuming the input lies in the nth segment.

$$\text{Out\_Pixel} = ((X - \text{ThrX}_n) \times \text{Slope}) + \text{ThrY}_n \quad (2)$$

The figure below shows the PWL block architecture



**Figure 6-56. RAWFE PWL Block Diagram**

The Wdr\_clip parameter can be set to clip the output at the desired bit width (typically 20 bits).

#### 6.9.4.3.4.1.3 RAWFE Offset/WB-1 Block

This block is used for applying White balance correction in linear domain as well as for subtracting the DC offset using the signed 24 bit offsets. The functionality is similar to the WB2 block. Note, there is a tap out point to H3A after the offset application, but prior to the gain block. This allows the H3A to receive DC subtracted (but not WB corrected) data.

#### 6.9.4.3.4.1.4 RAWFE LUT Based compression

The LUT based compression is used for addressing multiple use cases.

- The primary function of the LUT is to take the PWL decompounded data (up to 24 bits and after DC subtraction) and compress it using an  $x^n$  curve to a bitwidth of 16 bits. This is required since the downstream path is only capable of supporting up to 16 bits of data.
- It can be used to decompand data when the compression curve cannot be addressed using 3 knee points or if the compression is true curve instead of PWL. In this scenario the DC subtraction IP is disabled and the LUT is used to implement both the decompression as well as the DC subtraction functionality.

The LUT compression is modified from the implementation in previous generation ISPs. The previous LUTs had 512 entries leading to a step size of  $2^{20}/512 = 2048$ . As such there were only 2 LUT points in the critical low range (0-4095) of the image. The LUT implementation is optimized for a 20-bit decompounded data and allows for a step size of 32 in the critical range (lower 12 bits) and a step size of 2048 in the successive ranges.

The Non Linear LUT implementation supports 3 zones for the input range

- For bitWidth < 12 bits, the LUT offers a linear range with a step size of 32. (Only the first 128 locations (plus 129th location for supporting interpolation) are utilized in this mode)
- For bitWidths between 13 and 20: This is the most common operating zone for the LUT. In this mode, the range between 0 and 4k is split in 128 locations with a step size of 32. Beyond 4k range, the step is variable and is set to 2k for the highest bit width of 20 and subsequently reduces by half as the bitwidth is reduced (E.g. 1k for 19 bits etc)
- For bitwidths between 21 and 24: In this operation zone, the first 128 locations are used with a varying step size based on bit width such that the step size is 64 for 21 bits of data and doubles with increase in bitWidth. (E.g. 128 for 22 bits etc). The remaining locations implement a varying step size between 4k (for 21 bits) and 32k (for 24 bits).
  - Example-1: Bitwidth=21 bits:
    - Range 0-8k-> Step size = 64;
    - Range 8k-  $2^{21}$  -> Step Size = 4096
  - Example-2: Bitwidth = 22 bits

- Range 0 -16k -> Step Size = 128
- Range 16k – 2<sup>22</sup> -> Step\_Size = 8192
- Example 3: Bitwidth = 24 bits
  - Range 0 – 64K -> Step Size = 512
  - Range 64K - 2<sup>24</sup> -> Step Size = 32k

The LUT logic provides a generic table which can support any bit width and can be used in different regions in the RAWFE as well as other modules.

The LUT3 supports shadowing for very short frame only. If the MMR shadow enable is set then the LUT2 is used on the LUT3 data.

#### 6.9.4.3.4.2 RAWFE WDR Merge Block

The merge block supports merging up to 3 exposures to generate a cumulative 20 bit frame.

The merge block is based on 2 instances of the Motion-Adaptive merge block, each of which is independently capable of supporting the merging of 2 exposures (10 – 16 bits each) to a combined bit width of up to 20 bits. In a more generic scenario the merge block should be designed and verified to support the following scenarios

- Merge 2 exposures of 10 bits each to 16 bits
- Merge 2 exposures of 12 bits each to 16 bits
- Merge 2 exposures of 12 bits each to 20 bits
- Merge 2 exposures (12 bit and 16 bit) to 20 bits.
- Merge 2 exposures of 16 bit each to 20 bits.

Using two instances of the motion adaptive merge block provides the functionality to merge up to 3 independent exposures, each of 12 bits, to a combined bitwidth of 20 bits. The block also provides capability to only merge 2 exposures using a combination of mux and clipping blocks.

The merge block comprises of two instances of the motion adaptive merge (MA1/MA2) as well as an LUT based companding block and mux structures to enable different data flow. To keep the design complexity under control some restrictions are applied to the data connectivity on the merge blocks. A key restriction is that for the two exposure merge only the MA-2 block should be used and the MA-2 should be used along with MA-1 for 3 exposure merge. Further the position of the long and short exposures on each block are fixed. [Table 6-118](#) summarizes some of the possible combinations which can be realized using the logic. (Note, this is not an exhaustive list from a DV perspective).

The MA block when set to bypass/disabled mode, will pass the input stream marked as 'short' (refer to [Figure 6-53](#)) to the output.

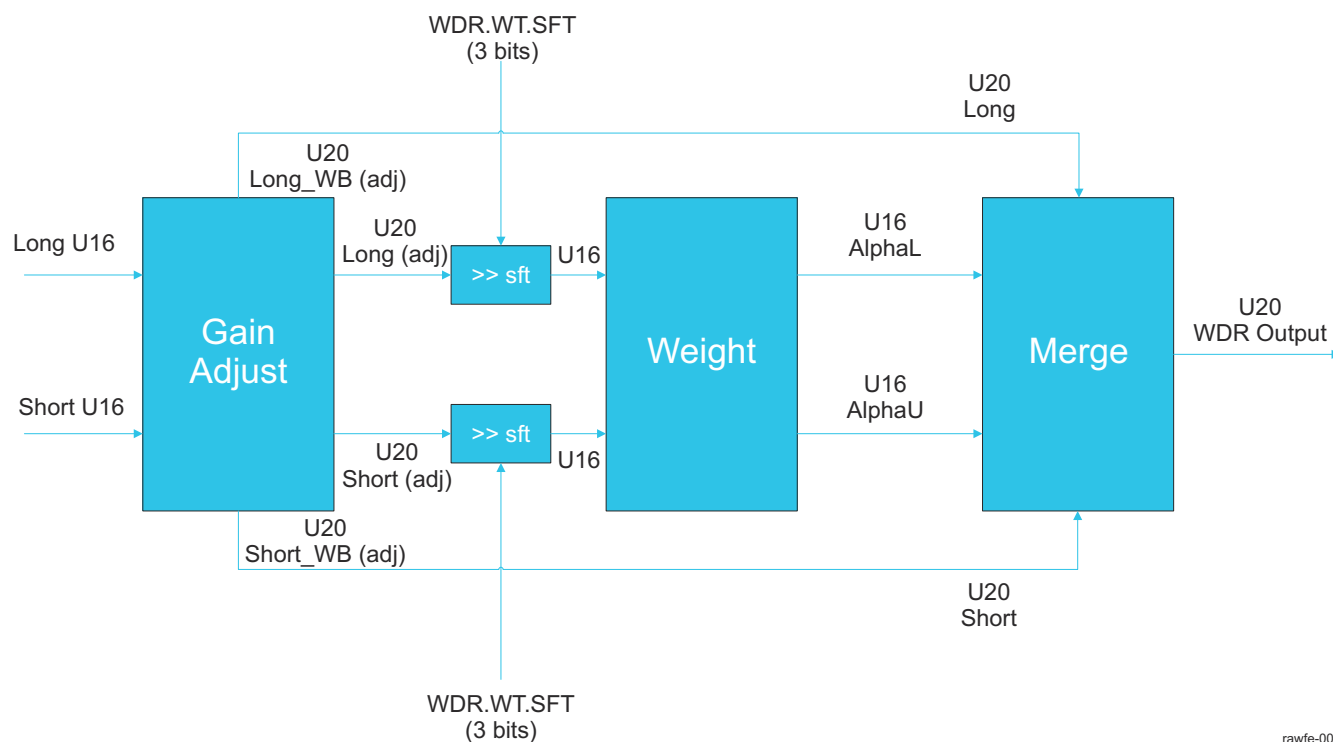
**Table 6-118. Figure 7:RAWFE Different merge modes**

Merge Mode	MA-1	MA-2	LUT
2 Exposures (12 bit) to 20 bit	Bypassed	Merge Long / Short	Use VS and Short Exposures Set to 20-16
2 Exposures (12 bit) to 16 bit	Bypassed	Merge Long /Short	Use VS and Short Exposures Bypassed
2 Exposures (12 bit) to 16 bit with tone mapping	Bypassed	Merge Long /Short	Use VS and Short Exposures Set to 16-16 mode
2 Exposures (12 & 16) to 20 bit	Bypassed	Merge Long /Short	Use VS and Short Exposures Set to 20-16
3 Exposures (12 bit each) to 20 bit	Merge Long and Short	Merge VS / MA-1 output	Use all 3 exposures Set to 20-16
Single exposure companded	Bypass	Bypass	Use VS exposure only Bypassed

#### 6.9.4.3.4.2.1 RAWFE WDR Motion Adaptive Merge (MA1 / MA2)

This section provides the detailed description of the two instances of the motion adaptive merge.

Figure 6-57 shows the high level block diagram of the MA merge block



rawfe-007

**Figure 6-57. RAWFE WDR Motion Adaptive Merge**

The MA block receives two inputs each of which can be up to 16 bits. As mentioned earlier, the ordering of the long and the short exposure cannot be changed and should be positioned as it's depicted in the figure above. The MA block consists of 3 main sub-blocks, which are described in the following sub-sections.

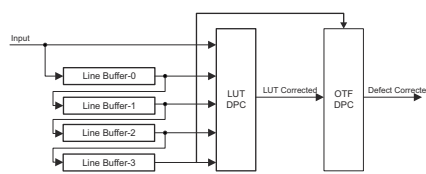
#### 6.9.4.3.4.2 RAWFE Companding LUT

The companding LUT is used to perform a global tone mapping on the image and reduce the bit depth from 20 bits down to 16 bits. The LUT is implemented as a non uniformly spaced memory, with higher precision in the lower range of the image intensity. The LUT architecture is similar to that used for the decompanding LUT in the previous section.

#### 6.9.4.3.4.3 RAWFE Defective Pixel Correction (DPC) Block

The defect pixel correction (DPC) block is responsible for correcting defective pixels present on the sensor as an artifact of the manufacturing process. The defect pixel detection can either be LUT based or on the fly (OTF – DPC). For LUT based DPC, the defect map is stored in a memory.

Figure 6-58 shows a high level block diagram of the DPC module. The DPC comprises of two different methods of performing the functionality, each of which is highlighted below. The LUT based defect correction mechanism requires the sensor to be calibrated and the defect positioned stored in memory, whereas the Adaptive DPC automatically detects defects and corrects them. If required the LUT DPC and adaptive DPC can be enabled together in sequence.



**Figure 6-58. RAWFE DPC Block Diagram**



#### 6.9.4.3.4.3.1 RAWFE LUT Based DPC

LUT based defect pixel correction provides the highest level of accuracy since all defects can be identified perfectly without any false positives. The LUT based DPC mechanism however suffers from the limitation that the sensor has to be calibrated (each sample has to be individually calibrated) and that can add cost to the testing and qualification process. Some automotive grade sensor vendors provide the defect pixel locations already in an on chip memory and that can additionally be utilized to program the defect LUT without adding the cost. The LUT size for the DPC is constrained to 256 entries, since it is assume for a sensor of 2 -3 Mpix resolution, 256 entries are more than sufficient. Each entry in the LUT is 29 bits with 13 bits for x coordinate, 13 bits for Y coordinate and 3 bits for the method. Figure 6-59 illustrates the organization of the LUT memory



rawfe-017

**Figure 6-59. RAWFE DPC LUT Memory Organization**

A sample table is provided below for 7 entries. The LUT is specified in left to right and top to down fashion (Raster Scan order). As such design only has to look at the current LUT pointer to be able to determine if the current location is a defect or not.

**Table 6-119. Figure 14: RAWFE Sample DPC Table**

10	1400	0	
236	1107	1	
236	1200	1	
488	10	1	
488	100	2	
800	138	3	
900	1000	1	
900	1100	3	

The method filed pertains to how the defect pixel is treated once it has been detected. Several possibilities arise including copying from one of the neighbors or using an average of the neighbors. The table below summarizes different methods that can be programmed for treating defect pixels.

**Table 6-120. Figure 15:RAWFE LUT DPC Method(s)**

Method Value	Functional equivalent
0	Replace with Black or white dot based on config
1	Dout = d4
2	Dout = d5
3	Dout = (d4 + d5)/2
4	Dout = (d2 + d7)/2
5	Dout = d2
6	Dout = d7
7	Dout = ((d2 + d7)/2 + (d4 + d5)/2)/2

The pixel arrangement and numbering convention is depicted in the image below with 'd0' being considered the center pixel.

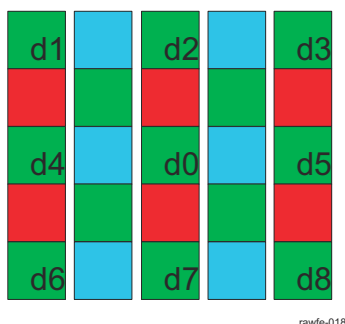


Figure 6-60. RAWFE DPC Pixel numbering convention

When the DPC method is set to '0', the output is either a black or a white pixel (depending on a separate configuration register). With this method, the LUT dpc can be used in conjunction with OTF DPC in a way that the LUT programmed defects are almost always identified by the adaptive DPC. This can be used to accurately correct for all known defects and then using adaptive DPC to correct for temperature or other operating condition based defects. (Defect pixels are strongly correlated with analog gain).

#### 6.9.4.3.4.3.2 RAWFE On-The-Fly (OTF) DPC

The LUT based DPC suffers from the drawback that each sensor die has to be calibrated for defects. The calibration process adds cost to the sensor dies, as such the preferred DPC method relies on automatically detecting the defects based on thresholds.

The OTF DPC algorithm detects defective pixels by comparing the current pixel (d0 on the left of the figure below) against its 8 neighboring pixels with the same color (d1 ~ d8 on the left of the figure below). Let's use dmax and dmin to denote the maximum and minimum of d1 ~ d8 respectively. If the current pixel d0 is greater than the sum of dmax and a detection threshold T (i.e.,  $d0 > d_{max} + T$ ) as shown on the right of the figure below, then the current pixel is considered a defect and its value is replaced by dmax. Similarly, if  $d0 < d_{min} - T$ , then the current pixel is replaced by dmin.

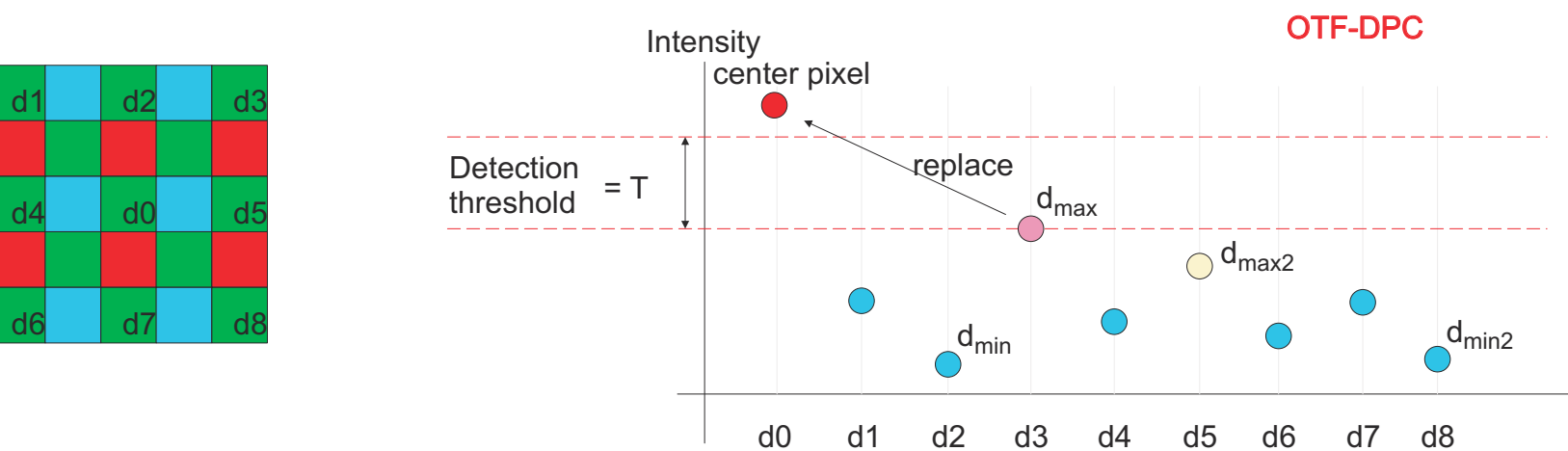
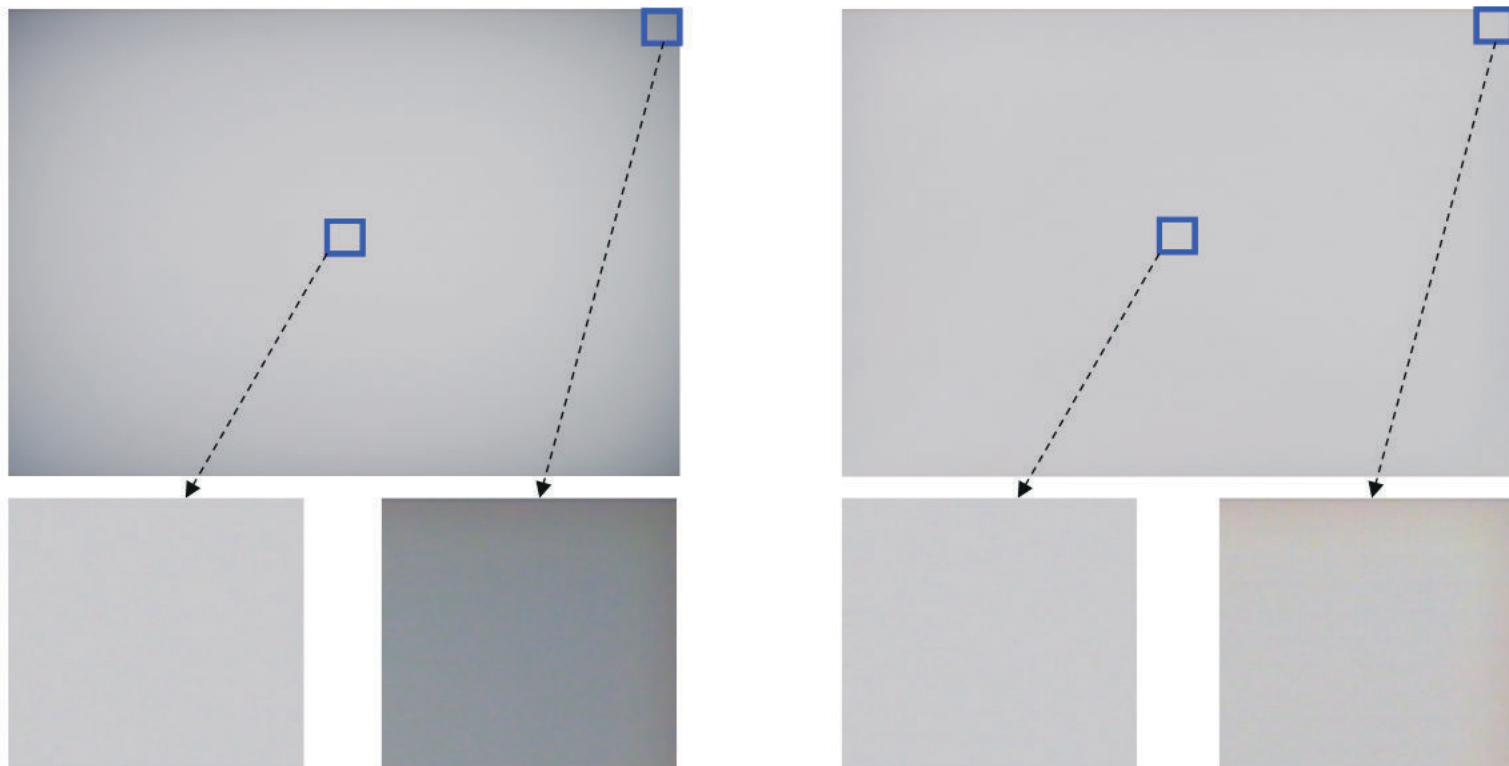


Figure 6-61. RAWFE OTF DPC

The detection threshold T is made adaptive by calculating the local image intensity and using a programmable look-up-table. The local intensity is approximated by the average of the second largest and the second smallest pixel values among d1 ~ d8 (dmax2 and dmin2 respectively in the figure above). With this average value, the threshold T is obtained from the look-up-table with linear interpolation. The look-up-table contains the detection thresholds (U16) at the average values of 0, 512, 1024, 2048, 4096, 8192, 16384, 32768 and the corresponding slope values (S12Q8) for linear interpolation.

#### 6.9.4.3.4.4 RAWFE Lens Shading Correction (LSC) and Digital Gain (DG) Block

The lens shading module corrects for the lens fall off (loss of light) at the corners of the lens. [Figure 6-62](#) shows a sample image of this phenomenon and the corresponding result after treating with LSC operation.



**Figure 6-62. RAWFE Before and After LSC**

The image on the left shows a block at the center of the lens and a corresponding block at the edge of the lens. Both blocks are equally illuminated, as such should appear equally bright on the image. However, due to lens fall off, the block at the edge appears significantly darker than the block at the center. The image on the right shows the corresponding result after LSC gains have been applied.

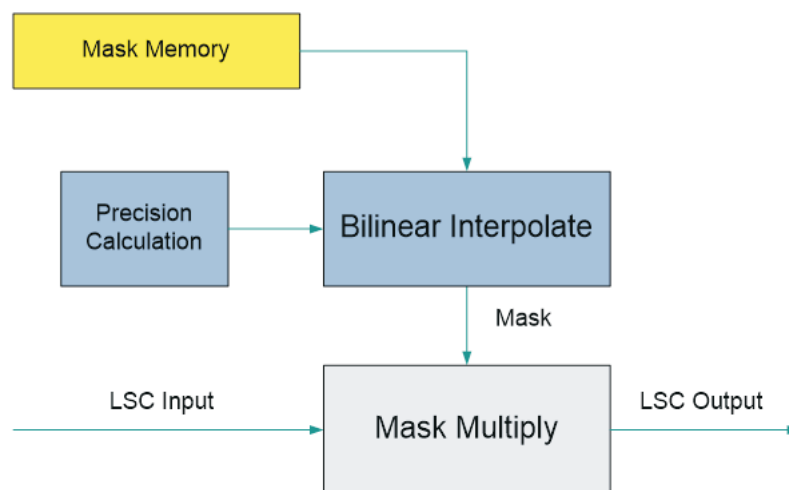
This block (2D-LSC) contains lens shading correction by multiplying an image with a gain factor 2-D map, pixel by pixel. The image is conceived to be in Bayer CFA format having a 2x2 color pattern. The gain factor map is stored in internal MEMORY down-sampled, and is accessed and up-sampled by the LSC module to pixel resolution before being applied to the pixel data. The key difference from previous generation LSC is that the gain map is stored internally in the memory instead of relying on DRAM fetch.

The LSC module does not implement any Region of Interest (ROI) functionality and the LSC gain mask is applied on the entire image. Further, only an 8 bit gain is stored in the mask. The gain can be stored in different formats depending on the range. The 8 bits of gain can be used to represent different ranges based on the `lscfg.GAIN_FORMAT` register as per the table below

- 0 : U8Q8 -> 0 to 0.996
- 1 : U8Q8 +1 -> 1 to 1.996
- 2 : U8Q7 -> 0 to 1.992
- 3 : U8Q7 +1 -> 1 to 2.992
- 4 : U8Q6 -> 0 to 2.984
- 5 : U8Q6 +1 -> 1 to 3.984
- 6 : U8Q5 -> 0 to 6.968
- 7 : U8Q5 +1 -> 1 to 7.968

### Note

The offset field is removed from the implementation of the LUT and only the gain field is retained. This is different than previous implementations.



rawfe-021

**Figure 6-63. RAWFE LSC block diagram**

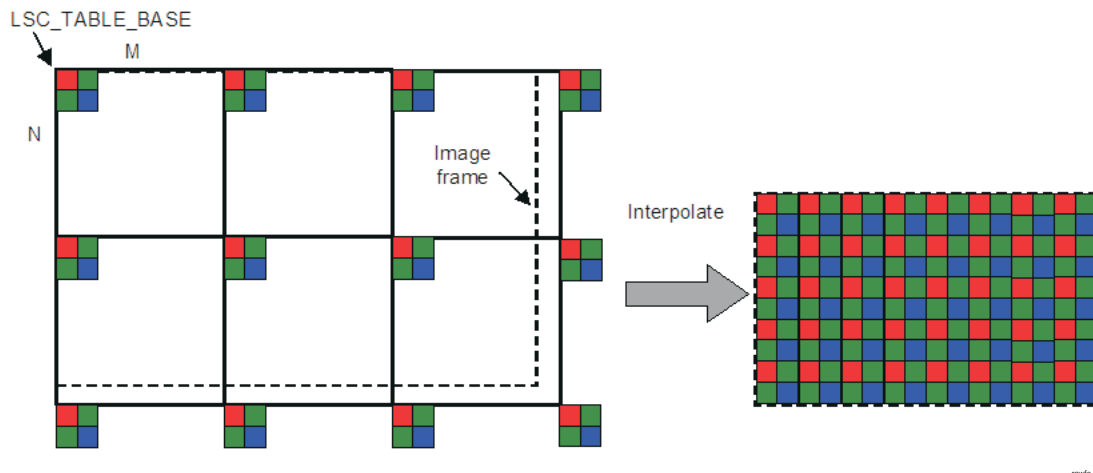
#### 6.9.4.3.4.4.1 RAWFE LSC Features Supported

- Gain map is MxN down-sampled, M being the horizontal sampling factor, N being the vertical sampling factor, M and N being {8,16,32,64,128 } independently.
- The size of the internal memory is based on a 2Mpix sensor with 16x32 downscaling ratio. As such the size is approximately 19 Kbytes-( 4758 locations with 4 Bytes/location) -> Each entry has gains for all 4 colors. This corresponds to a sensor resolution of 1928 x 1208
- Support 8-bit entry in the gain map (in U8Q8, U8Q7, U8Q6, and U8Q5 format with optional base of 1.0 to shift the range up)

Figure 6-62 illustrates the LSC active region and map up-scaling feature. (The maps are stored independently per color channel)

#### 6.9.4.3.4.4.2 RAWFE LSC Image Framing with Respect to Gain Map Samples

The gain maps are stored MxM downsampled, and needs to be upsampled by LSC hardware before being applied to the image. It is most straightforward for LSC hardware if the gain map grid is aligned with the input image grid. In other words, first pixel corresponds to a source gain map entry, rather than being upsampled through interpolation. Figure 6-64 shows the LSC active region with respect to the gain map grid.



**Figure 6-64. RAWFE LSC active region with respect to gain map samples**

For an image frame size of  $W \times H$  (output and input are the same size), gain map being  $M \times N$  downsampled, it needs  $(\text{ceil}(W / M) + 1) \times (\text{ceil}(H / N) + 1) \times 4$  bytes of gain map data in internal memory. (NOTE:  $\text{ceil}()$  represents the ceiling function).

The LSC module is designed to work with Bayer CFA data, having R/Gr/Gb/B color pattern. For the purpose of functional description, we assume Red is the starting color, but any other starting color or other  $2 \times 2$  color pattern can be used by placing color gains in the appropriate order.

Each  $2 \times 2$  set of samples is stored together in a 32-bit word in internal memory. For R/Gr/Gb/B CFA data and  $8 \times 8$  downsampled gains, the following order is assumed in the gain map, using conventional (X, Y) coordinate notation (X for horizontal offset and Y for vertical offset):

Line 0: R(0,0) Gr(1,0) Gb(0,1) B(1,1) R(8,0) Gr(9,0) Gb(8,1) B(9,1)...

Line 1: R(0,8) Gr(1,8) Gb(0,9) B(1,9) R(8,8) Gr(9,8) Gb(8,9) B(9,9)...

#### 6.9.4.3.4.5 RAWFE Gain & Offset Block

The gain block allow 2 primary functions

- Allow for digital gain to be applied if the image is too dark even after analog gain and exposure time have been set to the maximum
- Independent gain setting for each color channel in 13 bit format U13Q9 with 9 bits of fraction.
  - Allows a gain value from 0 to 31.996 in steps of  $1/256$ .

The Gain block can be configured to apply an offset on top of the gain. This is to support the log compressed image mode, where in the traditional WB Gain becomes an offset.

#### 6.9.4.3.4.6 RAWFE H3A

##### 6.9.4.3.4.6.1 RAWFE H3A Overview

The H3A module supports the control loops for autofocus, auto white balance, and auto exposure by collecting statistics about the raw image. The metrics are used to adjust parameters for processing the imaging/video data. There are two main blocks in the H3A module:

- Autofocus (AF) engine:

The AF submodule extracts and filters the red, green, and blue data from input image data and provides the accumulation or peaks of the data in a specified region. The specified region is a 2D block of data referred to as a paxel. The AF engine supports the following features:

- Peak mode in a paxel: Accumulation of the maximum focus value (FV) of each line in a paxel
- Accumulation mode in a paxel

- Accumulation of horizontal and vertical focus value in a paxel
- Up to 12 paxels in the horizontal direction and up to 12 paxels in the vertical direction with vertical focus
- Up to 36 paxels in the horizontal direction and up to 128 paxels in the vertical direction with horizontal focus only
- Programmable width and height for the paxel/window
- Programmable red, green, and blue position within a  $2 \times 2$  matrix
- Separate horizontal start for paxel and filtering
- Programmable vertical and horizontal line increments within a paxel
- Horizontal FV uses parallel infinite impulse response (IIR) filters configured in a dual-biquad configuration with individual coefficients (two filters with 11 coefficients each). The filters are intended to compute the sharpness/peaks in the frame on which to focus.
- Vertical FV uses a 5-tap FIR filter with 8-bit coefficients. With horizontal steps each paxel has up to 32 columns to be maintained for vertical FV calculation.

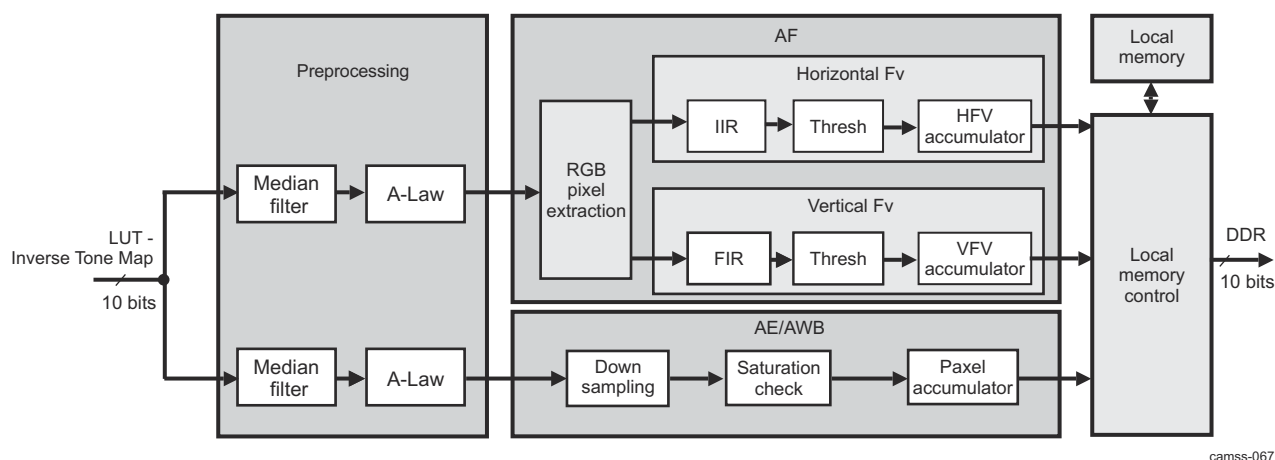
• Auto exposure and auto white balance (AE/AWB) engine:

The AE/AWB engine accumulates values and checks for saturated values in a subsampling of the video data. In the case of the AE/AWB, the 2D block of data is referred to as a window. Thus, other than having different names, paxels and windows are essentially the same. However, the numbers, dimensions, and starting positions of AF paxels and AE/AWB windows are programmable separately. AE/AWB supports the following features:

- Accumulate clipped pixels along with all nonsaturated pixels in each window per color
- Accumulate the sum of squared pixels in each window per color
- Minimum and maximum pixel values in each window per color
- Support for up to 36 horizontal windows with sum + { sum\_sq or min+max } output
- Support for up to 56 horizontal windows with sum output
- Support for up to 128 vertical windows
- Programmable width and height for the windows. All windows in the frame are the same size.
- Separate vertical start coordinate and height for a black row of paxels that is different than the remaining color paxels
- Programmable horizontal sampling points in a window
- Programmable vertical sampling points in a window
- Double-buffer for paxel/window accumulation
- H3A data path is 10 bits.
- Maximum input size is 4096 pixels.

#### 6.9.4.3.4.6.2 RAWFE H3A Top-Level Block Diagram

The block diagram in Figure 6-65 shows the process of the AF and AE/AWB data paths through the H3A module.



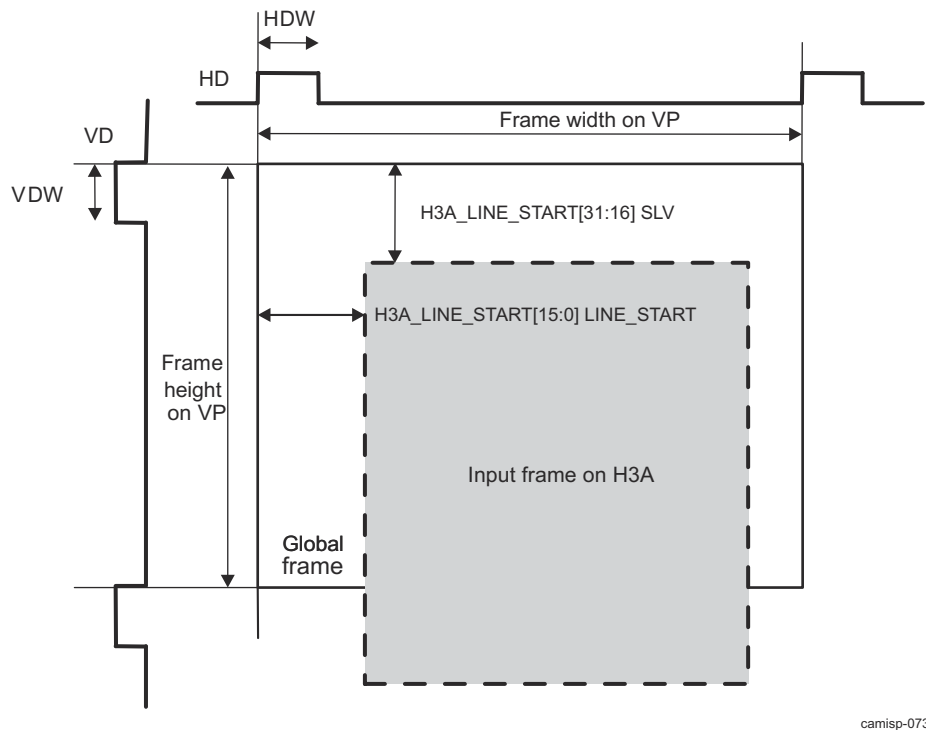
camss-067

**Figure 6-65. RAWFE H3A Top-Level Block Diagram**

#### 6.9.4.3.4.6.3 RAWFE H3A Line Framing Logic

In certain cases the number of clock cycles between HD pulses is greater than the line buffer included in the H3A. To solve this problem a framing module was added before the line buffer. The framing module uses the VISS\_RAWFE\_H3A\_LINE\_START register to find the position of the first pixel to place into the line buffer. All other registers reference this point as the 0 pixel for their start positions. The line size is 4096 pixels. After 4096 clock cycles the framing logic disables the line buffer and waits until the next HD. If the next HD comes before 4096 clock cycles, then the active region ends immediately and the counter waits for the VISS\_RAWFE\_H3A\_LINE\_START register count to be reached again. For the vertical position the VISS\_RAWFE\_H3A\_LINE\_START[31:16] SLV bit field can be used to determine where the start point of the frame is relative to the rising edge of VD. This logic allows for an active frame to cross VD boundaries and remain in the same frame.

Figure 6-66 shows the RAWFE H3A frame format settings.



**Figure 6-66. RAWFE H3A Frame Format Settings**

#### Note

(Frame width on VP) - (VISS\_RAWFE\_H3A\_LINE\_START[15:0] LINE\_START) must be less than or equal to 4096, because the H3A memory lines are limited to 4096 pixels.

#### 6.9.4.3.4.6.4 RAWFE H3A Optional Preprocessing

The input to the H3A module is 10-bit RAW data from the IPIPEIF. A 10-bit to 8-bit A-Law compression step can be enabled and disabled separately for the AF engine (the VISS\_RAWFE\_H3A\_PCR[1] AF\_ALAW\_EN bit) and the AE/AWB engine (the VISS\_RAWFE\_H3A\_PCR[17] AEW\_ALAW\_EN bit). A-Law compression offers added protection against overflowing the accumulators.

If the A-Law table is enabled, the output is 10 bits, with the upper two bits filled with 0.

For the AF process, a horizontal median filter can be enabled and disabled (the VISS\_RAWFE\_H3A\_PCR[2] AF\_MED\_EN bit) before A-Law compression. This filter is useful for reducing temperature-induced noise. The

horizontal median filter calculates the absolute difference between the current pixel (i) and pixel (i – 2), and between the current pixel (i) and pixel (i + 2). If the absolute difference exceeds a threshold, and the sign of the differences is the same, the average of pixel (i – 2) and pixel (i + 2) replaces pixel (i). The threshold of the horizontal median filter can be set in the VISS\_RAWFE\_H3A\_PCR[10:3] MED\_TH bit field.

#### **6.9.4.3.4.6.5 RAWFE H3A Autofocus Engine**

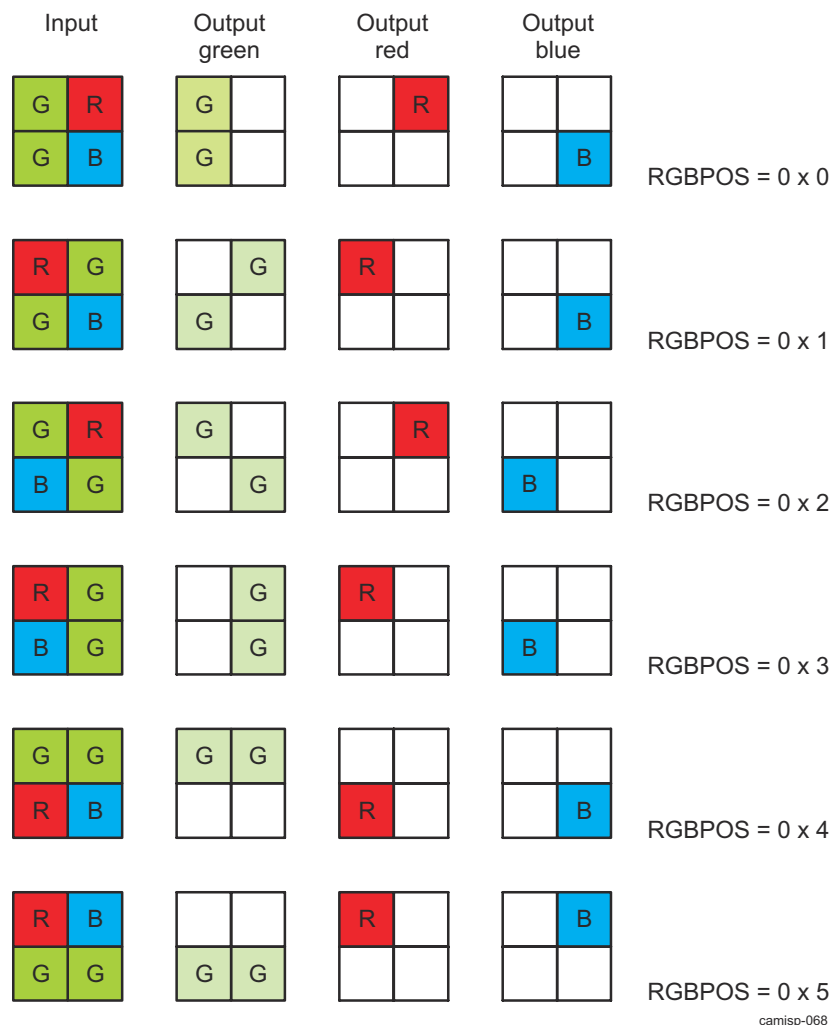
The AF engine works by extracting each green (Gr or Gb) pixel from the video stream and subtracts a fixed offset of 128 or 512 (depending of whether A-Law is enabled or disabled) from the pixel value. The offset value is then passed through an IIR filter and the absolute value of the filter output is the focus value (FV). Both FV and FV<sup>2</sup> are produced. The FV and FV<sup>2</sup> values can be accumulated or the maximum for each line/column can be accumulated. The following sections describe this process in more detail.

##### **6.9.4.3.4.6.5.1 RAWFE H3A Paxel Extraction**

From the paxel starting coordinate (the VISS\_RAWFE\_H3A\_AFPAXSTART[27:16] PAXSH and VISS\_RAWFE\_H3A\_AFPAXSTART[11:0] PAXSV bit fields) specifies the starting point of the paxel grid, with respect to first pixel of the input image frame.

The paxel starting coordinate also indicates which color pixels are extracted if VF is enabled (that is, if VISS\_RAWFE\_H3A\_PCR[20] AF\_VF\_EN = 1). Normally, either Gr or Gb is used for AF, but it is not important to the hardware whether it is red, green, or blue. If VF is not enabled, then the red, green, and blue pixel extraction is controlled by the VISS\_RAWFE\_H3A\_PCR[13:11] RGBPOS bit field to extract the correct colors from the input stream. [Figure 6-67](#) shows the available options for this bit field. The red and blue pixel positions are interchangeable. For each 2 × 2 grid, the green pixels are summed to create a single value. Because of this, the amplitude of the green output contains 2 pixels, while the red and blue outputs each contain 1 pixel.

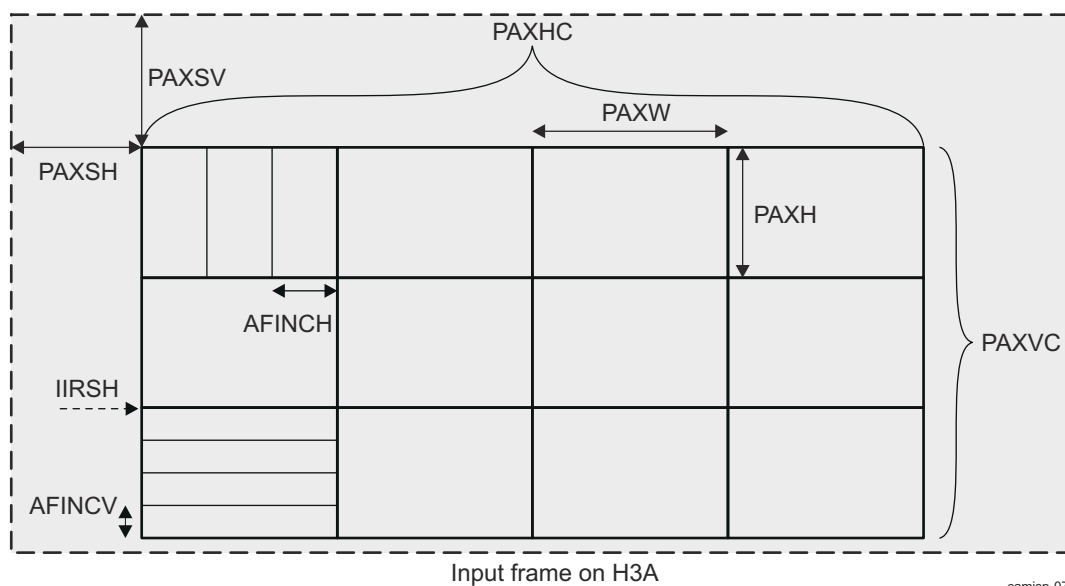




**Figure 6-67. RAWFE H3A Red, Green, and Blue Pixel Extraction Examples**

Each paxel is VISS\_RAWFE\_H3A\_AFPAX1[23:16] PAXW × VISS\_RAWFE\_H3A\_AFPAX1[7:0] PAXH (width × height) pixels. Inside each paxel, horizontal FV can skip lines, operating on one every VISS\_RAWFE\_H3A\_AFPAX2[16:13] AFINCV lines. Vertical FV can skip columns, operating on one every VISS\_RAWFE\_H3A\_AFPAX2[20:17] AFINCH columns. Up to 32 columns are supported for each paxel. If floor (PAXW/AFINCH) ≥ 32, only the first 32 designated columns are operated on. Because PAXW, PAXH, AFINCV, and AFINCH are all even numbers, AF always operates on the same green color, Gr or Gb. IIR filters for the horizontal FVs start operation at column VISS\_RAWFE\_H3A\_AFIIRSH[11:0] IIRSH.

Figure 6-68 shows the RAWFE H3A horizontal/vertical fv paxel configuration.



**Figure 6-68. RAWFE H3A Horizontal/Vertical FV Poxel Configuration**

**Note**

$$(\text{VISS\_RAWFE\_H3A\_AFPAXSTART}[27:16] \text{ PAXSH}) + (\text{VISS\_RAWFE\_H3A\_AFPAX2}[5:0] \text{ PAXHC}) \\ \times (\text{VISS\_RAWFE\_H3A\_AFPAX1}[23:16] \text{ PAXW}) \leq [(\text{Frame width on VP}) - \\ (\text{VISS\_RAWFE\_H3A\_LINE\_START}[15:0] \text{ LINE\_START})] \leq 4096$$

Table 6-121 lists the bit fields that configure the size and number of paxels.

**Table 6-121. RAWFE H3A Poxel Register Field Descriptions**

Bit Field	Bit Width	Description
VISS_RAWFE_H3A_AFPAX1[23:16] PAXW	8	Poxel width (in pixels)
VISS_RAWFE_H3A_AFPAX1[7:0] PAXH	8	Poxel height (in lines)
VISS_RAWFE_H3A_AFPAX2[5:0] PAXHC	6	Poxel count for horizontal direction
VISS_RAWFE_H3A_AFPAX2[12:6] PAXVC	7	Poxel count for vertical direction
VISS_RAWFE_H3A_AFPAX2[16:13] AFINCV	4	Line increments in a poxel
VISS_RAWFE_H3A_AFPAX2[20:17] AFINCH	4	Column increments in a poxel
VISS_RAWFE_H3A_AFPAXSTART[27:16] PAXSH	12	Poxel start position H
VISS_RAWFE_H3A_AFPAXSTART[11:0] PAXSV	12	Poxel start position V
VISS_RAWFE_H3A_AFIIRSH[11:0] IIRSH	12	IIR filter start position

The H3A AF engine also has an option for an advanced or normal stats collection mode. When 0xCA00 is written to the VISS\_RAWFE\_H3A\_ADVANCED[31:15] ID bit field, then VISS\_RAWFE\_H3A\_ADVANCED[0] AF\_MODE can be used to toggle between normal and advanced AF stats collection mode. When the advanced AF stats collection mode is enabled, the ZEROS section of the AF poxel packet is filled with the sum of the maximum FVs, regardless of the color, from HFV\_1 and HFV\_2.

#### 6.9.4.3.4.6.5.2 RAWFE H3A Horizontal FV Calculator

The FV calculator takes the unsigned red/green/blue extracted data and subtracts 128 or 512 (depending on whether A-Law is enabled) to place the data in the range –128 to 127 or –512 to 511.

After removing the offset, the data is sent through two parallel IIR filters configured in a dual-biquad configuration. Each filter uses a unique set of 11 programmable coefficients. Each coefficient is 12-bits-wide

with 6 bits of decimal, S12Q6 (VISS\_RAWFE\_H3A\_AFCOEF010 to VISS\_RAWFE\_H3A\_AFCOEF0010 for SET0, and VISS\_RAWFE\_H3A\_AFCOEF110 to VISS\_RAWFE\_H3A\_AFCOEF1010 for SET1). The filter-shift registers are cleared on each horizontal line at the position set by the register IIR horizontal start register (the VISS\_RAWFE\_H3A\_AFIIRSH[11:0] IIRSH bit field). The absolute values of the output (16 bits wide with 4 bits of decimal, U16Q4) of both filters are then sent to the AF accumulator module. Signed clipping is performed during the FV calculation. If the input value is  $m$  bits (signed) and the required output value is  $n$  bits, clipping transforms the input to between  $-2^1$  and  $2^1$ . Values lower than  $-2^1$  are set to  $-2^1$ , and values higher than  $2^1$  are set to  $2^1$ .

#### 6.9.4.3.4.6.5.3 RAWFE H3A HFV Accumulator

The horizontal focus value (HFV) accumulator takes the output of the horizontal IIR filter and accumulates values for each paxel. The size and number of paxels is configurable by registers.

Table 6-121 lists the register fields that configure the size and number of paxels:

- In peak mode (VISS\_RAWFE\_H3A\_PCR[14] FVMODE = 0x1), the maximum value is accumulated.
- In sum mode (VISS\_RAWFE\_H3A\_PCR[14] FVMODE = 0x0), all HFV<sub>n</sub> are accumulated in a paxel.

The following equations detail the calculation for:

- Sum of pixel values used in HFV: The pixel values that are used for filtering and accumulation of HFV are also accumulated in this sum of pixel values.
- HFV<sub>n</sub> (HFV<sub>n\_peak</sub> for peak mode or HFV<sub>n\_sum</sub> for sum mode)
- HFV\_count<sub>n</sub>
- HFV\_sq<sub>n</sub> (HFV\_sq<sub>n\_peak</sub> for peak mode or HFV\_sq<sub>n\_sum</sub> for sum mode)

$n = 1$  or  $2$  for IIR1 and IIR2, respectively.

For each paxel, these six values are available for each R, G, and B component.

```
for (k=0; k<PAXH) // Loop on paxel rows
{
    rowpeak_n = 0;
    for (l=0; l<PAXW; l++) // Loop on values within a row
    {
        aIIRout_n = ABS(IIRout_n);
        if (aIIRout_n >= threshold_n)
        {
            hfval = aIIRout_n - threshold_n;
            HFV_count_n++;
        }
        else hfval = 0;
        if (hfval > rowpeak_n)
        {
            rowpeak_n = HFV_n;
        }
        HFV_n_sum += hfval;
        HFV_sq_n_sum += (hfval* hfval + RNDADD) >> RNDSHIFT;
    } // Finished looping on values in a row
    HFV_n_peak += rowpeak_n;
    HFV_sq_n_peak += (rowpeak_n * rowpeak_n + RNDADD) >> RNDSHIFT;
}
```

- threshold<sub>n</sub> is VISS\_RAWFE\_H3A\_HVF\_THR[15:0] HTHR1 and VISS\_RAWFE\_H3A\_HVF\_THR[31:16] HTHR2, respectively.
- IIRout<sub>n</sub> is the IIRout<sub>1</sub> and IIRout<sub>2</sub> outputs, respectively.
- HFV\_count<sub>n</sub> and HFV\_sq<sub>n</sub> are not sent to the DMA interface if VF is disabled.
- RNDADD and RNDSHIFT depend on whether input pixels are 8-bit or 10-bit, and achieves rounding. This is automatically performed by the module.
- If VF is enabled, only the green color channel values are output to the DMA interface.
- In sum mode:
  - HFV<sub>n</sub> = HFV<sub>n\_sum</sub>
  - HFV\_sq<sub>n</sub> = HFV\_sq<sub>n\_sum</sub>
- In peak mode:

- HFV\_n = HFV\_n\_peak
- HFV\_sq\_n = HFV\_sq\_n\_peak

#### 6.9.4.3.4.6.5.4 RAWFE H3A VFV Calculator

The VFV calculator takes the unsigned extracted data through two FIR filters, each with a set of five coefficients (VCOEF1\_x, where x = 0 to 4, in the VISS\_RAWFE\_H3A\_VFV\_CFG1 and VISS\_RAWFE\_H3A\_VFV\_CFG2 registers for FIR 1, and VCOEF2\_x, where x = 0 to 4, in the VISS\_RAWFE\_H3A\_VFV\_CFG3 and VISS\_RAWFE\_H3A\_VFV\_CFG4 registers). Each coefficient is 8 bits wide with 4 bits of decimal (S8Q4). The filter outcome is downshifted by 4 bits and takes an absolute value to produce a 16-bit unsigned value. This is then sent to threshold VISS\_RAWFE\_H3A\_VFV\_CFG2[31:16] VTHR1 for FIR 1, and VISS\_RAWFE\_H3A\_VFV\_CFG4[31:16] VTHR2 for FIR 2, and square logic to produce VFV\_n and VFV\_sq\_n.

#### 6.9.4.3.4.6.5.5 RAWFE H3A VFV Accumulator

The VFV accumulator takes the output of the vertical FIR filters and accumulates values for each paxel. The size and number of paxels is configurable by registers.

[Table 6-121](#) lists the bitfields that configure the size and number of paxels.

The following equations detail the calculation for:

- VFV\_n
- VFV\_count\_n
- VFV\_sq\_n

n = 1 or 2 for FIR1 and FIR2, respectively.

For each paxel, these six values are available for each R, G, and B component.

```
FIR_coef_n = [VCOEFn_0, VCOEFn_1, VCOEFn_2, VCOEFn_3, VCOEFn_4]; /* coefficient values in S8.4
format */
aFIRout_n = (ABS(inner_product(extracted_G, FIR_coef_n)) + 8) >> 4;
if (aFIRout_n >= threshold_n)
{
    VFV_n = aFIRout_n - threshold_n
    VFV_count_n++;
}
else VFV_n = 0;
VFV_sq_n = (VFV_n * VFV_n + RNDADD) >> RNDSHIFT;
```

- threshold\_n is VISS\_RAWFE\_H3A\_VFV\_CFG2[31:16] VTHR1 and VISS\_RAWFE\_H3A\_VFV\_CFG4[31:16] VTHR2, respectively.
- FIRout\_n is the FIRout\_1 and FIRout\_2 outputs, respectively.
- RNDADD and RNDSHIFT depend on whether the input pixels are 8-bit or 10-bit, and achieves rounding. This is automatically performed by the module.

#### 6.9.4.3.4.6.6 RAWFE H3A AE/AWB Engine

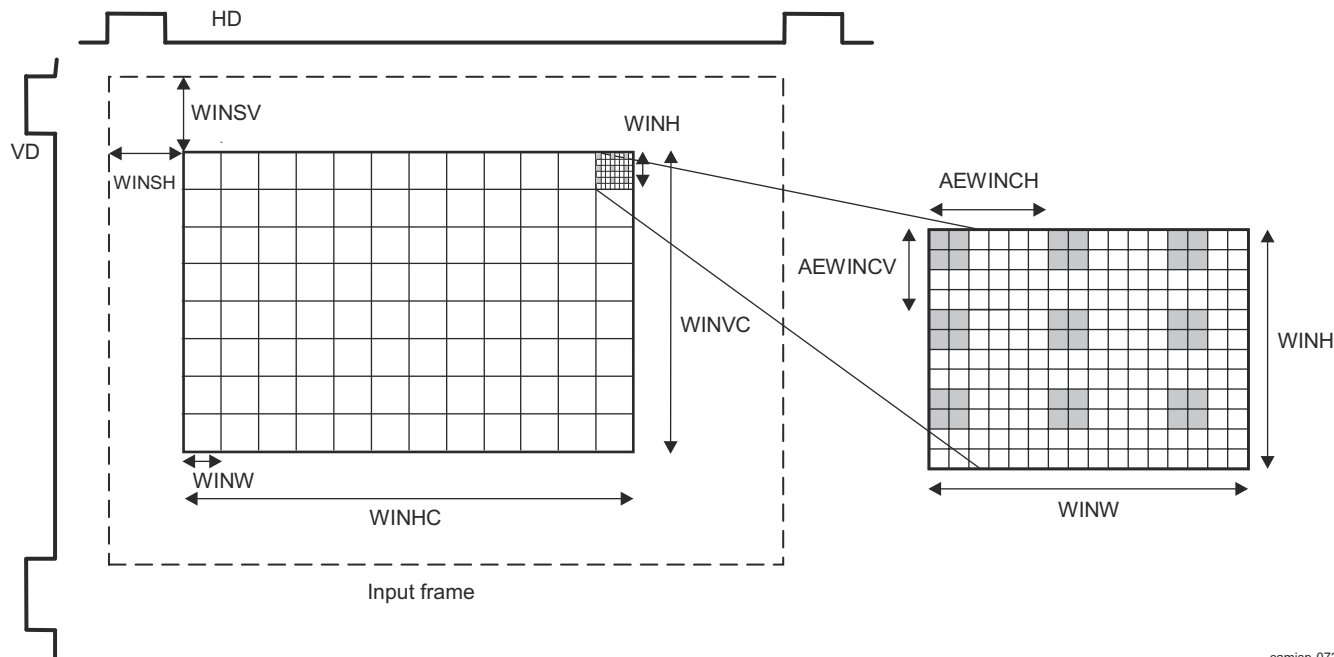
The AE/AWB engine starts by dividing the frames into windows, and then subsamples each window into 2 × 2 blocks. For each subsampled 2 × 2 block, each pixel is accumulated. Also, each pixel is compared to a limit set in a register. If any pixels in a 2 × 2 block are greater than or equal to the limit, the block is not counted in the unsaturated block counter. Pixels greater than the limit are replaced by the limit, and the value of the pixel is accumulated.

The AE/AWB module has three output format modes, which are set through the VISS\_RAWFE\_H3A\_AEWCFG[9:8] AEFMT bit field:

- Sum of square mode: VISS\_RAWFE\_H3A\_AEWCFG[9:8] AEFMT = 0x0
- Min/max mode: VISS\_RAWFE\_H3A\_AEWCFG[9:8] AEFMT = 0x1
- Sum-only mode: VISS\_RAWFE\_H3A\_AEWCFG[9:8] AEFMT = 0x2

#### 6.9.4.3.4.6.1 RAWFE H3A Sub\_sampler

The subsampler partitions the frame into windows using the size, count, and starting location parameters shown on the left in Figure 6-69. Each window is further sampled down to a set of  $2 \times 2$  blocks. The horizontal and vertical distances between the start of blocks within a window is programmable using the parameters shown on the right in Figure 6-69.



camisp-072

**Figure 6-69. RAWFE H3A AE/AWB Window Configurations**

Table 6-122 lists the bit fields that configure the window and block sizes, counts, and starting positions.

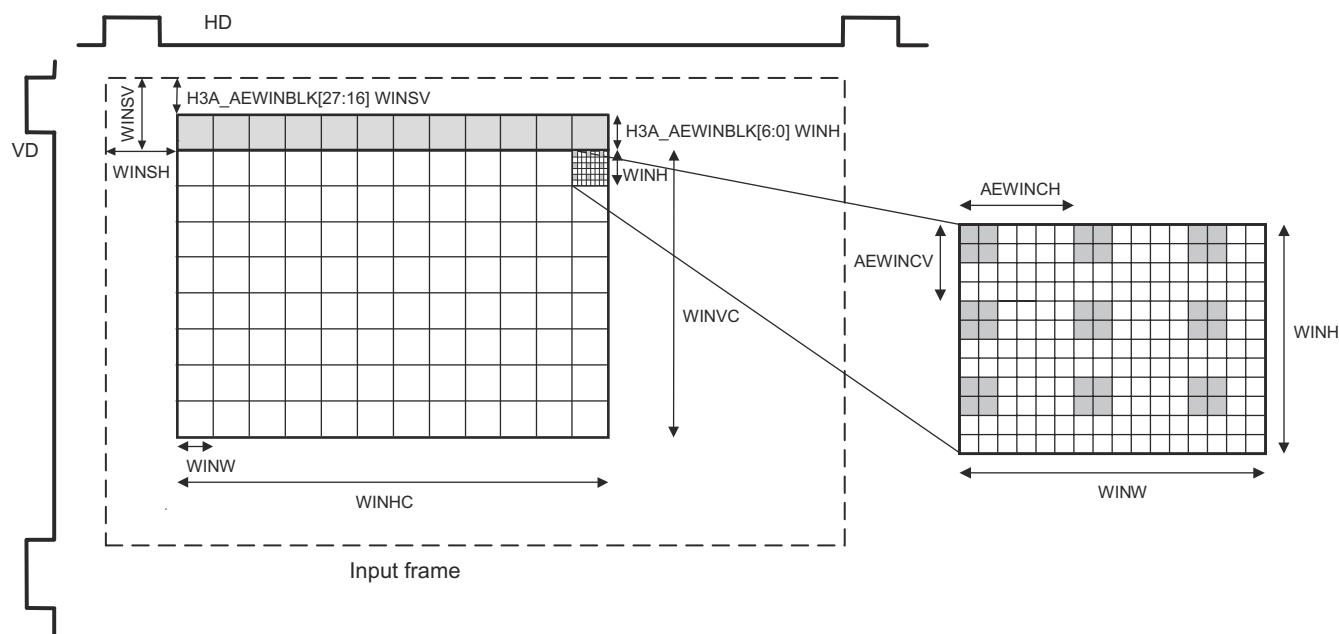
**Table 6-122. RAWFE H3A AE/AWB Window Register Field Descriptions**

Bit Field	Bit Width	Description
VISS_RAWFE_H3A_AEWWIN1[20:13] WINW	7	Window width (in pixels)
VISS_RAWFE_H3A_AEWWIN1[31:24] WINH	7	Window height (in lines)
VISS_RAWFE_H3A_AEWWIN1[5:0] WINHC	6	Window count for horizontal direction
VISS_RAWFE_H3A_AEWWIN1[12:6] WINVC	7	Window count for vertical direction
VISS_RAWFE_H3A_AEWINSTART[11:0] WINSH	12	Window start position H
VISS_RAWFE_H3A_AEWINSTART[27:16] WINSV	12	Window start position V
VISS_RAWFE_H3A_AEWSUBWIN[3:0] AEWINCH	4	Horizontal distance between subsamples
VISS_RAWFE_H3A_AEWSUBWIN[11:8] AEWINCV	4	Vertical distance between subsamples

#### 6.9.4.3.4.6.2 RAWFE H3A Additional Black Row of AE/AWB Windows

In addition to the 128 rows of windows, the AE/AWB module provides support for an additional row of windows for black data. This data may be useful in determining the DC offset noise of the rest of the data. The black row of windows can be before or after the regular rows of windows. The vertical start line for the black row of windows is specified in the VISS\_RAWFE\_H3A\_AEWINBLK[27:16] WINSV bit field, and the height is specified in the VISS\_RAWFE\_H3A\_AEWINBLK[6:0] WINH bit field. The horizontal starting pixel and horizontal width of the black row of windows are the same as for the regular rows of windows.

Figure 6-70 shows a black row of windows before rows of windows.



camisp-075

**Figure 6-70. RAWFE H3A Black Row of Windows Before Regular Rows of Windows**

Table 6-123 lists the bit fields that configure the window and block sizes, counts, and starting positions.

**Table 6-123. RAWFE H3A AE/AWB Window With Additional Black Row Register Field Descriptions**

Bit Field	Bit Width	Description
VISS_RAWFE_H3A_AEWIN1[20:13] WINW	7	Window width (in pixels)
VISS_RAWFE_H3A_AEWIN1[31:24] WINH	7	Window height (in lines)
VISS_RAWFE_H3A_AEWIN1[5:0] WINHC	6	Window count for horizontal direction
VISS_RAWFE_H3A_AEWIN1[12:6] WINVC	7	Window count for vertical direction
VISS_RAWFE_H3A_AEWINSTART[11:0] WINSH	12	Window start position H
VISS_RAWFE_H3A_AEWINSTART[27:16] WINSV	12	Window start position V
VISS_RAWFE_H3A_AEWSUBWIN[3:0] AEWINCH	4	Horizontal distance between subsamples
VISS_RAWFE_H3A_AEWSUBWIN[11:8] AEWINC	4	Vertical distance between subsamples
VISS_RAWFE_H3A_AEWINBLK[27:16] WINSV	12	Window start position H for single black line
VISS_RAWFE_H3A_AEWINBLK[6:0] WINH	7	Window height (in lines) for single black line

#### 6.9.4.3.4.6.3 RAWFE H3A Saturation Check

The saturation check module compares the data from the subsampler to the value programmed in the VISS\_RAWFE\_H3A\_PCR[31:22] AVE2LMT bit field. This value is the maximum clipping value. If all 4 pixels in the  $2 \times 2$  block are less than the AVE2LMT value, the value of the unsaturated block counter is incremented. There is one unsaturated block counter per window. The unsaturated block counters are later written to memory.

#### 6.9.4.3.4.6.4 RAWFE H3A AE/AWB Accumulators

The output from the saturation check module and the subsampler module are separately accumulated for each pixel in every  $2 \times 2$  pixel block for each window. Therefore, there are eight accumulators per window (one accumulator for each pixel in a  $2 \times 2$  pixel block, times two sets of accumulators: clipped/saturated data and presaturated data). Each of the 4 pixels in the  $2 \times 2$  pixel grid is associated with a color R, Gr, B, Gb); however, the output of these accumulators is referenced by position in the grid, not color.

The accumulators are 16 bits wide, and the accumulated data is 10 bits wide. Therefore, when a window contains more than 64 pixels of the same color, an overflow risk exists. This risk can be reduced by enabling the A-Law conversion in the preprocessing stage. See [Section 6.9.4.3.4.6.4](#) for details.

The AE/AWB module has a shift value for the accumulation of pixel values that is set in the VISS\_RAWFE\_H3A\_AEWCFG[3:0] SUMSHFT bit field.

#### 6.9.4.3.4.6.7 RAWFE H3A DMA Interface

The DMA interface module takes the data from the AF engine and AE/AWB engine and builds packets to be sent to the memory through the BL module.

The data interface has separate start pointers for the AF and AE/AWB engines.

- The starting address for the AF engine is the VISS\_RAWFE\_H3A\_AFBUFST[31:5] AFBUFST bit field.
- The starting address for the AE/AWB engine is the VISS\_RAWFE\_H3A\_AFBUFST[31:5] AEWBUFST bit field.

The DMA interface module continuously loops through this data as it builds the packets. To optimize the transfer sizes, the DMA interface sends out an AF or AE transfer for each row of pixels or windows. This requires that each horizontal row of pixels or windows starts and ends on a 32-byte boundary. If a horizontal row of pixels or windows ends on a non-32 byte boundary, the hardware packs zeroes. The counts for the AEW that occur every eight windows is sent in the row with the eighth consecutive window.

[Table 6-124](#) shows the packet formats for AF with vertical AF enabled.

**Table 6-124. RAWFE H3A AF Packet Format With Vertical AF Enabled**

Buffer Start Address (Byte Address)	31	16	15	0
VISS_RAWFE_H3A_AFBUFST				
Sum of pixel values used in HFV				(Paxel 0)
HFV_1 (peak or sum)				(Paxel 0)
HFV_sq_1 (peak or sum)				(Paxel 0)
HFV_count_1				(Paxel 0)
HFV_2 (peak or sum)				(Paxel 0)
HFV_sq_2 (peak or sum)				(Paxel 0)
HFV_count_2				(Paxel 0)
Zeroes				(Paxel 0)
VFV_1				(Paxel 0)
VFV_sq_1				(Paxel 0)
VFV_count_1				(Paxel 0)
Zeroes				(Paxel 0)
VFV_2				(Paxel 0)
VFV_sq_2				(Paxel 0)
VFV_count_2				(Paxel 0)
Zeroes				(Paxel 0)
Sum of pixel values used in HFV				(Paxel 1)
HFV_1 (peak or sum)				(Paxel 1)
HFV_sq_1 (peak or sum)				(Paxel 1)
HFV_count_1				(Paxel 1)
...				

Table 6-125 lists the packet formats for AE/AWB for sum of square mode (VISS\_RAWFE\_H3A\_AEWCFG[9:8] AEFMT = 0x0) .

**Table 6-125. RAWFE H3A AE/AWB Packet Format for Sum of Square Mode**

	31	16 15	0
Buffer address (byte address) VISS_RAWFE_ H3A_AEWBUF ST	Subsample Accum[1]		Subsample Accum[0]
			Window 0 data
	Subsample Accum[3]		Subsample Accum[2]
	Saturator Accum[1]		Saturator Accum[0]
	Saturator Accum[3]		Saturator Accum[2]
	Sum of squares[0]		
	Sum of squares[1]		
	Sum of squares[2]		
VISS_RAWFE_ H3A_AEWBUF ST + 32 bytes	Subsample Accum[1]		Subsample Accum[0]
			Window 1 data
	Subsample Accum[3]		Subsample Accum[2]
	Saturator Accum[1]		Saturator Accum[0]
	Saturator Accum[3]		Saturator Accum[2]
	Sum of squares[0]		
	Sum of squares[1]		
	Sum of squares[2]		
VISS_RAWFE_ H3A_AEWBUF ST + 64 bytes	Subsample Accum[1]		Subsample Accum[0]
			Window 2 data
	Subsample Accum[3]		Subsample Accum[2]
	Saturator Accum[1]		Saturator Accum[0]
	Saturator Accum[3]		Saturator Accum[2]
	Sum of squares[0]		
	Sum of squares[1]		
	Sum of squares[2]		
VISS_RAWFE_ H3A_AEWBUF ST + 96 bytes	Subsample Accum[1]		Subsample Accum[0]
			Window 3 data
	Subsample Accum[3]		Subsample Accum[2]
	Saturator Accum[1]		Saturator Accum[0]
	Saturator Accum[3]		Saturator Accum[2]
	Sum of squares[0]		
	Sum of squares[1]		
	Sum of squares[2]		
	Sum of squares[3]		



**Table 6-125. RAWFE H3A AE/AWB Packet Format for Sum of Square Mode  
(continued)**

	31	16 15	0
VISS_RAWFE_ H3A_AEWBUF ST + 128 bytes	Subsample Accum[1]		Subsample Accum[0]
			Window 4 data
	Subsample Accum[3]		Subsample Accum[2]
	Saturator Accum[1]		Saturator Accum[0]
	Saturator Accum[3]		Saturator Accum[2]
	Sum of squares[0]		
	Sum of squares[1]		
	Sum of squares[2]		
VISS_RAWFE_ H3A_AEWBUF ST + 160 bytes	Subsample Accum[1]		Subsample Accum[0]
			Window 5 data
	Subsample Accum[3]		Subsample Accum[2]
	Saturator Accum[1]		Saturator Accum[0]
	Saturator Accum[3]		Saturator Accum[2]
	Sum of squares[0]		
	Sum of squares[1]		
	Sum of squares[2]		
VISS_RAWFE_ H3A_AEWBUF ST + 192 bytes	Subsample Accum[1]		Subsample Accum[0]
			Window 6 data
	Subsample Accum[3]		Subsample Accum[2]
	Saturator Accum[1]		Saturator Accum[0]
	Saturator Accum[3]		Saturator Accum[2]
	Sum of squares[0]		
	Sum of squares[1]		
	Sum of squares[2]		
VISS_RAWFE_ H3A_AEWBUF ST + 224 bytes	Subsample Accum[1]		Subsample Accum[0]
			Window 7 data
	Subsample Accum[3]		Subsample Accum[2]
	Saturator Accum[1]		Saturator Accum[0]
	Saturator Accum[3]		Saturator Accum[2]
	Sum of squares[0]		
	Sum of squares[1]		
	Sum of squares[2]		
VISS_RAWFE_ H3A_AEWBUF ST + 256 bytes	Unsaturated count, win 1		Unsaturated count, win 0
			Unsaturated block count for the above 8 windows
	Unsaturated count, win 3		Unsaturated count, win 2

**Table 6-125. RAWFE H3A AE/AWB Packet Format for Sum of Square Mode (continued)**

31	16 15	0
Unsaturated count, win 5	Unsaturated count, win 4	
Unsaturated count, win 7	Unsaturated count, win 6	
Data for next eight windows, and so on. If the total number of windows is not a multiple of 8, the unsaturated counters are written immediately following the last window data. For example, if the total number of windows (including the black row) is 43, the first 40 windows are written out as per the 272-byte boundary above. Then the remaining three windows are written at +0, +32, and +64 bytes. The counts are written out at +96 instead of +256-byte boundary.		

Table 6-126 lists the packet formats for AE/AWB in minimum-maximum mode (VISS\_RAWFE\_H3A\_AEWCFG[9:8] AEFMT = 0x1).

**Table 6-126. RAWFE H3A AE/AWB Packet Format for Minimum-Maximum Mode**

31		16 15		0	
Buffer address (byte address) VISS_RAWFE_H3A_AEWBUF ST	Subsample Accum[1]		Subsample Accum[0]		Window 0 data
	Subsample Accum[3]		Subsample Accum[2]		
	Saturator Accum[1]		Saturator Accum[0]		
	Saturator Accum[3]		Saturator Accum[2]		
	Minimum[1]		Minimum[0]		
	Minimum[3]		Minimum[2]		
	Maximum[1]		Maximum[0]		
	Maximum[3]		Maximum[2]		
VISS_RAWFE_H3A_AEWBUF ST + 32 bytes	Subsample Accum[1]		Subsample Accum[0]		Window 1 data
	Subsample Accum[3]		Subsample Accum[2]		
	Saturator Accum[1]		Saturator Accum[0]		
	Saturator Accum[3]		Saturator Accum[2]		
	Minimum[1]		Minimum[0]		
	Minimum[3]		Minimum[2]		
	Maximum[1]		Maximum[0]		
	Maximum[3]		Maximum[2]		
VISS_RAWFE_H3A_AEWBUF ST + 64 bytes	Subsample Accum[1]		Subsample Accum[0]		Window 2 data
	Subsample Accum[3]		Subsample Accum[2]		
	Saturator Accum[1]		Saturator Accum[0]		
	Saturator Accum[3]		Saturator Accum[2]		
	Minimum[1]		Minimum[0]		
	Minimum[3]		Minimum[2]		
	Maximum[1]		Maximum[0]		
	Maximum[3]		Maximum[2]		

**Table 6-126. RAWFE H3A AE/AWB Packet Format for Minimum-Maximum Mode  
(continued)**

	31	16 15	0
VISS_RAWFE_ H3A_AEWBUF ST + 96 bytes	Subsample Accum[1]	Subsample Accum[0]	Window 3 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
	Minimum[1]	Minimum[0]	
	Minimum[3]	Minimum[2]	
	Maximum[1]	Maximum[0]	
	Maximum[3]	Maximum[2]	
VISS_RAWFE_ H3A_AEWBUF ST + 128 bytes	Subsample Accum[1]	Subsample Accum[0]	Window 4 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
	Minimum[1]	Minimum[0]	
	Minimum[3]	Minimum[2]	
	Maximum[1]	Maximum[0]	
	Maximum[3]	Maximum[2]	
VISS_RAWFE_ H3A_AEWBUF ST + 160 bytes	Subsample Accum[1]	Subsample Accum[0]	Window 5 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
	Minimum[1]	Minimum[0]	
	Minimum[3]	Minimum[2]	
	Maximum[1]	Maximum[0]	
	Maximum[3]	Maximum[2]	
VISS_RAWFE_ H3A_AEWBUF ST + 192 bytes	Subsample Accum[1]	Subsample Accum[0]	Window 6 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
	Minimum[1]	Minimum[0]	
	Minimum[3]	Minimum[2]	
	Maximum[1]	Maximum[0]	
	Maximum[3]	Maximum[2]	
VISS_RAWFE_ H3A_AEWBUF ST + 224 bytes	Subsample Accum[1]	Subsample Accum[0]	Window 7 data
	Subsample Accum[3]	Subsample Accum[2]	

**Table 6-126. RAWFE H3A AE/AWB Packet Format for Minimum-Maximum Mode  
(continued)**

31		16 15		0	
VISS_RAWFE_H3A_AEWBUF ST + 256 bytes	Saturator Accum[1]		Saturator Accum[0]		Unsaturated block count for the above 8 windows
	Saturator Accum[3]		Saturator Accum[2]		
	Minimum[1]		Minimum[0]		
	Minimum[3]		Minimum[2]		
	Maximum[1]		Maximum[0]		
	Maximum[3]		Maximum[2]		
	Unsaturated count, win 1		Unsaturated count, win 0		
	Unsaturated count, win 3		Unsaturated count, win 2		
	Unsaturated count, win 5		Unsaturated count, win 4		
	Unsaturated count, win 7		Unsaturated count, win 6		
	Data for next eight windows, and so on. If the total number of windows is not a multiple of 8, the unsaturated counters are written immediately following the last window data. For example, if the total number of windows (including the black row) is 43, the first 40 windows are written out as per the 272-byte boundary above. Then the remaining three windows are written at +0, +32, and +64 bytes. The counts are written out at +96 instead of +256-byte boundary.				

Table 6-127 lists the packet formats for AE/AWB in sum-only mode (VISS\_RAWFE\_H3A\_AEWCFG[9:8] AEFMT = 0x2).

**Table 6-127. RAWFE H3A AE/AWB Packet Format for Sum-Only Mode**

	31	16 15	0
Buffer address (byte address) VISS_RAWFE_H3A_AEWBUF ST	Subsample Accum[1]	Subsample Accum[0]	Window 0 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
VISS_RAWFE_H3A_AEWBUF ST + 32 bytes	Subsample Accum[1]	Subsample Accum[0]	Window 1 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
VISS_RAWFE_H3A_AEWBUF ST + 64 bytes	Subsample Accum[1]	Subsample Accum[0]	Window 2 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	

**Table 6-127. RAWFE H3A AE/AWB Packet Format for Sum-Only Mode  
(continued)**

	31	16 15	0
VISS_RAWFE_H3A_AEWBUF ST + 96 bytes	Subsample Accum[1]	Subsample Accum[0]	Window 3 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
VISS_RAWFE_H3A_AEWBUF ST + 128 bytes	Subsample Accum[1]	Subsample Accum[0]	Window 4 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
VISS_RAWFE_H3A_AEWBUF ST + 160 bytes	Subsample Accum[1]	Subsample Accum[0]	Window 5 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
VISS_RAWFE_H3A_AEWBUF ST + 192 bytes	Subsample Accum[1]	Subsample Accum[0]	Window 6 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
VISS_RAWFE_H3A_AEWBUF ST + 224 bytes	Subsample Accum[1]	Subsample Accum[0]	Window 7 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
VISS_RAWFE_H3A_AEWBUF ST + 256 bytes	Unsaturated count, win 1	Unsaturated count, win 0	Unsaturated block count for the above 8 windows
	Unsaturated count, win 3	Unsaturated count, win 2	
	Unsaturated count, win 5	Unsaturated count, win 4	
	Unsaturated count, win 7	Unsaturated count, win 6	
	Data for next eight windows, and so on. If the total number of windows is not a multiple of 8, the unsaturated counters are written immediately following the last window data. For example, if the total number of windows (including black row) is 43, the first 40 windows are written out as per the 272-byte boundary above. Then the remaining three windows are written at +0, +32, and +64 bytes. The counts are written out at +96 instead of +256-byte boundary.		

#### 6.9.4.3.4.6.8 RAWFE H3A Events and Status Checking

The AF and AEW engines generate an interrupt event at the end of processing each frame. However, these two interrupts are internally tied together so that only one H3A interrupt signal is generated. If the AF engine and AEW engine do not process the same frame concurrently, this should not be an issue. However, if they do run concurrently, one of two outcomes may occur:

- The H3A interrupt may seem to trigger only once for each frame. This can happen when the processing for the AF and AEW engines finishes at or near the same time. The interrupt service routine (ISR) does not have enough time to clear the interrupt flag for the first interrupt before the second interrupt occurs.
- The H3A interrupt may trigger twice for each frame. This can happen when the AF engine or the AEW engine finishes processing the frame much earlier than the other engine. In this case, the ISR does have enough time to clear the interrupt flag for the first interrupt by the time the second interrupt occurs.

The outcome depends on the difference in location of the last paxel/window in the frame (determines when processing is finished), the frequency of the relative clocks in the system, the occurrence and triggering of other interrupts in the system, and the latencies of the context switching and ISR execution.

The VISS\_RAWFE\_H3A\_PCR[15] BUSYAF and/or VISS\_RAWFE\_H3A\_PCR[18] BUSYAEAWB status bits are set when the start of frame occurs (if the VISS\_RAWFE\_H3A\_PCR[0] AF\_EN and/or VISS\_RAWFE\_H3A\_PCR[16] AEW\_EN bits are 1 at that time). They are automatically reset to 0 at the end of processing a frame. The VISS\_RAWFE\_H3A\_PCR[15] BUSYAF and/or VISS\_RAWFE\_H3A\_PCR[18] BUSYAEAWB status bits may be polled to determine the end of frame status.

#### 6.9.4.3.4.6.9 RAWFE H3A Interface Mux

The H3A logic also implements an LUT or Shift /clip block. The LUT logic is same as the decompanding LUT in the previous section(s) with some minor modifications. The input bit width to the LUT cannot exceed 16 and the LUT entry size is only 10 bits, since the H3A logic works on 10 bit data.

The logic also implements a shift and clipU10 functionality for converting data from up to 24 bits down to 10. The shift parameter has to be programmed to implement the desired shift value. The shift register programming must ensure that the output of the shift is not greater than 16bits when the LUT is enabled and it should not be greater than 10 bits when the LUT is disabled.

#### 6.9.4.3.4.6.10 RAWFE H3A interface to LSE

The H3A interface to LSE is VBUSP. It uses channel id to indicate data or control information. Please refer to LSE functional spec on control information mapping to channel id bits.

#### 6.9.4.3.4.6.11 RAWFE H3A Erratas

The following hardware restrictions apply and need to be taken care of by software:

1. When AF is enabled software can not use the first or last pixels in the window. Start at position 2 end at line -2.
2. When AEW is enabled software can not use the first pixel in a window. Start at position 2.
3.  $(PAXW+1) \% (AFINCH+1) \neq 1$  when VF enabled (PAXW and AFINCH are raw MMR parameters)

#### 6.9.4.3.5 RAWFE Programmer's Guide

##### 6.9.4.3.5.1 RAWFE Core programming details

There are multiple use cases that can be enabled with this module, however broadly the use case are broken in to two categories, one for Companded Data sensors and the other for Separate Exposures sensors. The following fields should be programmed correctly for the IP to function

- PWL Core
- Set slopes and thresholds as per sensor settings only for Companded sensors
- Decompanding LUT
- This LUT is used for global tone map of decompounded data from 24 bits to 16 bits. (This is typically not used for separate exposures sensor)

- Merge Logic
- Program the merge block based on the exposure ratios of the incoming data.
- This block is typically not used for companded sensors.
- LUT after merge should be programmed to tone map from 20 bits down to 16 (Separate exposure sensors only)
- DPC/ LSC
- Set as per tuning parameters for both companded/separate exposures sensors
- H3A LUT and settings
- Use LUT if H3A input comes after LSC. Program LUT to apply inverse global curve.
- Use shift if data is coming from exposures directly.

#### **6.9.4.3.5.2 RAWFE Initialization Sequence**

See [Section 6.9.3](#), *VPAC Subsystem* for details.

#### **6.9.4.3.5.3 RAWFE Real-time Operating Requirements**

See [Section 6.9.3](#), *VPAC Subsystem* for details.

#### 6.9.4.4 VISS Spatial Noise Filter (NSF4V)

##### 6.9.4.4.1 NSF4V Introduction

###### 6.9.4.4.1.1 NSF4V Features

For a list of features supported by the NSF4V module, see Section *VPAC Features*.

Some additional NSF4V parameters include:

- Input image format: any 2x2 raw color pattern
- Frame Width: limited to 4096
- Throughput:
  - 1 pixel in/out per clock
- Back Pressure Mechanism:
  - Input clock gating
  - Vert line sync
    - 7 extra lines of clocks plus enough clocks to flush pipeline

###### 6.9.4.4.1.2 NSF4V Not Supported Features

NSF4V does not support the following features:

- Edge Enhancement
- YUV formats
- Module bypass (it is implemented at top subsystem level)

###### 6.9.4.4.2 NSF4V Overview

The advanced noise filter NSF4v decomposes the input image into low-frequency, mid-frequency, and high-frequency bands first and then filter out the noise in frequency domain. NSF4v can adapt its noise filtering strength to local image intensity according to the user programmable noise threshold registers. NSF4 is also able to adjust its strength according to the amount of previously applied lens shading correction gains with a user programmable approximate radial model.

###### 6.9.4.4.2.1 Decomposition Kernel Representation

With combination of just level 1, levels 1+2, and levels 1+2+3, horizontally and vertically, the result feature detection filter kernels are shown in [Figure 6-71](#). Note that all coefficients are powers of 2, and therefore do not require multiplication.



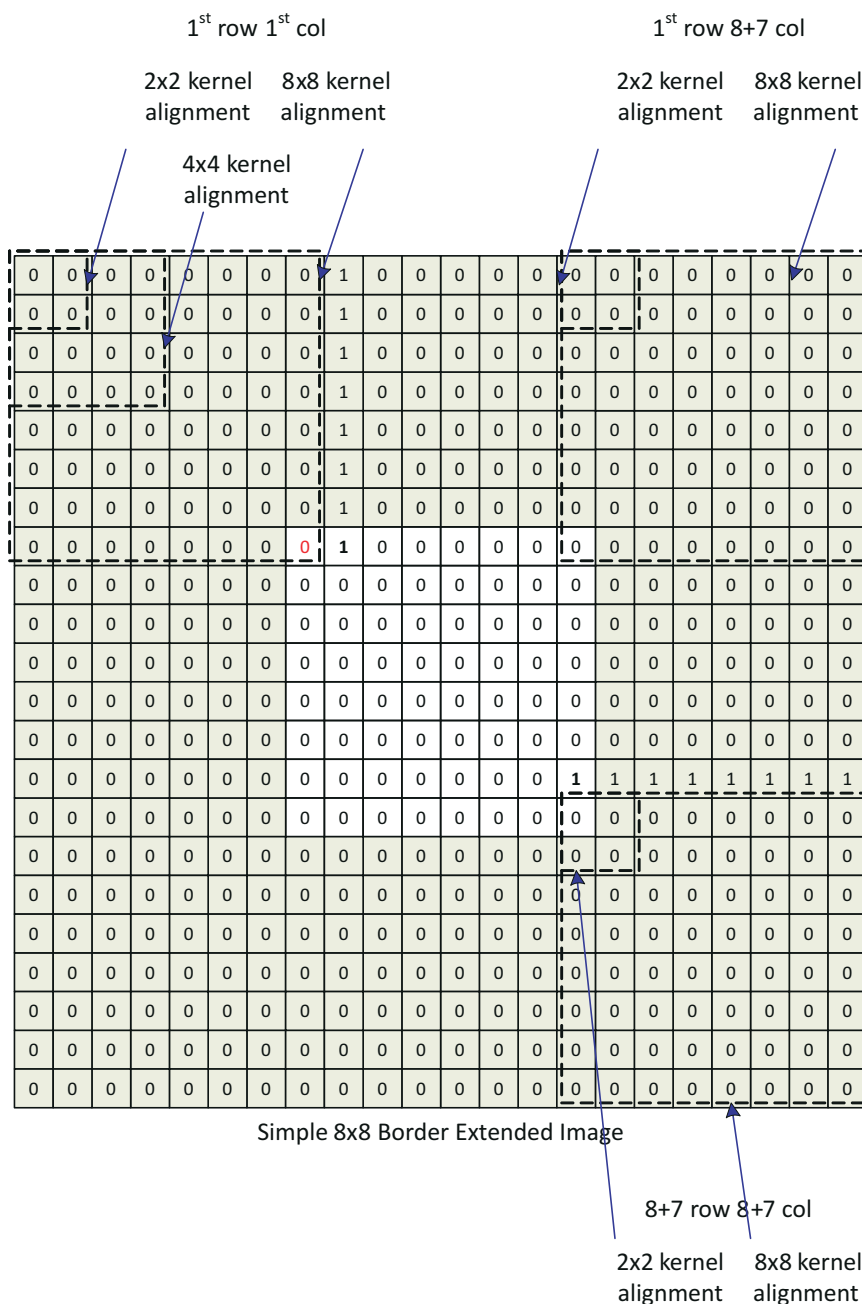
$$\begin{aligned}
 F1 &= \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix} /4 \\
 F2 &= \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} /4 \\
 F3 &= \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} /4 \\
 F4 &= \begin{bmatrix} 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 \end{bmatrix} /16 \\
 F5 &= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \end{bmatrix} /16 \\
 F6 &= \begin{bmatrix} 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \end{bmatrix} /16 \\
 F7 &= \begin{bmatrix} 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \end{bmatrix} /64 \\
 F8 &= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix} /64 \\
 F9 &= \begin{bmatrix} 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \end{bmatrix} /64
 \end{aligned}$$

nsf4v-002

**Figure 6-71. NSF4V Kernel Representation for Decomposition Filters**

G1...9 are the counter-part reconstruction kernels. Each of G1...9 is a row-wise and column-wise mirror of the counterpart F function.

The kernel representation in [Figure 6-72](#) illustrates a filter alignment.



nsf4v-003

**Figure 6-72. NSF4V Kernel Representation Filter Alignment**

Figure 6-72 is showing an academic 8x8 pixel image frame which has been border extended on all 4 edges. Superposed onto this border extended frame (shown in dotted lines) is the kernel alignment for all 3 different kernel sizes for each of the three levels:

- Level 1: 2x2 kernel
- Level 2: 4x4 kernel
- Level 3: 8x8 kernel

For each of the 9 different kernels, a matrix of (width + 7) x (height + 7) decomposition values result. All of these result matrixes are exactly aligned.

0	0	0	0	0	0	0	0	-2	2	0	0	0	0	0	0
0	0	0	0	0	0	0	0	-2	2	0	0	0	0	0	0
0	0	0	0	0	0	0	0	-2	2	0	0	0	0	0	0
0	0	0	0	0	0	0	0	-2	2	0	0	0	0	0	0
0	0	0	0	0	0	0	0	-2	2	0	0	0	0	0	0
0	0	0	0	0	0	0	0	-2	2	0	0	0	0	0	0
0	0	0	0	0	0	0	0	-2	2	0	0	0	0	0	0
0	0	0	0	0	0	0	0	-2	2	0	0	0	0	0	0
0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

nsf4v-004

**Figure 6-73. NSF4V Example F1 Kernel Result**

The kernel alignment is top-left aligned for the first pixel. Which translates in the recursive approach as a X-Y shift. In order to be mathematically aligned in the Recursive approach, the Level 1 output needs to be shifted 6 positions in both the X and Y direction relative to the Level 3 output. Additionally, the Level 2 output needs to be shifted 4 positions in both the X and Y direction relative to the Level 3 output. These shifts are critical such that Level 1, 2, and 3 values are cycle aligned when presented to the thresholding circuit.

One can reach the same conclusion by looking at the figure and thinking about which line delay or which horizontal pipelined pixel position is used in order for the kernels to output their results in the same pipeline cycle.

#### 6.9.4.4.3 NSF4V Lens Shading Correction Compensation

The Lens Shading Correction (LSC) is implemented prior to NSF. This is not an optimal order of operation and therefore an adjustment factor is implemented to compensate for the order of operations. The LSCC (Lens Shading Correction Compensation) first approximates the gain of LSC and calculates both that gain and it's inverse. Within the T<sub>n</sub> calculation, the inverse gain is applied to the LL2 average, then the T<sub>n</sub> is calculated. Last, the result is multiplied by the gain. In this way, the LSC is approximately reversed temporarily in order that T<sub>n</sub> is calculated properly.

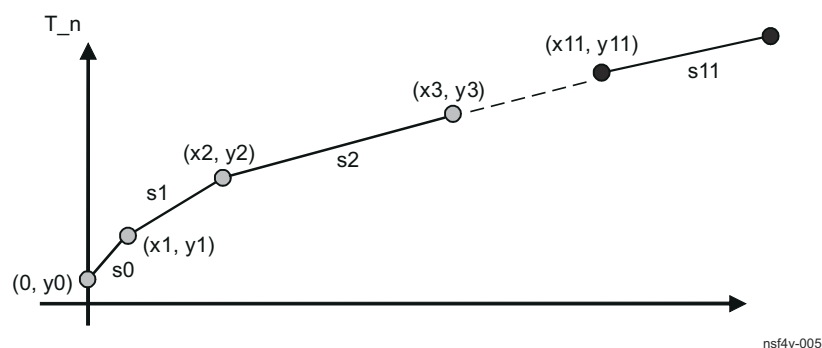
LSCC calculates the radius (distance) from the center of the image, then uses a programmable curve to look up the gain value.

While calculating the radius, the center on the frame is programmable. Also a correction gain (kh and kv) separately for X and Y are programmable which allows for elliptical shaped lenses. Because the SQRT is such an expensive function, the resulting radius is used in the SQR form and as such has a very large dynamic range. A programmable shift value effectively selects which portion of this dynamic range is to use for the curve lookup.

The curve lookup is a 16 segment curve. The SQR(Radius) is lookup on the X axis and the resulting gain is supplied on the Y axis.

#### 6.9.4.4.4 NSF4V Noise Threshold Adaptation to Local Image Intensity

Figure 6-74 shows the local image intensity to noise threshold  $T_n$  mapping via the 12-segment interpolated lookup. Twelve sets of  $(X, Y, S)$ ,  $S$  = slope, are provided per curve, one set per color plane, with the first  $X$  coordinate fixed at zero.



**Figure 6-74. NSF4V Local Image Intensity to Noise Threshold Piece-wise Linear Function**

The slopes should be implemented with signed value, as some image sensors might require higher noise threshold at the low intensity range than high intensity range.

### 6.9.4.5 VISS Global/Local Brightness and Contrast Enhancement (GLBCE) Module

#### 6.9.4.5.1 GLBCE Overview

The Global Local Brightness Contrast Enhancement module performs dynamic range compression or tone mapping of an input image based on a model of the human visual system. It is used to produce most natural images under a wide range of capture conditions, typically by revealing shadow detail which would otherwise be under-exposed in high contrast situations.

#### 6.9.4.5.2 GLBCE Interface

Data flow control of GLBCE module is done by PCLK. GLBCE processes the image cycle by cycle, and output the data after certain cycles of latency.

#### 6.9.4.5.3 GLBCE Core

The GLBCE Core is an iridix® core which performs "dynamic range compression" or "tone mapping" on an input image based on a model of the human visual system. It is used in a camera pipeline to produce the most natural images under a wide range of capture conditions, typically by revealing shadow detail which would otherwise be under-exposed in high contrast situations. The purpose of the transform is to map the image content from an input source such that it remains fully visible on an output display without loss of content. Note that iridix is specifically designed to preserve color, sharpness and boost contrast of the source image.

The iridix core is based on space-variant or pixel-by-pixel processing algorithms which construct a family of transforms based on the analysis of image content. The algorithms inside iridix are adaptive, meaning that a single set of parameters can be set to process any source image or any video sequence under different lighting conditions.

Iridix generates effective tone curves by analyzing the regions surrounding each pixel. The tone curves generated by iridix are based on an asymmetry curve which is then shaped and controlled by a number of parameters that control the gain limitation weighting towards shadows, mid tones and highlights.

##### 6.9.4.5.3.1 GLBCE Core Key Parameters

[Table 6-128](#) shows the key parameters within the iridix core. It is recommended that all parameters be set statically to their recommended values other than the iridix strength parameter, which should be controlled dynamically with scene content. The calculation of iridix strength is explained in [Section 6.9.4.5.3.2](#).

**Table 6-128. GLBCE Core Key Parameters**

Parameter	Description
iridix strength	This register sets iridix processing strength.
	0x00: Video data will not be processed at all, and will pass to the output unchanged.
	0xFF: Maximum processing strength.
Variance space	This register affects the sensitivity of the transform to different areas of the image, and can be increased in order to emphasize small regions (e.g. faces). If this parameter is set to zero, the sensitivity to small areas is maximized (i.e. the transform becomes more local). When this parameter is set to 0xF, the transform deemphasizes small local details, and the transform becomes more global
Variance intensity	For a high-contrast image with shadows and highlights, a small value will cause iridix to adjust shadows and highlights almost independently. A large variance causes iridix to become progressively more global, meaning that the presence of highlights affects the processing of shadows and vice-versa.
Slope min limit	This register is used to restrict the slope of the tone curve generated by iridix. When this parameter is set to 0x00, the tone curve slope generated by iridix is not limited.
	When this value is set to 0xFF, iridix will not generate a tone curve with a slope less than 3/4. Intermediate values of slope limit are linear interpolation between minimum and maximum values.
	The recommended range of this parameter is 0-128
Slope max limit	This register is used to restrict the slope of the tone-curve generated by iridix. When this parameter is set to 0xFF, the tone curve slope generated by iridix is not limited.
	When this value is set to 0x00, iridix will not generate a tone-curve with slope greater than 16/15 (48°). Intermediate values of slope limit are linear interpolation between minimum and maximum values.
	The recommended range of this parameter is 0-160. For noisy images, lower values should be used.

**Table 6-128. GLBCE Core Key Parameters (continued)**

Parameter	Description
Black level	The value stored in this register will be used as the zero level for iridix processing in all data channels. Data below this value will not be processed and remain unchanged.
White level	The value stored in this register will be used as the white level for iridix processing in all data channels. Data above this value will not be processed and remain unchanged.
Asymmetry LUT	This look-up-table is 33 words, each of which is 16-bits. The asymmetry function is used to balance the iridix effect between the dark and bright regions of the image.
Forward perceptual LUT	The iridix core works in a perceptually uniform space and thus when linear data is presented to the core, this look-up-table must be used to map the linear data to the perceptual space so that iridix can perform its perceptual functions.
Reverse perceptual LUT	The iridix core works in a perceptually uniform space and thus when linear data is required from the output of the core, this look-up-table must be used to map the data from the perceptual space back to linear space for further processing in the pipeline.

#### 6.9.4.5.3.2 GLBCE Iridix Strength Calculation

This section describes how the dynamic parameter, iridix strength, is calculated.

#### Note

Dynamic range of a digital camera can be described as the ratio of maximum light intensity measurable (at pixel saturation), to minimum light intensity measurable (but above read-out noise). This is a very important concept to consider when computing iridix strength, because the gain applied by iridix is also proportional to the visible boost of noise in the image. The maximum gain applied by iridix should be determined by the dynamic range of the camera system. This is governed by the equation presented in this section.

With iridix strength is zero, all tone curves are linear for all images. As iridix strength is increased, the variation between curves in shadows, mid tones and highlights increases, resulting in increasing compression of global dynamic range.

Iridix strength is calculated as follows:

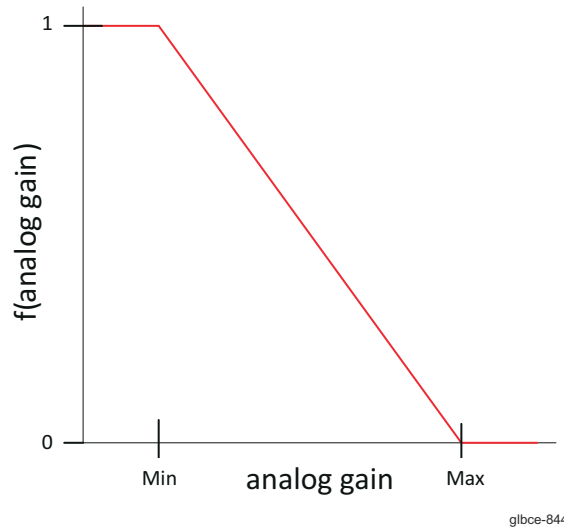
$$\text{iridix strength} = \text{Max} \left( 0, \text{Min} \left( \frac{\text{IR\_Gain\_Target} - 1}{\text{IR\_Gain\_Max} - 1}, f(\text{analog gain}) \right) \right) * 255$$

glbce-843

**Figure 6-75. GLBCE Iridix Strength Formula**

Where:

- IR\_Gain\_Target is the desired iridix gain to be applied to the image. This is calculated from the image's histogram on a per-frame basis by the auto exposure algorithm (in RAWFE H3A module). The histogram is analysed to compute an exposure ratio for the desired global histogram shift. This computed exposure ratio is then used to feed a look-up-table which in turn generates the IR\_Gain\_Target parameter.
- IR\_GainMax is the maximum iridix gain. If the fwd\_percep\_lut is disabled, then the IR\_GainMax should be set to 64, and if the fwd\_percep\_lut is enabled, then the IR\_Gainmax should be set to 16.
- f(analog gain) is a function that determines the maximum iridix strength in relation to noise. f(analog gain) is characterised as follows:



**Figure 6-76. GLBCE Iridix f(Analog Gain) Function**

$f(\text{analog gain})$  should only be used when the RAWFE is in linear mode. Otherwise, it should be set to 1

#### **6.9.4.5.3.3 GLBCE Iridix Configuration Registers**

##### **6.9.4.5.3.3.1 GLBCE Iridix Frame Width**

Frame Width is the number of pixels in an active line and is controlled from GLBCE\_FRAME\_WIDTH[15:0] VAL. Recommended value 2560 (decimal)

##### **6.9.4.5.3.3.2 GLBCE Iridix Frame Height**

Frame Height is the number of active lines in a frame and is controlled from GLBCE\_FRAME\_HEIGHT[15:0] VAL. Recommended value 1920 (decimal)

##### **6.9.4.5.3.3.3 GLBCE Iridix Control 0**

This is the main control register for the iridix core is GLBCE\_CONTROL0 where GLBCE is enabled/disabled and the type of processing algorithm is selected.

##### **6.9.4.5.3.3.4 GLBCE Iridix Control 1**

This register is reserved for debugging purposes. Recommended value 6 (decimal)

##### **6.9.4.5.3.3.5 GLBCE Iridix Strength**

GLBCE\_STRENGTH\_IR[7:0] VAL register sets the processing strength of the iridix core. Recommended value 128 (decimal)

##### **6.9.4.5.3.3.6 GLBCE Iridix Variance**

The GLBCE\_VARIANCE parameter affects the sensitivity of the transform to different areas of the image, and can be increased in order to emphasize small regions (e.g. faces). GLBCE\_VARIANCE[7:4] VARIANCEINTENSITY recommended value 12 (decimal). GLBCE\_VARIANCE[3:0] VARIANCESPACE recommended value 7 (decimal).

##### **6.9.4.5.3.3.7 GLBCE Iridix Dither**

When the number of color gradations of a display device is small (for example 6 bits per color), pixel dithering can make gradients look smother. Dithering of the least significant bits is applied and then these bits can be later truncated. The three least significant bits (D2, D1, D0) of this GLBCE\_DITHER register are responsible for the strength of dithering. There are 4 possible levels of dithering. Recommended value 0 (decimal)

#### 6.9.4.5.3.3.8 GLBCE Iridix Amplification Limit

The parameters dark amplification limit and bright amplification limit are used to restrict the luminance space in which iridix can adaptively generate tone curves for each pixel. GLBCE\_LIMIT\_AMPL[7:4] BRIGHTAMPLIFICATIONLIMIT recommended value 0 (decimal). GLBCE\_LIMIT\_AMPL[3:0] DARKAMPLIFICATIONLIMIT recommended value 0 (decimal).

#### 6.9.4.5.3.3.9 GLBCE Iridix Slope Min and Max

GLBCE\_SLOPE\_MIN[7:0] SLOPEMINLIMIT register is used to restrict the slope of the tone curve generated by iridix. Recommended value 64 (decimal).

GLBCE\_SLOPE\_MAX[7:0] SLOPEMAXLIMIT register is used to restrict the slope of the tone curve generated by iridix. Recommended value 72 (decimal).

#### 6.9.4.5.3.3.10 GLBCE Iridix Black Level

The value stored in the Black Level register GLBCE\_BLACK\_LEVEL[15:0] VAL will be used as the zero level for iridix processing in all data channels. Data below the Black level will not be processed and remain unchanged. Recommended value 0 (decimal).

#### 6.9.4.5.3.3.11 GLBCE Iridix White Level

The value stored in White Level register GLBCE\_WHITE\_LEVEL[15:0] VAL will be used as white level for iridix processing in all data channels. Data above White level will not be processed and stay unchanged. Recommended value 65535 (decimal).

#### 6.9.4.5.3.3.12 GLBCE Iridix Asymmetry Function Look-up-table

The Asymmetry function is used to balance the iridix effect between the dark and bright regions of the image. The Asymmetry Function Lookup Table geometry is 33 words, each of which is 16-bits wide.

GLBCE\_LUT\_FI\_00

GLBCE\_LUT\_FI\_01

... repeat until...

GLBCE\_LUT\_FI\_32

Recommended values (decimal):

0:5377:10218:14600:18585:22224:25561:28631:31466:34092:36530:38801:40921:42904

:44764:46511:48156:49706:51171:52557:53870:55116:56299:57425:58498:59520:60497

:61429:62322:63176:63995:64781:65535

The C-code for generating the Asymmetry LUT is shown below:

```
#include <math.h>
int AsymmetryTable[33];
void GenerateAsymmetry(int Asymmetry, int SecondPole)
{
    double x = (double(Asymmetry)+1)/257 * 2 - 1;
    int ai = (int)floor(0.5 + 255 * (1- 1/(1000*x*x*x) + x - ((x >= 0)*2)));
    double as = fabs(double(ai) / 255);
    double dp = double(SecondPole) / 255;
    int ii;
    for (ii = 0 ; ii < 33 ; ++ii)
    {
        if(ai >= 0)
        {
            x = double(ii) / 32;
        }
        else
        {
            x = double(32 - ii) / 32;
        }
        int y = (int)floor((dp+(1-dp)*pow((fabs(1-dp-x)/dp), 3)) * (x*(as+1)/(as+x) ) *
            65535.0 + 0.5);
        y = y < 0 ? 0 : y > 65535 ? 65535 : y;

        if (ai < 0)
        {

```



```

        y = 65535 - y;
    }
    AsymmetryTable[ii] = y;
}

```

#### 6.9.4.5.3.3.13 GLBCE Iridix Forward and Reverse Perceptual Functions Look-up-tables

The iridix core works in a perceptually uniform space and thus when linear data is presented to the core it needs to be processed with special perceptual functions. These are implemented as two perceptual LUTs namely: Forward Perceptual LUT and Reverse Perceptual LUT. Each LUT is made up of sixty-five 32-bit words.

GLBCE\_REV\_PERCEPT\_LUT\_00  
GLBCE\_REV\_PERCEPT\_LUT\_01  
... repeat untill ...  
GLBCE\_REV\_PERCEPT\_LUT\_64

Recommended values (in decimal) for Reverse LUT:

0:228:455:683:910:1138:1369:1628:1912:2221:2556:2916:3304:3717:4158:4626:5122:  
5645:6197:6777:7386:8024:8691:9387:10113:10869:11654:12471:13317:14194:15103  
:16042:17012:18014:19048:20113:21210:22340:23501:24696:25922:27182:28475  
:29800:31159:32552:33977:35437:36930:38458:40019:41615:43245:44910:46609  
:48343:50112:51916:53755:55630:57539:59485:61466:63482:65535

GLBCE\_FWD\_PERCEPT\_LUT\_00  
GLBCE\_FWD\_PERCEPT\_LUT\_01  
... repeat untill ...  
GLBCE\_FWD\_PERCEPT\_LUT\_64

Recommended values (in decimal) for Forward LUT:

0:4622:8653:11684:14195:16380:18335:20118:21766:23304:24751:26119:27422:28665  
:29857:31003:32108:33176:34209:35211:36185:37132:38055:38955:39834:40693:41533  
:42355:43161:43951:44727:45488:46236:46971:47694:48405:49106:49795:50475:51145  
:51805:52456:53099:53733:54360:54978:55589:56193:56789:57379:57963:58539:59110  
:59675:60234:60787:61335:61877:62414:62946:63473:63996:64513:65026:65535

#### 6.9.4.5.3.3.14 GLBCE Iridix WDR Look-up-table

If an image sensor with built-in WDR functionality is used, and that sensor outputs data in a companded format, this LUT can be used to reverse the companding function in the sensor and feed linear data into the iridix core. This LUT has 257 entries.

GLBCE\_WDR\_GAMMA\_LUT\_00  
GLBCE\_WDR\_GAMMA\_LUT\_01  
... repeat untill ...  
GLBCE\_WDR\_GAMMA\_LUT\_256

Recommended values (in decimal):

0:5887:10263:13117:15287:17058:18564:19881:21055:22117:23088:23984:24818:25597:26330  
:27021:27677:28301:28896:29466:30011:30536:31040:31527:31997:32451:32891:33318:33733  
:34135:34527:34908:35280:35643:35996:36342:36680:37010:37333:37650:37960:38264:38562  
:38854:39141:39423:39700:39972:40239:40502:40761:41016:41266:41513:41756:41996:42232  
:42465:42694:42921:43144:43364:43582:43797:44009:44218:44425:44629:44831:45030:45228  
:45423:45615:45806:45995:46181:46366:46548:46729:46908:47085:47260:47434:47606:47776  
:47945:48112:48277:48441:48604:48765:48924:49083:49239:49395:49549:49702:49854:50004  
:50153:50301:50448:50593:50738:50881:51024:51165:51305:51444:51582:51719:51855:51990  
:52124:52257:52389:52521:52651:52781:52909:53037:53164:53290:53415:53540:53663:53786  
:53908:54030:54150:54270:54389:54507:54625:54742:54858:54974:55089:55203:55316:55429  
:55541:55653:55764:55874:55984:56093:56202:56310:56417:56524:56630:56736:56841:56945  
:57049:57153:57256:57358:57460:57561:57662:57763:57863:57962:58061:58159:58257:58355

:58452:58548:58645:58740:58835:58930:59025:59118:59212:59305:59398:59490:59582:59673  
 :59764:59855:59945:60035:60124:60213:60302:60390:60478:60566:60653:60740:60827:60913  
 :60999:61084:61169:61254:61338:61422:61506:61589:61673:61755:61838:61920:62002:62083  
 :62164:62245:62326:62406:62486:62566:62645:62724:62803:62882:62960:63038:63115:63193  
 :63270:63347:63423:63500:63576:63651:63727:63802:63877:63952:64026:64101:64175:64248  
 :64322:64395:64468:64541:64613:64685:64757:64829:64901:64972:65043:65114:65185:65255  
 :65325:65395:65465:65535

#### 6.9.4.5.4 GLBCE Embedded Memory

GLBCE needs 2 lines of line memories and 1 set of cache memories.

**Table 6-129. GLBCE Memories**

Memory Type	Qty	Size	Clock	ECC Supported	Description
DELAY LINE	1	2122 x 64 (GLBCE_LINE_SIZE /2+10)x64	FCLK	No	GLBCE line memory
STATMEM	8 banks	16 bit x 1024	FCLK	Yes	GLBCE Cache memory (computes and stores the statistics from the frame)

#### 6.9.4.5.5 GLBCE General Processing

GLBCE block needs appropriate statistics information to be stored to its cache memory in prior to processing incoming frame. Usually, this takes a two-pass process.

1. GLBCE is configured for input frame-A
2. Frame-A is given to GLBCE
3. GLBCE processes the input frame-A
4. GLBCE generates statistic (statistics-A) data based on frame-A in vertical blanking period
5. (Optional process) statistics-A is read from cache memory and moved to SDRAM
6. GLBCE is configured for input frame-B
7. Frame-B is given to GLBCE
8. GLBCE processes the input frame-B with the statistics-A generated from frame-A
9. GLBCE generates statistics-B based on frame-B in vertical blanking period
10. (Optional process) statistics-B is read from cache memory and moved to SDRAM

Alternatively, GLBCE can process in the following manner

1. Pre-calculated Statistics-C is copied to GLBCE cache memory from SDRAM
2. GLBCE is programmed for input frame-D
3. Frame-D is given to GLBCE
4. GLBCE processes frame-D based on statistics-C
5. GLBCE generates statistics-D based on frame-D in vertical blanking period
6. (Optional process) statistics-D is read from cache memory and moved to SDRAM

In [Section 6.9.4.5.6](#), some examples closer to real world use cases are explained based on the above two processes.

#### 6.9.4.5.6 GLBCE Continuous Frame Processing

This is a very basic case, which does not need transferring statistics between cache memory and SDRAM. The most common example is video capture. In this case, the most straight forward way to run GLBCE, is to generate statistics from N-th frame, and apply it to (N+1)-th frame. The effect from using statistics of different frame should be small if the difference between frames is small. For example, it is expected this does not generate artifact, if the video frame rate is 30fps or larger. Below this rate, the single frame processing programming model explained in [Section 6.9.4.5.7](#) should be considered.

In the case of continuous frame processing, GLBCE is operated in the following manner:

1. GLBCE is programmed for incoming frame

2. Input frame comes in
3. GLBCE processes Input frame based on the statistics from the previous frame
4. GLBCE generates statistics based on the current input frame in the vertical blanking period
5. Repeat Step 1 through 4 as required

For the first frame of a set of sequential frames, Strength parameter should be 0, so GLBCE does not affect the image based on wrong statistics. SW may need to increase the strength value gradually after the first frame.

#### **6.9.4.5.7 GLBCE Single Image Processing**

For a single image, GLBCE needs to process the image twice, once for statistics generation, and once for actual GLBCE process.

1. GLBCE is configured for the input image.
2. The image is given to GLBCE. (Output image from GLBCE should be suppressed, for example, by disabling the next module in the path)
3. Statistics is generated in vertical blanking period.
4. The input image is given to GLBCE again. Output image with right GLBCE is generated.

Also, GLBCE can take in small size image in the first pass to reduce processing time.

5. A small size version (IMG-S) of the input image (IMG-IN) is generated.
6. GLBCE is configured for the small size image (IMG-S).
7. The small size image (IMG-S) is given to GLBCE.
8. Statistics for the small size image is generated in vertical blanking period.
9. GLBCE is configured for the original input image (IMG-IN).
10. The original input image (IMG-IN) is given to GLBCE.

### 6.9.4.6 VISS Flexible Color Processing (FCP) Module

#### 6.9.4.6.1 FCP Overview

The Flexible Color Processing module is responsible for performing CFA Interpolation as well as performing a host of color processing functions for generating different color outputs.

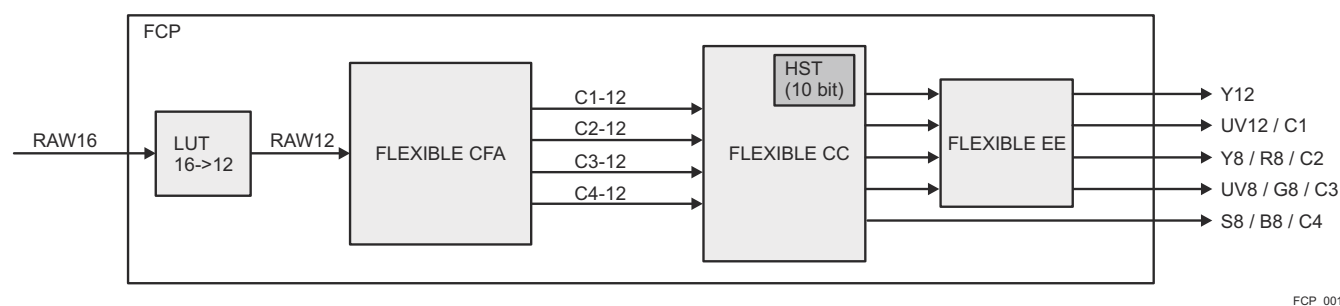
##### 6.9.4.6.1.1 FCP Features Supported

The FCP include the following main features:

- Flexible CFA for generating multi-plane output for any 2x2 raw format sensor configuration.
  - Supports RCBC, RCCC, Bayer, RGBC as well as other formats
  - Can generate up to 4 independent color planes
  - Support for up to 3 directions per color plane
  - Adaptive Threshold based on intensity measure
- Support for multiple color format output generation:
  - Support 12-bit YUV output
  - Supports RGB output
  - Supports Saturation outputs (8 bits only)
  - Supports Grayscale/Luma/IR/Clear output.
- Support for 16 to 12 compression (LUT based) at the entry of the block
- Support for flexible output format generation including YUV as well as RGB, SV, and so forth.
- Single cycle/pixel performance

##### 6.9.4.6.2 FCP Functional Description

The Flexible Color Processor (FCP) receives data from RAW-FE and does demosaicing and color conversion. The output of FCP is sent to external memory.



**Figure 6-77. FCP Block Diagram**

The FCP consists of the following major sub-blocks for supporting different sensor and output formats:

- LUT based compression: used to reduce the bit width from 16 bits to 12 bits while still preserving the dynamic range.
- Flexible CFA: this block takes in as input any 2x2 raw sensor pattern and is capable of generating up to 4 full resolution color planes.
- Flexible Color Conversion (Flexible CC): this block takes in the output of the Flexible CFA and generates multiple standard as well as custom data formats.
- Flexible Edge Enhancer (Flexible EE): this block can take the output of the Color Conversion and align the Y and UV channels as well as enhance the edges in the luma channel.

##### 6.9.4.6.3 FCP Submodule Details

###### 6.9.4.6.3.1 Flexible CFA / Demosaicing

###### 6.9.4.6.3.1.1 Feature-set

The Flexible CFA include the following main features:

- Support for Flexible CFA for generating multi-plane output for any 2x2 raw format sensor configuration.
  - Supports RCBC, RCCC, Bayer, RGBC as well as other formats.

- Can generate up to 4 independent color planes.
- Supports Clear (C) pixel.
- Supports Infrared (IR) pixels.
- Edge aware CFA interpolation.
- Software programmable filter for interpolation up to 4 color channels.
- Support for up to 4 phases and 3 directions per color channel.
- Software controlled algorithm for direction selection.
- Adaptive threshold calculation for direction selection.
- Support of up to 2 sets of programmable mask based gradient selection

#### 6.9.4.6.3.1.2 Block Diagram of Flexible CFA

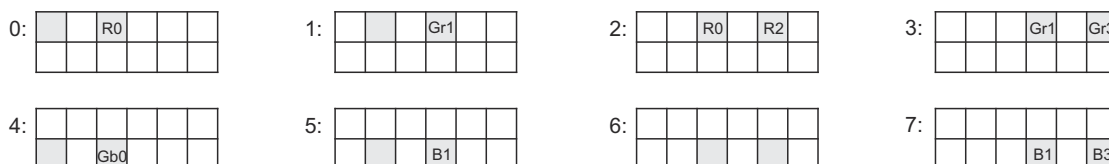
The Flexible CFA module follows the VPORT (internal) interface for both the input/output to the RAW-FE or Flexible CC module.

##### 6.9.4.6.3.1.2.1 Gradient/Threshold Calculation

The Flexible CFA incorporates 2 sets of gradient and direction selection logic. Typically for symmetric 2x2 patterns like Bayer, only a single set is sufficient; however in sensors wherein one of the channels is working on a different data type, the second set can be used for that channel. An example of such sensors is the RGB-IR sensor, wherein the 2x2 pattern consists of one pixel of each color channel and one pixel of the IR channel. In this case, mixing the gradients and intensity of the color channels with the IR channel could be meaningless.

The gradient selection uses bit masks to select the desired pixel position for calculating the horizontal and the vertical gradient.

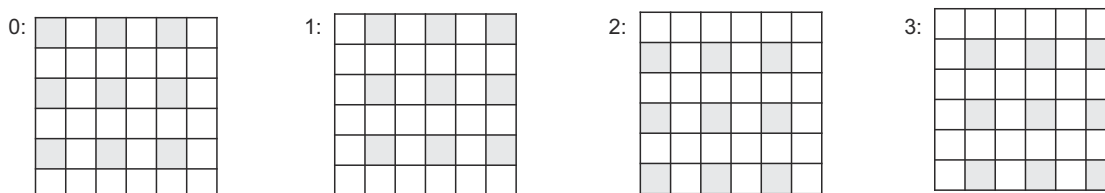
Figure 6-78 shows the notation for the bit-mask values. The same notation is used for vertical masks as well and the notation implies that setting a '1' in any field of the bit Mask enables the absolute difference of those 2 pixels to be summed in the gradient calculation.



FCP\_007

**Figure 6-78. Gradient Bit Masks**

Similar masks are also used to calculate the intensity (see Figure 6-79), however instead of taking the absolute difference, the bit masks are used to sum up the pixels in those locations.



FCP\_008

**Figure 6-79. Intensity Bit Mask**

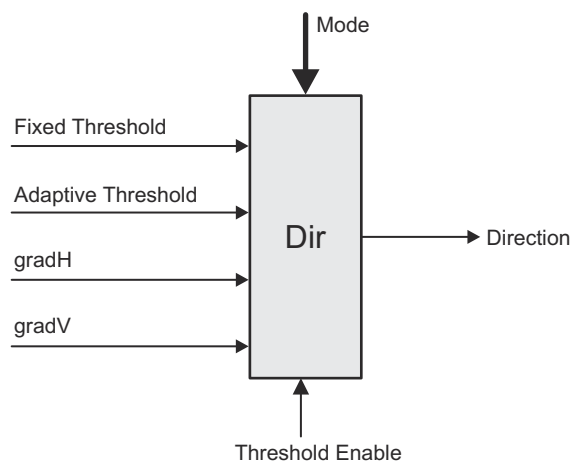
The 4 bit Intensity mask is used to choose which pixels in the 6 rows should be used for creating the intensity sum. The same masks are used for every 2x2 window within the 6x6 buffer. (Thus setting all 4 bits to '1' will be equivalent to summing up the entire 6x6 window.) A programmable shift is then used to bring the intensity down to the 0-16k range. The maximum value of intensity can be 6x6 x4k = 144k, so the maximum value of shift is 4 (divide by 16). The shift value can be programmed to any value between 0 and 4 and should be chosen based

on the mask. Note that dependent on the shift value, it's possible that the full range of 0-16k of the intensity may not be reached, hence the adaptive threshold should be programmed accordingly.

The adaptive threshold calculation uses the calculated intensity. For more information, see [Section 6.9.4.6.3.1.2.2, Software Controlled Direction Selection](#).

#### 6.9.4.6.3.1.2.2 Software Controlled Direction Selection

Software can control or guide direction selection at frame or sequence level as shown in [Figure 6-80](#).



FCP\_009

**Figure 6-80. Software Controlled Direction Selection**

[Table 6-130](#) summarizes number of option available to software.

**Table 6-130. Summary of HSX Spaces**

Mode	Name	Details
0	Horizontal	Dir = 0
1	Vertical	Dir = 1
2	HV	Dir = 2
3	HV_Threshold (Fixed threshold is MMR)	in Dir = 0: $\text{gradH} \geq \text{gradV} + \text{threshold}$ in Dir = 1: $\text{gradV} \geq \text{gradH} + \text{threshold}$ in Dir = 2: Otherwise
4	HV_AdaptiveThreshold	in Dir = 0: $\text{gradH} \geq \text{gradV} + \text{threshold}$ in Dir = 1: $\text{gradV} \geq \text{gradH} + \text{threshold}$ in Dir = 2: Otherwise

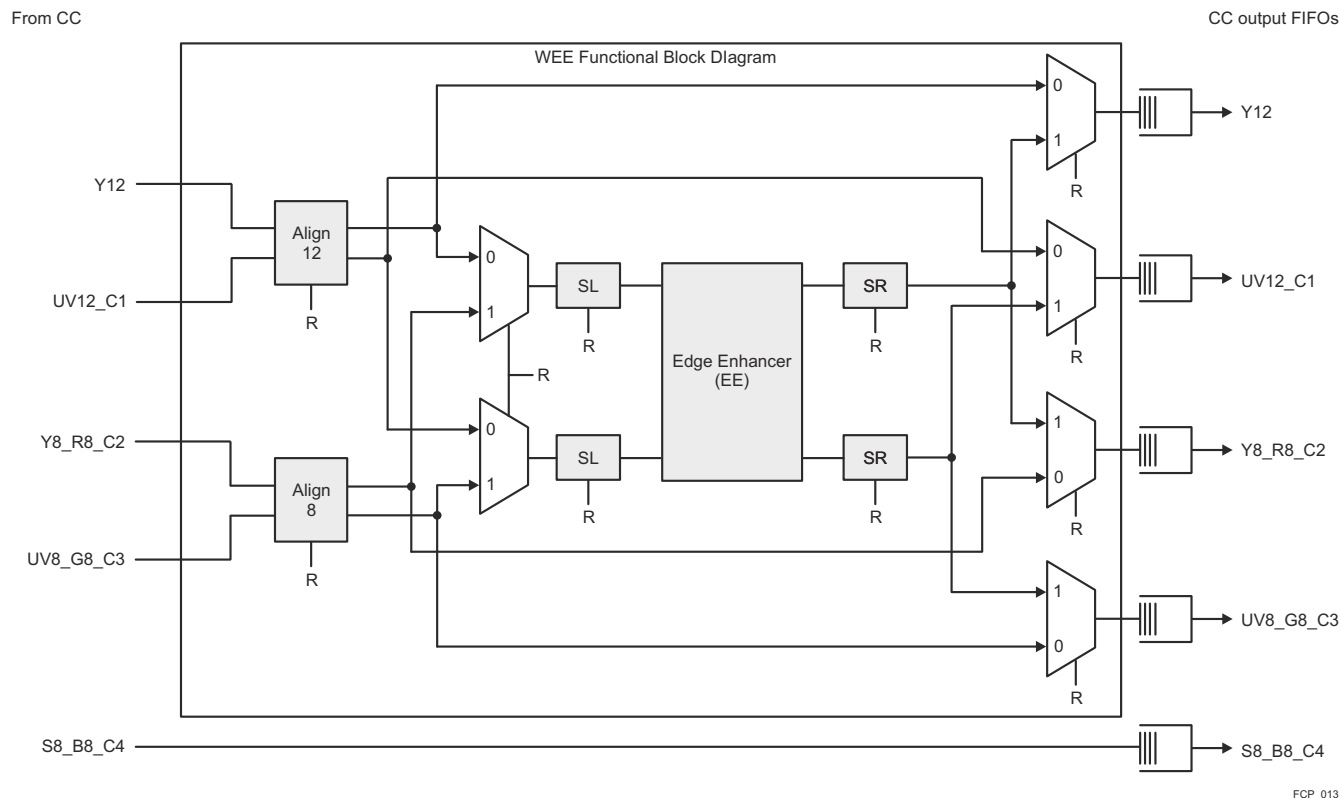
For calculation of adaptive threshold, 7 options are used. Each one defines a threshold to be used as a different intensity level as it follows:

- Entry-0 -> Intensity = 0
- Entry-1 -> Intensity = 512
- Entry-2 -> Intensity = 1024
- Entry-3 -> Intensity = 2048
- Entry 4 -> Intensity = 4096
- Entry 5 -> Intensity = 8192
- Entry 6 -> Intensity = 16384

The adaptive threshold can then be calculated as a linear interpolation between 2 successive entries based on the given intensity.

#### 6.9.4.6.3.2 Edge Enhancer Module Wrapper (WEE)

Figure 6-81 shows a high level block diagram of the Edge Enhancer (EE) module wrapper. For more information about the Edge Enhancer algorithm, see *Edge Enhancer (EE)*. The wrapper places the Edge Enhancer within the FCP structure along with associated muxes.



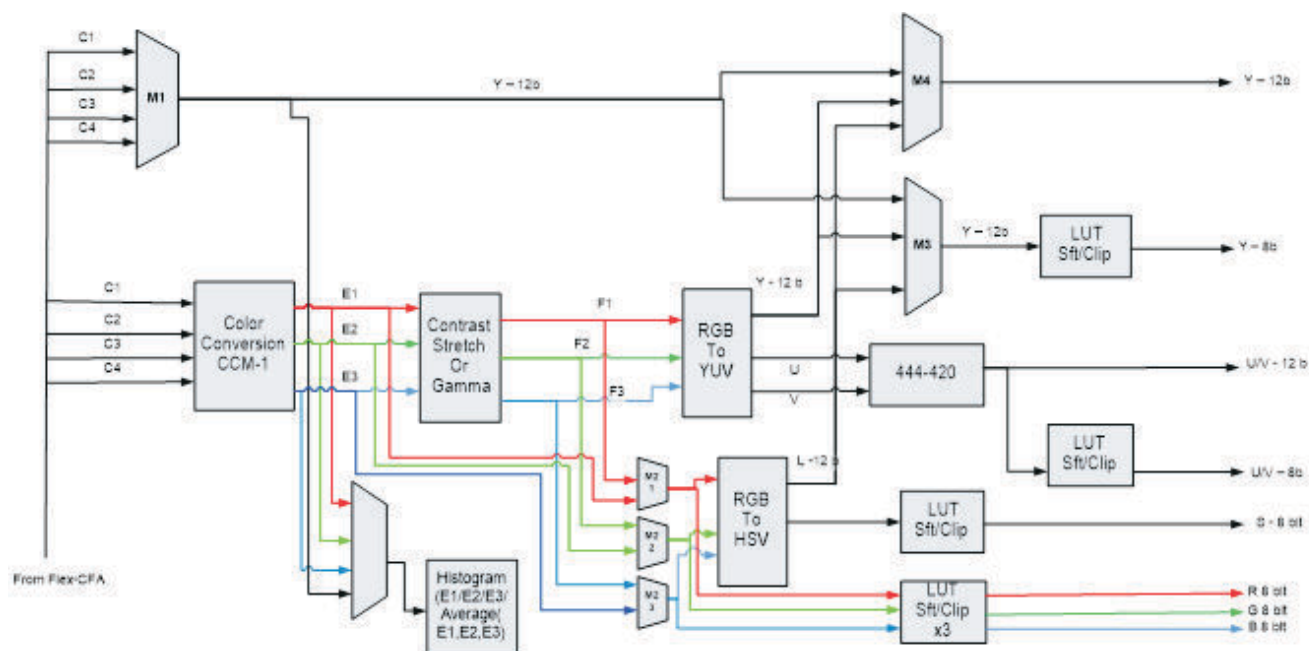
**Figure 6-81. Block Diagram of Edge Enhancer Module Wrapper**

##### 6.9.4.6.3.2.1 EE - Edge Enhancer Block

The Edge Enhancer module is used to enhance the visual quality of the image by increasing its sharpness. The module only works on Y (Luma) data and uses a combination of 2D High-Pass Filtering to detect edges and then apply them on the input image.

##### 6.9.4.6.3.3 Flexible Color Conversion (CC)

Figure 6-82 shows a high level block diagram of the Flexible Color Conversion (CC) module.



**Figure 6-82. Flexible CC Block Diagram (Logical View)**

The Flexible CC enables multiple use cases and allows different color formats to be generated.

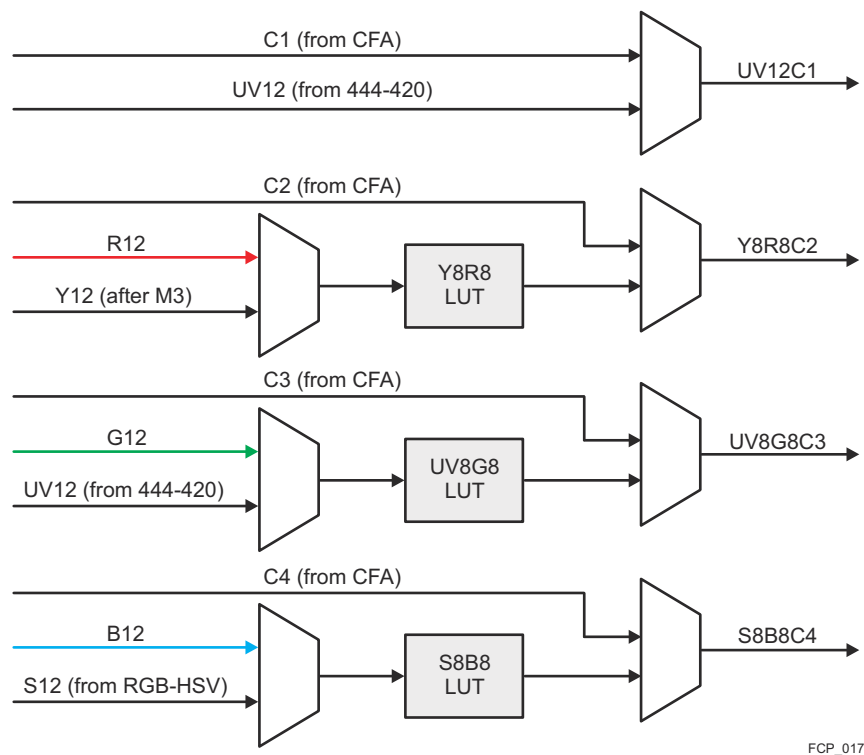
The Flexible CC module follows the VPORT (Internal) Interface for both the input to the Flexible CC module.

#### 6.9.4.6.3.3.1 Interface Mux

The Flexible CC logical block diagram has 5 primary outputs connecting to the LSE modules in the VISS. For more information about LSE modules, see *Load Store Engine (LSE)*. A multiplex scheme is used to connect multiple outputs on to 5 interfaces using the interface Mux. The Flexible CC output can optionally also allow data from Flexible CFA to be tapped out on to one of the primary interfaces. Note that 12-8 LUTs are shared between the RGB generation and the Y/UV 8-bit generation logic.

Figure 6-83 shows the details of the interface mux. There are only 3 LUTs shared in different mode and only 5 concurrent output streams can be activated at any point of time. For 8-bit outputs, the data is stored in the MSB 11-4 bits of the 12-bit bus. Further, the Y12 output is dedicated and not multiplexed with other interface signals.





**Figure 6-83. Flexible CC Interface Mux**

#### 6.9.4.6.3.3.2 Color Conversion (CCM-1)

The Color Conversion (CCM-1) is defined using the mathematical equations from [Figure 6-84](#).

$$\begin{aligned} E1(x, y) &= W11 \times C1(x, y) + W12 \times C2(x, y) + W13 \times C3(x, y) + W14 \times C4(x, y) + Offset\_1 \\ E2(x, y) &= W21 \times C1(x, y) + W22 \times C2(x, y) + W23 \times C3(x, y) + W24 \times C4(x, y) + Offset\_2 \\ E3(x, y) &= W31 \times C1(x, y) + W32 \times C2(x, y) + W33 \times C3(x, y) + W34 \times C4(x, y) + Offset\_3 \end{aligned}$$

**Figure 6-84. CCM-1**

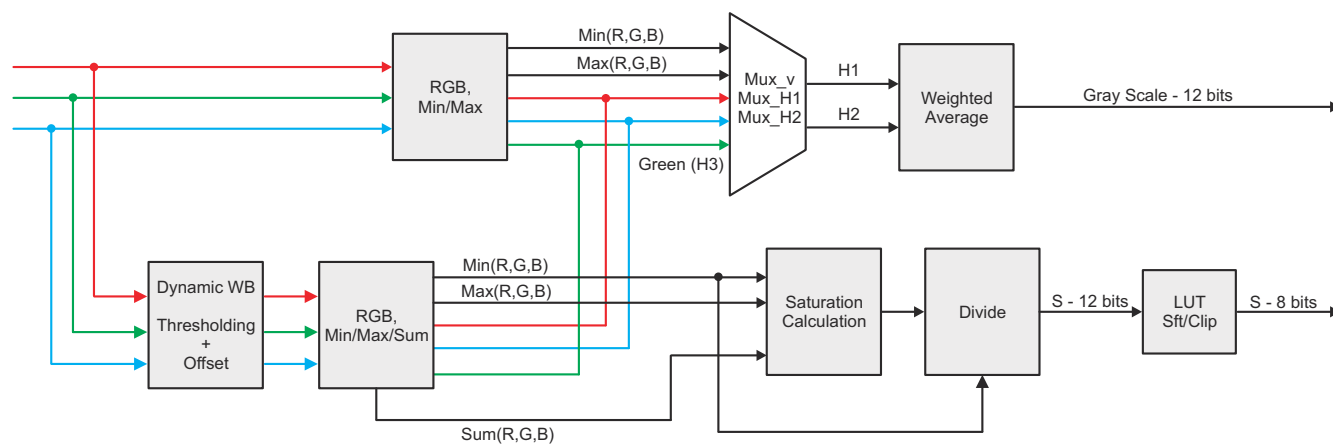
The CCM matrices weights ( $W^{**}$ ) are 12 bit signed each (S12Q8) representing a range from -8 to +7.996 with 8 bits of fraction. The offsets are 13 bit signed notation (S13Q11) representing a range of -4096 to +4095. The offset is effectively shifted by one (left shifted/ multiplied by 2) before being applied. The effective range of offset is -2 to +1.995 with 11 bits of fraction.

The 4 component CCM allows for a power combination of color format generation to be addressed.

#### 6.9.4.6.3.3.3 RGB to HSX Conversion

The RGB-HSV module is used for converting color spaces and creating custom luminance data plane. This way the 8-bit Saturation and 12-bit Luminance (L or V) plane are generated.

[Figure 6-85](#) shows the block which generates the Luminance / Gray Scale Plane as well as the Saturation (8-bit) plane.



FCP\_019

Figure 6-85. S & V Generation

#### 6.9.4.6.3.3.3.1 Weighted Average Block

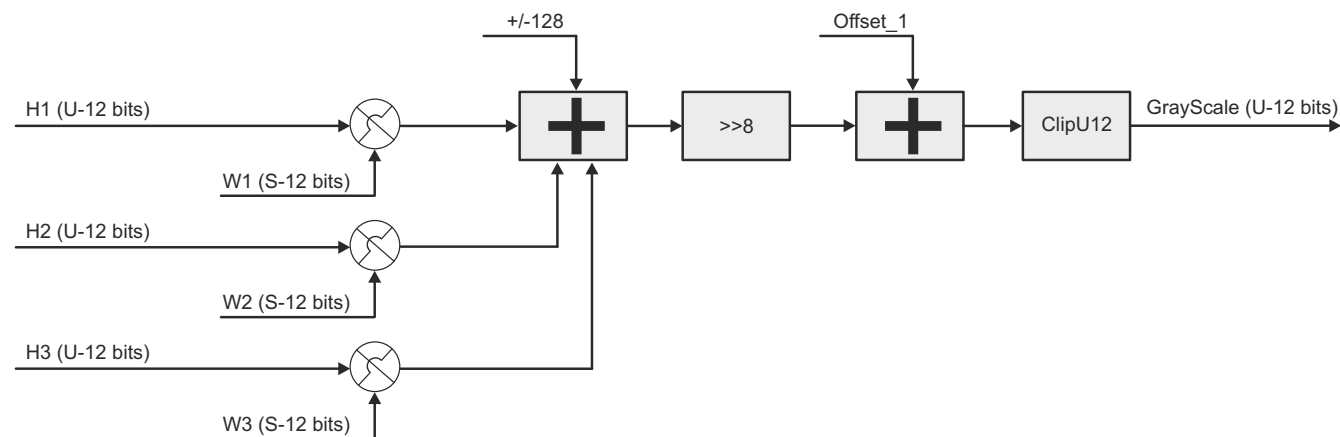
The Weighted Average block implements the equation shown on Figure 6-86.

$$GrayScale(x, y) = W1 \times H1(x, y) + W2 \times H2(x, y) + W3 \times H3(x, y) + Offset\_1$$

Figure 6-86. Gray Scale Computation

In the equation on Figure 6-86 the Weights (W1\*) are 12 bits signed with a range of -8 to +7.9996 in S12Q8 format. The offset is Signed 13 bits in S13Q11 format with a range of -2 to +1.995.

Figure 6-87 depicts the architecture of the Weighted Average block:



FCP\_020

Figure 6-87. Weighted Average Block

The mux before the matrix chooses whether the native R/B channels are transmitted or alternatively the Min/Max (RGB) from the Min-Max block are transmitted. Further, the calculation of V can work on either the non-WB corrected data or the WB corrected version. For more information about WB correction, see Section 6.9.4.6.3.3.3.2, Saturation Block.

The combination of the Min-Max block, the mux and the Weighted Matrix allows any of the Gray Scale calculation as shown in Table 6-131.

**Table 6-131. Summary of HSX Spaces**

No	Formula	Configuration
Grey scale (HSI)	$(R+G+B) / 3$	H1 = R, H2 = B; Offset_1 = 0; W1 = W2 = W3 = 1/3 = 85;
Gray Scale (HSV)	Max(RGB)	H1 = Max(RGB); H2 = NA Offset_1 = 0; W2/W3 = 0; W1 = 1 (256);
Gey Scale HSL	$(\text{Max}(\text{RGB}) + \text{Min}(\text{RGB})) / 2$	H1 = Max(RGB), H2 = Min(RGB); W1 = W2 = 0.5 (128) Offset_1 = 0; W3 = 0;
		H1 = R, H2 = G; W1 = W3 = 0.25 (64); W2 = 0.5 (128) Offset_1 = 0

**Table 6-132. Grey Scale and Saturation Calculation**

No	HSI Color Space	HSV Color Space	SHL Color Space	Customer Requirements
Grey Scale Computation	$(R+G+B) / 3$	Max(R,G,B)	$(\text{Max}(\text{R,G,B}) + \text{Min}(\text{R,G,B})) / 2$	$(R+2*G+B) / 3$
Saturation (S) Computation	$1 - (\text{Min}(\text{R,G,B}) / (R+G+B))$	$(\text{Max}(\text{R,G,B}) - \text{Min}(\text{R,G,B})) / \text{Max}(\text{R,G,B})$	$(\text{Max}(\text{R,G,B}) - \text{Min}(\text{R,G,B})) / 255 - \text{Gray value}$	S = Max(R,G,B) - Min(R,G,B)
Hue (H) Computation	Complex formula involving (R-G) (R-B) & (G-B)	Division formula livolving (R-G) (R-B) & (G-B)	Complex formula involving all component	Separate output for (R-G) and Yellowness

#### 6.9.4.6.3.3.2 Saturation Block

The Saturation block first applies a dynamic white balance offset to correct the saturation plane when the image is in log domain and is used for analytics data flow. Once the WB offset is applied, the saturation calculation proceeds as described below. Note that the White balance offset is only applied if the independent pixel values are below a threshold (VISS\_FCP\_FCC\_RGBHSV\_WB\_LINLOGTHR\_1/2). Further, even the minimum of RGB is compared against a threshold (VISS\_FCP\_FCC\_RGBHSV\_WB\_LINLOGTHR\_2[27-16] SATMINTHR bit field) and the higher of the two is used. The saturation calculation always uses the data with the WB correction applied, however the V calculation can choose between WB corrected or uncorrected data using VISS\_FCP\_FCC\_CFG\_1[26] MUXRGBHSV\_MUX\_V.

Table 6-132 shows the condensed form for the Grey Scale and Saturation calculation. In the Flexible CC the saturation calculation is supported using two modes based on the VISS\_FCP\_FCC\_CFG\_2[6] HSVSATMODE bit as follows.

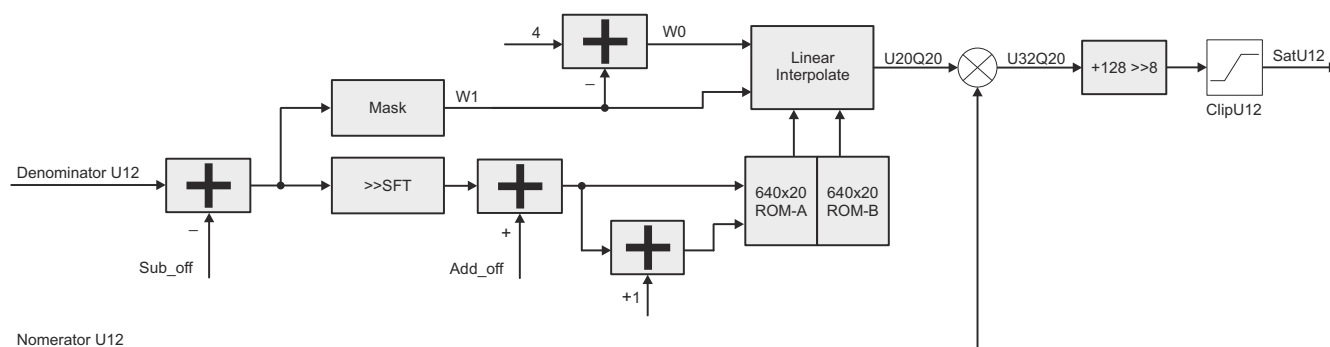
- HSVSATMODE = 0: Max(RGB) - Min(RGB)
- HSVSATMODE = 1: SUM(RGB) - Min(RGB)

Similarly the denominator for the dvision is chosen using the following selection on the VISS\_FCP\_FCC\_CFG\_2[5-4] HSVSATDIVMODE bit field.

- HSVSATDIVMODE = 0: No division, denominator = 1
- HSVSATDIVMODE = 1: Max(RGB)
- HSVSATDIVMODE = 2: 4095 – Gray value (computed in Figure 6-86)
- HSVSATDIVMODE = 3: Sum(RGB)

#### 6.9.4.6.3.3.3 Division Block

The division operation is implemented to calculate 1/x followed by a multiplication with the value. The input to the LUT is 12-bit unsigned value and the output is 8-bits fraction representing 1/x calculation. The divide LUT is implemented as a non-linear ROM with 1216 (1280 for aligning to 128 size boundary) entries and up to 2 segments to reduce error. Ideally to implement a full ROM for an input space of 12 bits would require 4k entires, however to save space and reduce error, the ROM is partitioned such that it is full resolution for the first 256 entries where the curve is highly non linear, whereas the next 4k - 256 entires have a step size of 4. Each ROM entry has a bit size of 20 bits in U20Q20.



FCP\_022

**Figure 6-88. Division LUT**

These are the parameters that need to be calculated based on the range of the input.

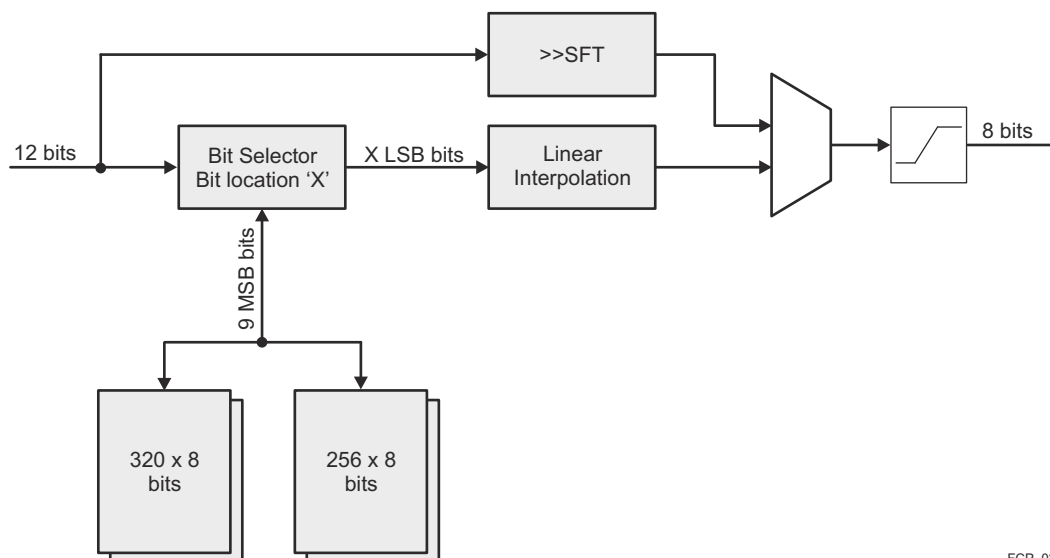
- SFT: '0' if input <256, 2 otherwise
- Mask: '0' if input <256, 4 otherwise
- Add\_off: '0' if input <256, 256 otherwise
- Sub\_off: '0' if input <256, 256 otherwise

The 'shift' parameter at the end is a configuration register which needs to be programmed. Typical programming value is 8 to generate a result in Q12 format (such that the range is 0 to1). This is usually the case when the denominator is greater than the numerator.

#### 6.9.4.6.3.3.4 LUT Based 12 to 8 Downsampling

In the final step the generated saturation value (12 bit) or the incoming RGB values (12 bit) needs to be scaled down to 8 bits using the linear LUT with 513 entries.

Figure 6-89 is a high level block diagram of the LUT block. It shows the weight calculation for the LUT assuming the incoming data can be of any bit depth between 8 and 12 bits.



FCP\_023

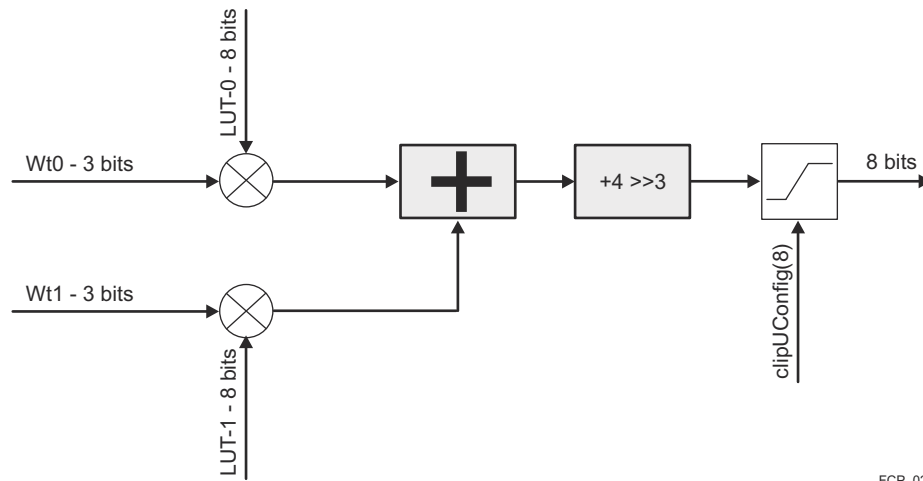
**Figure 6-89. LUT Based 12-8 Compression**

The LUTs are sized to provide 513 locations of data, to enable linear interpolation for all locations. The internal weights are sized as 3 bits since the maximum delta between 2 steps is only  $4095/512 = 8$ .

Figure 6-90 shows the high level block diagram of the interpolation process. The logic is designed to scale for bit width and can support anything from 8 to 12 bits of input to support different usecases. However, since the

step size is different depending on the input bit width, the bit selection and weight calculation logic is designed to account for that.

A shift operation can be performed instead of LUT to reduce bitwidth from 12 down to 8 bits.



FCP\_024

**Figure 6-90. Linear Interpolation on LUT**

The code below shows the addressing for the LUT as well as the weight calculation logic for supporting multiple bit widths at the input from 8 – 12 bits.

#### 12-8 Bit LUT Curve

```

CASE BIT_SEL
12 (12 bit data): Addr = Inp[3:11]; wt1 = Inp[2:0]
11 (11 bit data): Addr = Inp[2:10]; wt1 = Inp[1:0] & '0'
10 (10 bit data): Addr = Inp[1:9]; wt1 = Inp[0] & '00'
9 (9 bit data): Addr = Inp[0:8]; wt1 = '000'
8 (8 bit data): Addr = Inp[0:7]; wt1 = '000';
//wt0 is always calculated at 8 - wt1
wt0 = 8 - wt1
  
```

The input bit width (BIT\_SEL) is specified using a dedicated register for generating Y8 output. However for other paths (RGB/UV/Sat) the 8 bit conversion is done using either 12 bits (if data is tapped prior to contrast block) as the bitwidth or the clipping value after contrast block (VISS\_FCP\_FCC\_CFG\_2[12-9] CONTRASTBITCLIP, if data is tapped after contrast block).

#### 6.9.4.6.3.3.4 Histogram

The histogram can work on either one of the color components after the CCM-1 or the fourth color component routed directly from the Flexible CFA. The Histogram can also work on internally generated Luma value using simple averaging of RGB components. The Histogram module supports a local memory which can be read by the host processor. The Histogram provides data which can be used by the software to generate tone curves for the Contrast Stretch LUT.

The input to the Histogram module is calculated based on the control MMR as it follows:

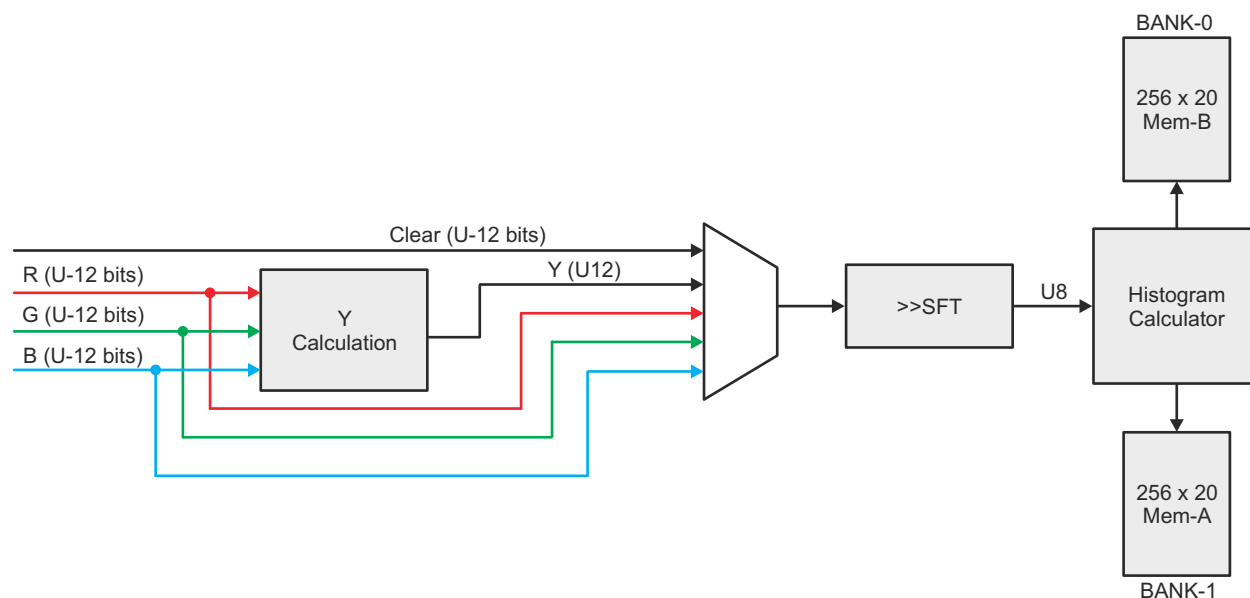
- 0: R
- 1: G
- 2: B
- 3: MuxC1\_4
- 4:  $(R+2 \cdot G+B) / 4$

The histogram module downsamples the image in the horizontal direction by a factor of 2 using simple averaging filter. This ensures that the memory access requirements of the module are limited to 1 read + 1 write access every 2 cycles (1 access/cycle) and thus can be achieved using a simple memory architecture.

The histogram module requires a ping-pong memory which is mapped to the configuration bus. This allows for histogram operation to run in concurrent with the Read operation from the host. For ease of software complexity, both the Ping/Pong memories map to the same address and it is the responsibility of the design to ensure that the muxing logic always take care of the correct access from both the host and the module. To detect error conditions, if the host is not able to read fast enough from the memory, an error interrupt should be generated whenever the host has initiated a read transfer from the first location of the memory but has not initiated a read transfer from the last location of the memory.

The Histogram module has 2 banks of memories each of 256x20 bits which are used in ping pong fashion. The input pixel has to be downshifted to match the bin size. The Histogram module has separate registers for creating an ROI using horizontal/vertical start postions as well as horizontal/vertical size.

Typically the ROI registers (VISS\_FCP\_FCC\_CFG\_HIST\_1 and VISS\_FCP\_FCC\_CFG\_HIST\_2) have to be set that at least one line is skipped for histogram calculation. The one line of time can be used by the histogram module to set the value of each of the bins to zero. This also puts a constraint on the histogram ROI to have a minmum width of 256 pixels.



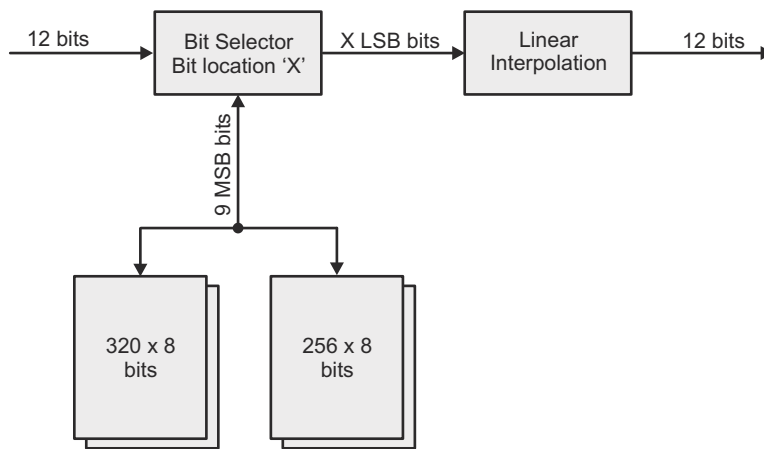
FCP\_026

**Figure 6-91. Histogram Module**

#### 6.9.4.6.3.3.5 Contrast Stretch / Gamma

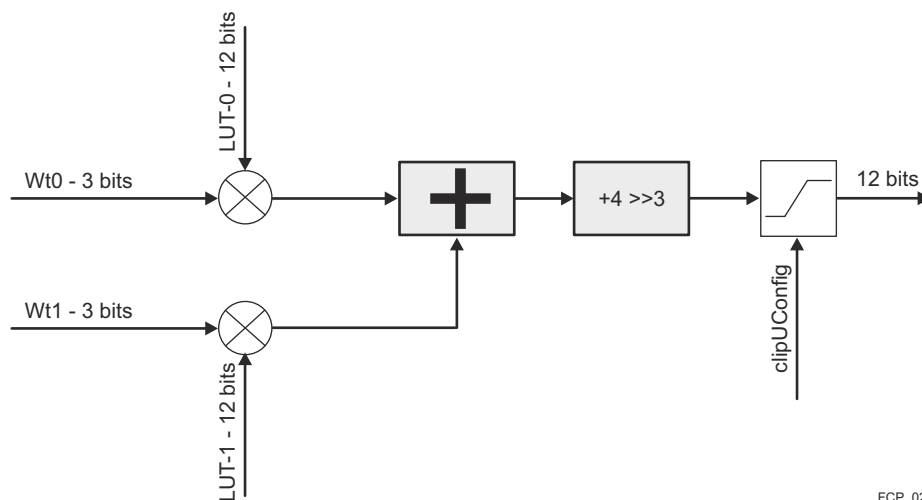
The logic of the Contrast/Gamma unit is based on the same as that of the 12-8 bit conversion LUTs. There are 3 copies of the logic, one for each color channel. The standard data flow allows for a 12 bit input and 12 bit unsigned output, however different bit width combinations are possible using the select bit of the LUT and the clip value. Each entry of the LUT table is 12 bits.

Figure 6-92 and Figure 6-93 show the block diagram and implementation of linear interpolation for the Contrast Stretch/Gamma module.



FCP\_027

**Figure 6-92. Contrast Enhancement Module**



FCP\_028

**Figure 6-93. Contrast Enhancement Linera Interpolation**

The code below shows the LUT addressing scheme which is similar to the 12-8 bit conversion module. The configurable clip at the end of processing allows for less than 12 bit output to be supported directly from the module.

#### Contrast Enhancement, LUT Addressing

```
CASE BIT_SEL
0 (12 bit data): Addr = Inp[3:11]; wt1 = Inp[2:0]
1 (11 bit data): Addr = Inp[2:10]; wt1 = Inp[1:0] & '0'
2 (10 bit data): Addr = Inp[1:9]; wt1 = Inp[0] & '00'
3 (9 bit data): Addr = Inp[0:8]; wt1 = '000'
4 (8 bit data): Addr = Inp[0:7]; wt1 = '000';
//Wt0 is always calculated at 8 - wt1
Wt0 = 8 - wt1
```

#### 6.9.4.6.3.3.6 RGB-YUV Conversion

The RGB-YUV conversion is a matrix based conversion from 12-bit RGB to 12-bit YUV. The output after conversion is YUV444.

The conversion is carried out using the equation on [Figure 6-94](#).

$$Y(x,y) = WRY \times R(x,y) + WGY \times G(x,y) + WBY \times B(x,y) + Offset\_1$$

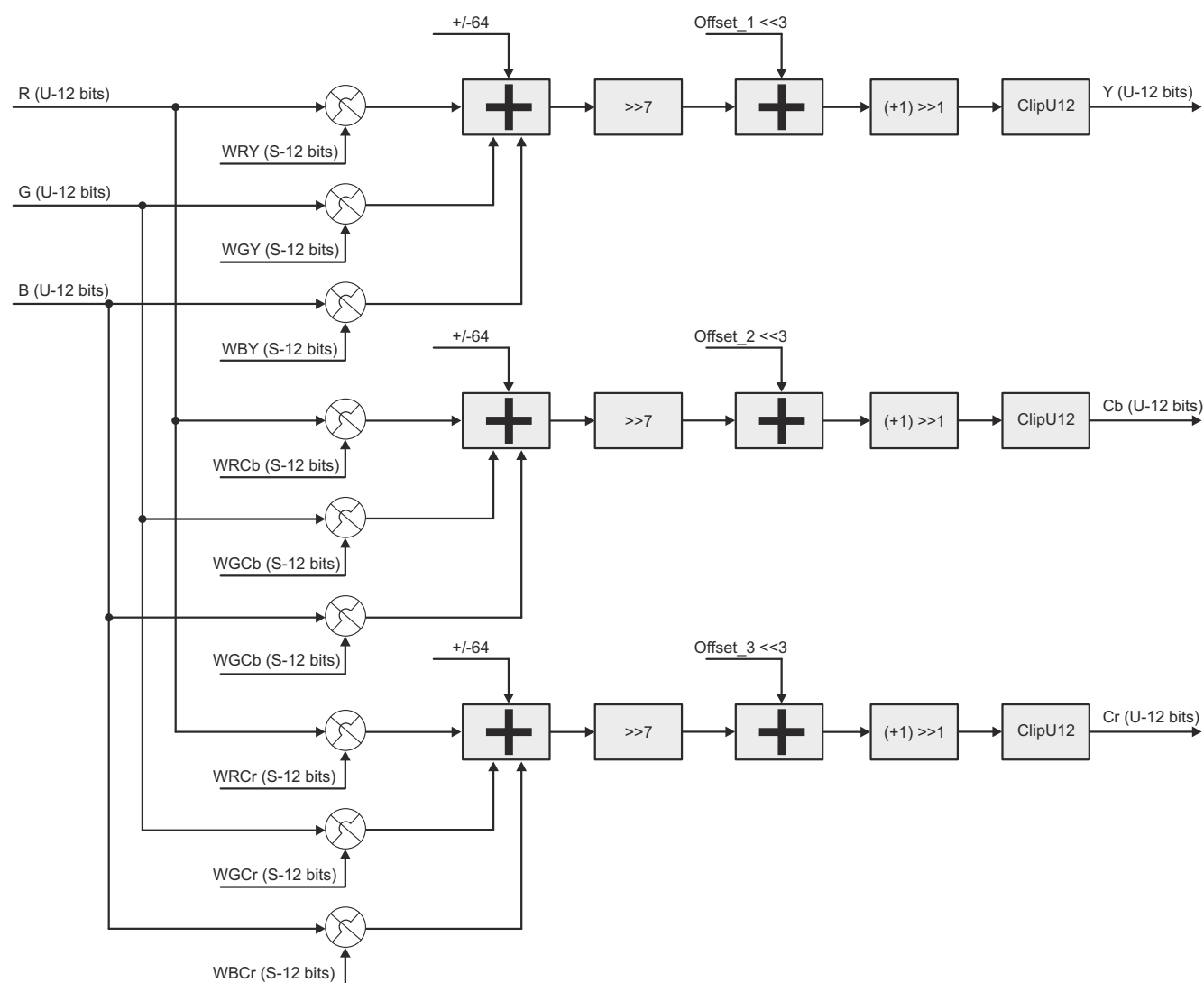
$$Cb(x,y) = WRCb \times R(x,y) + WGCb \times G(x,y) + WBCb \times B(x,y) + Offset\_2$$

$$Cr(x,y) = WRCr \times R(x,y) + WGCr \times G(x,y) + WBCr \times B(x,y) + Offset\_3$$

**Figure 6-94. RGB-to-YUV Conversion**

In this equation, the weights are signed 12 bits (S12Q8) with 8 bit of fraction precision, representing a range of -8 to +7.996. The offsets are signed 13 bits.

Figure 6-95 shows the implementation of the RGB to YUV module. The output after conversion is YUV444, as such it is followed by a chroma downsampling stage where in the Chroma resolution is reduced by half in both the horizontal and vertical direction.



FCP\_030

**Figure 6-95. RGB to YUV**

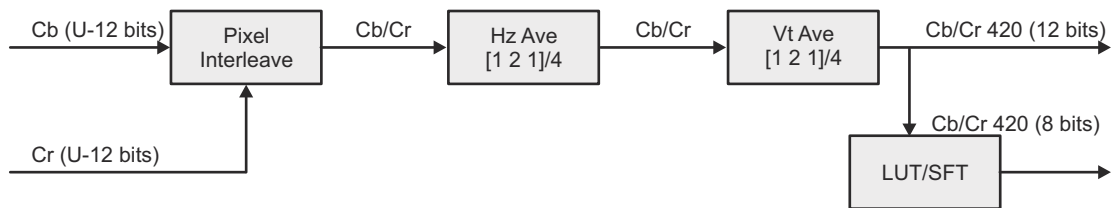


#### 6.9.4.6.3.4 444-422/420 Chroma Down-sampler

This module down-samples the chroma components from 444-420 format; effectively reducing the chroma resolution to 50% in both the horizontal and vertical direction.

The downsampling is performed using a 3 Tap filtering [1 2 1] followed by decimation. The conversion logic generates chroma in co-sited/co-located format in both the horizontal and vertical direction.

Figure 6-96 shows the block diagram of the format conversion module. The Chroma output can be optionally downshifted from 12 bits to 8 bits using the LUT/SFT module. Only one copy of the Horizontal averaging filter is required since there is a decimation step following the filtering. Only even locations of both Cb/Cr are retained while the odd locations are decimated, however from design perspective the Hz Averager module works on Cb on even cycles and Cr on odd cycles.



FCP\_031

**Figure 6-96. 444 to 420 Module**

Additionally another mode of operation is to support planar YUV422 mode. In this case the chroma is only downsampled in the horizontal direction and the vertical averaging/downsampling is disabled.

The mode is referred to pseudo 422 since actual 422 mode requires the data to be interleaved between the Y and the chroma channels. In this case the interleaving of Y & C is handled by LSE whereas the Flexible CC module will only generate Y & Cb/Cr (interleaved) planes separately.

#### 6.9.4.6.4 FCP Interrupts

The FCP module has three interrupts, as it follows:

- LUT\_CFG\_ERR - 16-12 LUT is written during active window
- CFA\_PIX\_ERR - Access to line memories by CFG during active window
- CFA\_MMR\_ERR - Non-Shadowed registers are written during active window

There are two additional events:

- SOL\_EVENT - Start of line processing for Flexible CFA, triggered when the line enters the Flexible CFA
- SOF\_EVENT - Start of frame processing for Flexible CFA, triggered when the frame enters the Flexible CFA

For information about the FCP interrupt events, see Section *VISS Top Level Functional Description*.

#### 6.9.4.6.5 FCP Programmer's Guide

##### 6.9.4.6.5.1 HWA Core Programming Details

The following fields should be programmed correctly for the FCP module to function.

- Input LUT
  - Use the LUT to reduce the bitwidth of the data to 12 bits so that it can pass through the Flexible CFA pipe.
  - If the GLBCE module is disabled, set the bit width and enable the LUT.
- Flexible CFA
  - Program the Flexible CFA kernels based on the input data pattern. Program only 3 channels for Bayer sensors and up to 4 for other non-standard formats like RGB-IR.
  - Set up the correct thresholds (use only one set for Bayer type sensors and two sets for non-standard formats like RGB-IR).
- Flexible CC
  - Set up the Flexible CC for either visual or analytics data flow.

- For visual, program the CCM-1 to be used as RGB2RGB and Contrast to be used as Gamma with clip at 10 bits. Only the 8-bit outputs are used and the Y8/C8 data is generated using shift down from 10 to 8 (instead of LUT).
- For analytics, CCM-1 is programmed depending on sensor input format and the histogram is used to populate the contrast table. 12-bit outputs are used however Y8/C8 can be generated using 12 to 8 LUT based downshifting. The correct Saturation generation parameters should also be set.

#### **6.9.4.6.5.2 HWA HTS Programming Details**

For details of programming, see *HWA Thread Scheduler (HTS)* and [Section 6.9.3](#), *VPAC Subsystem*.

#### **6.9.4.6.5.3 HWA Data Transfer Programming Details**

For details of programming for PARAM space, see *Section UDMA Specification*.

#### **6.9.4.6.5.4 Initialization Sequence**

For information, see [Section 6.9.3](#), *VPAC Subsystem*.

#### **6.9.4.6.5.5 Real-time Operating Requirements**

For information, see [Section 6.9.3](#), *VPAC Subsystem*.

#### **6.9.4.6.5.6 Power Up/Down Sequence**

For information, see [Section 6.9.3](#), *VPAC Subsystem*.

### 6.9.4.7 VISS Edge Enhancer (EE)

This section describes the Edge Enhancer module.

#### 6.9.4.7.1 Edge Enhancer Introduction

The Edge Enhancer module is used to enhance the visual quality of the image by increasing the sharpness of the image. The module only works on Y (Luma) data and uses a combination of 2D HPF (High-Pass Filter) filtering to detect edges and then apply them on the input image.

The Edge Enhancer consists of two separate blocks internally, the edge enhancer filter and the edge sharpener filter. The module can be programmed to either select the output of one of the filters or to use a blend of both.

Figure 6-97 below shows a conceptual block diagram of the edge enhancer filter. The input data is assumed to be a 12 bits/pixel. The EE module is tuned to process 8-bit images (MSB aligned) but has support to process 10-bit and 12-bit images as well.

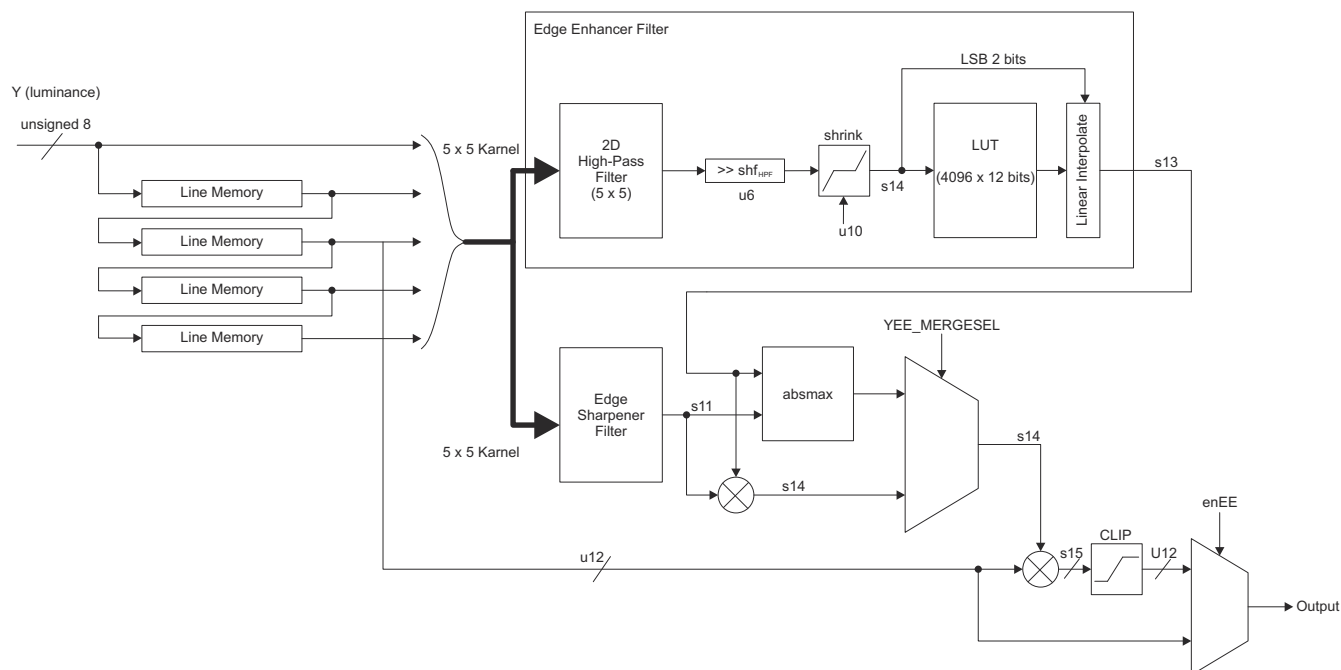


Figure 6-97. Edge Enhancer Block Diagram

#### 6.9.4.7.1.1 Edge Enhancer Filter

The edge enhancer filter works on a  $5 \times 5$  HPF filter (M) with programmable coefficients. The filter is symmetric as such only  $3 \times 3$  coefficients are programmable.

Here, M is a  $5 \times 5$  matrix with programmable coefficients specified by VISS\_FCP\_EE\_YEE\_COEF\_Ri\_Cj (i and j are 0, 1, or 2). The shift value (shf<sub>HPF</sub>) is specified by VISS\_FCP\_EE\_YEE\_SHIFT[5-0] YEE\_SHIFT.

$$HPF(h, v) = \left( \sum_{j=-2}^2 \sum_{i=-2}^2 M_{i,j} Y(h+i, v+j) \right) \gg shf_{HPF}$$

$$M = \begin{pmatrix} M_{2,2} & M_{1,2} & M_{0,2} & M_{1,2} & M_{2,2} \\ M_{2,1} & M_{1,1} & M_{0,1} & M_{1,1} & M_{2,1} \\ M_{2,0} & M_{1,0} & M_{0,0} & M_{1,0} & M_{2,0} \\ M_{2,1} & M_{1,1} & M_{0,1} & M_{1,1} & M_{2,1} \\ M_{2,2} & M_{1,2} & M_{0,2} & M_{1,2} & M_{2,2} \end{pmatrix}$$

**Figure 6-98. Edge Enhancer Linear Filter**

The HPF value is shrink by a threshold value,  $threshold_{HPF}$  ( $u6 = 10$ ), specified by VISS\_FCP\_EE\_YEE\_E\_THR register, and clipped to signed 14 bits to get the index for the LUT. The MSB 12 bits of the index are used for the LUT address (4k locations) whereas the LSB 2 bits acts as weight to the linear interpolation logic.

$$index = clip(shrink(HPF, threshold_{HPF}), -8k, 8k)$$

$$shrink(x, threshold) = \begin{cases} x + threshold & x < -threshold \\ 0 & -threshold \leq x \leq threshold \\ x - threshold & threshold < x \end{cases}$$

$$clip(x, limit_{LOW}, limit_{HIGH}) = \begin{cases} -limit_{LOW} & x < -limit_{LOW} \\ x & -limit_{LOW} \leq x \leq limit_{HIGH} \\ limit_{HIGH} & limit_{HIGH} < x \end{cases}$$

**Figure 6-99. Edge Enhancer Indexing**

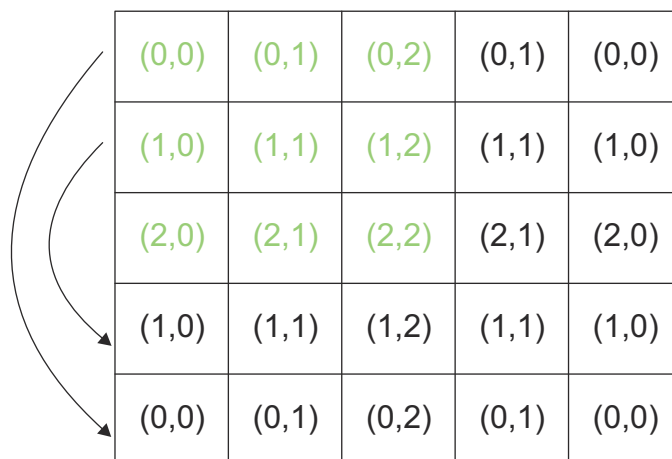
The edge enhancement intensity is looked up from the LUT, where A is the MSB 12 bits of the index.

$$E_{int} = Interpolate(LUT[A], LUT[A+1])$$

**Figure 6-100. Edge Intensity LUT Formula**

The  $5 \times 5$  HPF filter is not fully programmable, rather only the top left quadrant is programmable coefficients. The rest of the kernel is symmetric along the y and the x axis.

**Figure 6-101** below depicts how the kernels are implemented, with the kernel locations marked inside the boxes. The green color shows the kernels that are programmable.



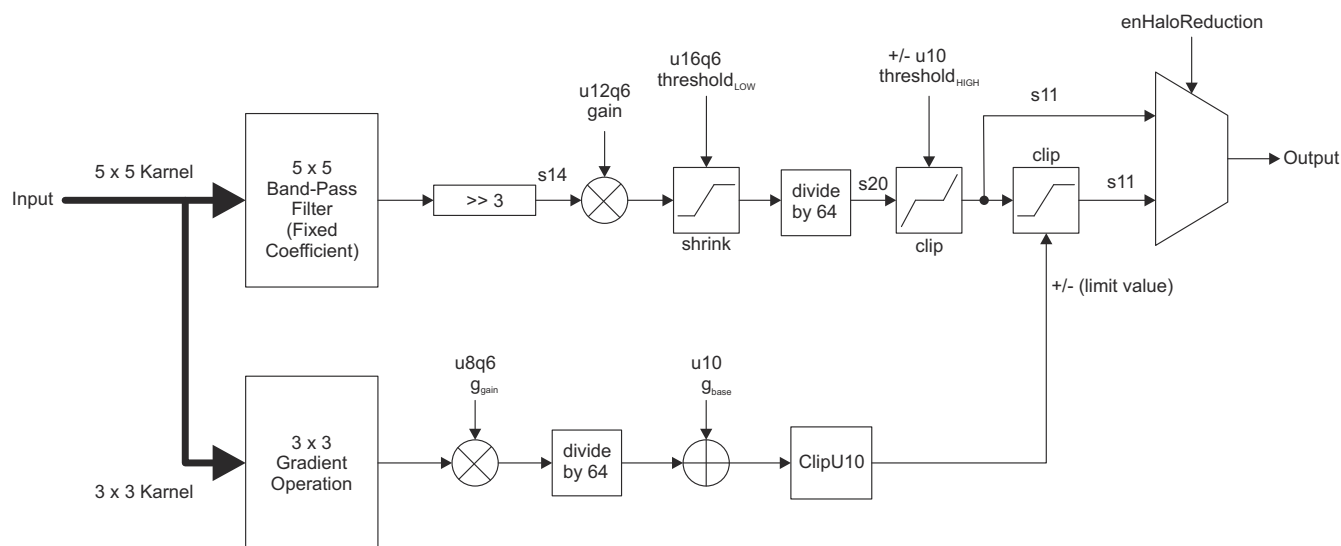
**Figure 6-101. 5 × 5 HPF Filter**

The border conditions are handled by data duplication on each of the borders when the desired pixel goes out of range.

The output of the LUT is the final output of the edge enhancer filter.

#### 6.9.4.7.1.2 Edge Sharpener Filter

Figure 6-102 below shows a high level block diagram of the edge sharpener function.



**Figure 6-102. Edge Sharpener Function**

In edge sharpener filter module, enabled when VISS\_FCP\_EE\_YEE\_MERGESEL[0] YEE\_MERGESEL = 1, edge clarity is enhanced without producing a halo artifact. In this module, edge intensity is derived by the following 2D linear filter with fixed coefficients shown in Figure 6-103.

$$S_{i,j} = \begin{pmatrix} 0 & -1 & -2 & -1 & 0 \\ -1 & 0 & 2 & 0 & -1 \\ -2 & 2 & 8 & 2 & -2 \\ -1 & 0 & 2 & 0 & -1 \\ 0 & -1 & -2 & -1 & 0 \end{pmatrix}$$

$$sharpness(h,v) = \text{clip} \left( \text{shrink} \left( g \times \left( \sum_{j=-2}^2 \sum_{i=-2}^2 S_{i,j} Y(h+i, v+j) \right) \gg 3 \right), threshold_{LOW} \right) \gg 6, -threshold_{HIGH}, threshold_{HIGH}$$

**Figure 6-103. Edge Sharpener Details**

The gain ( $g$ ) and threshold values for shrink/clip function ( $threshold_{LOW}$ ,  $threshold_{HIGH}$ ) are determined by register values (VISS\_FCP\_EE\_YES\_E\_GAIN, VISS\_FCP\_EE\_YES\_E\_THR1, VISS\_FCP\_EE\_YES\_E\_THR2). The precision of  $g$  is in U12Q6,  $threshold_{LOW}$  is in U16Q6 and that of  $threshold_{HIGH}$  is in U10.

This edge intensity is then clipped by a threshold value in the formula shown in [Figure 6-104](#).

$$S_{int} = \begin{cases} \text{clip}(sharpness, -grad, grad) & \text{Halo reduction on} \\ sharpness & \text{Halo reduction off} \end{cases}$$

**Figure 6-104. Edge Intensity Clipping Formula**

The threshold value ( $grad$ ) is a function of the activity around the target pixel, which is derived from gradient values. Gain ( $g_{grad}$ ) and offset ( $g_{base}$ ) are specified by VISS\_FCP\_EE\_YES\_G\_GAIN and VISS\_FCP\_EE\_YES\_G\_OFT.

$$grad = g_{grad} (\min(gx_L, gx_R) + \min(gy_T, gy_B)) \gg 6 + g_{base}$$

$$gx_L = \text{abs} \left( \sum_{j=-1}^1 \sum_{i=-1}^1 G_{xL}(i,j) \cdot y(h+i, v+j) \right)$$

$$gx_R = \text{abs} \left( \sum_{j=-1}^1 \sum_{i=-1}^1 G_{xR}(i,j) \cdot y(h+i, v+j) \right)$$

$$gy_T = \text{abs} \left( \sum_{j=-1}^1 \sum_{i=-1}^1 G_{yT}(i,j) \cdot y(h+i, v+j) \right)$$

$$gy_B = \text{abs} \left( \sum_{j=-1}^1 \sum_{i=-1}^1 G_{yB}(i,j) \cdot y(h+i, v+j) \right)$$

$$G_{xL} = \frac{1}{4} \begin{pmatrix} 1 & -1 & 0 \\ 2 & -2 & 0 \\ 1 & -1 & 0 \end{pmatrix}$$

$$G_{xR} = \frac{1}{4} \begin{pmatrix} 0 & 1 & -1 \\ 0 & 2 & -2 \\ 0 & 1 & -1 \end{pmatrix}$$

$$G_{yT} = \frac{1}{4} \begin{pmatrix} 1 & 2 & 1 \\ -1 & -2 & -1 \\ 0 & 0 & 0 \end{pmatrix}$$

$$G_{yB} = \frac{1}{4} \begin{pmatrix} 0 & 0 & 0 \\ 1 & 2 & 1 \\ -1 & -2 & -1 \end{pmatrix}$$

**Figure 6-105. Edge Threshold Value Formula**

Capping with gradient value prevents overly enhancing edges, and suppresses halo artifacts around edges.

#### 6.9.4.7.1.3 Merge Block

The output from edge enhancer filter and edge sharpener filter are merged with the following function.

$$E_{merge} = \begin{cases} E_{int} + S_{int} & VISS\_FCP\_EE\_YEE\_MERGESEL[0]YEE\_MERGESEL = 0 \\ \text{absmax}(E_{int}, S_{int}) & VISS\_FCP\_EE\_YEE\_MERGESEL[0]YEE\_MERGESEL = 1 \end{cases}$$

$$\text{absmax}(x, y) = \begin{cases} x & \text{abs}(y) \leq \text{abs}(x) \\ y & \text{otherwise} \end{cases}$$

**Figure 6-106. Edge Enhancer and Sharpener Merger Formula**

The  $E_{merge}$  value is added to the Y input value to make the final output.

The contribution of each of the filters can be modified. The edge sharpener filter block can be turned off or reduced in strength using the gain fields. The edge enhancer filter block can be disabled using the LUT.

#### 6.9.4.7.2 Edge Enhancer Programming Model

Table 6-133 captures the programming parameters and the programming model of the Edge Enhancer module.

#### Note

UMQN implies M bits with N bits of fraction and M-N bits of Integer.

For more information, see *Edge Enhancer Registers*.

**Table 6-133. Edge Enhancer Programming Parameters**

Parameter Name	New Precision	Tuning Methodology
VISS_FCP_EE_EE_CFG_0[12-0] WIDTH	13 bits	Width of the image
VISS_FCP_EE_EE_CFG_0[28-16] HEIGHT	13 bits	Height of the image
VISS_FCP_EE_EE_ENABLE[0] YEE_ENABLE	1	Enable Module
VISS_FCP_EE_YEE_SHIFT[5-0] YEE_SHIFT	U6	-
VISS_FCP_EE_YEE_COEF_R0/1/2_C0/1/2[9-0] YEE_COEF_R0/1/2_C0/1/2	Signed 10	-
VISS_FCP_EE_YEE_E_THR[9-0] YEE_E_THR	U10	Shrink Threshold before LUT, scale by 16×
VISS_FCP_EE_YEE_MERGESEL[0] YEE_MERGESEL	1 bit	Mergesel: Off(0), On(1)
VISS_FCP_EE_YES_E_HAL[0] YES_E_HAL	1 bit	Halo Reduction; Off(0)/On(1)
VISS_FCP_EE_YES_G_GAIN[7-0] YES_G_GAIN	U8Q6	Usually 1.5 times YES_E_GAIN
VISS_FCP_EE_YES_G_OFT[9-0] YES_G_OFT	U10	Scale by 16
VISS_FCP_EE_YES_E_GAIN[11-0] YES_E_GAIN	U12Q6	Same as previous
VISS_FCP_EE_YES_E_THR1[15-0] YES_E_THR1	U16Q6	Scale by 16
VISS_FCP_EE_YES_E_THR2[9-0] YES_E_THR2	U10	Scale by 16



### 6.9.5 VPAC Lens Distortion Correction (LDC) Module

This section describes the Lens Distortion Correction (LDC) module.

#### 6.9.5.1 LDC Overview

The LDC module deals with lens geometric distortion issues in the camera system. The distortion can be common optical distortions, such as barrel, pin-cushion, or fisheye distortion. LDC is not limited to just these types of distortions. Affine transformation support is also added to support the image and video stabilization application, where multiple images of the same scene need to be aligned. Perspective warp is an extension to affine transform and offers additional capabilities. Perspective transformations can align two images that are captured from different camera viewpoints or locations. This can also be used to rectify the left and right images of a stereo camera to reduce the complexity of a disparity computation. Perspective transformations can also generate new viewpoints.

##### 6.9.5.1.1 LDC Features

- Autonomous memory-to-memory operation
  - Tile based processing
- Generic mesh based distortion model to correct multiple distortion types
- Perspective warp transformation for perspective correction; affine transform is a subset of perspective warp and supports scaling and rotation
- Support multiple input and output YUV data formats:
  - Among the supported data formats are UYVY (YCbCr 422), NV12 (YCbCr 420) and NV21 (YCbCr 420)
- Support up to 8192 x 8192 image dimension
- Pixel Interpolation
  - Bi-cubic interpolation for Y and bilinear interpolation for Cb/Cr
  - Bilinear interpolation mode to offer double throughput
- Performance
  - 1 cyc/pixel (bilinear)
  - 2 cyc/pixel (Bi-cubic)
- Support to process either Luma only or Chroma only modes in YUV420 input data format to save bandwidth
- Support for independent block size in multiple regions (up to 9 regions)
- Dual output channels for programmable output pixel size
- ECC support on Mesh Data internal storage memories and width conversion LUTs on dual output channel
- Interface to HTS module for block level synchronization with other IPs
- Circular addressing support for Output Write (for SL2 interface) and Input Read
- Event for Block/Frame completion and Error scenarios

#### 6.9.5.2 LDC Functional Description

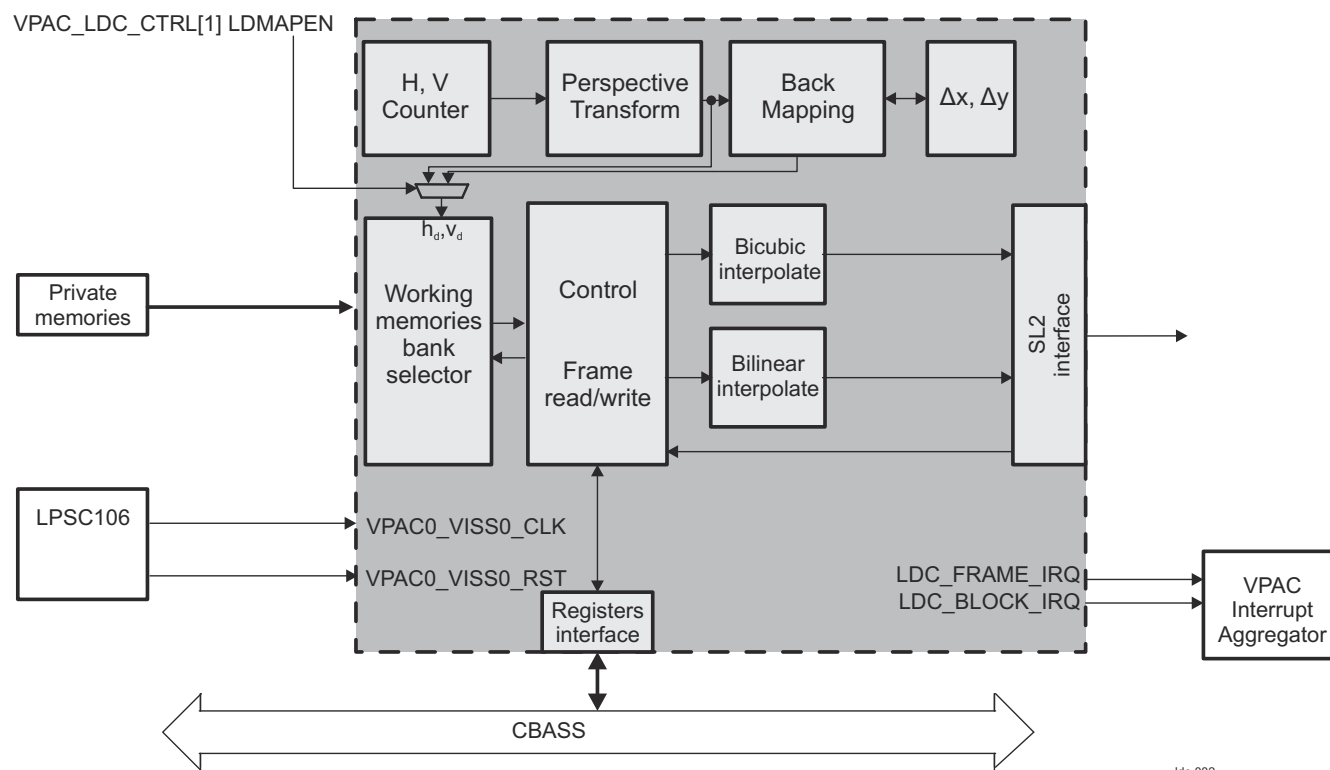
##### 6.9.5.2.1 LDC Block Diagram

Most digital cameras suffer from some degree of nonlinear geometric distortion. A spatial transformation is required to correct the distortion. In automotive applications, cameras use wide angle lenses, including fisheye lenses to provide 180+° field of view. To visually present the scene to the user in an easy-to-consume representation, these distortions need to be corrected

Back mapping gives coordinates of the distorted image as a function of coordinates of the undistorted output image. Correction involves back-mapping each output pixel to a location in the source distorted image, and thus the corrected image is fully populated. As the distorted pixel locations mostly fall onto fractional coordinates, correction involves interpolation.

As shown in [Figure 6-107](#), the LDC consists of a back mapping block, a x/y offset table, image buffer interface, buffer, an interpolation block, and SL2 interface.

[Figure 6-107](#) is a block diagram of the LDC.



**Figure 6-107. LDC Block Diagram**

Given the coordinates of the undistorted image, the corresponding coordinates of the distorted image are calculated by combining the output coordinates and the offsets from the offset table. Distorted pixels are read from the image buffer, and buffered for the bilinear interpolation. After the interpolation, corrected image is written back to the SL2 memory.

The LDC processes the image in small two-dimensional (2D) blocks. The software configures appropriate parameters, then initiates the LDC function by writing to an LDC register. The LDC controls the sequencing through 2D blocks, DMA transfers, and computation to process an entire image autonomously. Interrupt, if enabled, is asserted at the completion of the image.

The LDC write is controlled by HTS at block level.

To start the LDC, software must set the VPAC\_LDC\_CTRL[0] LDC\_EN bit to 1. The VPAC\_LDC\_CTRL[2] BUSY bit is a status that reflects LDC activity.

#### 6.9.5.2.2 LDC Clocks

Complete VPAC LDC operates on single clock (that is, VPAC0\_LDC0\_CLK). Except the signals that are coming from LPSC, all other LDC sub-block clock domains are synchronous to VPAC0\_LDC0\_CLK.

#### 6.9.5.2.3 LDC Interrupts

LDC generates interrupt events and these are made available to the VPAC. VPAC implements compatible interrupt function via the interrupt aggregator, see chapter [Section 6.9.3 VPAC Subsystem](#).

All the interrupts are active high and pulsed for one VPAC0\_VISS0\_CLK cycle. Following interrupts events are generated. Out of these only ECC interrupt should be treated as real interrupt and others are used as events for the interrupt aggregator at VPAC level.

**Table 6-134. LDC Interrupt Events**

Interrupt Event	Type	Category	Description
PIX_IBLK_OUTOFBOUND	pulse	func	PIX_IBLK_OUTOFBOUND pulse interrupt event

**Table 6-134. LDC Interrupt Events (continued)**

Interrupt Event	Type	Category	Description
MESH_IBLK_OUTOFBOUND	pulse	func	MESH_IBLK_OUTOFBOUND pulse interrupt event
IFR_OUTOFBOUND	pulse	func	IFR_OUTOFBOUND pulse interrupt event
INT_SZOVF	pulse	func	INT_SZOVF pulse interrupt event
VPAC_LDC_FR_DONE_EVT	pulse	func	LSE frame done event
VPAC_LDC_SL2_WR_ERR	pulse	func	LSE SL2 VBUSM Write Error interrupt event
PIX_IBLK_MEMOVF	pulse	func	Input pixel block internal memory overflow event
MESH_IBLK_MEMOVF	pulse	func	Input mesh block internal memory overflow event
VPAC_LDC_VBUSM_RD_ERR	pulse	func	Error event on VBUSM Read Interface
UNCORR_PULSE	pulse	error	ECC Aggregator uncorrectable error, pulse
UNCORR_LEVEL	level	error	ECC Aggregator uncorrectable error, level
CORR_PULSE	pulse	error	ECC Aggregator correctable error, pulse
CORR_LEVEL	level	error	ECC Aggregator correctable error, level

#### 6.9.5.2.3.1 LDC Interrupt Events Description

##### 6.9.5.2.3.1.1 PIX\_IBLK\_OUTOFBOUND

Generated when back mapping of block non-corner pixel co-ordinate goes out of the pre-computed input bounding box (i.e. data required for processing is not available in private pixel storage). This event is generated for the first occurrence of error in each frame. Upon this error, PIX\_PAD setting needs to be reviewed. Upon pixel out of bound, pixel co-ordinate will be clipped to the pre-fetched block boundary. Usually no special treatment is required on this error event. Only few pixels are expected to be have this condition..

##### 6.9.5.2.3.1.2 MESH\_IBLK\_OUTOFBOUND

Generated when mesh data required for non-corner mesh co-ordinate goes out of the pre-computed mesh bounding box (i.e. mesh data required for processing is not available in internal mesh storage). This event is generated for the first occurrence of error in each frame. This error condition is handled inside design by clipping the mesh data location to the pre-fetched block boundary. No special treatment is expected on this error event.

##### 6.9.5.2.3.1.3 IFR\_OUTOFBOUND

Generated when back mapped input pixel co-ordinate goes out of valid input frame boundary or co-ordinates post Perspective warping goes out of valid Mesh frame size (Frame size for which mesh data is available). This event is generated for the first occurrence of error in each frame. In the event of error condition, design will clip the co-ordinates to valid frame boundary. It is recommended to review the source of the error to make sure that it is expected.

##### 6.9.5.2.3.1.4 INT\_SZOVF

Generated when intermediate variables of affine or perspective transform operation goes above what hardware supports (U16Q3). This event is generated for the first occurrence of error in each frame. In the event of error condition, design will clip the variables the size to what HW supports. Upon this error, Affine and Perspective coefficients need to be reviewed.

##### 6.9.5.2.3.1.5 VPAC\_LDC\_FR\_DONE\_EVT

Generated on once complete output frame is written into SL2.

#### 6.9.5.2.3.1.6 VPAC\_LDC\_SL2\_WR\_ERR

This event is generated whenever the LDC SL2 Output VBUSM Interface write status is something other than 0 (success).

#### 6.9.5.2.3.1.7 PIX\_IBLK\_MEMOVF

Generated when input pixel block memory storage requirement is more than internal pixel memory or each side of input block size goes above 1023. This event will be generated once per block in case of error. In the event of error condition, one or more output pixel blocks corruption. Upon this error, Output Block size setting needs to be reviewed.

#### 6.9.5.2.3.1.8 MESH\_IBLK\_MEMOVF

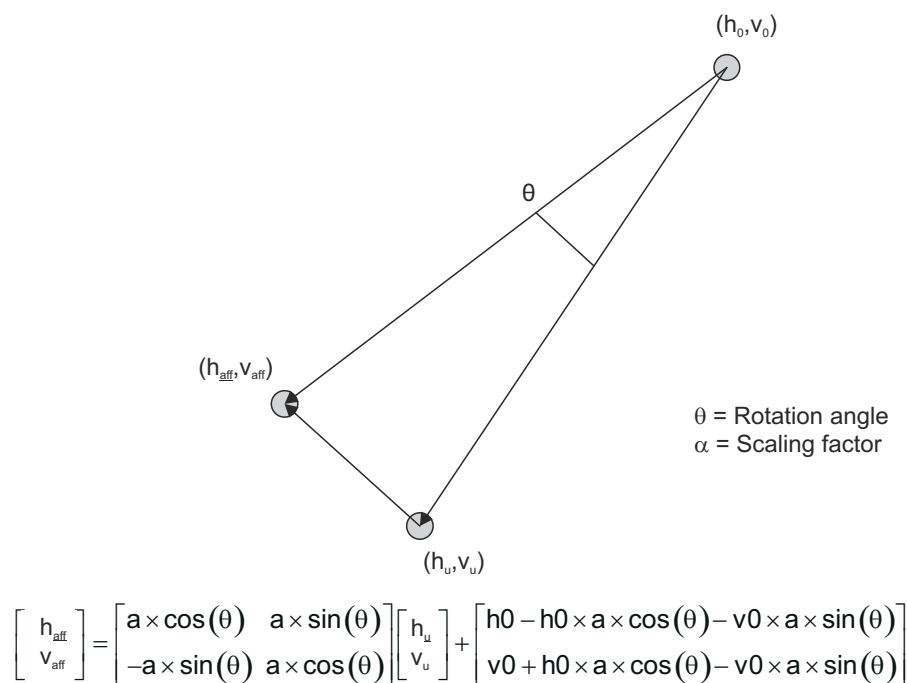
Generated when Mesh block internal storage requirement is more than internal Mesh memory or either side of input block size goes above 1010. This event will be generated once per block in case of error. In the event of error condition, one or more output pixel blocks corruption. Upon this error, Output Block size or Affine/Pwarp parameters or mesh sub-sampling factor setting needs to be reviewed.

#### 6.9.5.2.3.1.9 VPAC\_LDC\_VBUSM\_RD\_ERR

This event is generated whenever the LDC Read VBUSM Interface status is something other than 0 (success).

#### 6.9.5.2.4 LDC Affine Transform

The LDC first performs an affine transform, which can be used for rotations and scaling, as shown in [Figure 6-108](#). The output of the pixel coordinate is used as the input address to the affine transform.



ldc-010

**Figure 6-108. LDC Affine Transformation**

The mapping from destination coordinate to the source coordinate is expressed as:

$$h_{aff} = a \times h_u + b \times v_u + c$$

$$v_{aff} = d \times h_u + e \times v_u + f$$

This transform allows full rotation, expansion, contraction, and translation. The parameters {a, b, d, e} are in S16Q12 (signed number on 14 bits with 2 bits for integer and 12 bits of fraction) format, and parameters {c, f}

are in S16Q3 format (signed number on 16 bits with 13 bits for integer and 3 bits of fraction). These parameters must be set in following bit fields:

- a = VPAC\_LDC\_AFF\_AB[15:0] A
- b = VPAC\_LDC\_AFF\_AB[31:16] B
- c = VPAC\_LDC\_AFF\_CD[15:0] C
- d = VPAC\_LDC\_AFF\_CD[31:16] D
- e = VPAC\_LDC\_AFF\_EF[15:0] E
- f = VPAC\_LDC\_AFF\_EF[31:16] F

#### 6.9.5.2.5 LDC Perspective Transformation

When a camera is viewing a scene from two different positions or when multiple cameras are viewing the scene from different positions, a transformation between the two viewing angles is needed to align the images. Under specific conditions, the class of geometric transformations known as homography, or planar-perspective transformation, will capture the geometric relationship between the images accurately. Common applications of homography transforms are to align (or stitch) multiple frames of the same scene to compute a panoramic output image. A second application is the alignment of planar surfaces in the world. Finally, perspective transforms are also useful in computing depth maps from a stereo image pair. By rectifying the two views, the search to compute disparity between the two views is simplified to a 1-D search problem. The homography is defined by a 3x3 transformation matrix, as in

$$h_{aff} = a * h_u + b * v_u + c \quad (3)$$

$$v_{aff} = d * h_u + e * v_u + f \quad (4)$$

$$z = \max(0, g * h_u + h * v_u + 1) \quad (5)$$

$$h_p = h_{aff} / z \quad (6)$$

$$v_p = v_{aff} / z. \quad (7)$$

The affine transform is a subset of the perspective transformation. By setting  $g = h = 0$ ,  $h_p = h_{aff}$  and  $v_p = v_{aff}$ .

In image alignment applications, the homography matrices are computed by locating corresponding points in the two frames and estimating the matrix parameters to transform the set of points in one frame onto the corresponding points in the second frame. In the stereo rectification application, the matrix is determined (pre-computed) at the calibration step and remains fixed.

When LDC is requested to provide a change in the frame size between the input and output, the affine/perspective parameters must be programmed to implement the coordinate scaling (see for programming details). In the following sections, the mesh table is used to program distortion correction. The table sizes are defined based on the total output frame size.

#### 6.9.5.2.6 LDC Lens Distortion Back Mapping

In YCbCr mode, the offset table defines a (x,y) vector for a regular grid of output points. The grid can be fully sampled or down sampled. A fully sampled grid will define an offset vector for every output pixel, defining exactly where to fetch the input data to compute the output pixel. This is the most precise definition and can capture rapidly changing offset tables. The drawback is that it will require a large amount of memory bandwidth as the LDC engine will be reading offset values for every output pixel. Since most offset tables are not expected to change rapidly in a small spatial region, LDC supports reading a subsampled offset table. Offset tables can be subsampled by powers of two in both horizontal and vertical directions and the subsampling factor is set in the register, VPAC\_LDC\_MESHTABLE\_CFG[2-0] M. This mode conserves memory bandwidth by reducing the amount of data read to describe the offset vectors, but requires more hardware to interpolate the missing offset vectors. LDC supports bilinear interpolation to interpolate missing offset vectors.

The mapping procedure is described by the following series of equations. Given an output pixel at location, we compute the input pixel location.

$$x_c = \text{CLIP}(x_o, 0, 8 \times W - 1), \text{ W: Mesh Frame Width} \quad (8)$$

$$y_c = \text{CLIP}(y_o, 0, 8 \times H - 1), \text{ H: Mesh Frame Height} \quad (9)$$

$$\text{Mask} = (1 \ll (3 + M)) - 1 \quad (10)$$

$$i_{x0} = (x_c \gg (3 + M)), f_x = x_c \& \text{Mask} \quad (11)$$

$$i_{y0} = (y_c \gg (3 + M)), f_y = y_c \& \text{Mask} \quad (12)$$

Clip new locations to previously fetched mesh block boundaries.

$$i_x = \text{CLIP}(i_{x0}, \text{block\_startmx}, \text{block\_endmx} - 1) \quad (13)$$

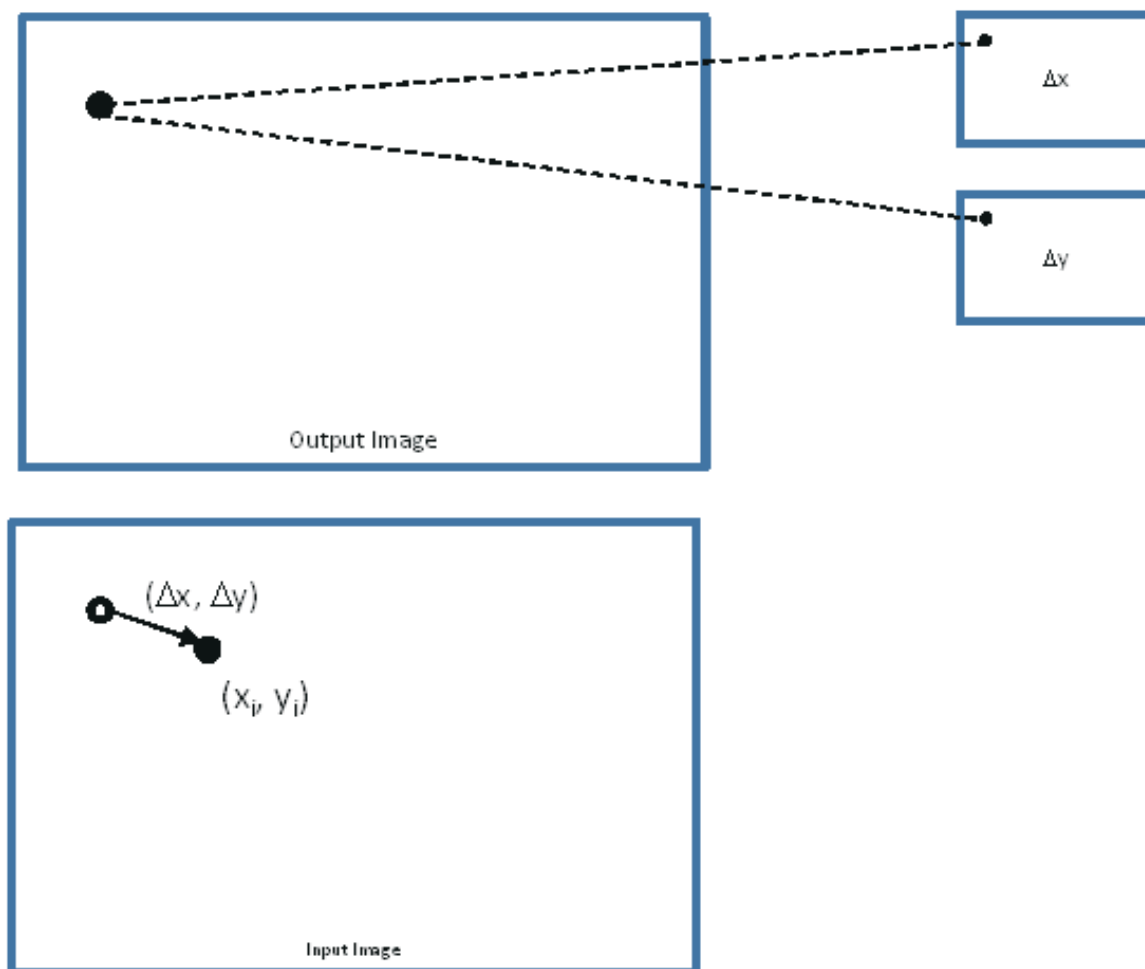
$$i_y = \text{CLIP}(i_{y0}, \text{block\_startmy}, \text{block\_endmy} - 1) \quad (14)$$

$$\begin{aligned}
 \text{Ofst0} &= \text{READ32}(\text{OfstTable}[i_x, i_y]) \\
 \Delta x_0 &= \text{MSB16}(\text{Ofst0}) \\
 \Delta y_0 &= \text{LSB16}(\text{Ofst0}) \\
 \\
 \text{Ofst1} &= \text{READ32}(\text{OfstTable}[i_x + 1, i_y]) \\
 \Delta x_1 &= \text{MSB16}(\text{Ofst1}) \\
 \Delta y_1 &= \text{LSB16}(\text{Ofst1}) \\
 \\
 \text{Ofst2} &= \text{READ32}(\text{OfstTable}[i_x, i_y + 1]) \\
 \Delta x_2 &= \text{MSB16}(\text{Ofst2}) \\
 \Delta y_2 &= \text{LSB16}(\text{Ofst2}) \\
 \\
 \text{Ofst3} &= \text{READ32}(\text{OfstTable}[i_x + 1, i_y + 1]) \\
 \Delta x_3 &= \text{MSB16}(\text{Ofst3}) \\
 \Delta y_3 &= \text{LSB16}(\text{Ofst3}) \\
 \\
 T_{x0} &= ((1 \ll (M + 3)) - f_x) \cdot \Delta x_0 + f_x \cdot \Delta x_1 \\
 T_{x1} &= ((1 \ll (M + 3)) - f_x) \cdot \Delta x_2 + f_x \cdot \Delta x_3 \\
 \\
 T_{y0} &= ((1 \ll (M + 3)) - f_y) \cdot \Delta y_0 + f_y \cdot \Delta y_1 \\
 T_{y1} &= ((1 \ll (M + 3)) - f_y) \cdot \Delta y_2 + f_y \cdot \Delta y_3 \\
 \\
 \Delta x &= (((1 \ll (M + 3)) - f_y) \cdot T_{x0} + f_y \cdot T_{x1} + (1 \ll 2M + 5)) \gg (2M + 6) \\
 \Delta y &= (((1 \ll (M + 3)) - f_x) \cdot T_{y0} + f_x \cdot T_{y1} + (1 \ll 2M + 5)) \gg (2M + 6) \\
 \\
 x_i &= x_o + \Delta x \\
 y_i &= y_o + \Delta y \\
 \\
 x_o, y_o &: \text{U16Q3} \\
 x_i, y_i &: \text{U16Q3} \\
 \Delta x, \Delta y &: \text{S16Q3}
 \end{aligned}$$

Idc-023

**Figure 6-109. LDC Lens Distortion Back Mapping Offset Calculation**

Graphically, this procedure is shown in [Figure 6-110](#).



ldc-024

**Figure 6-110. LDC Back Mapping Procedure Using Offset Table**

The input coordinate  $(x_i, y_i)$  refers to final input frame co-ordinates and is used to fetch the appropriate input pixels for interpolation. The interpolation procedure is described in . Before fetching  $data(x_i, y_i)$ , is clipped to the input frame boundary.

#### 6.9.5.2.6.1 LDC Mesh Table Storage Format

The size of the mesh table is dependent on the Total output frame size. The size must be:

$$\text{Table Width} = \text{CEIL}((\text{Mesh Frame Width})/2^M) + 1 \quad (15)$$

$$\text{Table Height} = \text{CEIL}((\text{Mesh Frame Height})/2^M) + 1. \quad (16)$$

Mesh frame width and height are specified by registers: VPAC\_LDC\_MESH\_FRSZ[13-0] W and VPAC\_LDC\_MESH\_FRSZ[29-16] H. The downsampling factor, M, is specified by register VPAC\_LDC\_MESHTABLE\_CFG[2] M.

The mesh table should be stored with delta(x) and delta(y) offsets interleaved and stored in raster order. Each offset is S16Q3, so a table entry is 32-bits, including both delta(x) and delta(y). The location of the mesh table is configured by registers VPAC\_LDC\_MESH\_BASE\_H and VPAC\_LDC\_MESH\_BASE\_I. The buffer width ( $\geq$  Table Width), is provided by VPAC\_LDC\_MESH\_OFST[15-0] OFST. The mesh table must be aligned on a 16-byte boundary.



### 6.9.5.2.7 LDC Pixel Interpolation

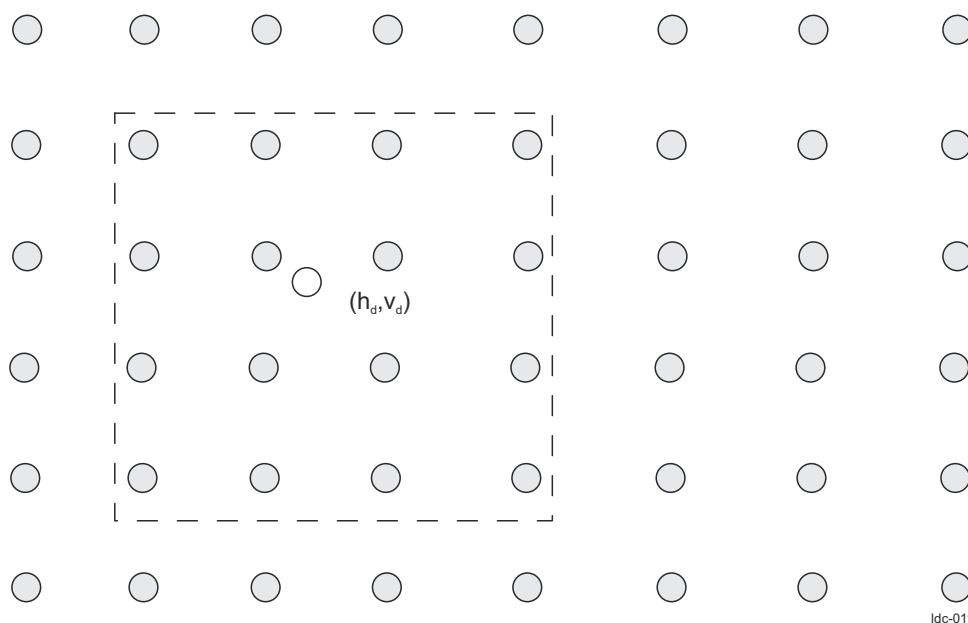
In UYVY and NV12 mode, the pixel interpolation type can be set to bi-cubic interpolation for best quality or bilinear interpolation for faster performance.

As the coordinates ( $h_d, v_d$ ) calculated by the back-mapping function are not generally integer values, bi-cubic or bilinear interpolation is applied to the distorted pixels.

Depending on register configuration, bi-cubic or bilinear interpolation is used to interpolate the output Y pixels:

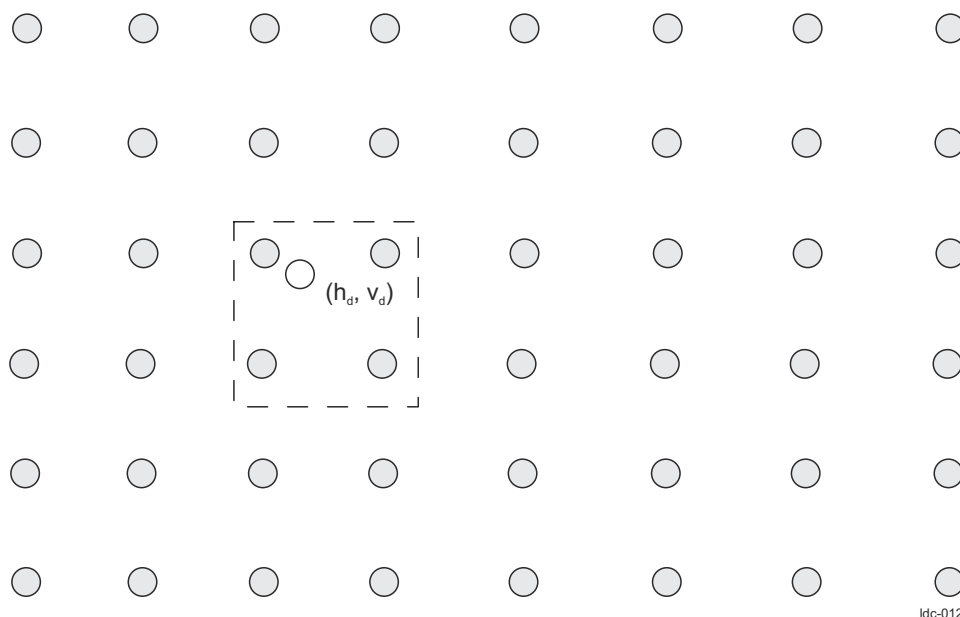
- VPAC\_LDC\_CFG[6] YINT\_TYP = 0, bi-cubic interpolation for Y, bilinear for other components
- VPAC\_LDC\_CFG[6] YINT\_TYP = 1, bi-linear interpolation for all components

In the case of bi-cubic interpolation, the distorted pixel is interpolated from the 16 Y pixels in the 4 x 4 grid around the distorted location, as shown in [Figure 6-111](#). Bi-cubic interpolation is used first along the horizontal direction, then the vertical direction.



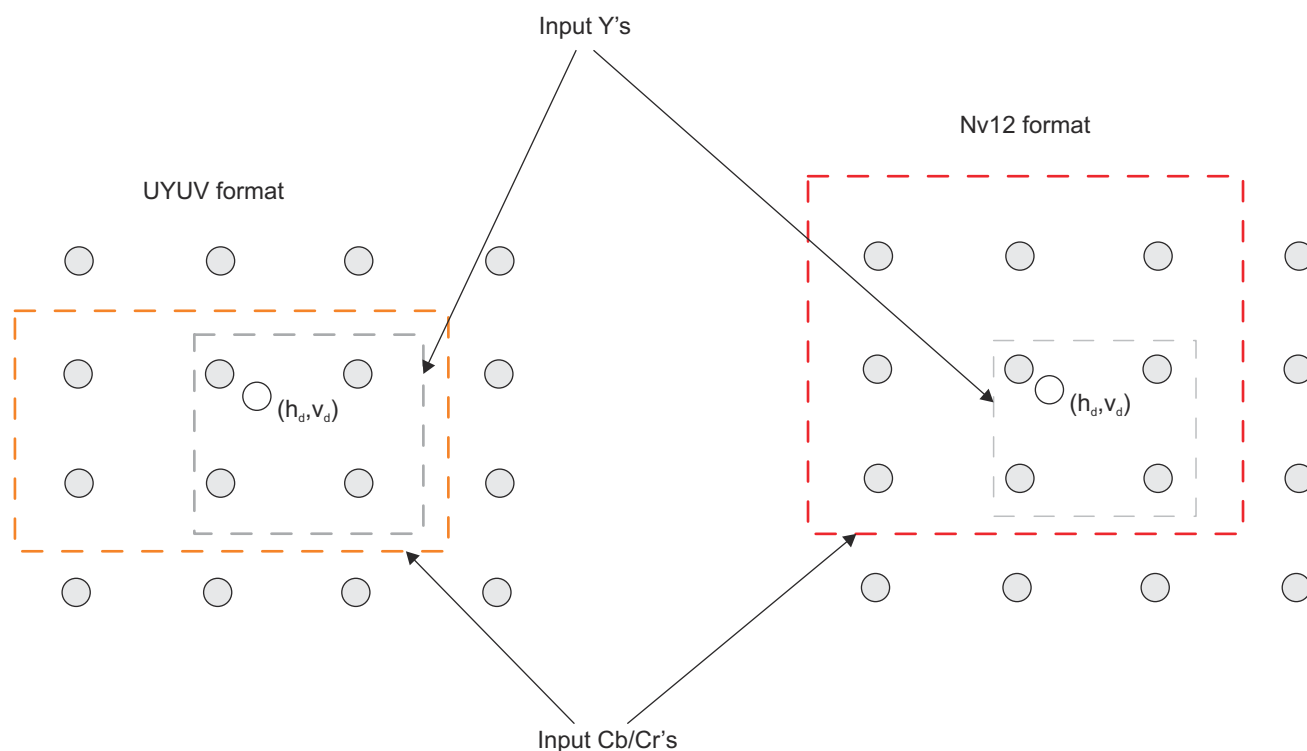
**Figure 6-111. LDC Bi-cubic Interpolation for Y**

If bi-linear interpolation is selected, the distorted pixel is interpolated from the four Y pixels in the 2 x 2 grid around the distorted location, as shown in [Figure 6-112](#).



**Figure 6-112. LDC Bi-linear Interpolation for Y**

For Cb and Cr components, simple bilinear interpolation is used. Each distorted pixel is interpolated from the four same-color pixels on the 2 x 2 grid around the distorted location. For UYVY data, the Cb/Cr grid is not square, but is 2x wider compared to the height. This is shown in [Figure 6-113](#).



**Figure 6-113. LDC Bilinear Interpolation for CB/Cr in UYVY and NV12 Format**

### 6.9.5.2.8 LDC Buffer Management

Nonlinear nature of the Back Mapping suggests 2-D processing to make more efficient use of external memory bandwidth. Since the mapping computation is non-trivial, the hardware requires the software to provide a number of data management parameters to assist hardware with data transfers.

#### 6.9.5.2.8.1 LDC Buffer Management

Once mesh block fetch is completed, final input co-ordinates are calculated by applying back mapping on previously calculated perspective warp corner pixel co-ordinates. Additional padding is applied on top of these back mapped corner co-ordinates based on the interpolation type.

The LDC processes a block of VPAC\_LDC\_OUT\_BLKSZ[7:0] OBW x VPAC\_LDC\_OUT\_BLKSZ[15:8] OBH (output block width by output block height) pixels at a time in a raster-scan order throughout the image. OBW is constrained to be a multiple of a base width, depending on the operating mode. This is described in [Table 6-135](#). OBH should be even. The LDC uses four corners of the output block plus some padding to fetch the input block. Due to extreme scenarios, there will be scenarios where above bounding box calculated by hardware doesn't cover all the input data required to generate particular output block. To cover those cases, software needs to supply additional PixelPad, the amount of padding in input block in all directions. Software must supply OBW, OBH, and VPAC\_LDC\_OUT\_BLKSZ[19:16] PIXPAD, the amount of padding in input pixels. Software must guarantee that, for EVERY output pixel in the OBW x OBH output block, the input pixels required are bounded by back mapping of the four corners plus/minus the padding. More precisely, the input block is determined by:

- $IBX\_start = \min(\text{truncate}(\text{distortx}(\text{corner1})), \text{truncate}(\text{distortx}(\text{corner3}))) - \text{HwPad} - \text{PixelPad}$
- $IBX\_end = \max(\text{truncate}(\text{distortx}(\text{corner2})), \text{truncate}(\text{distortx}(\text{corner4}))) + \text{HwPad} + \text{PixelPad}$
- $IBY\_start = \min(\text{truncate}(\text{distorty}(\text{corner1})), \text{truncate}(\text{distorty}(\text{corner2}))) - \text{HwPad} - \text{PixelPad}$
- $IBY\_end = \max(\text{truncate}(\text{distorty}(\text{corner3})), \text{truncate}(\text{distorty}(\text{corner4}))) + \text{HwPad} + \text{PixelPad}$

where corner1, corner2, corner3, and corner4 are upper-left, upper-right, lower-left, and lower-right corners of the OBW x OBH output block, and distortx(.), distorty(.) are X and Y distorted coordinates of the corners. Distorted coordinates computed by the back-mapping process described in [Section 6.9.5.2.6, LDC Lens Distortion Back-Mapping](#). [Figure 6-114](#) shows the input block bound calculation. [Table 6-135](#) lists the OBW requirements.

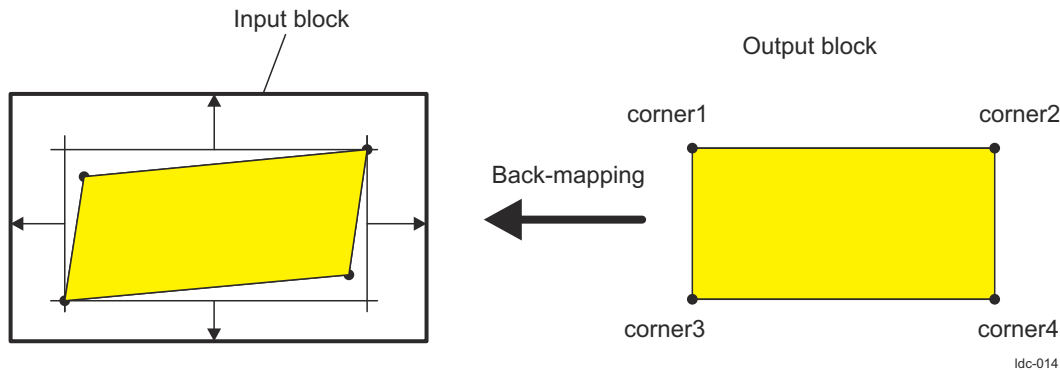


Figure 6-114. LDC Input Block Bound

Table 6-135. LDC OBW Requirements

Data Format	OBW must be a multiple of
YUV422	8
YUV420	8

Typically for geometric distortion, the LDC\_BLOCK[19:16] PIXPAD bit field must be 4 to accommodate neighbor sets for all colors. Software must set the PIXPAD bit field correctly.

OBH and OBW must be large enough for efficient operation of the lens distortion operation. As shown in [Table 6-135](#), OBW is constrained to ensure efficient external memory write. Another constraint is that input block size,  $(IBX\_end - IBX\_start + 1) \times (IBY\_end - IBY\_start + 1)$ , for EVERY input block of the image, must fit the allocated input buffer. These parameters must not be specified pessimistically (OBH, OBW too small, or PixelPad too large). Pessimistic parameter setting degrades performance and incurs unnecessary external memory transfer.

The LDC can process a portion of the image, rather than the entire image. This saves time by letting an image process through multiple software/LDC interactions to correct only a portion of the image (See Multi-pass Frame processing).

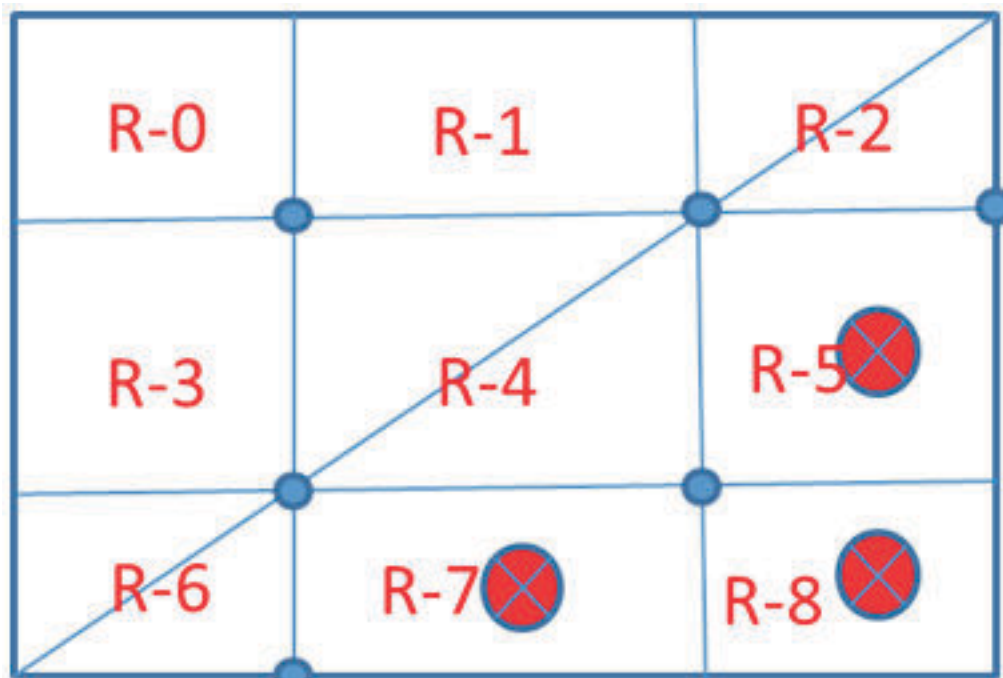
An intermediate event, *hts\_tdone*, is also provided on completion of each macro block output write. This allows the LDC operation to be pipelined with other tasks. The LDC output write stall after this event, waiting for a pulse on the *hts\_tstart* signal to begin writing the next macro block.

#### 6.9.5.2.9 LDC Multi Region with Variable Block size

Distortion in wide angle lenses causes output block size to be very small due to extreme scaling in particular regions of the Image. This could be caused by high field-of-view or viewing position. The range of “Ratio of Output block to input pixel block size” is very high across the frame.

In order to solve this, LDC supports dividing the frame into multiple regions and output block size can be programmed independently for each region. Frame can be divided up to 9 regions. Frame can be divided into maximum of three slices horizontally and maximum of three slices vertically as shown in [Figure 6-115](#). All the regions in a vertical slice share the common horizontal start location and all the regions in a horizontal slice share the common vertical start location. Width of vertical slices can be configured using VPAC\_LDC\_REGN\_W12\_SZ[13-0] W1, VPAC\_LDC\_REGN\_W12\_SZ[29-16] W2 and PAC\_LDC\_REGN\_W3\_SZ[13-0] W3 register fields. Height of horizontal slices can be configured using VPAC\_LDC\_REGN\_H12\_SZ[13-0] H1, VPAC\_LDC\_REGN\_H12\_SZ[29-16] H2 and VPAC\_LDC\_REGN\_H3\_SZ[13-0] H3 register fields.





Idc-027

**Figure 6-116. LDC Region Skipping Example**

#### 6.9.5.2.9.2 LDC Support for sub-set of 3x3 regions

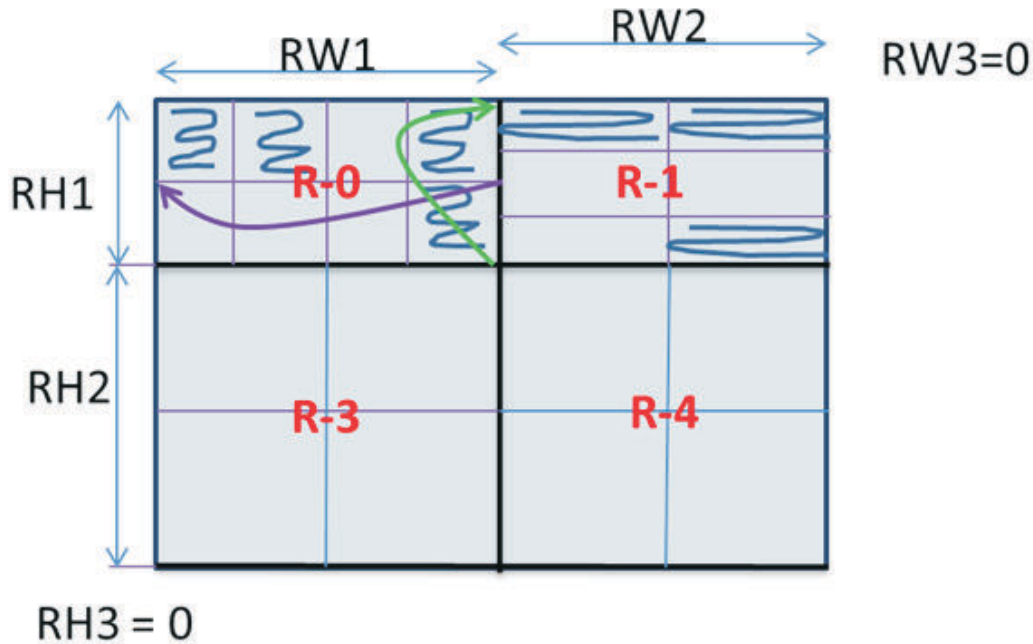
HW supports dividing the frame into sub-set of 3x3 regions. Following combinations of regions are supported.

- 3 vertical slices
  - $RW1 = x$ ,  $RW2 = y$  and  $RW3 = z$
- 2 vertical slices
  - $RW1 = x$ ,  $RW2 = y$  and  $RW3 = 0$
  - Last region width has to be zero
- 1 vertical slice
  - $RW1 = x$ ,  $RW2 = 0$  and  $RW3 = 0$
  - Last two regions width has to be zero
- 3 horizontal slices
  - $RH1 = x$ ,  $RH2 = y$  and  $RH3 = z$
- 2 horizontal slices
  - $RH1 = x$ ,  $RH2 = y$  and  $RH3 = 0$
  - Last region height has to be zero
- 1 horizontal slice
  - $RH1 = x$ ,  $RH2 = 0$  and  $RH3 = 0$
  - Last two regions height has to be zero

As shown in [Figure 6-117](#), 2x2 region partitioning can be done by programming  $RW3=RH3=0$ .

#### Note

Region numbering remains as in 3x3 region partitioning with R3, R6-R8 being null regions.



ldc-028

**Figure 6-117. LDC 2x2 Region Partitioning Example**

#### 6.9.5.2.9.3 LDC Limitations of Multi Region Scheme

Following features can't be supported when multi region processing is enabled.

- Circular buffer support can't be enabled on Input data fetch
- HTS synchronization can't be enabled on Input data fetch
- LDC can't be joined with other HWAs in the VPAC. LDC output needs to be written to external memory.
- HTS can't make use of multiple BPR buffers available in SL2 circular buffer

#### 6.9.5.2.9.4 LDC Multi Region Block Constrains

- $RW1 + RW2 + RW3$  must be equal to OFW
- $RH1 + RH2 + RH3$  must be equal to OFH
- SW to program the LSE buffer Stride, at-least the size of largest output block width among all regions.
- SW to program buffer height at-least twice the maximum block height among all regions

#### 6.9.5.2.10 LDC Multi-pass Frame processing

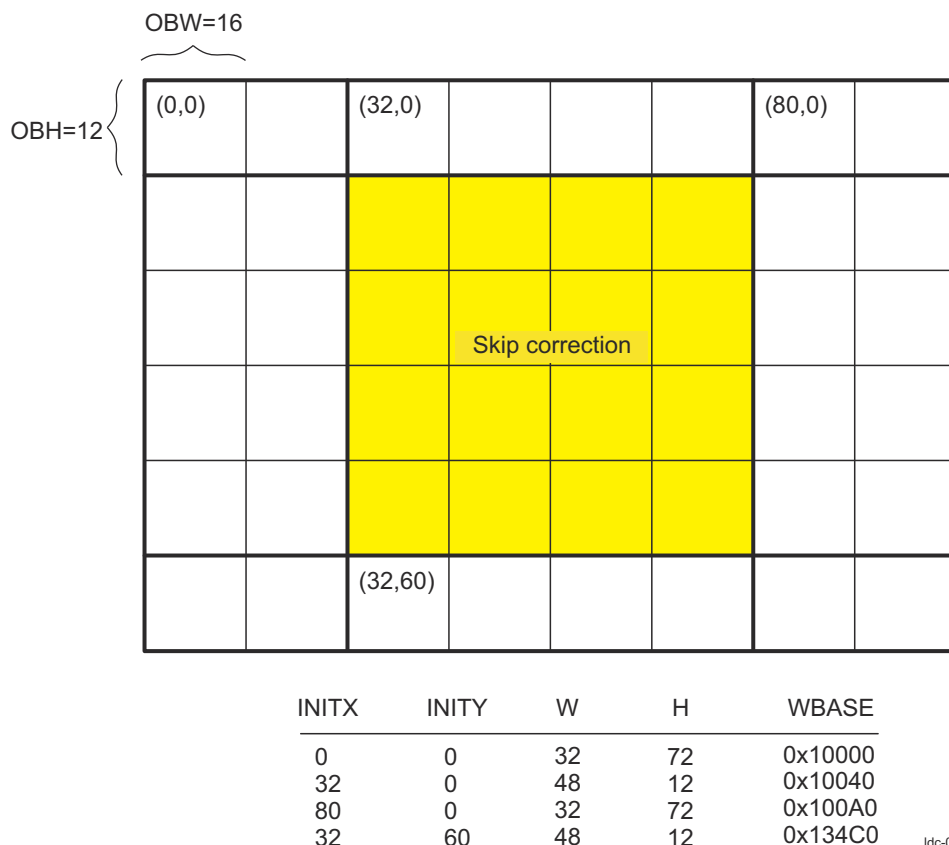
The LDC can process a portion of the image, rather than the entire image. This saves time by letting an image process through multiple software/LDC interactions to correct only a portion of the image.

An intermediate interrupt can occur after processing each macro block. This lets the LDC operation be pipelined with other tasks.

To process only a portion of the image, the following parameters are required:

- VPAC\_LDC\_INITXY[12:0] INITX: X coordinate of upper-left corner of output frame
- VPAC\_LDC\_INITXY[28:16] INITY: Y coordinate of upper-left corner of output frame
- VPAC\_LDC\_INPUT\_FRSZ[13:0] W: Width of output frame
- VPAC\_LDC\_INPUT\_FRSZ[29:16] H: Height of output frame
- FrameBase SDRAM address of upper-left corner of output frame (Y and Cb/Cr base needed in case of YCbCr420)
- FrameOfst SDRAM frame width (in bytes)

Figure 6-118 is an example of multiple-pass processing with middle-of-the-image skipped.



**Figure 6-118. LDC Multiple-Pass Correction Example**

The LDC does not copy skipped blocks from the input frame to the output frame (necessary task unless output frame = input frame); software must set up and initiate this memory copy.

A note on in-place operation to conserve external memory: Depending on the offset table and configuration parameters, it may not be feasible to point the output image at the input image and have corrected pixels overwriting the source image. Software must determine whether it is feasible to overwrite, and allocate a separate frame buffer when it is not feasible. The LDC does not care if the input and output image pointers coincide, nor does it check any dependency violations.

The starting address of the input frame, corresponding to input coordinate (0, 0), is specified in the LDC\_RD\_BASE[31:0] RBASE bit field and offset in the LDC\_RD\_OFST[15:0] ROFST bit field ((Y and Cb/Cr base required for YCbCr4:2:0: LDC\_420C\_RD\_BASE[31:0] RBASE)). The LDC does not clips input block to input frame size if any of the input block falls outside the input frame.

#### Note

- Any number of input blocks CAN fall partially or totally outside the input frame by assuming coordinates that are outside valid range (negative or exceeding maximum image width defined by the line offset).
- If any output pixel refers to any input pixel outside the input frame, the computed value for that output pixel would be same as neighbor pixel which falls into valid input frame. The other output pixels can still have correct outcomes.

#### 6.9.5.2.11 LDC Input/Output Data Formats

Table 6-136 captures valid number of LDC input and output data mode/format combinations.



**Table 6-136. LDC Supported Data Format Combinations**

Input		Output	
Data Mode	Data format	Data Mode	Data format
420 420_Y 420_UV	8-bit	Same as input	8bit
	12-bit - packed		12-bit - packed
	12-bit - unpacked		12-bit - packed
	12-bit - packed		12-bit - unpacked
422	8-bit	420	8-bit (on channel[2/3] using LUT)
			8-bit (on channel[0/1] with shift)
			8-bit (on channel[2/3] using LUT)
			8-bit
422	8-bit	422	12-bit – packed
			8-bit

Output data mode will be same as input data mode except when input data is 422, output data can also be converted to 420. In the case of input data format is 8-bit and output data format is 12-bit packed, valid data range is only 8-bit. Output data formatted into 12-bit packed format with 4 MSB bits being all zeros.

The following list contains the register fields used to configure data modes and formats:

- Input data mode - VPAC\_LDC\_CTRL[4-3] IP\_DATAMODE
- Input data format - VPAC\_LDC\_CTRL[6-5] IP\_DFMT
- Output Luma data format – VPAC\_LDC\_LSE\_BUF\_CFG\_j (j = 0, 2) PIX\_FMT\_PW
- Output Chroma data format – VPAC\_LDC\_LSE\_BUF\_CFG\_j (j = 1, 3) PIX\_FMT\_PW

LDC supports YCbCr 422 in the UYVY input data format. The sample grid for YCbCr 422\_UYVY is shown in [Figure 6-119](#).



ldc-029

**Figure 6-119. LDC Sample Grids for UYVY**

UYVY data are stored in 8-bit per color components as shown in [Table 6-137](#).

**Table 6-137. LDC UYVY 8-bit Per Color Components**

31			0
Y(1)	Cr(0)	Y(0)	Cb(0)

The sample grid for YUV420 NV12 is shown in [Figure 6-120](#).



ldc-029

**Figure 6-120. LDC Sample Grids for NV12**

YCbCr 420 data are stored 8-bit per color components, and in two planes, Y by itself and Cb/Cr interleaved as shown in [Table 6-138](#)

**Table 6-138. LDC YCbCr 420 data stored 8-bit per color components**

31			0
(Y plane) Y(3)	Y(2)	Y(1)	Y(0)
(Cb/Cr plane) Cr(2)	Cb(2)	Cr(0)	Cb(0)

3	1	3	0	2	9	2	8	2	7	2	6	2	5	2	4	2	3	2	2	1	2	0	1	9	8	7	6	5	4	3	2	1	0			
Y2[7:0]												Y1												Y0												+0x0
Y5[3:0]				Y4												Y3												Y2[11:8]				+0x4				
Y7								Y6												Y5[11:4]								+0x8								
Y10[7:0]												Y9												Y8												+0C
Y13[3:0]				Y12												Y11												Y10[11:8]				+10				
Y15												Y14												Y13[11:4]								+14				

3	1	3	0	2	9	2	8	2	7	2	6	2	5	2	4	2	3	2	2	1	2	0	1	9	8	7	6	5	4	3	2	1	0					
Cb1[7:0]																Cr0										Cb0												+0x0
Cr2[3:0]				Cb2										Cr1										Cb1[11:8]				+0x4										
Cr3										Cb3										Cr2[11:4]								+0x8										
Cb5[7:0]										Cr4										Cb4								+0C										
Cr6[3:0]				Cb6										Cr5										Cb5[11:8]				+10										
Cr7										Cb7										Cr6[11:4]								+14										

ldc-031

**Figure 6-121. LDC YUV420 (2-plane 12-bit Fully Packed Format)**

#### 6.9.5.2.12 LDC YUV422 to YUV420 Conversion

Supporting input YUV422 format enables external sensors providing the YUV data directly. YUV422 to YUV420 conversion is done after pixel interpolation and achieved by dropping odd Chroma lines in the block. This retains the Chroma co-sited relation with Luma as at LDC input.



Idc-032

**Figure 6-122. LDC YUV422 to YUV420 Conversion**

#### 6.9.5.2.13 LDC SL2 Interface (LSE)

LDC Output write interface is handled by LSE sub-module. LSE (Load Store Engine) module is a common component integrated in VPAC to perform data load and store tasks on the SL2 memory for the HWA algorithm core.

VBUSP interface is be used to transfer the data from LDC Core to LSE.

For more LSE details, see *Load Store Engine (LSE)*.

##### 6.9.5.2.13.1 LDC PSA (Parallel Signature Analysis)

LSE integrates a CRC signature capture mechanism on each output channel data to verify the integrity of the HWA's internal paths. When enabled, every valid pixel data from the CORE is sampled by a CRC module to update the signature for the channel before the data is packed and sent to the SL2 buffer. Note that data used for PSA is raw data from the CORE rather than data written into SL2 with container alignment. For LDC, PSA signature is calculated in the same order as CORE data transfer to LSE which is 3D block order as against frame order. At the end of frame condition, the final signature is saved in a separate read-only signature register (VPAC\_LDC\_LSE\_PSA\_SIGNATURE\_y y=0..#\_of\_Output Channel-1) till the next end of frame condition for software to read and compare it against the golden reference signature. The read-only register keeps the last saved value until a reset (set to 0) or is replaced by another frame data signature.

##### 6.9.5.2.14 LDC LUT Mapped Dual Output

LDC produces parallel 8-bit and 12-bit outputs to support surround view and ADAS applications concurrently. LDC produces parallel 8-bit output from 12-bit input using LUT based implementation. This 8-bit output is written into SL2 using LSE output channels (channel[2] for Luma and channel[3] for Chroma). Though use case requires 8-bit output, hardware is designed to be generic which supports any 8-12bit to 8-12bit conversion.

12 or 8 bit generated by data interpolation block is mapped to 8 to 12 bits using a LUT based mapping. The LUT is similar to the LUTs used in VISS and is a linear LUT with 513 entries.

The LUTs are sized to provide 513 locations of data, to enable linear interpolation for all locations. The internal weights are sized as 3 bits since the maximum delta between 2 steps is only  $4095/512 = 8$ . The logic is designed to scale for bit width and can support anything from 8 to 12 bits of input to support different use cases. However, since the step size is different depending on the input bit width, the bit selection and weight calculation logic is designed to account for that. LSE Shift operation can be instead of LUT to reduce bit width from 12 down to 8.

The input bit width (IN\_BITDEPTH explained above) is specified using a dedicated register. Output is clipped to programmable clip value ( $2^{\text{BitClip}-1}$ ). Combination of IN\_BITDEPTH, LUT values and Clip value programation allows any 8-12bit to 8-12bit conversion.

#### 6.9.5.2.15 LDC Band Width Controller

A Bandwidth Limiter is required to limit the mean BW that LDC can request over the VBUSM Read interface. The software sets the maximum allowed bytes per cycle in the register, VPAC\_LDC\_VBUSMR\_CFG[27-16] BW\_CTRL.

To determine the maximum allowed bytes per cycle, the software needs to know the configuration of LDC, which determines the maximum input block size. Given a specific configuration, the `ldc_findMaxInputBuffer` utility provided with the functional C model, can be used to give the maximum input block size. The performance requirements determine the maximum number of cycles allowed per block.

#### 6.9.5.2.16 LDC Input Block Fetch Limit

In order to limit the band width hogging by LDC in case of wrong mesh table data, maximum amount of input block pixel data being fetched will be capped at maximum internal memory available for respective data (Note: Luma and Chroma has different size internal buffers).

In YUV420\* modes, limit is imposed by Luma data. Pixel data fetch is stopped on detecting the overflow condition and interrupt event is generated. Maximum of 2 requests can be issued upon overflow detection. In case of mesh data, only interrupt event is generated on detecting overflow condition and there is no stalling of fetch. These events are generated for the first overflow case per frame.

#### 6.9.5.2.17 LDC HTS Interface

The HTS interface is used to synchronize between blocks in VPAC Subsystem. In the case of LDC, it controls the start of LDC operation by issuing `hts_init`. LDC will be waiting for `hts_init` once it is enabled, so LDC is expected to be enabled before issuing `hts_init`.

HTS also controls the LDC output write which will indirectly control all LDC operations (that is, fetch and processing). LDC will start writing output pixel data when only it receives `hts_tstart` for that particular block. Once all output data is written, LDC will generate `hts_tdone` (also `hts_eop` if it is last block of the frame).

#### 6.9.5.2.18 LDC VBUSM Read Interface

VBUSM Read interface has following features

- Data Width – 128 Bit
- Address width – 48 Bit
- Maximum of number of Tags – 32
  - Design can be constrained to use lesser number of Tags
- Maximum Command Size: Programmable
  - 32/64/128/256 Bytes
  - Command will split at maximum command size address boundary
- Chanid Encode: `chanid[1:0]` identify the source/type of the data interface is requesting. `Chanid[11:2]` is tied low.
  - Chanid[1:0] decoding
    - 00 – Mesh Data.
    - 01 – Luma data in 420 mode/422 data.
    - 10 – Chroma data in 420 mode.

### 6.9.5.3 LDC Programmers Guide

#### 6.9.5.3.1 LDC Programming Geometric Distortion Mode

Geometric distortions, barrel and pincushion distortion, can be corrected on UYVY and NV12 data formats. The Mesh based back mapping is used in this mode. An additional affine transform can be configured at the same time.

1. Check and wait for VPAC\_LDC\_CTRL[2] BUSY to become IDLE (0).
2. Set VPAC\_LDC\_CTRL[4-3] IP\_DATAMODE for UYVY (0) or NV12 data (2). Program VPAC\_LDC\_CTRL[6-5] IP\_DFMT to required data storage format.
3. Set VPAC\_LDC\_CTRL[1] LDMAPEN = 1 to enable lens distortion back mapping.

4. Set the input frame base address in VPAC\_LDC\_RD\_BASE\_H / VPAC\_LDC\_RD\_BASE\_L registers. Note that the frame base address must be aligned on a 16-byte boundary. In case of YUV420, YUV422SP, Y1\_Y2 and Y1\_Y2Y3 modes, configure VPAC\_LDC\_RD\_420C\_BASE\_H / VPAC\_LDC\_RD\_420C\_BASE\_L registers for Chroma Data.
5. Set the input frame line offset in VPAC\_LDC\_RD\_OFST[15-0] OFST.
6. If reading the input image from a circular buffer:
  - a. Set VPAC\_LDC\_CTRL[9] IP\_CIRCEN = 1.
  - b. LDC computes the row number and applies modulo operation. The absolute address in the buffer is computed from this wrap-around row and the column number and then data is fetched from the circular buffer. Set the circular buffer size in VPAC\_LDC\_RD\_OFST[29-16] MOD register field. In YUV420 modes, the number of rows in the buffer is MOD/2. In others modes, the number of rows in the buffer is MOD for both Y buffer and Cb/Cr buffer.
7. Set the tile size in VPAC\_LDC\_REGION\_OUT\_BLKSZ\_j[15-8] OBH and VPAC\_LDC\_REGION\_OUT\_BLKSZ\_j[7:0] OBW. Note the constraints on OBW in [Table 6-135](#).
8. Set the pixel pad in VPAC\_LDC\_REGION\_OUT\_BLKSZ\_j[19-16] PIXPAD.
9. Set the input frame size in VPAC\_LDC\_INPUT\_FRSZ[29-16] H and VPAC\_LDC\_INPUT\_FRSZ[13-0] W.
10. Set the output frame size in VPAC\_LDC\_MESH\_FRSZ[29-16] H and VPAC\_LDC\_MESH\_FRSZ[13-0] W. Mesh data has to be present for entire frame after transform.
11. Set the output compute frame size in VPAC\_LDC\_COMPUTE\_FRSZ[29-16] H and VPAC\_LDC\_COMPUTE\_FRSZ[13-0] W.
12. Set the starting output point in VPAC\_LDC\_INITXY[12-0] INITX and VPAC\_LDC\_INITXY[28-16] INITY.
13. Configure SL2 Interface programming registers. Details of LSE configuration is captured in [Section 6.9.5.3.4](#).
14. Set the mesh offset table pointer to the correct address (VPAC\_LDC\_MESH\_BASE\_H / VPAC\_LDC\_MESH\_BASE\_I[31-0] ADDR and VPAC\_LDC\_MESH\_OFST[15-0] OFST). Set the table down sampling factor for MxM down sampling in VPAC\_LDC\_MESHTABLE\_CFG[2-0] M.
15. Set the Y plane interpolation type to bilinear or bi-cubic in VPAC\_LDC\_CFG[6] YINT\_TYP.
16. If also using affine transform, set the six affine transform parameters in VPAC\_LDC\_AFF\_AB.A, VPAC\_LDC\_AFF\_AB.B, VPAC\_LDC\_AFF\_CD.C, VPAC\_LDC\_AFF\_CD.D, VPAC\_LDC\_AFF\_EF.E, and VPAC\_LDC\_AFF\_EF.F. If affine transform is not used, then these need to be set to the following values: A = 4096, B = 0, C = 0, D = 0, E = 4096, and F = 0.
17. If also using perspective transform, enable VPAC\_LDC\_CTRL[7] PWARPEN and set VPAC\_LDC\_PWARP\_GH[15-0] G and VPAC\_LDC\_PWARP\_GH[31-16] H. If perspective transform is not used, disable VPAC\_LDC\_CTRL[7] PWARPEN and set VPAC\_LDC\_PWARP\_GH[15-0] G = 0 and VPAC\_LDC\_PWARP\_GH[31-16] H = 0.
18. Set VPAC\_LDC\_CTRL[0] LDC\_EN = 1 to start the LDC operation.
19. Start the HTS init sequencing. LDC fetch/processing is gated with hts\_init.
20. Wait for VPAC\_LDC\_CTRL[2] BUSY to become IDLE (0) or wait for the end of frame completion interrupt.

#### 6.9.5.3.2 LDC Programming Rotational Video Stabilization (Affine Transformation)

In the rotational video stabilization use case, consecutive frames capturing the same scene are aligned to minimize camera shake. Camera motions cause the frames to have translation or rotation differences. Thus, the frames need to be aligned to maintain a stable output video. The affine transform offers control to account for scaling, rotation, and translation transforms. For example, the rotation use case can be parameterized with the following affine transform parameters:

$$\begin{bmatrix} h_{aff} \\ v_{aff} \end{bmatrix} = \begin{bmatrix} a & b \\ d & e \end{bmatrix} \begin{bmatrix} h_d \\ v_d \end{bmatrix} + \begin{bmatrix} c \\ f \end{bmatrix}$$

$$\begin{bmatrix} h_{aff} \\ v_{aff} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} h_d \\ v_d \end{bmatrix} + \begin{bmatrix} h_o - h_o \cos(\theta) - v_o \sin(\theta) \\ v_o - v_o \cos(\theta) - h_o \sin(\theta) \end{bmatrix}$$

ldc-035

where theta is the angle of rotation. In the scaling case, the affine transform uses:

$$\begin{bmatrix} h_{aff} \\ v_{aff} \end{bmatrix} = \begin{bmatrix} \alpha & 0 \\ 0 & \alpha \end{bmatrix} \begin{bmatrix} h_d \\ v_d \end{bmatrix} + \begin{bmatrix} h_o(1-\alpha) \\ v_o(1-\alpha) \end{bmatrix}$$

Idc-036

where alpha is the scale factor. Translation uses an identity matrix along with the translation vector as the affine transform parameters, as in:

$$\begin{bmatrix} h_{aff} \\ v_{aff} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} h_d \\ v_d \end{bmatrix} + \begin{bmatrix} T_h \\ T_v \end{bmatrix}$$

Idc-037

where  $T_h$  and  $T_v$  are the translation parameters. More complex transforms can be derived and the affine transformation is general so that optimization routines may be used to provide the best parameters to align two frames. In all affine transforms, the perspective warp parameters,  $g$  and  $h$ , should be set to 0.

The following steps need to be performed by the application to initialize and run LDC to perform affine transform.

1. Check and wait for VPAC\_LDC\_CTRL[2] BUSY to become IDLE (0).
2. Set VPAC\_LDC\_CTRL[4-3] IP\_DATAMODE and VPAC\_LDC\_CTRL[6-5] IP\_DFMT to required data mode and format.
3. If lens distortion correction is performed at the same time as the affine transform, set VPAC\_LDC\_CTRL[1] LDMAOPEN = 1. Otherwise, set VPAC\_LDC\_CTRL[1] LDMAOPEN = 0.
4. Set the input frame base address in VPAC\_LDC\_RD\_BASE\_H / VPAC\_LDC\_RD\_BASE\_I. Note that the frame base address must be aligned on a 16-byte boundary.
5. Set the input frame line offset in VPAC\_LDC\_RD\_OFST[15-0] OFST.
6. If reading the input image from a circular buffer:
  - a. Set VPAC\_LDC\_CTRL[9] IP\_CIRCEN = 1.
  - b. If using row address, LDC computes the row number and applies a modulo operation. The absolute address in the buffer is computed from this wrap-around row and the column number and then data is fetched from the circular buffer. Set the circular buffer size in VPAC\_LDC\_RD\_OFST[29-16] MOD. In YUV420 modes, the number of rows in the Y buffer is MOD and the number of rows in the Cb/Cr buffers is MOD/2. In others modes, the number of rows in the buffer is MOD for both Y buffer and Cb/Cr buffer.
7. Set the tile size in VPAC\_LDC\_OUT\_BLK SZ[15-8] OBH and VPAC\_LDC\_OUT\_BLK SZ[7-0] OBW. Note the constraints on OBW in [Table 6-135](#).
8. Set the pixel pad in VPAC\_LDC\_OUT\_BLK SZ[19-16] PIXPAD.
9. Set the input frame size in VPAC\_LDC\_INPUT\_FRSZ[29-16] H and VPAC\_LDC\_INPUT\_FRSZ[13-0] W.
10. Set the output frame size in VPAC\_LDC\_MESH\_FRSZ[29-16] H and VPAC\_LDC\_MESH\_FRSZ[13-0] W. Mesh data has to be present for entire frame after transform.
11. Set the output compute frame size in VPAC\_LDC\_COMPUTE\_FRSZ[29-16] H and VPAC\_LDC\_COMPUTE\_FRSZ[13-0] W.
12. Set the starting output point in VPAC\_LDC\_INITXY[12-0] INITX and VPAC\_LDC\_INITXY[28-16] INITY.
13. Configure SL2 Interface programming registers. Details of LSE configuration is captured in [Section 6.9.5.3.4](#).
14. If the data format is NV12, set the 420 UV input plane base address in VPAC\_LDC\_RD\_420C\_BASE\_H / VPAC\_LDC\_RD\_420C\_BASE\_I. This address must be 16-byte aligned.
15. Set the mesh offset table pointer to the correct address (VPAC\_LDC\_MESH\_BASE\_H / VPAC\_LDC\_MESH\_BASE\_I and VPAC\_LDC\_MESH\_OFST. Set the table down sampling factor for MxM down sampling in MESHTABLE\_CFG.M.
16. Set the Y plane interpolation type to bilinear or bicubic in VPAC\_LDC\_CFG[6] YINT\_TYP.
17. Set the six affine transform parameters in VPAC\_LDC\_AFF\_AB.A, VPAC\_LDC\_AFF\_AB.B, VPAC\_LDC\_AFF\_CD.C, VPAC\_LDC\_AFF\_CD.D, VPAC\_LDC\_AFF\_EF.E, and VPAC\_LDC\_AFF\_EF.F. If



affine transform is not used, then these need to be set to the following values: A = 4096, B = 0, C = 0, D = 0, E = 4096, and F = 0.

18. Disable perspective warp transform with VPAC\_LDC\_CTRL[7] PWARPEN = 0. Set the two perspective transform parameters VPAC\_LDC\_PWARP\_GH[15-0] G = 0 and VPAC\_LDC\_PWARP\_GH[31-16] H = 0.
19. Set VPAC\_LDC\_CTRL[0] LDC\_EN = 1 to start the LDC operation.
20. Start the HTS init sequencing. LDC fetch/processing is gated with hts\_init.
21. Wait for VPAC\_LDC\_CTRL[2] BUSY to become IDLE (0) or wait for the end of frame completion interrupt.

#### 6.9.5.3.3 LDC Programming Perspective Transformation

Perspective transforms are used to align the image data of two different cameras viewing the same scene from different positions. It can also be used in the case of stereo image rectification to align the epipolar lines of the left/right image pair with their scan lines.

The following steps need to be performed by the application to initialize and run LDC to perform perspective transform.

1. Check and wait for VPAC\_LDC\_CTRL[2] BUSY to become IDLE (0).
2. Set VPAC\_LDC\_CTRL[4-3] IP\_DATAMODE and VPAC\_LDC\_CTRL[6-5] IP\_DFMT to required data mode and format.
3. If lens distortion correction is performed at the same time as the perspective transform, set VPAC\_LDC\_CTRL[1] LDMAPEN = 1. Otherwise, set LDMAPEN = 0.
4. Set the input frame base address in VPAC\_LDC\_RD\_BASE\_H / VPAC\_LDC\_RD\_BASE\_I. Note that the frame base address must be aligned on a 16-byte boundary.
5. Set the input frame line offset in VPAC\_LDC\_RD\_OFST[15-0] OFST.
6. Set the tile size in VPAC\_LDC\_OUT\_BLKSZ[15-8] OBH and VPAC\_LDC\_OUT\_BLKSZ[7-0] OBW. Note the constraints on OBW in [Table 6-135](#).
7. Set the pixel pad in VPAC\_LDC\_OUT\_BLKSZ[19-16] PIXPAD.
8. Set the input frame size in VPAC\_LDC\_INPUT\_FRSZ[29-16] H and VPAC\_LDC\_INPUT\_FRSZ[13-0] W.
9. Set the output frame size in VPAC\_LDC\_MESH\_FRSZ[29-16] H and VPAC\_LDC\_MESH\_FRSZ[13-0] W. Mesh data has to be present for entire frame after transform.
10. Set the output compute frame size in VPAC\_LDC\_COMPUTE\_FRSZ[29-16] H and VPAC\_LDC\_COMPUTE\_FRSZ[13-0] W.
11. Set the starting output point in VPAC\_LDC\_INITXY[12-0] INITX and VPAC\_LDC\_INITXY[28-16] INITY.
12. Configure SL2 Interface programming registers. Details of LSE configuration is captured in [Section 6.9.5.3.4](#).
13. If the data format is NV12, set the 420 UV input plane base address in VPAC\_LDC\_RD\_420C\_BASE\_H / VPAC\_LDC\_RD\_420C\_BASE\_I. This address must be 16-byte aligned.
14. Set the mesh offset table pointer to the correct address (VPAC\_LDC\_MESH\_BASE\_H / VPAC\_LDC\_MESH\_BASE\_I and VPAC\_LDC\_MESH\_OFST. Set the table down sampling factor for MxM down sampling in MESHTABLE\_CFG.M.
15. Set the Y plane interpolation type to bilinear or bicubic in VPAC\_LDC\_CFG[6] YINT\_TYP.
16. Set the eight perspective transform parameters in VPAC\_LDC\_AFF\_AB.A, VPAC\_LDC\_AFF\_AB.B, VPAC\_LDC\_AFF\_CD.C, VPAC\_LDC\_AFF\_CD.D, VPAC\_LDC\_AFF\_EF.E, and VPAC\_LDC\_AFF\_EF.F. If affine transform is not used, then these need to be set to the following values: A = 4096, B = 0, C = 0, D = 0, E = 4096, F = 0, G = 0 and H = 0. Enable VPAC\_LDC\_CTRL[7] PWARPEN = 1.
17. Set VPAC\_LDC\_CTRL[0] LDC\_EN = 1 to start the LDC operation.
18. Start the HTS init sequencing. LDC fetch/processing is gated with hts\_init.
19. Wait for VPAC\_LDC\_CTRL[2] BUSY to become IDLE (0) or wait for the end of frame completion interrupt.

#### 6.9.5.3.4 LDC Programming LSE

Following provides the LSE general programming guideline.

It also captures an additional secondary specific use case configuration values for some registers.

- Frame Size = 1920x1080
- Output Block size = 64x32
- Luma data being consumed by MSC and Chroma is written out by UDMA
  - Block to line conversion done for Luma data via SL2.

- Chroma data is written out at block level.
  - Circular buffer size just to do ping-pong between LDC – MSC(Y) and LDC – UDMA(C)
  - 12-bit pixel data
1. Pixel Data Format
    - a. VPAC\_LDC\_LSE\_BUF\_CFG\_j (PIX\_FMT\_PW, PIX\_FMT\_CNTRSZ, and PIX\_FMT\_ALIGN) parameters define the pixel output data format for this output channel.
    - b. Common data formats:
      - i. Fully packed 12-bit : PIX\_FMT\_PW=1, PIX\_FMT\_CNTRSZ=1, PIX\_FMT\_ALIGN=0
      - ii. 12-bit unpacked (MSB aligned) : PIX\_FMT\_PW = 1, PIX\_FMT\_CNTRSZ=2, PIX\_FMT\_ALIGN=1
  2. Output SL2 Circular Buffer Configuration
    - a. VPAC\_LDC\_LSE\_BUF\_ATTR0\_j (buf\_stride) – define the line stride size (must be 64 byte multiple)
      - i. VPAC\_LDC\_LSE\_BUF\_ATTR0\_j[15-6] BUF\_STRIDE = 2880 (1920 \* 1.5)
      - ii. VPAC\_LDC\_LSE\_BUF\_ATTR0\_j[15-6] BUF\_STRIDE = 96 (64 \* 1.5)
    - b. VPAC\_LDC\_LSE\_BUF\_ATTR0\_j (cbuf\_size) – define the size of the circular buffer in the SL2 (number of lines)
      - i. VPAC\_LDC\_LSE\_BUF\_ATTR0\_j[24-16] CBUF\_SIZE = 68 (2\*OBH+4)
      - ii. VPAC\_LDC\_LSE\_BUF\_ATTR0\_j[24-16] CBUF\_SIZE = 64 (2\*OBH)
  3. Output SL2 Circular Buffer 2D mode Configuration (applicable for LDC only - TBD)
    - a. VPAC\_LDC\_LSE\_BUF\_ATTR1\_j (buf\_blk\_width) – defines the number of bytes equivalent to OBW (output block width) pixels.
      - i. VPAC\_LDC\_LSE\_BUF\_ATTR1\_j.buf\_blk\_width = 96 (64\*1.5)
    - b. VPAC\_LDC\_LSE\_BUF\_ATTR1\_j (cbuf\_bpr) – defines the number of 2D blocks per row in the SL2 circular buffer.
      - i. VPAC\_LDC\_LSE\_BUF\_ATTR1\_j[25-16] CBUF\_BPR\_CHAN = 30
      - ii. VPAC\_LDC\_LSE\_BUF\_ATTR1\_j[25-16] CBUF\_BPR\_CHAN = 1
  4. Base Address and Channel Enable:
    - a. VPAC\_LDC\_LSE\_BUF\_BA\_j[23-6] ADDR – defines the SL2 base address of the Circular buffer for this output channel.
    - b. VPAC\_LDC\_LSE\_BUF\_BA\_j[31] ENABLE – enables this output channel.

#### 6.9.5.3.5 LDC Programming Restrictions and Special Cases

The following list of LDC programming restrictions should be considered:

1. VPAC\_LDC\_LSE\_BUF\_ATTR1\_j[25-16] CBUF\_BPR\_CHAN supports 1023 number of block per row. Hence, with output block width of 8, maximum line size supported in SL2 memory is 8184 pixels.
2. In 422 mode, always program as data format as 8-bit (design does not select data format automatically based on mode)
3. Due to hardware limitations, make sure that input block size (height and width) does not cross 1023 (pixels/lines).
  - The hardware generates either PIX\_IBLK\_MEMOVF or MESH\_IBLK\_MEMOVF events in this case.
4. VPAC\_LDC\_CTRL[0] LDC\_EN should be high before issuing hts\_init.



## 6.9.6 VPAC Multi-Scaler (MSC)

### 6.9.6.1 MSC Overview

VPAC\_MSC (VPAC Multi-Scaler, hereinafter simply referred to also as MSC) is a Hardware Acceleration (HWA) module in Vision Pre-processing Accelerator (VPAC) . MSC supports two sets of multi-output scaling filters to accelerate generation of pyramid (Octave) and intra-octave scales required by Optical Flow processing and DSP vision algorithms.

#### 6.9.6.1.1 MSC Features

- 10 scaled outputs independently mapped to 1 or 2 processing threads (any number up to 10 scaler outputs can be mapped to one thread, while the remaining scaler outputs can be mapped to the other thread)
- Programmable scaling engine:
  - Pyramid or intra-octave scale generation
  - $1x \sim x/4$  image downscaling
  - $1x$  generic separable convolution filtering
  - On-the-fly (OTF) or memory-to-memory (M2M) operation with line buffers in SL2 (thus, frame width not constrained by internal line buffer width).
- Polyphase filters:
  - 5-tap separable filter kernel structure
  - Programmable kernel sizes for vertical/horizontal filter (3, 4, or 5)
  - Polyphase implementation with support of 64 or 32 phases
  - Independent programming of vertical/horizontal filters with initial phase programming option
- Filter Coefficients (shared by all scalers):
  - Coefficients in signed 10-bit number (typically 1-bit integer with 8 fractional bits)
  - Programmable Coefficient shift size to set precision of coefficient number (shift size represents the fractional bit width)
  - 2 dedicated sets of 5-tap single phase filter coefficients (for example, Gaussian) for Pyramid (Octave) generation or custom convolution filtering.
  - 4 sets of 5-tap x 32 phase coefficients
    - Two can be combined to support 5-tap x 64-phase filters
    - First can be configured to hold additional 5-tap single-phase convolution filters for non-resizing application.
- Input/output format support:
  - Each thread can support the following input source formats:
    - One plane (uniform or interleaved format - Y, CbCr interleaved, R)
  - Each Input channel supports 8/12-bit pixel data in 8/12/16-bit pixel container with programmable LSB/MSB alignment
  - Each Output channel supports 8/12-bit pixel data in 8/12/16-bit pixel container with programmable LSB/MSB alignment
- Performance:
  - 1 cycle/pixel per plane
  - Input line skip for thread performing only pyramid-generation (50% less processing time)
- Other features:
  - Programmable Input and Output Region of Interest (ROI) (or clipping) for each scaler
  - Support for Vertical and Horizontal edge Padding (replication) for both luma/chroma data plane inputs
- Functional Safety Support: CRC frame data signature capture on all output channels for fault detection in the data path.

#### 6.9.6.1.2 MSC Not Supported Features

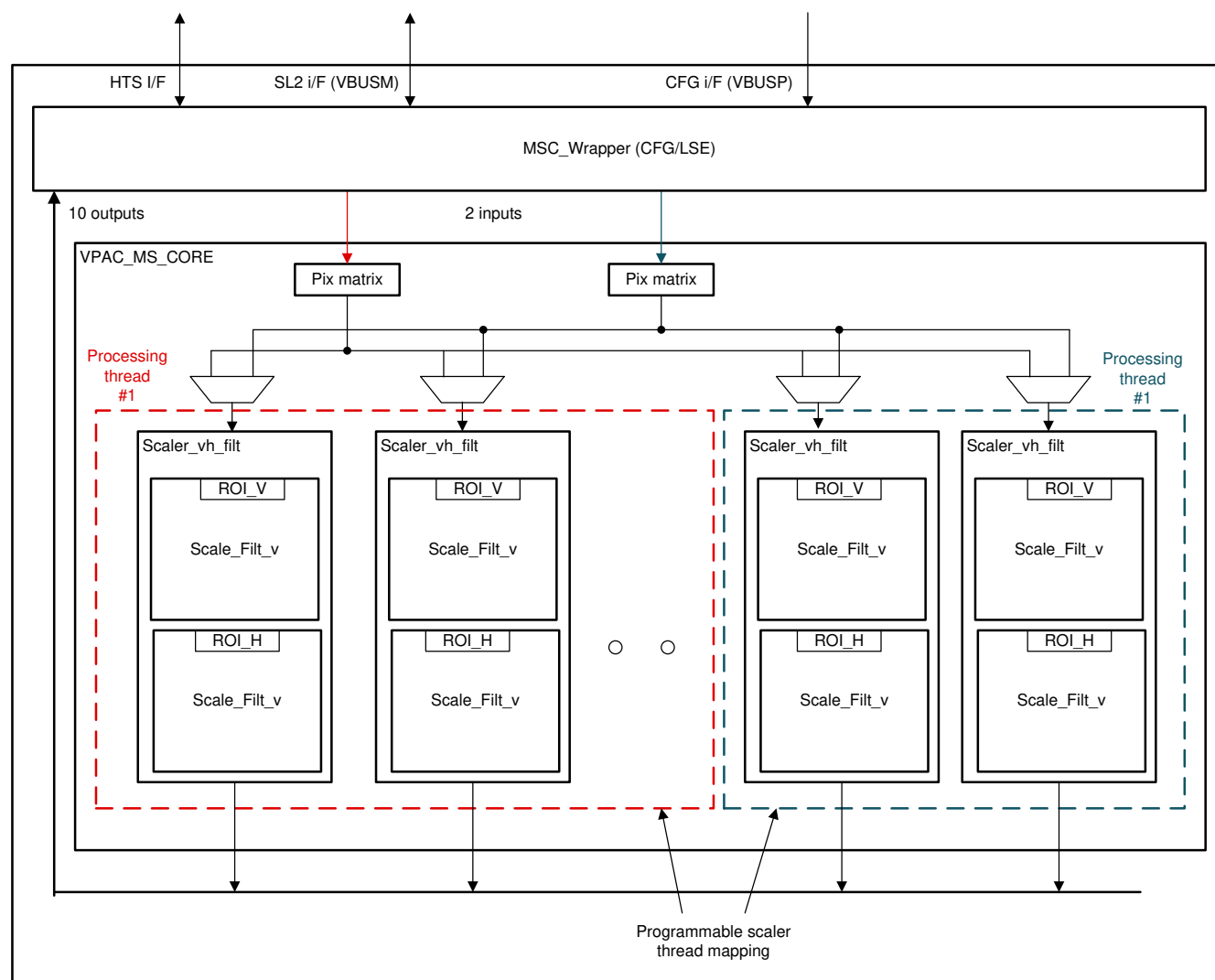
- More than 2 input threads or 10 outputs
- Block based operation for scalar
- Multi-component (RGB) Scaling
- Error checks for out of bound parameters

## 6.9.6.2 MSC Functional Description

### 6.9.6.2.1 MSC Functional Overview

VPAC\_MSC consists of 10 programmable resizers performing multi-thread/multi-scaling operations (1-input to N-outputs and 1-input to M-outputs where  $N+M$  is 10 or less). Each processing thread of MSC HWA reads its input plane data from a Shared Memory buffers (SL2) circular line buffers, performs multi-scaling operations (ratios between X and 0.25X) on the selected thread channel input, and writes out results to SL2 circular line buffers. In case of OTF operation, the source data is generated from another HWA. In case of M2M operation, the source data is read from the DDR memory. Data transfers from/to SL2 to/from external memory (or another HWA) are handled by VPAC level DMA controller with transfer request events coming from VPAC top level HWA Thread Scheduler (HTS), see *HWA Threads Scheduler (HTS)*.

A typical configuration of the MSC module uses one input plane data to generate 7 intra-octave scales and the next octave image. Multi-pass processing is then utilized to generate up to next 6 or 7 sets of scales/octaves. Since the resolution after each octave is essentially reduced by 4x, the performance requirement for generating an infinite number of scales is bounded by 1.33x of the base input image pixel rate. The second input can be used to enable a pyramid generation processing for another input data or a set of scales/pyramid generation for chroma data as long as the total number of scales needed by both threads is 10 or less. Each resizer in a processing thread can access any coefficient set.



**Figure 6-123. Block Diagram of Multi-Scaler Filter Core**

For details on MSC integration in VPAC, see [Section 6.9.3, VPAC Subsystem Level](#).

#### **6.9.6.2.1.1 MSC Submodule Details**

The MSC\_LSE (Load Store Engine) handles all interfacing to the SL2 memory space and performs following functions:

- Provides interface to SL2 circular buffers via a common VBUSM master interface
- Manages input and output channel updates in sync with HTS
- Provides data unpacking for core and packing for SL2 interface
- Manages LSE specific configuration registers

The MSC\_CORE performs all filtering operations for various 2-in/10-out multi-scaling/multi-thread configurations. It has no direct connection to external interface and it works on a frame data coming in and out in raster scan order.

The internal data transfers between the MSC\_LSE and MSC\_CORE are done over VBUSP write-only streaming interfaces.

##### **6.9.6.2.1.1.1 MSC Load Store Engine (MSC\_LSE)**

##### **6.9.6.2.1.1.1.1 MSC\_LSE Overview**

[Figure 6-124](#) shows the MSC\_LSC block diagram.



The MSC LSE supports following VPAC LSE features:

- Up to 2 processing threads supported (a thread is a processing chain with its own HTS start/done).
- Input channel features:
  - One SL2 input channel per thread with following features for the input channel:
    - Single data plane support
    - 8/12 packed and 12-bit unpacked source format support
    - Up to 5 lines - input kernel height support
    - 1D addressing mode only with CBUF address handling
    - Input Line Skip Support
    - Vertical boundary Edge Padding (Replication only) support
    - Channel enable
    - Video frame data with frame sync signals over a common vbusp streaming write-master interface

- Output Channel Features:
  - Up to 10 SL2 output channels (pixel data output only)
  - Single data plane support for each output channel
  - Programmable thread mapping for each output channel
  - 8/12 packed and 12-bit unpacked output pixel data format support
  - 1D addressing mode support with CBUF address handling
  - Channel enable
  - Video frame data with frame sync signals (along with optional in-band control signals) over a separate vbusp streaming write-slave interface
- HTS synchronization support
- LSE internal data bypass mode support
  - Input channel 0 can be configured to bypass/loopback mode the data to the core and/or directly to the output channel 9 (bypassing the core).

#### 6.9.6.2.1.1.2 MSC\_LSE Internal Data Loopback Channel

The MSC\_LSE provides a data loopback channel.

When enabled (VPAC\_MSC\_LSE\_CFG\_LSE[0] LOOPBACK\_EN), the LSE sends the “loopback-enabled” input channel unpacked data directly to the designated output channel packing logic without going through the HWA\_CORE. The looped back data can be packed in a different format in the output channel to perform a data format conversion or can be packed in the same format as the input.

When the loopback mode is enabled, the “loopback” designated output channel is no longer available for HWA\_CORE output data transfer. But, the input channel can still operate normally to read in the data for the HWA\_CORE (provided that the other output channel is used for transferring out the HWA\_CORE output data) by setting VPAC\_MSC\_LSE\_CFG\_LSE[1] LOOPBACK\_CORE\_EN. In this mode, the input channel flow is throttled by both the core and the loopback data path.

The data flow of the loopback mode is also controlled by the HTS synchronization signals.

#### 6.9.6.2.1.1.3 MSC\_LSE PSA Support

As a safety function, LSE integrates a CRC signature capture mechanism on each output channel data to verify the integrity of the HWA internal paths. When enabled (VPAC\_MSC\_LSE\_CFG\_LSE[8] PSA\_EN), every valid transfer frame data from the MSC is passed through a CRC module to update the signature for the channel before the data is packed and sent to the SL2 buffer. At the end of frame condition, the final signature is saved in a separate read-only signature register (VPAC\_MSC\_LSE\_PSA\_SIGNATURE\_y[31-0] VAUE = 0,...,Number\_of\_Output\_Channel-1) till the next end of frame condition for software to read and compare it against the golden reference signature.

The PSA signature register is based on the following IEEE 802.3 standard 32-bit CRC polynomial:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

The PSA module is inserted in the LSE data path as shown in [Figure 6-124](#).

#### 6.9.6.2.1.1.4 MSC\_LSE Feature Detailed Description

For detailed LSE feature description, see *Load Store Engine (LSE)*.

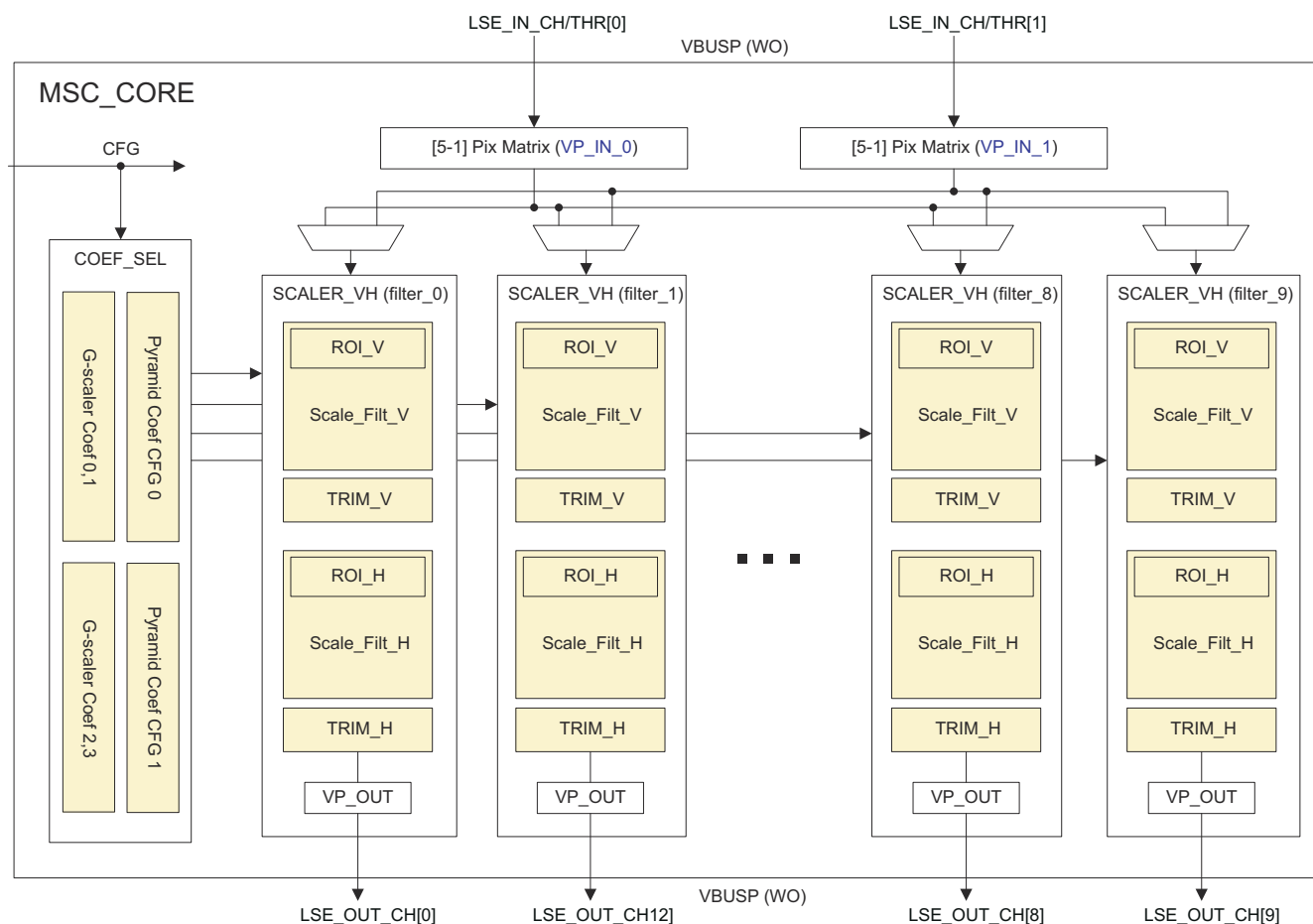
#### 6.9.6.2.1.1.2 MSC\_CORE (HWA Core)

The MSC\_CORE (HWA\_CORE) is configured as two multi-scaling engines each of which can perform 1-to many (multi) scaling operations on an independent input source. The number of outputs (scaler filters) assigned to each multi-scaling engine is user-configurable.

#### 6.9.6.2.1.1.2.1 MSC\_CORE Overview

[Figure 6-125](#) shows the architectural overview of the MSC\_CORE. Each reconfigurable multi-scaling engine in the MSC\_CORE receives its input frame data from its own dedicated VP (vbusp write only slave) interface. VP\_IN\_0 port is used for the thread #0 (multi-scaling engine-0) and VP\_IN\_1 port is used for the thread

#1 (multi-scaling engine-1). The `THREAD_MAP` parameter of the `MSC_LSE` output channel configuration register (`VPAC_MSC_LSE_DST_BUF_CFG_j[7] THREAD_MAP`) controls which input port is connected to which scaling filter (filter\_0 ~ filter\_9).



**Figure 6-125. MSC\_CORE Block Diagram**

The scaling filter implements the vertical-before-horizontal architecture in order to avoid intermediate result line buffers that would be needed in the horizontal-before-vertical configuration. This architecture also allows the source frame width to be not restricted by internal line buffer sizes.

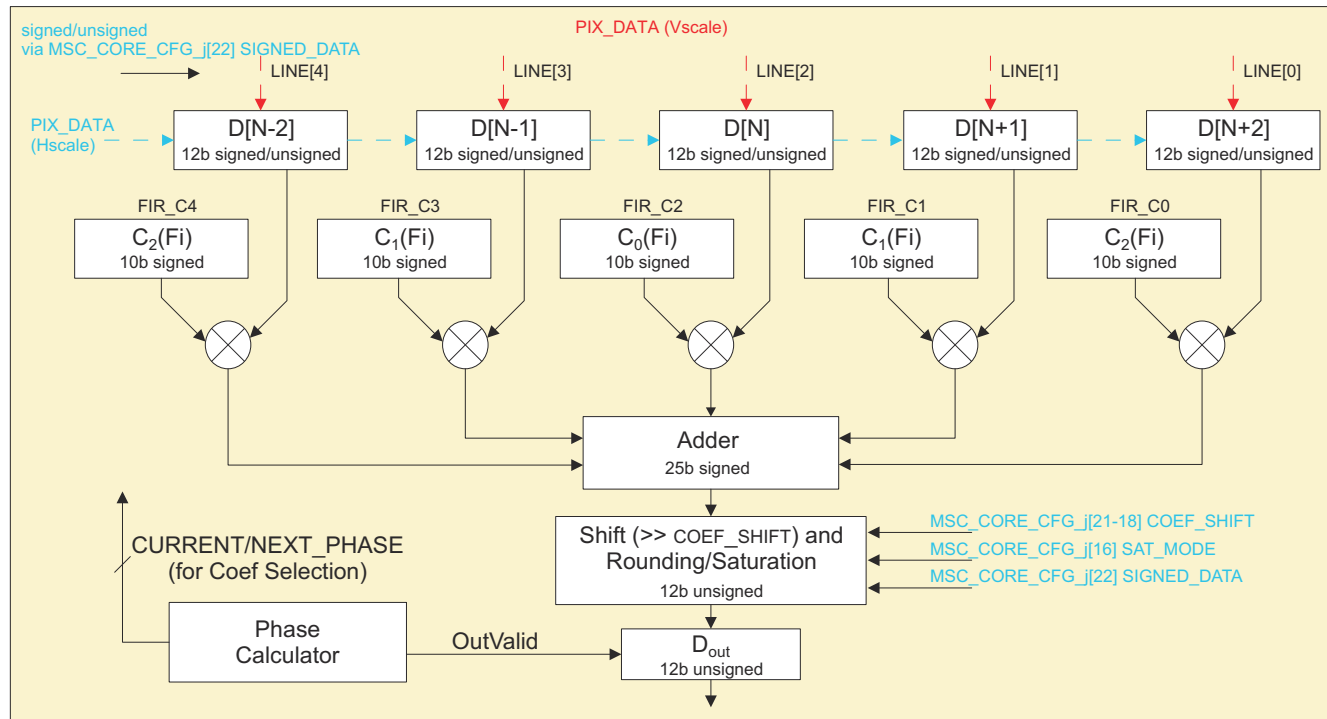
Each scaling filter in the `MSC_CORE` consists of 5-tap (32/64-phase) Polyphase filter based vertical and horizontal resizers. At the input of each resizer, a ROI checker determines the input start position (that is, marks the input data as valid). Output of each resizer is then trimmed to the configured output size of the scaling filter in each respective direction.

For `MSC`, the vertical input edge padding is performed in the `MSC_LSE` while the horizontal input boundary padding is done within the `MSC_CORE` before the horizontal resizer.

#### 6.9.6.2.1.1.2.2 Polyphase Filter of Vertical/Horizontal Resizers

##### 6.9.6.2.1.1.2.2.1 Filter Data Path Logic

Figure 6-126 shows the micro-architecture of the polyphase filter data path. The 5-tap filter structure can be programmed to perform a 5-tap Gaussian filter (for Octave generation) or a 4-tap bi-cubic downscaling filter (for scale generation). Or, it can be configured to perform a custom convolution filter without resizing. Filter coefficients for any unused taps should be set to 0. For example, in case of a 4-tap filtering, `C2(Fi)` filter coefficient should be set to 0. For 3-tap filtering, `C-2(Fi)` and `C2(Fi)` filter coefficients should be set to 0.



**Figure 6-126. Polyphase Filter Micro-Architecture**

### Kernel Size Configuration

There is no separate configuration required to set the filter kernel size (number of taps) within the polyphase filter. The filter always works as a 5-tap filter but requires unused taps to be masked using the zero coefficients.

### Coefficient Precision Selection

The precision of the coefficients is user-configurable using VPAC\_MSC\_CORE\_CFG\_j[21-18] COEF\_SHIFT which determines the number of fractional bits of the coefficient.

### Unsigned/Signed Data Type support

Typically for image resizing, the input and output data are in unsigned (positive) integer numbers in the range of [0..4095]. But, in some non-image convolution filtering (for example, Sobel filter), the input and output data may be in signed integer data format. The MSC supports a user-selectable data type mode bit (VPAC\_MSC\_CORE\_CFG\_j[22] SIGNED\_DATA) per resizer to specify whether the input/output data is in signed or unsigned data format. The selection determines the input range and output clipping limits as shown below:

VPAC\_MSC\_CORE\_CFG\_0[22] SIGNED\_DATA = 0 -> [0..4095]

VPAC\_MSC\_CORE\_CFG\_0[22] SIGNED\_DATA = 1 -> [-2048..2047]

Note that all filters in the same processing thread should have the SIGNED\_DATA bit set to a same value.

### Rounding Logic

The rounding is done by “adding 0.5 and dropping fractional bits”. This achieves the following:

For data  $\geq 0$ : “round to nearest, ties away from zero”

For data  $< 0$ : “round to nearest, ties toward zero”

### Output Saturation Logic

For some non-image unsigned data filtering, the result of the final adder could be negative numbers. To avoid clipping all negative numbers to 0, the MSC filter data processing supports a mode

(VPAC\_MSC\_CORE\_CFG\_j[22] SAT\_MODE = 1) to offset the result by 2048 and then clip the final result to [0..4095] in the output saturation logic to preserve the full data range.

### Edge Pixel Replication

Pixel replication in vertical direction (edge line replication) is done in the MSC\_LSE.

Pixel replication in horizontal direction (edge pixel replication) is performed in the horizontal filter. For chroma data plane, Cb/Cr pixel replication is done independently with edge Cb/Cr data respectively.

#### 6.9.6.2.1.2.2.2 Filter Parameters

[Table 6-139](#) lists the parameters used to define the configuration of the resizing (downsampling) vertical/horizontal filters.

**Table 6-139. Parameters of the Resizing (Downsampling) Vertical/Horizontal Filters**

Parameter	Bit Precision	Description
FIRINC_V VPAC_MSC_CORE_FIRINC_j[30-16] VS	U15Q12 (Unsigned 15b number with 12-bit fraction)	Vertical Resizing Ratio value (vertical filter increment) Valid Range: 4096 (1x scaling) < FIRINC_V < 16384 (1/4x scaling) +1 in the above equation is optional. +1 forces non-repeating coefficients in integer ratio scaling cases.
FIRINC_H VPAC_MSC_CORE_FIRINC_j[14-0] HS	U15Q12 (Unsigned 15b number with 12-bit fraction)	Horizontal Resizing Ratio value (Horizontal filter increment) Valid Range: 4096 (1x scaling) < FIRINC_V < 16384 (1/4x scaling)
ACC_INIT_V VPAC_MSC_CORE_ACC_INIT_j[27-16] VS	U12Q12 (Unsigned 12b number with 12b fraction)	Initial Vertical Resizer Phase Valid Range: 0 < ACC_INIT_V < 4095
ACC_INIT_H VPAC_MSC_CORE_ACC_INIT_j[11-0] HS	U12Q12 (Unsigned 12b number with 12b fraction)	Initial Horizontal Resizer Phase Valid Range: 0 < ACC_INIT_V < 4095

Additionally, [Table 6-140](#) lists the mode parameters defined the filter operation and coefficient selection.

**Table 6-140. Filter Mode and Coefficient Selection via VPAC\_MSC\_CORE\_CFG\_j Registers**

Parameter	Bit Width	Description
SIGNED_DATA VPAC_MSC_CORE_CFG_j[22] SIGNED_DATA	1-bit	Integer type of input and output frame data: 0 : Unsigned 12-bit (default) 1 : Signed 12-bit
UV_MODE VPAC_MSC_CORE_CFG_j[17] UV_MODE	1-bit	Source data interleave format: 0 - non-interleaved (Y data) 1 - interleaved (UV data)
SP_VS_COEF_SEL VPAC_MSC_CORE_CFG_j[15-12] SP_VS_COEF_SEL	4-bits	Single Phase – Vertical Filter Coef Selection N : 5-tap/32-phase Filter Coef Set #0 (Entry N) where N=0-9 10 : Gaussian Filter #0 11 : Gaussian Filter #1 12-15 : Invalid When FilterMode=0 and N < 10, then the multi-phase coef set #0 is dedicated for one or more single phase filters.



**Table 6-140. Filter Mode and Coefficient Selection via VPAC\_MSC\_CORE\_CFG\_j Registers (continued)**

Parameter	Bit Width	Description
SP_HS_COEF_SEL VPAC_MSC_CORE_CFG_j[10-7] SP_HS_COEF_SEL	4-bits	Single Phase – Horizontal Filter Coef Selection N : 5-tap/32-phase Filter Coef Set #0 (Entry N+10) where N=0-9 10 : Gaussian Filter #0 11 : Gaussian Filter #1 12-15 : Invalid When FilterMode=0 and N < 10, then the multi-phase coef set #0 is dedicated for one or more single phase filters.
VS_COEF_SEL VPAC_MSC_CORE_CFG_j[5-4] VS_COEF_SEL	2-bits	Multi-phase Vertical Coef Selection 00 : 5-tap/32-phase Filter coef set #0 01 : 5-tap/32-phase Filter coef set #1 10 : 5-tap/32-phase Filter coef set #2 11 : 5-tap/32-phase Filter coef set #3
HS_COEF_SEL VPAC_MSC_CORE_CFG_j[3-2] HS_COEF_SEL	2-bits	Multi-phase Horizontal Coef Selection 00 : 5-tap/32-phase Filter coef set #0 01 : 5-tap/32-phase Filter coef set #1 10 : 5-tap/32-phase Filter coef set #2 11 : 5-tap/32-phase Filter coef set #3
PHASE_MODE VPAC_MSC_CORE_CFG_j[1] PHASE_MODE	1-bit	Filter Phase mode selection 0 - 64 phases 1 - 32 phases
FILTER_MODE VPAC_MSC_CORE_CFG_j[0] FILTER_MODE	1-bit	Filter Mode 0 – Single Phase Filter (for example, Gaussian Filter for Pyramid) 1 – Multi-phase Scaling Filter
FILT_SAT_MODE VPAC_MSC_CORE_CFG_j[16] SAT_MODE	1-bit	Filter Output Saturation Mode 0 - [0..4095] clipping 1 - [-2048.. 2047] clip followed by +2048 This parameter is used only when signed_data = 0 (unsigned data type).
COEF_SHIFT VPAC_MSC_CORE_CFG_j[21-18] COEF_SHIFT	4-bits	Coef Shift Size: configures the precision of the 10-bit signed filter coefficients (valid Shift range: 5~9) : 5: Shift by 5 (5-bit fraction) 6: Shift by 6 (6-bit fraction) 7: Shift by 7 (7-bit fraction) 8: Shift by 8 (8-bit fraction) 9: Shift by 9 (9-bit fraction) Integer size = 9 - #_of_fraction_bits

#### 6.9.6.2.1.1.2.2.3 Single-Phase Filter Parameters

For a single-phase filter configuration (VPAC\_MSC\_CORE\_CFG\_j[0] FILTER\_MODE = 0), software must ensure that FIRINC parameters (VPAC\_MSC\_CORE\_FIRINC\_j[30-16] VS and VPAC\_MSC\_CORE\_FIRINC\_j[14-0] HS) are programmed with one of the following values only:

- 0x4000 (1/4 X resizing)
- 0x2000 (1/2 X resizing)
- 0x1000 (1X – no scaling)

VPAC\_MSC\_CORE\_ACC\_INIT\_j[27-16] VS and VPAC\_MSC\_CORE\_ACC\_INIT\_j[11-0] HS configuration values are ignored by the MSC hardware (internally set to 0x0) when VPAC\_MSC\_CORE\_CFG\_j[0] FILTER\_MODE = 0. If any phase offsets are needed, the starting pixel/line locations of the frame and the filter coefficient values should be adjusted to implement the desired phase shift.

#### **6.9.6.2.1.2.2.4 Interleaved Mode Handling**

When the input is in the interleaved data format (for example, YUV420 Chroma data), the horizontal filter supports two sets of 5-input buffers (Cb and Cr buffers) and alternates filtering operation between two. Phase calculation and coefficient updates are done every other sample. Edge replications at the start of the line and at the end of the line are handled separately for Cb and Cr data.

There is no change in how the vertical filter handles an interleaved data format source.

#### **6.9.6.2.1.2.2.5 Input Skip Line Support**

When a processing thread is only performing an octave generation (1/2x single phase resizing), the next input lines can be offset by 2 lines since the output is only generated every other input line times. By skipping lines (VPAC\_MSC\_LSE\_SRC\_CFG\_j[7] SRC\_LN\_INC\_2), the SL2 access is reduced in half for this mode and the cycle time to complete an Octave generation is equal to 1/2 of the input frame size.

Line N: Read Line N-2, N-1, N, N+1, N+2

Line N+1: Skip

Line N+2: Read Line N, N+1, N+2, N+3, N+4

Line N+3: Skip

Even though “N+1” line processing is skipped, all source lines are still required for proper filtering. Therefore, the DMA transfer from DDR to SL2 should include all lines. VPAC\_MSC\_CORE\_FIRINC\_j[30-16] VS should be set to 4096 (1x resizing) to compensate for source line skip (if it is doing a 1/2 resizing).

Horizontal Skip is done normally as 1/2 x resizing by the core HScale Filter.

#### **6.9.6.2.1.2.3 Scaler Filter Thread Mapping**

The scaling filter\_# (# = 0..9) is enabled and its input is connected to one of the processing thread source VP ports (VP\_IN\_0 or VP\_IN\_1) by configuring the output channel #n buffer configuration:

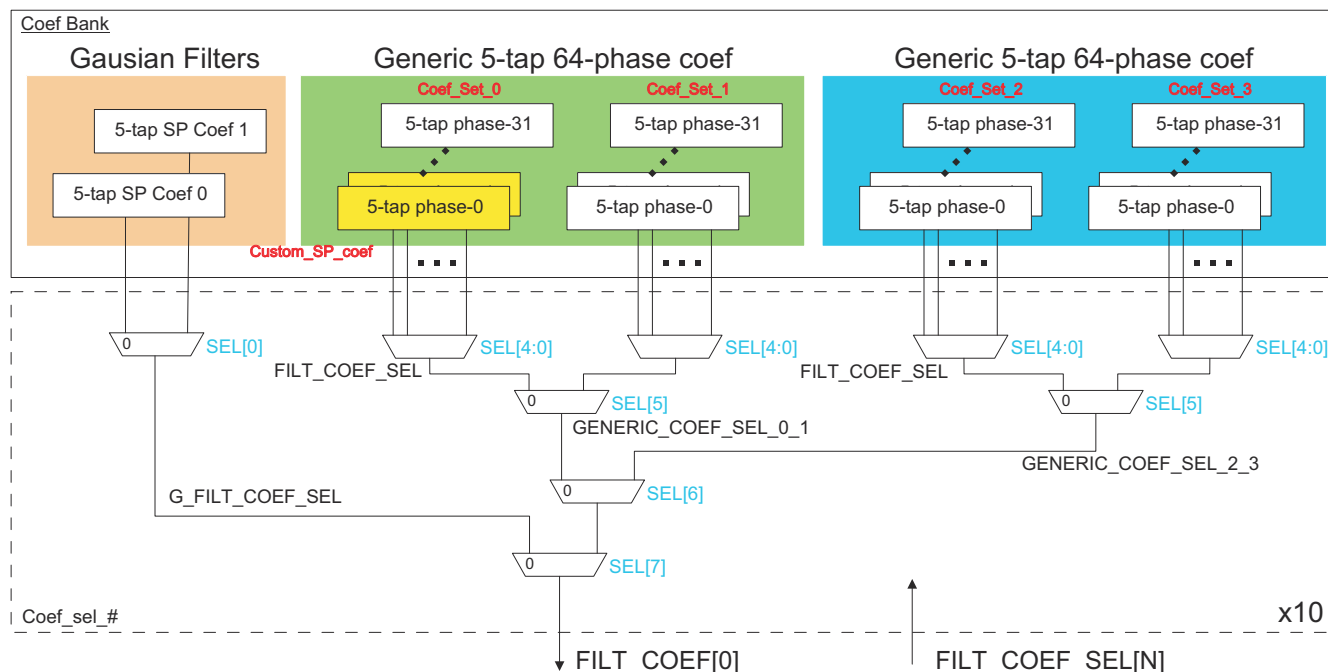
VPAC\_MSC\_LSE\_DST\_BUF\_CFG\_j[7] THREAD\_MAP: 0 maps the filter to VP\_IN\_0, 1 maps to the VP\_IN\_1.

VPAC\_MSC\_LSE\_DST\_BUF\_CFG\_j[TBD] ENABLE: enables both filter and output channel

By mapping each output to a thread explicitly in the output channel configuration register, MSC guarantees that a filter can only be connected to one processing thread.

#### **6.9.6.2.1.2.4 Filter Coefficients**

Since each resizer needs to access the common coefficients independently, the coefficients are stored in registers (instead of memory). Each resizer selects directly from these register using the coefficient selection scheme as shown in [Figure 6-127](#). The same mux is used to select either vertical or horizontal coefficients.



**Figure 6-127. Coefficient Selection**

#### 6.9.6.2.1.2.4.1 Filter Coefficient Parameter Configuration

Each coefficient table entry consists of 5 coefficients stored in two memory mapped configuration registers - VPAC\_MSC\_CORE\_C210\_j and VPAC\_MSC\_CORE\_C43\_j. FIR\_C4-0 parameters map to C-2, C-1, C0, C1, and C2 coefficients (see [Figure 6-125](#)).

#### 6.9.6.2.1.2.4.2 3/4/5-Tap Filter Configuration

All filters operate in 5-tap filter mode – requiring 5-tap coefficients regardless of the filter usage. For less-than-5 tap filter configurations, coefficients corresponding to the unused taps should be set to 0.

For example, FIR\_C0 and FIR\_C4 will need to be set to 0x0 for a 3-tap configuration.

For 4-tap configuration, FIR\_C0 will need to be set to 0x0.

#### 6.9.6.2.1.2.5 Input/Output ROI Trimmers

Each scaler filter implements the ROI scaling as shown below with input and output trimmers. The input trimmer simply marks the beginning of the ROI to indicate where the valid source pixels and lines start in the source image. Doing the source ROI marker allows the exact specification of a sub-image in a common source image (no position or size restriction) while keeping the original boundary pixels (that is, not use the replicated pixels/lines).

The output trimmers then uses destination size parameters (VPAC\_MSC\_CORE\_OUT\_SIZE\_j[28-16] HEIGHT and VPAC\_MSC\_CORE\_OUT\_SIZE\_j[12-0] WIDTH) to trim out the exact final output image sizes.

VPAC\_MSC does not perform output size correction by stuffing extra pixels in the case the programmed VPAC\_MSC\_CORE\_OUT\_SIZE\_j[28-16] HEIGHT and VPAC\_MSC\_CORE\_OUT\_SIZE\_j[12-0] WIDTH exceeds the actual output size of the scaler. In this case, VPAC\_MSC simply outputs the actual size.

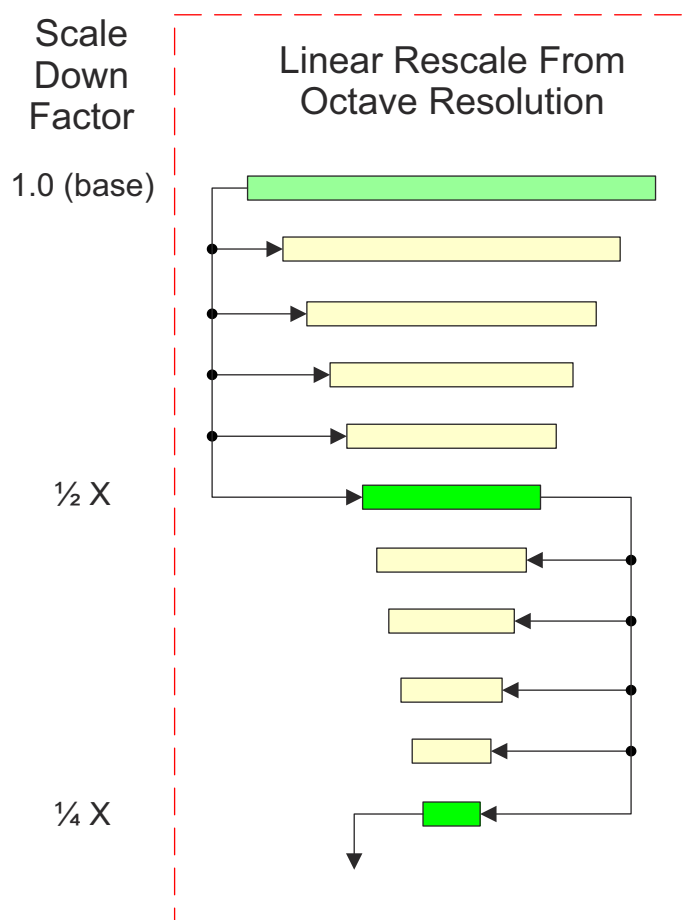
#### 6.9.6.2.2 Resizer Algorithm Details

This section describes the algorithm details of the Horizontal and Vertical Down-Scaling engines.

##### 6.9.6.2.2.1 Multiple Scales Generations

For a high number of scales needed by vision image processing algorithm (for example, 32 or more), there are many options in generating the multiple scales as shown in [Figure 6-128](#). Among the options, the “linear rescale

from octave resolution” gives the best tradeoff in video quality and design simplicity. The MSC implements 1-to-N multi-rescaling (that is, from one base image to multi-intra-scales and the next octave image generation) as shown as option B in [Figure 6-128](#).



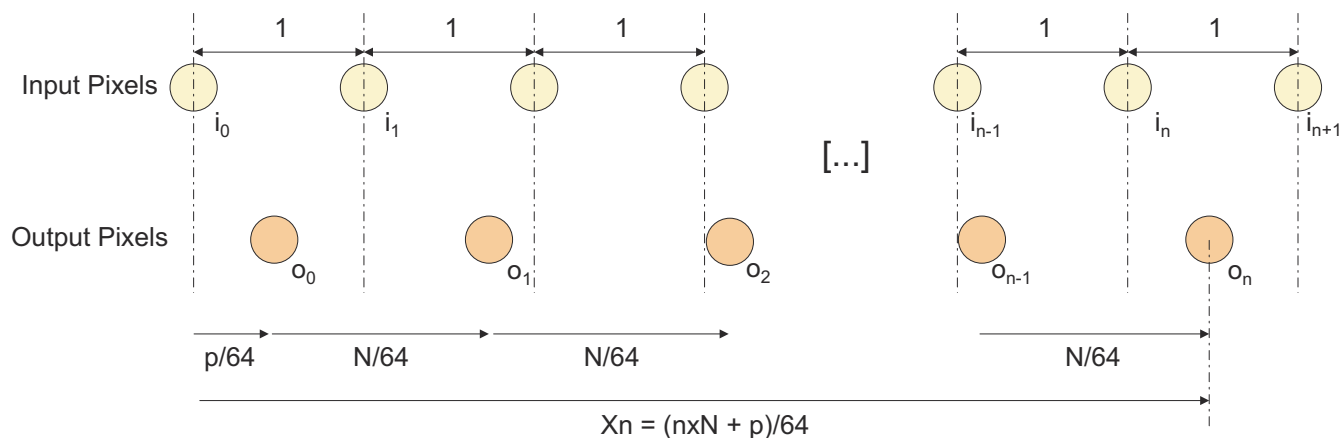
**Figure 6-128. Multiple Scale Generation**

#### 6.9.6.2.2.2 Polyphase Filter

Both vertical and horizontal resizers in MSC hardware are implemented using a programmable Polyphase filter structure supporting 5-tap kernel with either 64 or 32 phases.

##### 6.9.6.2.2.2.1 Interpolation/Resampling

[Figure 6-129](#) illustrates the interpolation principle. The assumptions are that the input pixels are evenly spaced. The distance between each input pixel is assumed to be 1. The magnification ratio is given by  $64/N$  and  $p/64$  is the initial phase of the output data. The output pixels are evenly spaced as well. The distance between each output pixel is given by  $N/64$  (in the example below,  $N > 64$ ). The position of the  $n$ -th output pixel is given by  $(n \times N + p) / 64$ .



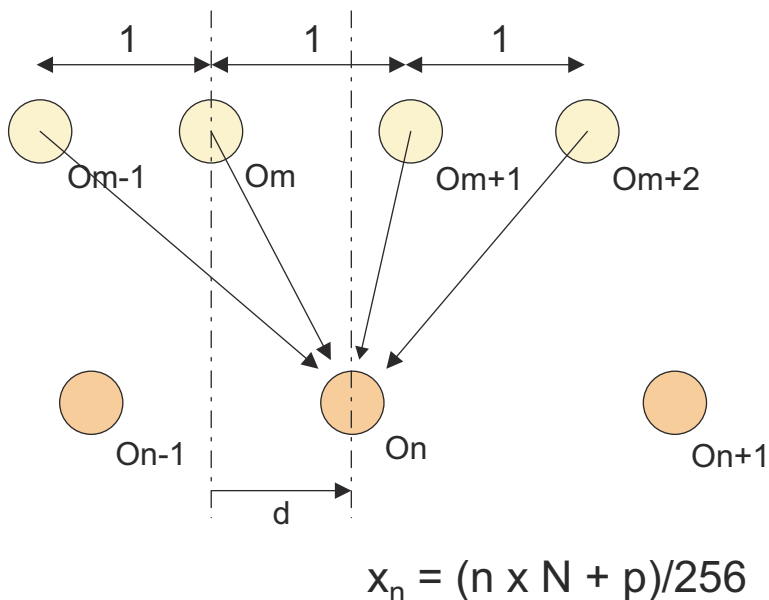
**Figure 6-129. Interpolation Principle**

Figure 6-130 illustrates the interpolation principle at the n-th output pixel ( $o_n$ ) at position  $x_n$ . Interpolation method with  $m$  and  $d$  parameters, are described as:

$$m = \text{floor}((n \times N + p) / 64)$$

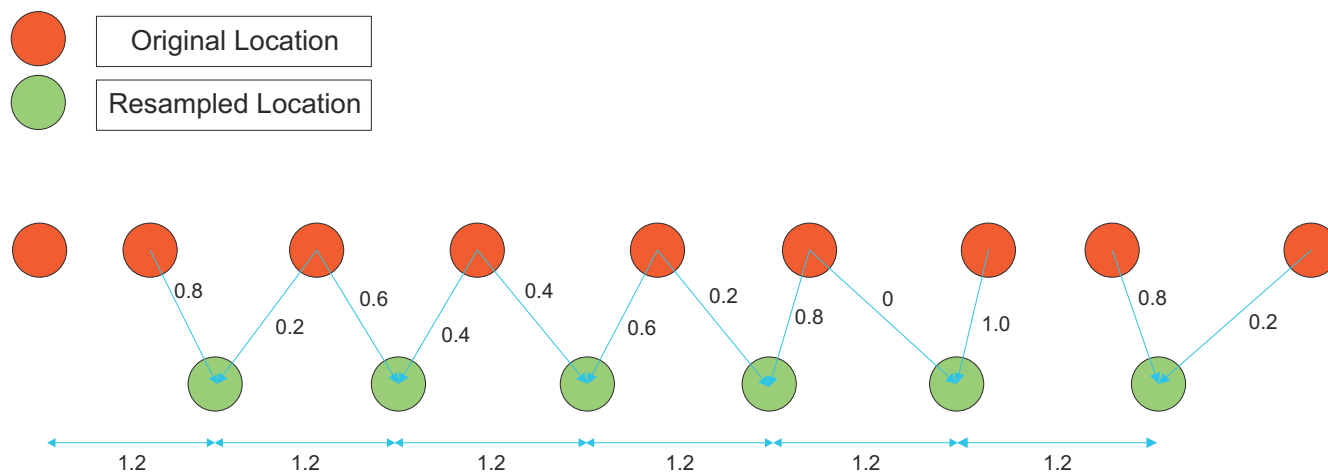
$$d = ((n \times N + p) / 64) - m$$

Figure 6-130 illustrates the concept when the filters are configured in 4-tap mode for non-integer rescaling (the first coefficient corresponding to  $O_{m-2}$  is set to zero).



**Figure 6-130. Interpolation Process**

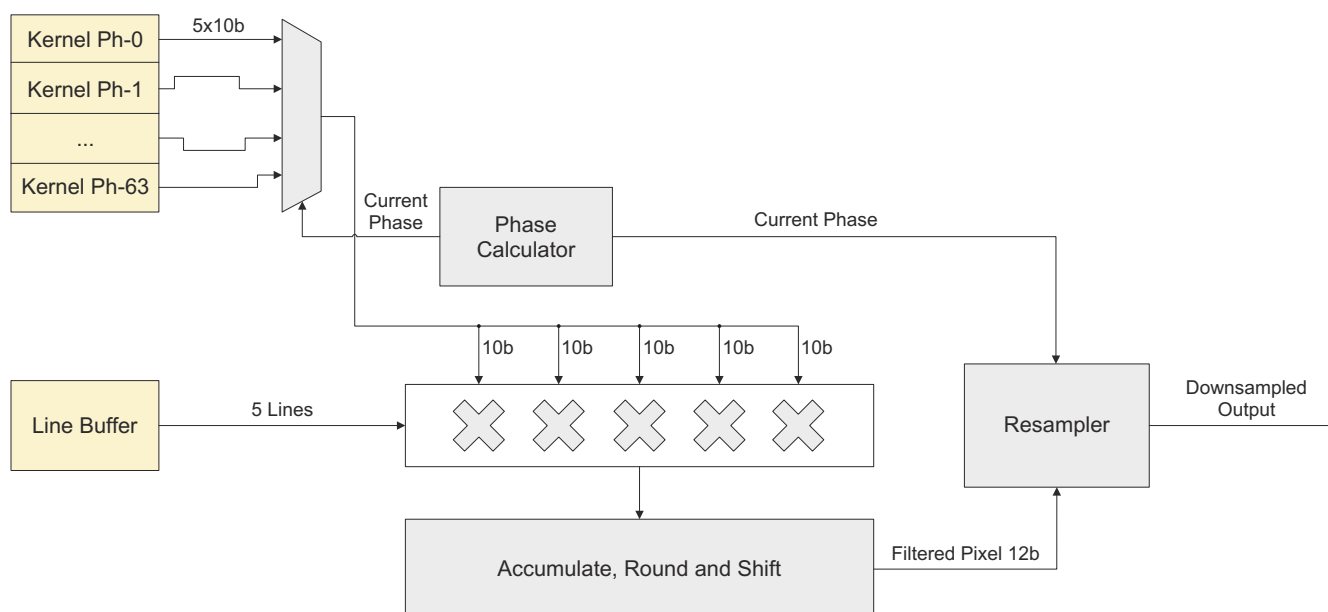
Figure 6-131 illustrates another explanation of the concept of a Polyphase filter using a simple 2-tap linear filter in a design which supports 5 phases.



**Figure 6-131. Polyphase Filtering**

For each one of the 5 phases, the weights (or filter coefficients) are different and after the 5 phases the weights start repeating, see [Figure 6-131](#).

[Figure 6-132](#) depicts the architecture for the independent polyphase filter unit.



**Figure 6-132. Polyphase Filter Unit**

Both the horizontal and vertical filter described earlier are an instance of the Polyphase filter unit, though only the vertical filter requires line memories (implemented in SL2 memory space). In all, there will be 20 instances of the Polyphase filter unit, 2 filters per channel and 10 total channels in the VPAC\_MSC module.

#### 6.9.6.2.2.2 Phase Calculation and Re-sampler

The phase calculation block calculates the current phase which in turn is used for selecting the right coefficient set and for the output qualifier in the Re-sampler block. The Re-sampler engine then decides whether or not the output created by the interpolation unit is a valid output or not.

#### 6.9.6.2.2.3 Shared Coefficient Buffers

The MSC module supports up to 4 sets of multi-phase coefficients for generic non-integer resizing and up to 10 sets of single-phase coefficients (5x1 entries) for Octave generation and integer resizing applications. The multi-phase coefficients can also be configured in one of two following options:

- 4 sets of 5-tap x 32 phase coefficients
- 1 set of 5-tap x 64 phase coefficients and 2 sets of 5-tap x 32 phase coefficients
- 2 sets of 5-tap x 64 phase coefficients

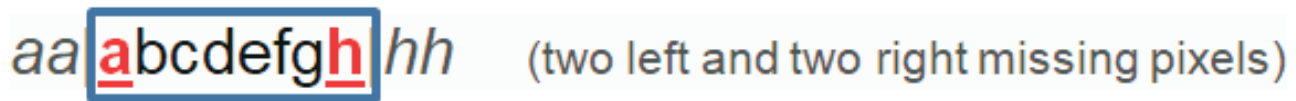
The 32 phase configuration option allows using independent coefficients for vertical and horizontal resizer to achieve aspect ratio change during the scaling.

#### 6.9.6.2.2.4 Border Pixel Padding

The MSC module replicates missing lines and pixels at top/bottom and left/right sides of the input frame for non-interleaved (luma) and interleaved (chroma) data.

The missing lines at the top/bottom are replicated in the LSE sub-module prior to the vertical scaler and the missing pixels at the left/right sides of each line are replicated in the core prior to the horizontal scaler. For more information about LSE module, see *Load Store Engine (LSE)*.

Replication scheme simply repeats the edge line or pixel to create missing lines/pixels. [Figure 6-133](#) shows horizontal pixel replication.



**Figure 6-133. Horizontal Pixel Replication**

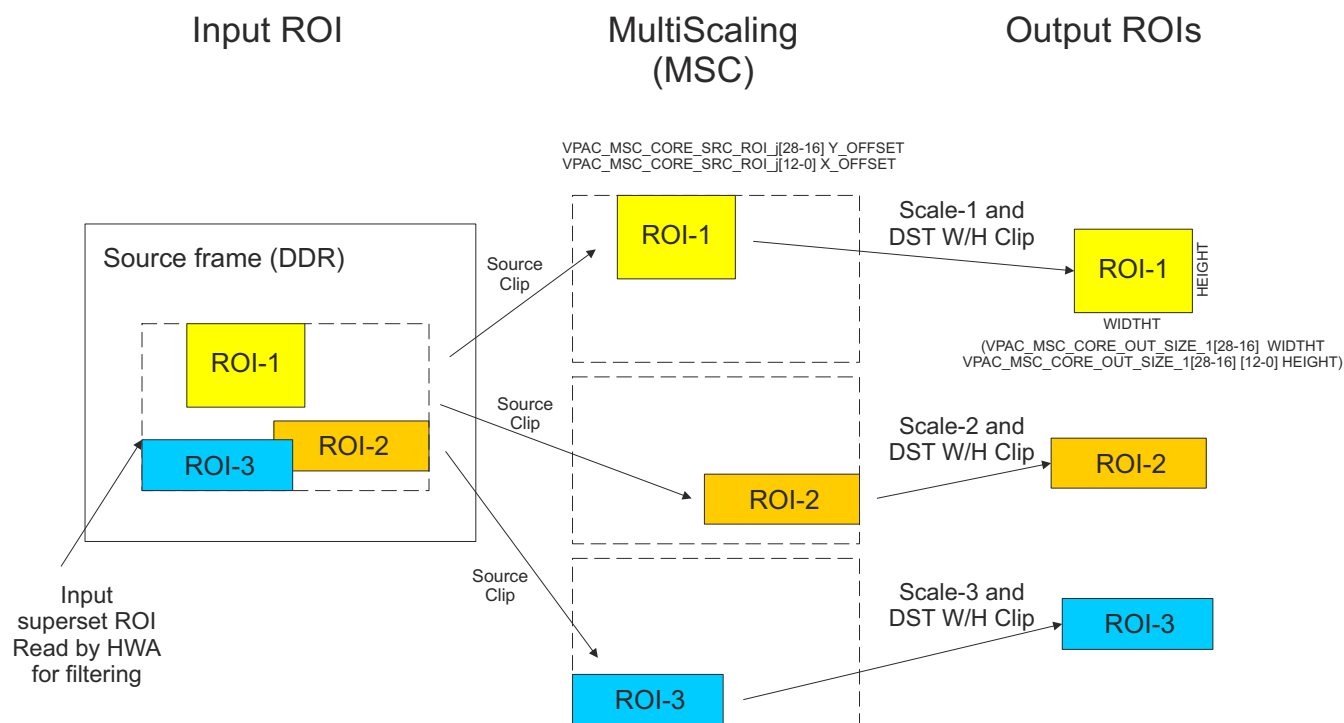
If a scaling is performed on a sub-frame (ROI) within the input frame, no replicated lines/pixels will be used since the lines/pixels outside the sub-frame are already present in the input frame.

For interleaved data format, the horizontal replication is done separately for U and V data. For U-data, edge U-pixel will be replicated. For V-data, edge V-pixel will be replicated. There is no distinction in the vertical edge padding.

#### 6.9.6.2.2.3 ROI Handling

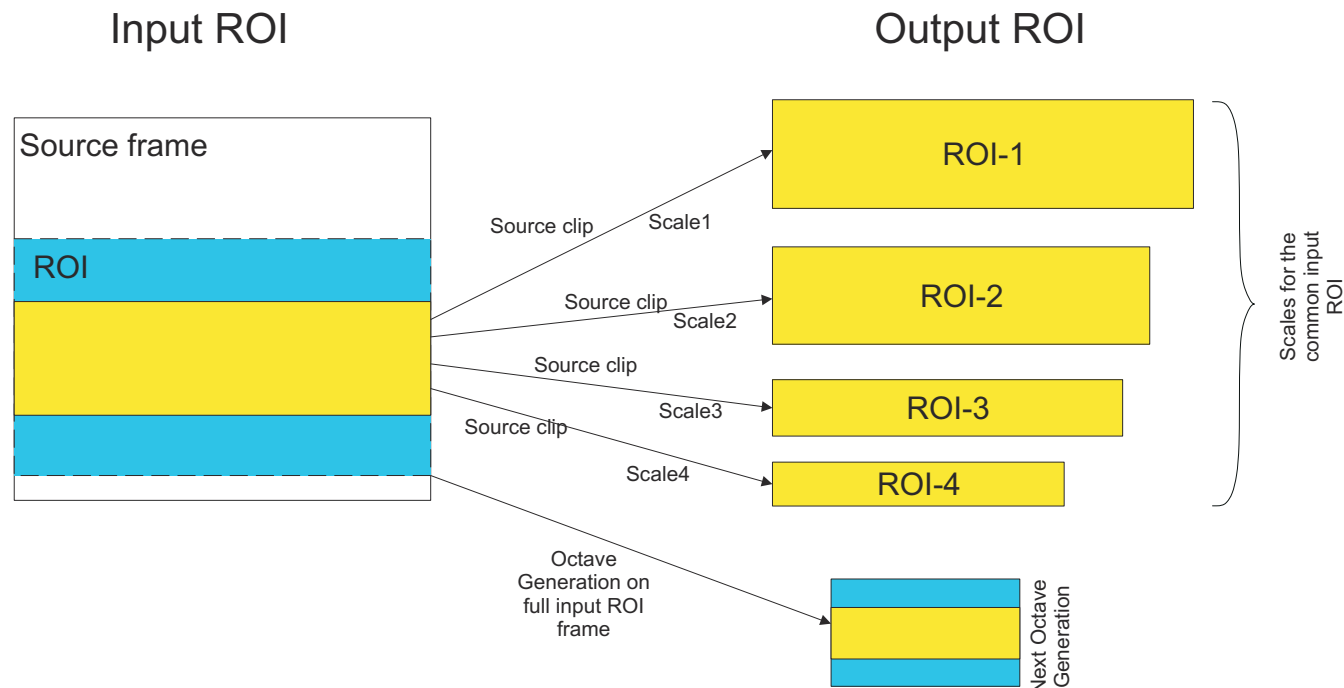
- ROI supported during scaling generation
  - Only one ROI at Input Frame and one ROI at Output frame per scale
  - ROI definitions (for input as well as all output scales) can be changed at every input frame under software control.
  - ROI for each scale (for output) can be different among scales (as well as with respect to the input). It is set under software Control.
  - The key purpose of ROI is DDR bandwidth reduction. There is additional help by reduction on processing power on DSP and hardware.
- The input ROI is a superset area considering following two scenarios.
  - Multiple ROIs in given Input Frame, if there are multiple ROI for input frame
  - Input Frame area from Overlapped ROI across scales

[Figure 6-134](#) shows an example of multiple ROIs in the source superset ROI area.



**Figure 6-134. Multiple-ROI Support Illustration**

Figure 6-135 illustrates another application of generating output ROIs with different scales from a common source region.



**Figure 6-135. Multiple-ROI From a Common Source ROI**

By defining output ROIs, only the data within ROI in the “superset ROI” scaled output image is saved thus reducing the DDR traffic significantly.



### 6.9.6.2.3 MSC Data Formats Supported

Table 6-141 summarizes data format supported by MSC.

**Table 6-141. MSC Input/Output Data Formats**

Module (IO)	Bit-Depth	Chroma Format	Packing
MSC Input	8-bit	YUV420	Fully Packed
	12-bit	YUV420	Fully Packed
	12-bit	YUV420	Unpacked in 16-bit
MSC Output	8-bit	YUV420	Fully Packed
	12-bit	YUV420	Fully Packed
	12-bit	YUV420	Unpacked in 16-bit

Table 6-142 and Table 6-143 show the pixel data memory organization for YUV420 (2-plane 12-bit fully packed format).

**Table 6-142. YUV420 (2-plane 12-bit Fully Packed Format), First Plane**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Y2[7:0]								Y1								Y0								+0 x0								
Y5[3:0]								Y4								Y3								Y2[11:8]								+0 x4
Y7								Y6								Y5[11:4]								+0 x8								
Y10[7:0]								Y9								Y8								+0 C								
Y13[3:0]								Y12								Y11								Y10[11:8]								+1 0
Y15								Y14								Y13[11:4]								+1 4								

**Table 6-143. YUV420 (2-plane 12-bit Fully Packed Format), Second Plane**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Cb1[7:0]								Cr0								Cb0																+0 x0
Cr2[3:0]								Cb2								Cr1								Cb1[11:8]								+0 x4
Cr3																Cb3								Cr2[11:4]								+0 x8
Cb5[7:0]								Cr4								Cb4																+0 C
Cr6[3:0]								Cb6								Cr5								Cb5[11:8]								+1 0
Cr7																Cb7								Cr6[11:4]								+1 4

Table 6-144 and Table 6-145 show the pixel data memory organization for YUV420 (2-plane 8-bit fully packed format)

**Table 6-144. YUV420 (2-plane 8-bit Fully Packed Format), YUV 4:2:0 – NV12, First Plane**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Y3								Y2								Y1								Y0								+0 x0
Y7								Y6								Y5								Y4								+0 x4

**Table 6-144. YUV420 (2-plane 8-bit Fully Packed Format), YUV 4:2:0 – NV12, First Plane (continued)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Y11								Y10								Y9								Y8								+0 x8
Y15								Y14								Y13								Y12								+0 xC

**Table 6-145. YUV420 (2-plane 8-bit Fully Packed format), YUV 4:2:0 – NV12, Second Plane**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Cr1								Cb1								Cr0								Cb0								+0 x0
Cr3								Cb3								Cb2								Cb2								+0 x4
Cr5								Cb5								Cr4								Cb4								+0 x8
Cr7								Cb7								Cb6								Cb6								+0 xC

**Note**

MSC supports both NV12 and NV21 chroma component ordering. Since there is no color processing, the ordering of Cb and Cr is irrelevant. Simply, if a NV12 chroma plane is the input, the output will be a NV12 chroma plane.

**12-bit unpacked format (16-bit container) with LSB/MSB alignments is shown below**

[Table 6-146](#) and [Table 6-147](#) show the pixel data memory organization for YUV420 (2-plane 12-bit unpacked format – LSB aligned)

**Table 6-146. YUV420 (2-plane 12-bit Unpacked Format – LSB aligned), YUV 4:2:0 – NV12 (12-bit) (0x3D), First Plane**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					
unused									Y1									unused									Y0									+0 x0
unused									Y3									unused									Y2									+0 x4

**Table 6-147. YUV420 (2-plane 12-bit Unpacked Format – LSB aligned), YUV 4:2:0 – NV12 (12-bit) (0x3D), Second Plane**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					
unused									Cr0									unused									Cb0									+0 x0
unused									Cr1									unused									Cb1									+0 x4

[Table 6-148](#) and [Table 6-149](#) show the pixel data memory organization for YUV420 (2-plane 12-bit unpacked format – MSB aligned)

**Table 6-148. YUV420 (2-plane 12-bit Unpacked Format – MSB aligned), YUV 4:2:0 – NV12 (12-bit) (0x3D), First Plane**

PROT-Plane																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						Y1						unused						Y0						unused						+0x0	
						Y3						unused						Y2						unused						+0x4	

**Table 6-149. YUV420 (2-plane 12-bit Unpacked Format – MSB aligned), YUV 4:2:0 – NV12 (12-bit) (0x3D), Second Plane**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
Cr0												unused												Cb0												unused	+0x0
Cr1												unused												Cb1												unused	+0x4

### 6.9.6.3 MSC Interrupt Conditions

#### 6.9.6.3.1 CPU Interrupts

Table 6-150 lists the interrupts generated by the MSC module.

**Table 6-150. Interrupts**

Interrupt	Type	Description
VPAC_MSC_LSE_FR_DONE_E VT_0	Pulse	Frame Processing complete for all filters in the processing thread 0.
VPAC_MSC_LSE_FR_DONE_E VT_1	Pulse	Frame Processing complete for all filters in the processing thread 1.
VPAC_MSC_LSE_SL2_RD_ERR	Pulse	Set whenever there is an error response on VBUSM read command request for any input channel
VPAC_MSC_LSE_SL2_WR_ER R	Pulse	Set whenever there is an error response on VBUSM write command request for any output channel

All interrupts are single pulse event signals that are mapped to the VPAC level interrupt aggregation logic. The MSC has no mask/set/clear registers for these events. For more information see [Section 6.9.3, VPAC Subsystem](#).

#### 6.9.6.3.2 Interrupt Event Description

##### 6.9.6.3.2.1 VPAC\_MSC\_LSE\_FR\_DONE\_EVT\_0/1 Events

These events are generated when all filters associated with the thread 0 or 1 complete the frame processing. This is set when input EOF (end of frame) event and output EOF (end of frame) events are detected.

##### 6.9.6.3.2.2 VPAC\_MSC\_LSE\_SL2\_RD\_ERR Interrupt Event

This event is generated whenever the VBUSM read status is something other than 0 (success).

The VPAC status register VPAC\_MSC\_STATUS\_ERROR[4-0] VM\_RD\_ERR stores the last non-zero RSTATUS value and the input channel number for user to see what type of error has occurred on which channel.

**Table 6-151. Encoding for RSTATUS**

RSTATUS	Meaning
0	Success
1	Addressing error
2	Protection error
3	Timeout error
4	Data error
5	Unsupported Addressing Mode error
6	RESERVED
7	Exclusive read fail

##### 6.9.6.3.2.3 VPAC\_MSC\_LSE\_SL2\_WR\_ERR Interrupt Event

This event is generated whenever the VBUSM write status is something other than 0 (success).

The VPAC status register VPAC\_MSC\_STATUS\_ERROR[14-8] VM\_WR\_ERR stores the last non-zero WSTATUS value and the output channel number for user to see what type of error has occurred on which channel.

**Table 6-152. Encoding for WSTATUS**

WSTATUS	Meaning
0	Success
1	Addressing error
2	Protection error
3	Timeout error
4	Data error
5	Unsupported Addressing Mode error
6	RESERVED
7	Exclusive write fail

#### 6.9.6.4 MSC Performance

Since the resizer performs only downscaling operations, the throughput of scaler is mostly dictated by the input processing rate – 1 input pixel per 1 clock cycle. Once the data is made available to the scaler filter after the initial set up and data fetching, the scaler should be fully active except for few cycles at the end of each line to flush the line.

If a processing thread is performing a pyramid generation only, the vertical line skip feature can be enabled to reduce the frame processing time approximately by ½.

#### 6.9.6.5 MSC Clocking

Complete VPAC\_MSC module operates on single clock - VPAC0\_MSC\_CLK.

#### 6.9.6.6 MSC Reset

VPAC\_MSC has one synchronous active low reset input. The entire module is reset by this reset.

#### 6.9.6.7 MSC Programmer's Guide

##### 6.9.6.7.1 Programming Model

##### 6.9.6.7.1.1 MSC Programming Guidelines

MSC does not support shadow registers for its configuration registers. All changes must be done prior to the HTS.INIT request for a frame processing.

Input and output channels are individually enabled. It is expected that at least one input and output channels are enabled for each HTS thread enabled.

The MSC does not check for proper programming of all configuration parameters. It is strictly software responsibility to ensure programming of MSC (Core and LSE) registers do not cause conflict with each other.

##### 6.9.6.7.1.2 MSC\_Core Programming Details

All filters enabled for a multi-scaling thread are programmed via:

- VPAC\_MSC\_CORE\_CFG\_j[0] FILTER\_MODE; Coefficient Set Selection - VPAC\_MSC\_CORE\_CFG\_j[3-2] HS\_COEF\_SEL, VPAC\_MSC\_CORE\_CFG\_j[5-4] VS\_COEF\_SEL, VPAC\_MSC\_CORE\_CFG\_j[10-7] SP\_HS\_COEF\_SEL, VPAC\_MSC\_CORE\_CFG\_j[15-12] SP\_VS\_COEF\_SEL; and filter tap size parameters VPAC\_MSC\_CORE\_CFG\_j[11] SP\_VS\_COEF\_SRC and VPAC\_MSC\_CORE\_CFG\_j[6] SP\_HS\_COEF\_SRC.
- Coefficients set in VPAC\_MSC\_CORE\_C210\_j, VPAC\_MSC\_CORE\_C43\_j, VPAC\_MSC\_CORE\_C210\_j\_k, VPAC\_MSC\_CORE\_C43\_j\_k registers.
- ROI source offset - VPAC\_MSC\_CORE\_SRC\_ROI\_j[28-16] Y\_OFFSET and VPAC\_MSC\_CORE\_SRC\_ROI\_j[12-0] X\_OFFSET, and size. The size of ROI depends on the X and Y

offsets. For full-size set the offsets to 0. Note that the ROI\_SIZE may or may not be same as the MSC frame size (set in MSC\_FRAME\_SIZE\_j[28-16] HEIGHT and MSC\_FRAME\_SIZE\_j[12-0] WIDTH) - depending on whether ROI is full or sub-frame sized.

- Output size - VPAC\_MSC\_CORE\_OUT\_SIZE\_j[28-16] HEIGHT and VPAC\_MSC\_CORE\_OUT\_SIZE\_j[12-0] WIDTH.
- Filter parameters - VPAC\_MSC\_CORE\_FIRINC\_j[14-0] HS, VPAC\_MSC\_CORE\_FIRINC\_j[30-16] VS, VPAC\_MSC\_CORE\_ACC\_INIT\_j[27-16] VS, and VPAC\_MSC\_CORE\_ACC\_INIT\_j[11-0] HS.

Filter thread mapping is done with output channel configuration parameters (MSC\_BUF\_CFG\_j[7] THREAD\_MAP). Specifying the mapping in one register for each channel prevents any resource conflict.

#### 6.9.6.7.1.3 MSC\_LSE Programming Details

Two processing threads can be activated independently.

##### 6.9.6.7.1.3.1 Input Thread Configuration:

1. Pixel Data Format
  - a. VPAC\_MSC\_LSE\_SRC\_CFG\_j[1-0] PIX\_FMT\_PW, VPAC\_MSC\_LSE\_SRC\_CFG\_j[3-2] PIX\_FMT\_CNTRSZ, and VPAC\_MSC\_LSE\_SRC\_CFG\_j[4] PIX\_FMT\_ALIGN parameters define the pixel input data format for all input channels of the thread.
  - b. Common data formats:
    - i. Fully packed 12-bit: PIX\_FMT\_PW = 1, PIX\_FMT\_CNTRSZ = 1, PIX\_FMT\_ALIGN = 0
    - ii. Fully packed 8-bit: PIX\_FMT\_PW = 0, PIX\_FMT\_CNTRSZ = 0, PIX\_FMT\_ALIGN = 0
2. Pixel Input Matrix Configuration
  - a. VPAC\_MSC\_LSE\_SRC\_CFG\_j[15-12] KERN\_LN\_OFFSET defines the starting line offset number of the input pixel matrix (for 5-tap filter configuration, offset = 0, for 4-tap configuration, offset = 1)
  - b. VPAC\_MSC\_LSE\_SRC\_CFG\_j[11-8] KERN\_SZ\_HEIGHT defines the height of the input pixel matrix (for example, MSC = 5, VISS = 1, NF = 5)
  - c. VPAC\_MSC\_LSE\_SRC\_CFG\_j[21-19] KERN\_TPAD\_SZ, and VPAC\_MSC\_LSE\_SRC\_CFG\_j[18-16] KERN\_BPAD\_SZ defines the number of padding lines required at top and bottom of the frame (for example, for 5-tap filter configuration for MSC, both of these parameters are set to 2. For 4-tap filter configuration for MSC, TPAD\_SZ = 1 while BPAD\_SZ = 2).
3. Source Frame Size
  - a. VPAC\_MSC\_LSE\_SRC\_FRAME\_SIZE\_j[28-16] HEIGHT and VPAC\_MSC\_LSE\_SRC\_FRAME\_SIZE\_j[12-0] WIDTH – define the width and height (pixels/lines) of the source video frame.
4. Source SL2 Circular Buffer Configuration
  - a. VPAC\_MSC\_LSE\_SRC\_BUF\_ATTR\_j[15-6] BUF\_STRIDE – defines the line stride size (must be 64 byte multiple)
  - b. VPAC\_MSC\_LSE\_SRC\_BUF\_ATTR\_j[24-16] CBUF\_SIZE – define the size of the circular buffer in the SL2 (number of lines)
  - c. VPAC\_MSC\_LSE\_SRC\_BUF\_ATTR\_j[31-25] START\_NIB\_OFFSET – defines the line start offset in smaller resolution (start address within the first SL2 512-bit word in 4-bit resolution).
5. Source BA and Enable Configuration
  - a. VPAC\_MSC\_LSE\_SRC\_i\_BUF\_BA\_y[23-6] ADDR – defines the SL2 base address of the Circular buffer for the input channel. VPAC\_MSC\_LSE\_SRC\_i\_BUF\_BA\_y[31] ENABLE – enables the input channel.

##### 6.9.6.7.1.3.2 Output Channel Configuration

1. Pixel Data Format
  - a. VPAC\_MSC\_LSE\_DST\_BUF\_CFG\_j[1-0] PIX\_FMT\_PW, VPAC\_MSC\_LSE\_DST\_BUF\_CFG\_j[3-2] PIX\_FMT\_CNTRSZ, and VPAC\_MSC\_LSE\_DST\_BUF\_CFG\_j[4] PIX\_FMT\_ALIGN parameters define the pixel output data format for this output channel.
  - b. Common data formats:
    - i. Fully packed 12-bit: PIX\_FMT\_PW = 1, PIX\_FMT\_CNTRSZ = 1, PIX\_FMT\_ALIGN = 0
    - ii. Fully packed 8-bit: PIX\_FMT\_PW = 0, PIX\_FMT\_CNTRSZ = 0, PIX\_FMT\_ALIGN = 0

2. Output Thread Map
  - a. VPAC\_MSC\_LSE\_DST\_BUF\_CFG\_j[7] THREAD\_MAP defines the output channel thread mapping.
3. Output SL2 Circular Buffer Configuration
  - a. VPAC\_MSC\_LSE\_DST\_BUF\_ATTR0\_j[15-6] BUF\_STRIDE – define the line stride size (must be 64 byte multiple)
  - b. VPAC\_MSC\_LSE\_DST\_BUF\_ATTR0\_j[24-16] CBUF\_SIZE – define the size of the circular buffer in the SL2 (number of lines)
4. Base Address and Channel Enable:
  - a. VPAC\_MSC\_LSE\_DST\_BUF\_BA\_j[23-6] ADDR – defines the SL2 base address of the Circular buffer for this output channel.
  - b. VPAC\_MSC\_LSE\_DST\_BUF\_BA\_j[31] ENABLE – enables this output channel.

#### 6.9.6.7.1.4 MSC HTS Programming Details

There is no configuration required for HTS interface in the MSC\_HWA. All programming is done in the separate HTS module. It is expected that at least one input and output channel is enabled for each HTS thread. For more details on programming HTS, see *Hardware Accelerator (HWA) Thread Scheduler (HTS)*.

#### 6.9.6.7.1.5 MSC Data Transfer Programming Details

Refer to *VPAC Subsystem Programmer's Guide*.

#### 6.9.6.7.1.6 LSE Interrupt Programming

MSC does not have any interrupt MASK/SET/CLEAR registers. Interrupts are pulse output events that go to the VPAC top level interrupt aggregation logic. For more information about the aggregation logic, see [Section 6.9.3](#), *VPAC Subsystem*.

MSC does, however, provide a set of statuses for VBUSM interface errors. These status are mapped to VPAC\_MSC\_STATUS\_ERROR and they are cleared by writing all 1's to the status bit field.

#### 6.9.6.7.2 Initialization Sequence

For details, see [Section 6.9.3](#), *VPAC Subsystem Level*.

#### 6.9.6.7.3 Real-Time Operating Requirements

For details, see [Section 6.9.3](#), *VPAC Subsystem Level*.

#### 6.9.6.7.4 Power Up/Down Sequence

For details, see [Section 6.9.3](#), *VPAC Subsystem Level*.

### 6.9.7 VPAC Noise Filter (NF)

#### 6.9.7.1 NF Overview

The noise filter in VPAC is used for filtering out spatial noise without impacting edges in image.

##### 6.9.7.1.1 NF Supported Features

- Bilateral filtering:
  - Supports filter size up to 5×5
  - Supports true 2D Bilateral filtering
- Generic filtering:
  - Supports filter size up to 5×5 of programmable static weights
- LUT based Bilateral weights generation:
  - 8-bit for weight
- Supported formats:
  - YUV420, 12-bit
  - One plan (interleaved plane and non-interleaved)
- Performance: 1 cycle/pixel
- Line based Input and Output
- Memory-to-memory operation
- Support for ROI:
  - Both Input and Output can be handled via DMA

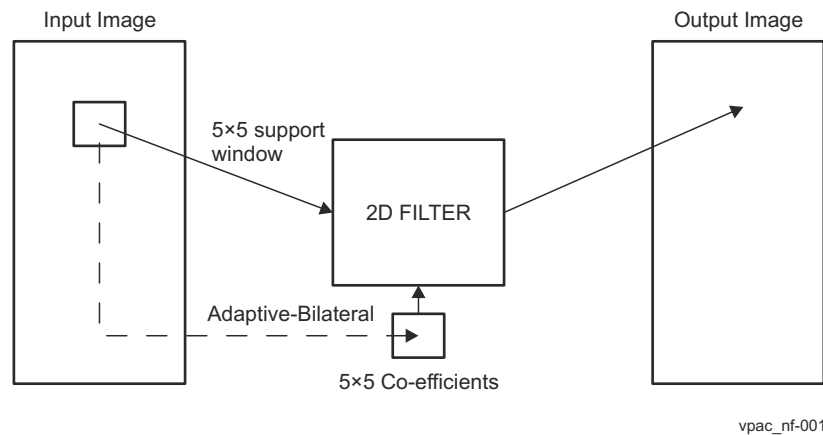
#### 6.9.7.2 NF Functional Description

##### 6.9.7.2.1 Functional Operation

The noise filter hardware (NF HW) block reads data from memory (DDR or on-chip) to shared memory (SL2 with the help of DMA) and does Bilateral filtering to remove noise. The output of NF HW block can be sent to external memory (DDR) from shared memory (SL2) or can be further re-sized using Scalar hardware.

##### 6.9.7.2.1.1 Overview

Figure 6-136 shows a block diagram of Bilateral noise filter with support for filtering size up to 5×5.



**Figure 6-136. Bilateral Noise Filter Block Diagram**

##### 6.9.7.2.1.2 Algorithm Details

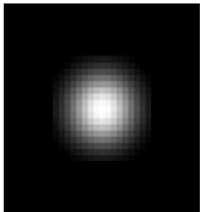

The basic equation for Bilateral filtering is shown in Figure 6-137.

$$BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(\|p - q\|) G_{\sigma_r}(\|I_p - I_q\|) I_q$$

new
not new
new

$\frac{1}{W_p}$ 
 $G_{\sigma_s}(\|p - q\|)$ 
 $G_{\sigma_r}(\|I_p - I_q\|)$

normalization factor
space weight
range weight






vpac\_nf-003

**Figure 6-137. Bilateral Filtering Equation**

The range and space parameters are shown in [Figure 6-138](#).

$$BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(\|p - q\|) G_{\sigma_r}(\|I_p - I_q\|) I_q$$

vpac\_nf-004

**Figure 6-138. Range and Space Parameters**

Range and space parameters:

- space ss: spatial extent of the kernel, size of the considered neighborhood.
- range sr: "minimum" amplitude of an edge.

### 6.9.7.3 NF Interrupts

#### 6.9.7.3.1 CPU Interrupts

The following interrupts are generated by the NF module (see [Table 6-153](#)):

**Table 6-153. Interrupts**

Interrupt	Description
NF_FRAME_DONE	Frame processing complete.
NF_SL2_READ_ERROR	Set whenever there is an error response on VBUSM read command request for any input channel.
NF_SL2_WRITE_ERROR	Set whenever there is an error response on VBUSM write command request for any output channel.

All interrupts are single pulse event signals that are mapped to the VPAC level interrupt aggregation logic.



### 6.9.7.3.2 Interrupt Event Description

#### 6.9.7.3.2.1 NF\_FRAME\_DONE Event

These events are generated when noise filtering is complete for entire frame. This is set when input end of frame (EOF) event and output EOF events are detected.

#### 6.9.7.3.2.2 NF\_SL2\_READ\_ERROR Event

This event is generated whenever the VBUSM read status is something other than '0' (success).

The VPAC status register (VPAC\_NF\_LSE\_STATUS\_ERROR) stores the last non-zero rstatus value for user to see which type of error has occurred.

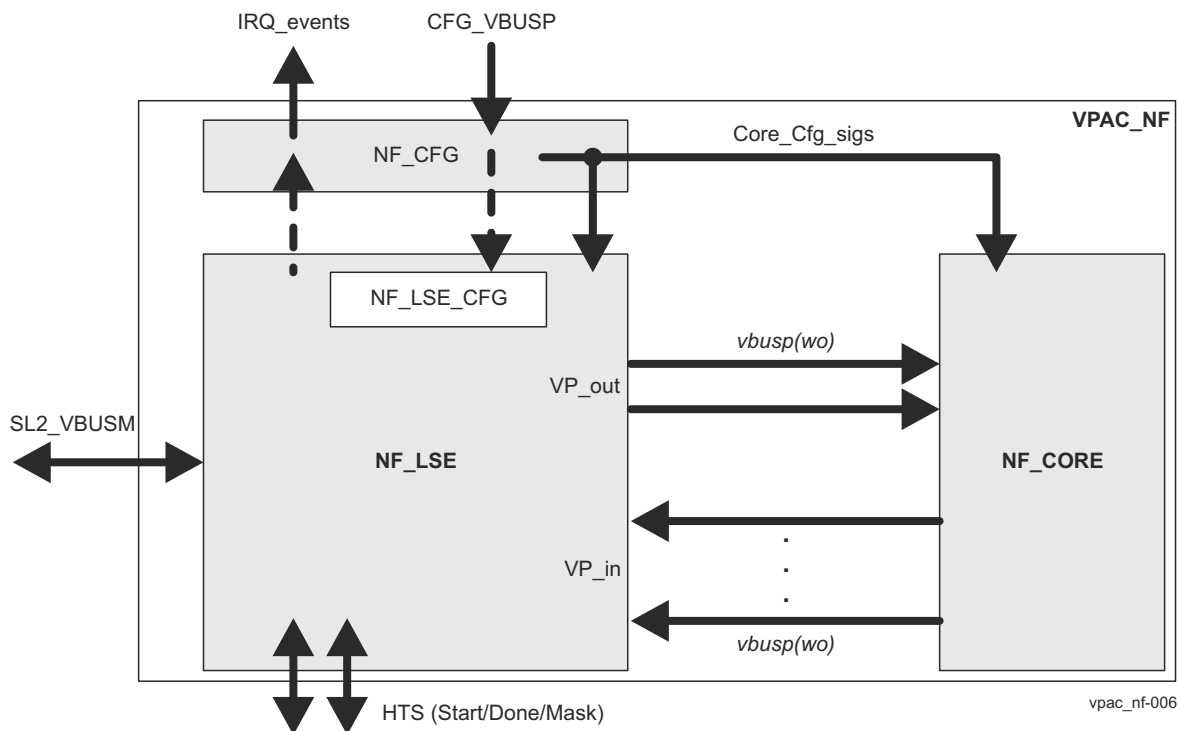
#### 6.9.7.3.2.3 NF\_SL2\_WRITE\_ERROR Event

This event is generated whenever the VBUSM write status is something other than '0' (success).

The VPAC status register (VPAC\_NF\_LSE\_STATUS\_ERROR) stores the last non-zero sstatus value for user to see which type of error has occurred.

### 6.9.7.4 NF Submodule Details

Following the common hardware partition strategy of VPAC HWA, the Noise Filter is partitioned as shown in Figure 6-139.



**Figure 6-139. NF Hardware Partition**

The NF\_CFG manages all configuration registers for the NF (top and core) via a common VBUSB slave interface.

The NF\_LSE (Load Store Engine) handles all interfacing to the SL2 memory space and performs the following functions:

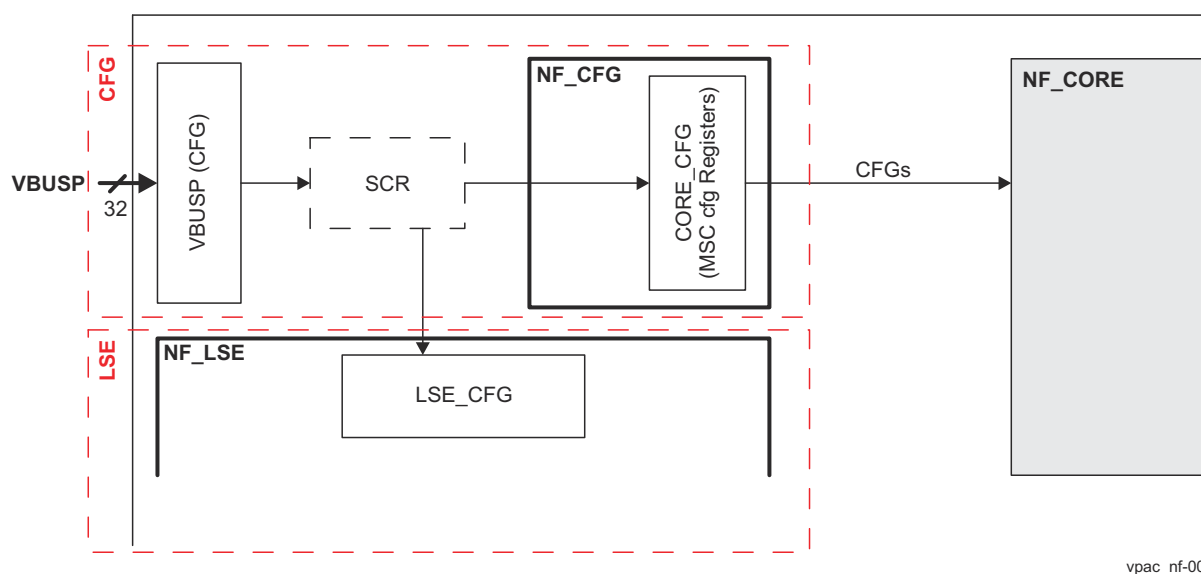
- Provides interface to SL2 circular buffers via a common VBUSM master interface.
- Manages input and output channel updates in sync with Hardware Thread Scheduler (HTS).
- Provides data unpacking for core and packing for SL2 interface.
- Manages LSE specific configuration registers.

The NF\_CORE performs noise filtering operations on given input and produces output. It has no direct connection to external interface and it works on a frame data coming in and out in raster scan order.

#### 6.9.7.4.1 NF\_CFG

The NF configuration registers consist of LSE configuration registers (see *VPAC\_NF\_LSE Registers*) and CORE configuration registers (see *VPAC\_NF\_CORE Registers*).

Figure 6-140 shows NF\_CFG block diagram.



**Figure 6-140. NF\_CFG Block Diagram**

#### 6.9.7.4.2 NF\_LSE

The NF integrates a VPAC\_LSE (Load Store Engine) .

##### 6.9.7.4.2.1 NF\_LSE Overview

The NF\_LSE supports the following VPAC\_LSE features:

- Input channel features:
  - One SL2 input channel with following features for the input channel:
    - Up to 5 lines - input kernel height support
    - Any source pixel start position support with Line Start Offset (in nibble-address resolution)
    - Input Line Skip support
    - Vertical boundary Edge Padding (Replication only) support
- Output Channel Features:
  - Only one SL2 output channels (pixel data output only)
  - Single data plane support for each output channel with no special chroma data interleaving support on the output
- Hardware Thread Scheduler (HTS) synchronization support
  - Start/Done task synchronization signal handling
  - Programmable Done-mask generation schemes
  - End of processing (EOP) generation

##### 6.9.7.4.2.2 NF\_LSE Feature Detailed Description

For detailed LSE feature description, refer to *Load Store Engine (LSE) Overview*.

#### 6.9.7.4.3 Synchronization With HTS

Scheduling of data transfer in the SL2 interface is controlled by the HTS controller integrated at the VPAC top level. A dedicated HTS is integrated per processing thread to manage start and done flow control signals and to issue DMA service events to trigger DDR from/to SL2 transfers and SL2 from/to HWA.

The following flow control scheme is used:

/\* initialization \*/

1. @ Frame Start up: (HTS) INIT → Frame Initialization

/\* First two line processing \*/

---

#### Note

'N' is input kernel height.

---

1. @ N-2 input lines available and output buffer line available: (HTS) START → Starts filter Line processing (initial line) tasks
2. @ Input Line Consumed and all output Done: (PixCntlFSM) DONE → Indicates completion of the line processing task
3. @ N-1 input lines available and output buffer line available: (HTS) START → Starts filter Line processing (initial line) tasks
4. @ Input Line Consumed and output Done: (PixCntlFSM) DONE → Indicates completion of the line processing task

/\* Middle Lines processing \*/

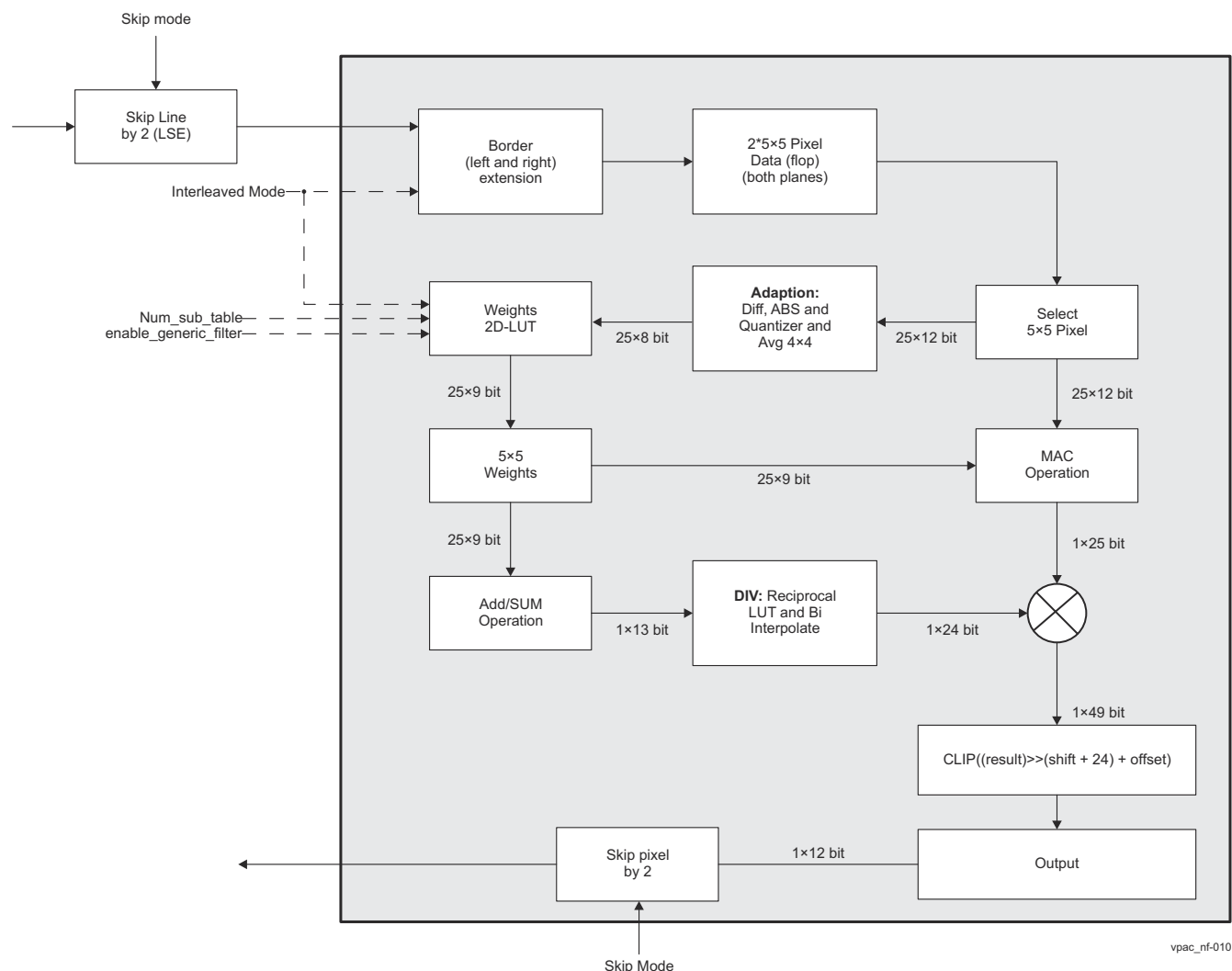
1. @ N-1 input lines available and output buffer line available: (HTS) START → Starts filter Line processing (initial line) tasks
2. @ Input Line Consumed and output Done: (PixCntlFSM) DONE → Indicates completion of the line processing task
3. Repeat (6) and (7) until Frame\_H-3 lines

/\* Last two line processing \*/

1. @ N-1 input lines available and output buffer line available: (HTS) START → Starts filter Line processing (initial line) tasks
2. @ Input Line Consumed and output Done: (PixCntlFSM) DONE → Indicates completion of the line processing task
3. @ N-2 input lines available and output buffer line available: (HTS) START → Starts filter Line processing (initial line) tasks
4. @ Input Line Consumed and all output Done: (PixCntlFSM) DONE → Indicates completion of the line processing task
5. @EOP: (PixCntlFSM) EOP

#### 6.9.7.4.4 Noise Filter Core Block Diagram

Figure 6-141 shows the core of the Noise filter. The line buffers are stored in SL2.



**Figure 6-141. NF-CORE Block Diagram**

There are two modes of filtering:

1. Bilateral filtering mode: The overall weights are computed based on center pixels. These are 8-bit unsigned weights. The center weight is read from MMR as 8-bit unsigned. The 9th bit for all weight is hardwired to '0'. The MAC operation is signed multiply with addition. As MSB for weight is set '0', it will be act as unsigned multiplier of 12-bit pixel to 8-bit unsigned weight.
2. Generic filtering mode: In this mode, all weights are 9-bit signed value read LUT from initial 24 values read as 9-bit signed value from every 2 bytes (16 bits) locations.

#### 6.9.7.4.4.1 Space Weight Details

Figure 6-142 shows spatial distance from central pixel to decide spatial weight.

4	3	2	3	4
3	1	0	1	3
2	0		0	2
3	1	0	1	3
4	3	2	3	4

vpac\_nf-011

**Figure 6-142. Space Distance For 5×5 Pixels**

- Total number pixels = 25
  1. Number of pixel at distance 0 = 4
  2. Number of pixel at distance 1 = 4
  3. Number of pixel at distance 2 = 4
  4. Number of pixel at distance 3 = 8
  5. Number of pixel at distance 4 = 4
- Space range (five, 0-4)
- Please note that Center pixel has 9-bit programmable MMR register for fixed weight


#### 6.9.7.4.4.2 Weight Calculation Logic

##### 6.9.7.4.4.2.1 Combined LUT For Space And Range Weights


In this approach, Bilateral weight is computed for each center pixel based on the input (guide) image:

- $i$ : center pixel index,  $j$ : neighbor pixel index
- $I_i$ : center pixel input image intensity,  $I_j$ : neighbor pixel input image intensity

$$W_{i,j}^{bf} = \frac{1}{K_i} \exp\left(-\frac{|i-j|^2}{\sigma_s^2}\right) \exp\left(-\frac{|I_i - I_j|^2}{\sigma_c^2}\right),$$



**Division LUT**



**Combined LUT**

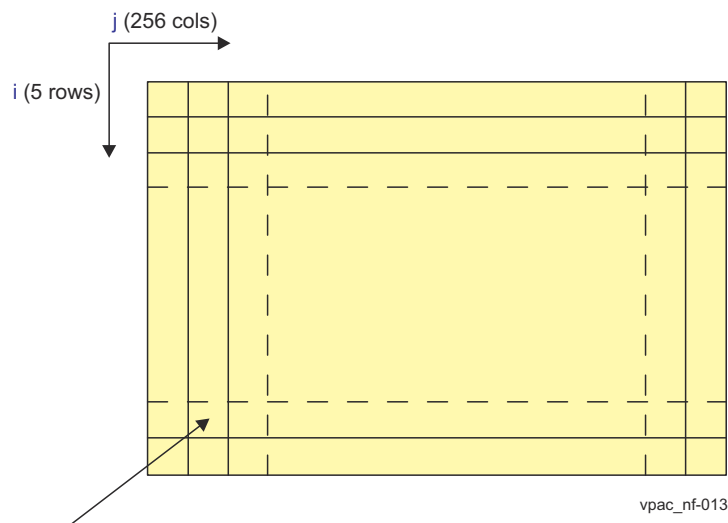
vpac\_nf-012

**Figure 6-143. LUT For Combined Weight For Space And Range**

The following are details of this approach:

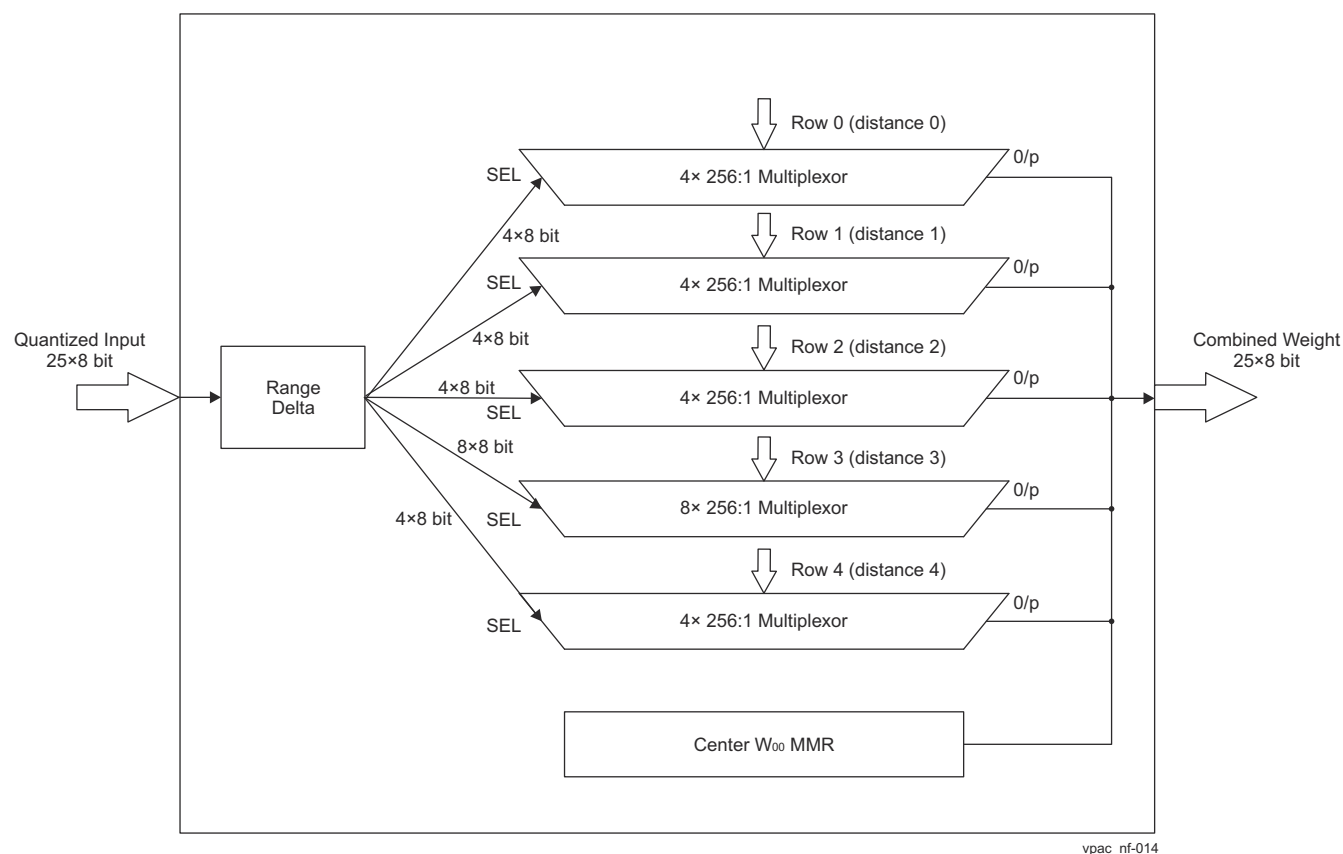
- $(i-j)$  is 3-bit for 5×5: use 3 bits
- $(I_i - I_j)$  is 13-bits: quantize to 8 bits
- Pre-compute  $W_{i,j}$  as 8-bit values including normalization and put in a 2D lookup table
- $LUT[i][j]$ :  $i$  is 3-bit pix index,  $j$  is 8-bit image intensity diff
- Lookup value is 8-bit
- Total storage:  $5 \times 256 \times 8b = 1280$  Bytes
- Can be reduced further based on symmetric nature of range
- For  $1/K_i$  (refer to [Section 6.9.7.4.4.3](#))

The LUT organization is shown in [Figure 6-144](#).



**Figure 6-144. LUT Organization For Combined Weights**

Figure 6-145 shows details of weight LUT logic for this approach.



**Figure 6-145. Weight LUT Logic**

#### 6.9.7.4.4.3 Reciprocal Calculation Logic

1/Ki calculation is done using Reciprocal table.

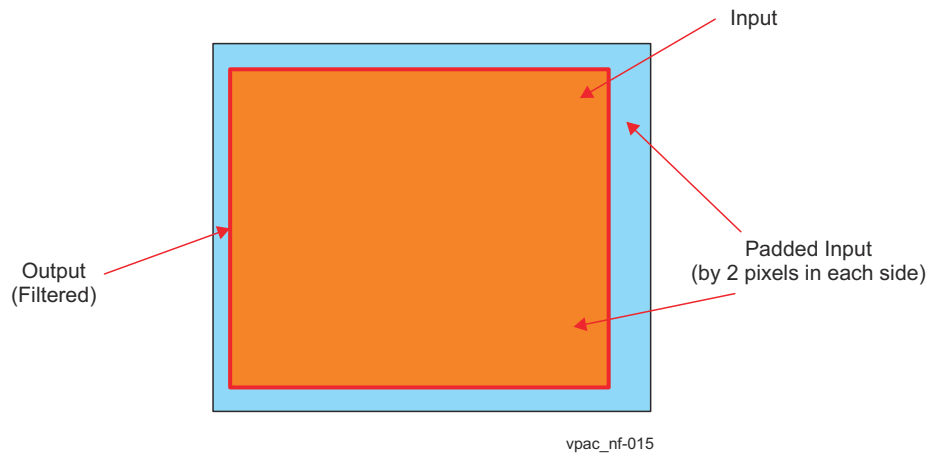
The following are details of Reciprocal table:

- Table is located in ROM (all values are hardwired)
- 24 bits per entry
- The first 512 entries are not quantized
- The remaining entries are quantized by 16 with Bilinear interpolation during lookup two neighbouring values
- Total of 993 entries

#### 6.9.7.4.4.4 Border Handling

##### 6.9.7.4.4.4.1 Border Handling (Simple)

The noise filter will support boarder extension. It will process two pixel copy in each directions for padding. The output frame size will be equal to input frame size as shown in [Figure 6-146](#). The output there will be border pixels (two in each direction) will be unfiltered, while rest of middle frame is noise filtered. Please note that the output frame size (or resolution) will be identical to input frame (resolution).

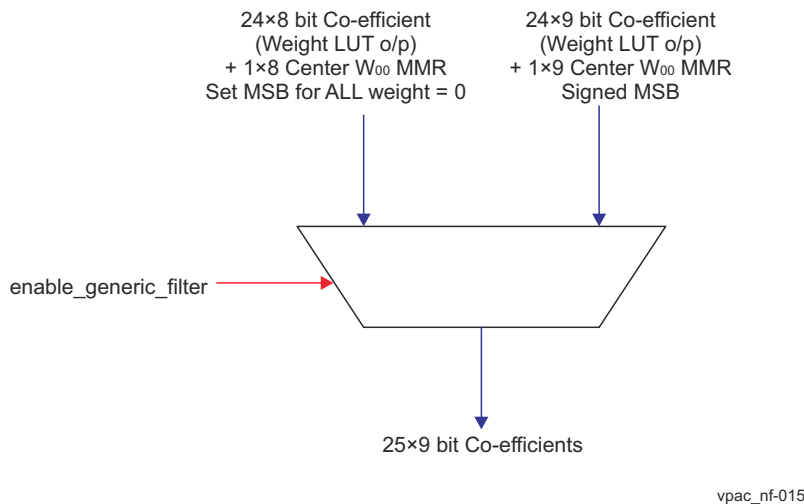


**Figure 6-146. Border Handling**

#### 6.9.7.4.5 Usage As Generic 2D Filter Engine

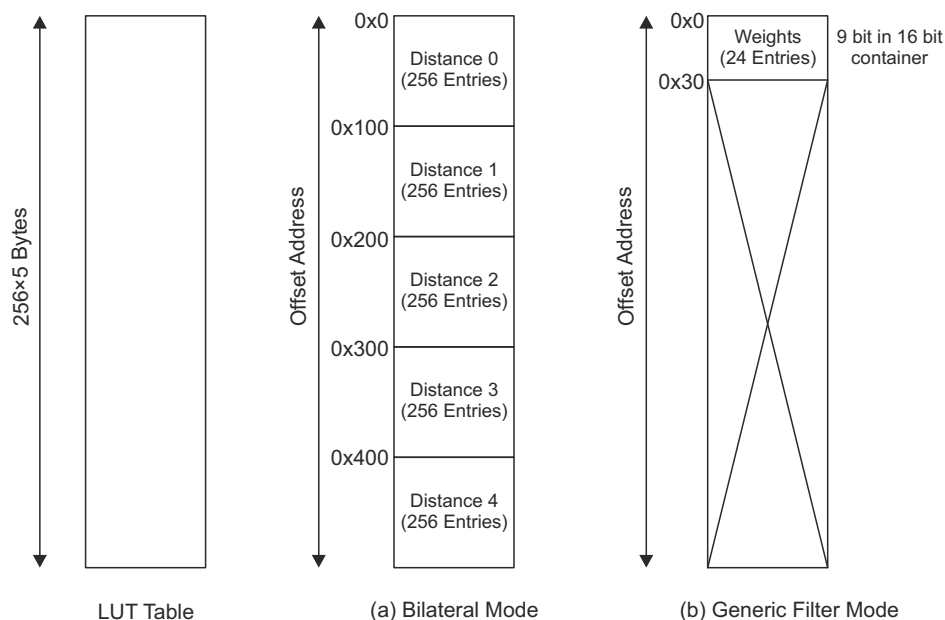
The given noise filter can be used as generic 2D 5×5 filter core. This mode is enabled by selecting `ENABLE_GENERIC_FILTER` in `CNTR` MMR. In this case, software needs to set necessary MMR to provide co-efficients for 2D convolution. The overall decision for 5×5 co-efficient is shown in [Figure 6-147](#).

Note the order of the weight is Column first then Row second `C0_R0`, `C1_R0`, ... , `C4_R5`, `C5_R5`.



**Figure 6-147. 5×5 Weight (Generic 2D Filtering)**

Figure 6-148 shows the usage of Weight or Range LUT for Generic Filtering Mode.



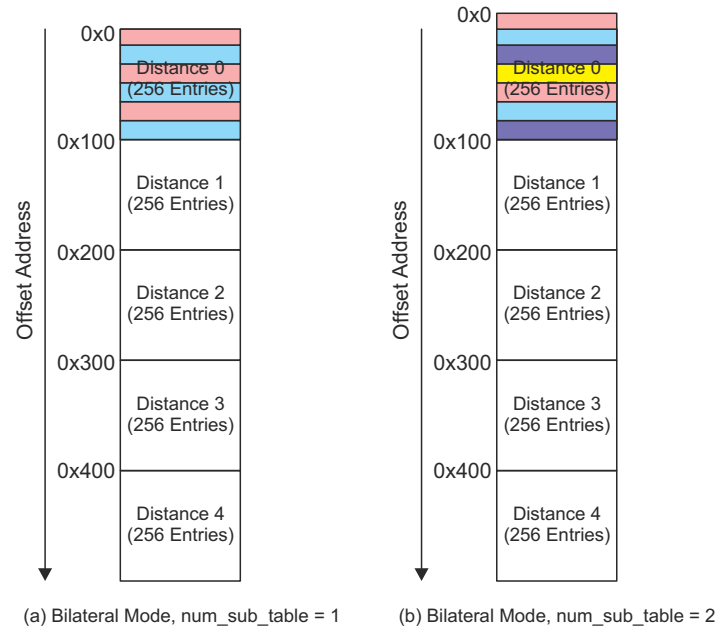
vpac\_nf-017

**Figure 6-148. Weight LUT Usage In Generic Mode**

#### 6.9.7.4.6 Adaptive Bilateral Weight Support

The hardware supports adaptive Bilateral filtering using `num_sub_tables > 0`. The `num_sub_table = 1` means 2 tables withing LUT per distance, `num_sub_table = 2` means 4 sub\_table within LUT per distance and `num_sub_table = 3` means 8 sub\_tables per distance. The tables are stored as interlaved within each distance. For example in case `num_sub_table = 1`, there are two LUT within each distance with even entries coming from LUT1 and odd entries coming from table 2 as shown in Figure 6-149.



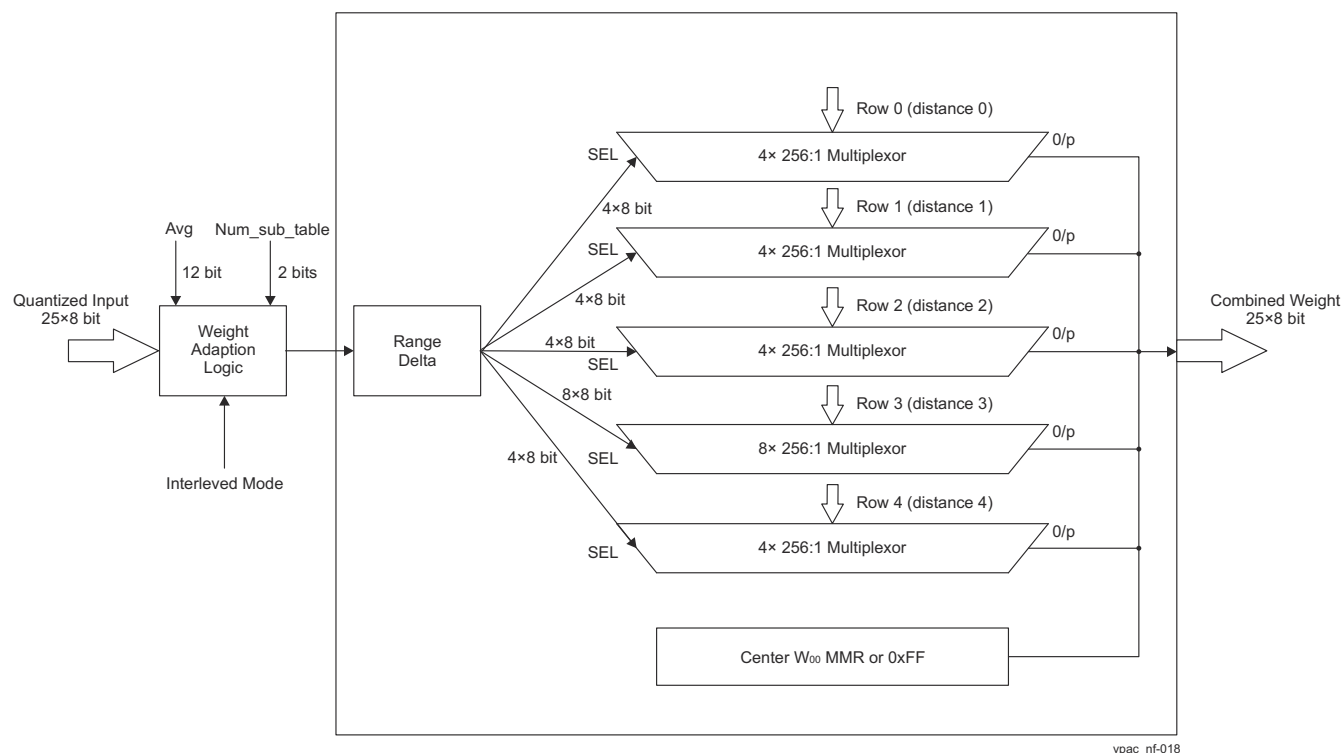


vpac\_nf-018

**Figure 6-149. Interleaved Entries In Table**

The adaptation logic consist using 4×4 average of 5×5 window (top left corner) and `num_sub_tables` from MMR as shown below using combo logic below:

- Mask "`num_sub_tables`" LSB bits out of 8 bits.
- $\text{LSB Bits} = \text{Average\_4x4} \gg (12 - \text{num\_sub\_tables})$
- 8 bits for LUT look up =  $(8 - \text{num\_sub\_table}) \ll \text{num\_sub\_table} + \text{LSB Bits}$
- Please note that this is constant LSB bits for all 25 pixels that will be replaced for all 25×8 pixels.



**Figure 6-150. Adaptive Bilateral Filtering**

#### 6.9.7.4.7 Chroma Handling (Interleaved Mode)

Please note that the given hardware module can support either Luma (Non-Interleaved mode) OR Chroma mode (Interleaved mode) at a time. It can't support two data plane (Luma + Chroma) simultaneously.

When the input is in the interleaved data format (for example: YUV420 Chroma data), the filter core keeps two sets of 5-input samples (Cb and Cr buffers) and alternates filtering operation between two. In this case, core does deinterleaving and maintains two set of data of 6x5 for both planes as shown in block diagram of core. The parameters for Chroma needs to be programmed at every frame differently compared to Luma.

Note software can always do a "force" select which sub table to use by clearing `adaptive_mode` then `sub_table_select` defines which table to use, otherwise below defines which sub table is selected.

- Mask "num\_sub\_tables" LSB bits out of 8 bits.
- $\text{LSB Bits} = \text{Average\_4x4} \gg (12 - \text{num\_sub\_tables})$
- If (`num_sub_tables != 1`):
  - 8 bits for LUT look up =  $(8 - \text{num\_sub\_table}) \ll \text{num\_sub\_table} + \text{LSB Bits}$

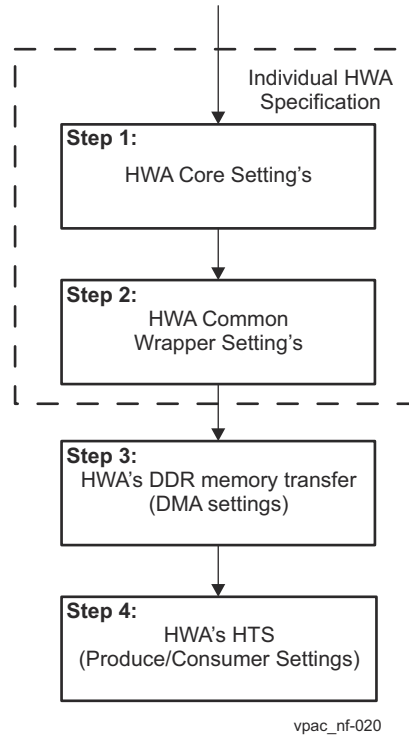
Else:

- 8 bits for LUT look up =  $(8 - \text{inputPlaneInterleaving}) \ll \text{inputPlaneInterleaving} + \text{LSB Bits}$
- Please note that this is constant LSB bits for all 25 pixels that will be replaced for all 25x8 pixels.

#### 6.9.7.5 NF Programmer's Guide

##### 6.9.7.5.1 Programming Model

The full programming of the module, consist of the following four steps (see [Figure 6-151](#)):



**Figure 6-151. Programming Steps**

#### 6.9.7.5.1.1 HWA Core Programming Details

There are two use-cases:

- Bilateral Noise Filter Mode: The following core MMR needs to be programmed to enable Bilateral noise filter mode:
  - Set Weight LUT (8-bit unsigned) tables (5×256 values corresponding to various distance from distance in MMR (0x100 onwards))
  - Set Control register fields (Center weight to maximum - 0×255, offset, shift, ...)
  - Set generic\_filter\_mode to '0'
- Generic 2D Convolution (filtering) Mode: The following core MMR needs to be programmed to enable Bilateral noise filter mode:
  - Set Weight LUT (9 bit unsigned) tables (24 values (each 9-bit signed) in 16 bit container from start of LUT
  - Set Control register fields (Center weight, offset, shift, ...)
  - Set generic\_filter\_mode to '1'

#### 6.9.7.5.1.2 NF SL2 Wrapper Interface Programming Details

The following MMR needs to be programmed as per overall Data flow as well as algorithm needs:

- Set Input (SRC) side SL2 configuration, buffer base address, attributes and sizes
- Set Output (DST) side SL2 configuration, buffer base address, attributes and sizes

#### 6.9.7.5.1.3 HWA HTS Programming Details

For more details of programming, refer to HTS and Section VPAC Subsystem.

Although following are Hardcoded configuration for given HWA (Noise filter (NF)) for HTS:

- Consumer score board: Transaction aggregation
- Number of lines of delay = 5
- ...

**6.9.7.5.1.4 HWA Data Transfer Programming Details**

For more details, refer to the VPAC Top/UTC specification.

## 6.10 Depth and Motion Perception Accelerator (DMPAC)

This chapter describes the Depth and Motion Perception Accelerator (DMPAC) in the device.

### 6.10.1 DMPAC Overview

The Depth and Motion Perception Accelerator (DMPAC) is a power efficient hardware accelerator that computes dense stereo depth maps (*depth*) and dense optical flow vectors (*motion*) from camera inputs.

The image/video sensor-based environmental perception (also known as scene understanding) is at the core of many emerging applications in automotive, industrial and consumer electronics. Typically, this involves detection of all objects in the scene along with their 3D position and motion with regards to the observer or the car by analyzing one or many related input video streams. Various computer vision algorithms are used to achieve these tasks.

A very robust method of obtaining the 3D depth from images is to use two cameras in a stereo setup - two cameras with known relative positions and camera parameters. The two images of the same scene, captured from two different camera poses/perspectives, are analyzed to find disparities among every pixel positions in the images. This is known as the Stereo Disparity map. The disparity values of every pixel can be used to obtain the 3D positions of the object/space they belong to via triangulation.

On the other hand, by analyzing two images from a single camera, captured at two different time instances (that is, two temporal frames in a video), one can determine where each pixel in a past frame moved to in the future frame. This is known as the Optical Flow vector. The flow vectors for each pixel position can be used to obtain 3D structure of the scene, identify moving objects and determine their relative speed and direction of motion.

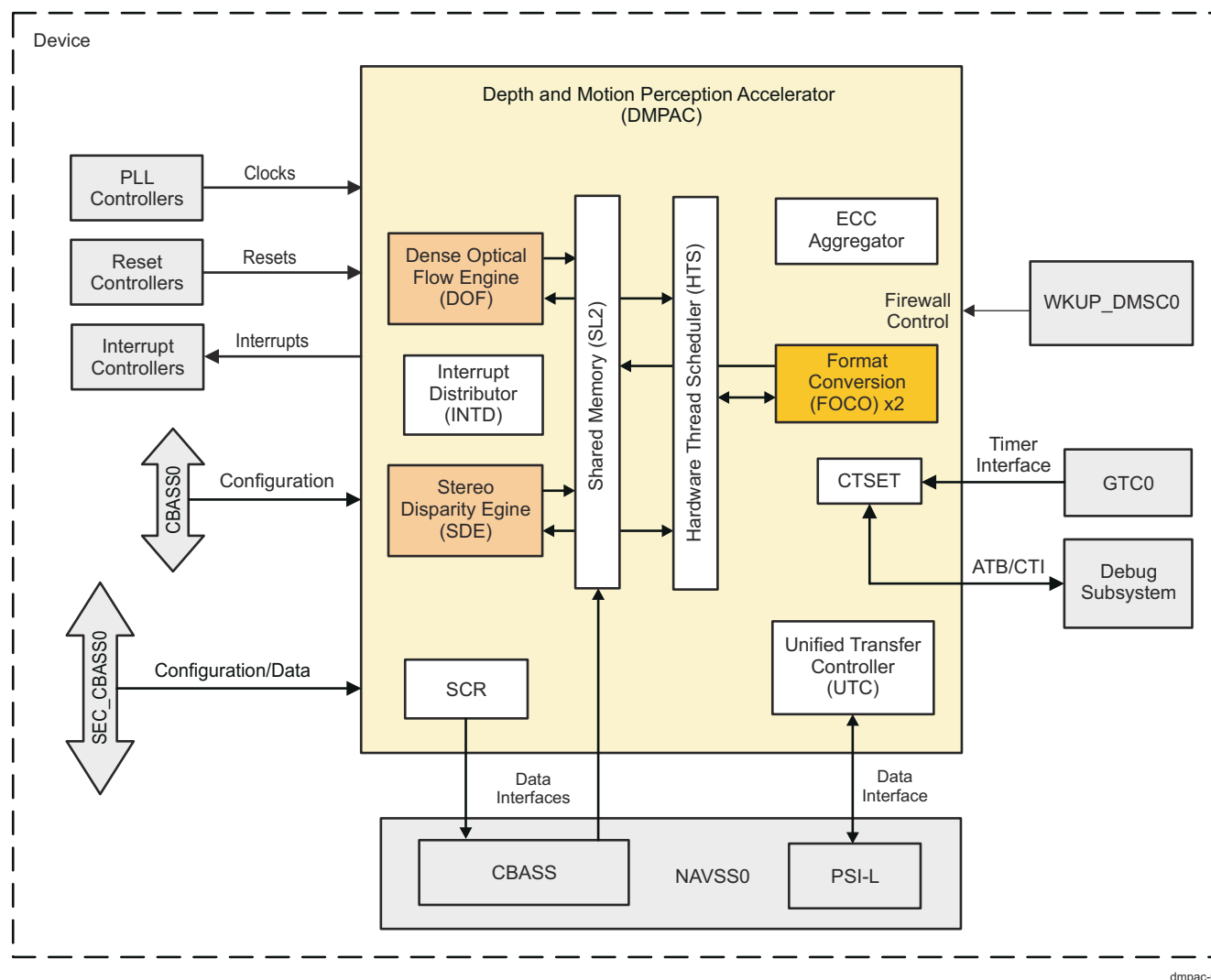
The DMPAC is dedicated to the aforesaid image processing tasks. The stereo and optical flow processing is partitioned into two top level sub-blocks: the Dense Optical Flow (DOF) engine and the Stereo Disparity Engine (SDE). The DOF and SDE blocks share a common shared local memory, DMA, external messaging and control infrastructure.

The device includes a single instantiations of the DMPAC. [Table 6-154](#) shows the DMPAC allocation across device domains.

**Table 6-154. DMPAC Allocation Across Device Domains**

Instance	Domain		
	WKUP	MCU	MAIN
DMPAC0	-	-	✓

[Figure 6-152](#) provides an overview of the DMPAC.



dmpac-001

**Figure 6-152. DMPAC Overview**

### 6.10.1.1 DMPAC Features

The DMPAC supports the following main features:

- Input image resolution up to 2 MPix with maximum horizontal resolution of up to 2048 pixels and maximum vertical resolution of up to 1024 pixels
- Input pixel format of 12-bit fully packed luminance data, and other formats using the integrated Format Conversion (FOCO) module
- Pixel level output packed in 16bpp and 32bpp formats for stereo disparity estimates and optical flow estimates, respectively
- Simultaneous operation of SDE and DOF, each processing inputs of up to 1 MPix image resolution (1 MPix = 1280 × 720 pixels)
- Resolution vs. frame rate scalability (higher frame rate at lower resolution)
- A Dense Optical Flow (DOF) engine: The DOF engine implements Texas Instruments' proprietary algorithm for estimation/calculation of the optical flow between the input image pair. The algorithm uses hierarchical coarse-to-fine block search strategy leveraging image pyramids in which proprietary binary pixel descriptors are used for pixel correspondence determination. In the scheme accuracy and smoothness of the flow map is enforced using optical flow estimates of the causal neighbors of a pixel in the given and higher (lower resolution) pyramid level (pixels are processed in raster scan order in a pyramid level to refine existing optical flow estimates). The DOF engine supports:

- Optical flow vector lengths up to +/- 191 pixels in horizontal direction and +/- 62 pixels in vertical direction for each input pixel
- Dense flow vector map for each input pixel
- Optical flow vector precision of 1/16<sup>th</sup> of inter pixel distance
- Post filtering of the optical flow estimates using 2D median filtering
- 16 level confidence score for every output flow vector
- A Stereo Disparity Engine (SDE): The SDE implements Texas Instruments' proprietary algorithm in order to find the disparity map between the input image pair. The SDE supports:
  - Disparity search range (SR) of 64/128/192. It can support either "0 to SR-1" or "-3 to SR-4"
  - 1/16<sup>th</sup> pixel accurate sub-pixel level disparity estimate
  - Disparity estimate post processing using 2D median filtering
  - 8 level confidence score for each disparity output
- Common top level infrastructure:
  - Shared Level 2 (SL2) memory sub-system, which serves both DOF and SDE blocks, along with the FOCO modules when required as an intermediate storage exchange data across sub-modules (FOCO to DOF/SDE) and from DDR/System Memory
  - A Unified Transfer Controller (UTC), which serves as a DMA engine
  - Messaging and control mechanism via a Hardware Thread Scheduler (HTS) block
  - A Counter, Timer and System Event Trace (CTSET) module, which provides event tracing capability for the SDE and DOF hardware threads

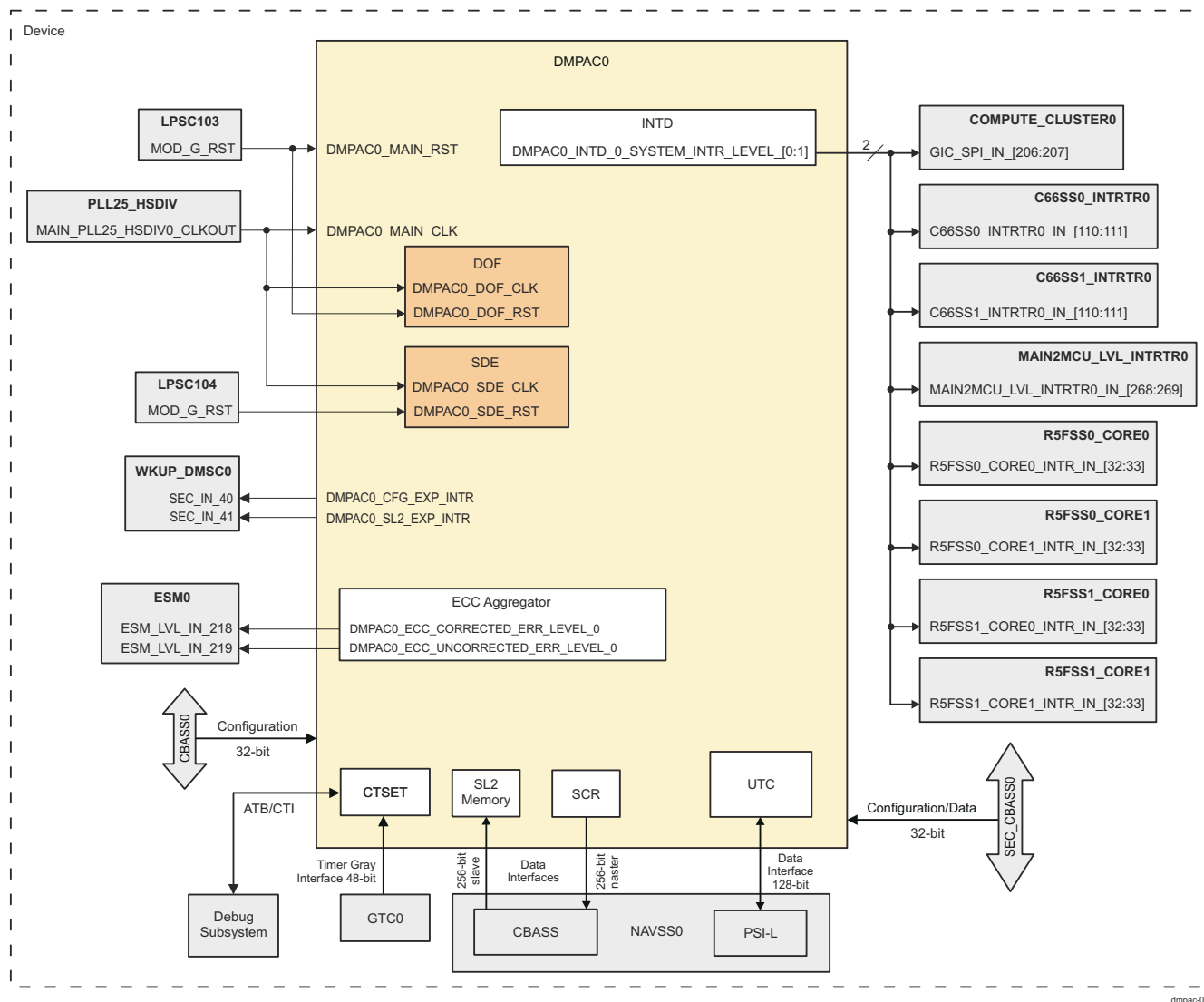
The DMPAC does not support (but relies on other HWAs/processors on the SoC to perform these tasks):

- Epipolar rectification of the stereo input images
- Image pyramid generation for the optical flow input images

### 6.10.2 DMPAC Integration

This section describes the DMPAC integration in the device MAIN domain, including information about clocks, resets, and hardware requests.

There is one DMPAC integrated in the device MAIN domain - DMPAC0. Figure 6-153 shows the integration of DMPAC0.



**Figure 6-153. DMPAC Integration**

Table 6-155 through Table 6-157 summarize the integration of DMPAC0 in the device MAIN domain.

**Table 6-155. DMPAC Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
DMPAC0 (TOP Level and DOF)			LPSC103	CBASS0
DMPAC0 (SDE)	PSC0	PD28	LPSC104	NAVSS0_CBASS NAVSS0_PSI_L SEC_CBASS0



**Table 6-156. DMPAC Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
DMPAC0	DMPAC0_MAIN_CLK	MAIN_PLL25_HSDIV0_CLKOUT	PLL25_HSDIV	DMPAC0 main functional clock.
	DMPAC0_SDE_CLK			DMPAC0 SDE functional clock.
	DMPAC0_DOF_CLK			DMPAC0 DOF functional clock.
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
DMPAC0	DMPAC0_MAIN_DOF_RST	MOD_G_RST	LPSC103	DMPAC0 main and DOF reset
	DMPAC0_SDE_RST	MOD_G_RST	LPSC104	DMPAC0 SDE reset

### Note

The DMPAC and VPAC clocks are derived from the same PLL and as a result, the following maximum clock frequency restrictions apply:

- When the VPAC is set to maximum clock frequency = 720 MHz, the DMPAC clock frequency is limited to 480 MHz.

**Table 6-157. DMPAC Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DMPAC0	DMPAC0_INTD_0_SYSTEM_INTR_LEVEL_0[0:1]	GIC_SPI_IN_206:207	COMPUTE_CLUSTER0	DMPAC0 interrupts	Level
		C66SS0_INTRTR0_IN_110 :111]	C66SS0_INTRTR0		Level
		C66SS1_INTRTR0_IN_110 :111]	C66SS0_INTRTR0		Level
		MAIN2MCU_LVL_INTRTR0_IN_268:269]	MAIN2MCU_LVL_INTRTR0		Level
		R5FSS0_CORE0_INTR_IN_32:33]	R5FSS0_CORE0		Level
		R5FSS0_CORE1_INTR_IN_32:33]	R5FSS0_CORE1		Level
		R5FSS1_CORE0_INTR_IN_32:33]	R5FSS1_CORE0		Level
		R5FSS1_CORE1_INTR_IN_32:33]	R5FSS1_CORE1		Level
	DMPAC0_ECC_CORRECTED_ERR_LEVEL_0	ESM_LVL_IN_218	ESM0	DMPAC0 ECC Aggregator interrupt for correctable error (SEC)	Level
	DMPAC0_ECC_UNCORRECTED_ERR_LEVEL_0	ESM_LVL_IN_219		DMPAC0 ECC Aggregator interrupt for non-correctable error (DED, parity, redundancy, timeout)	Level

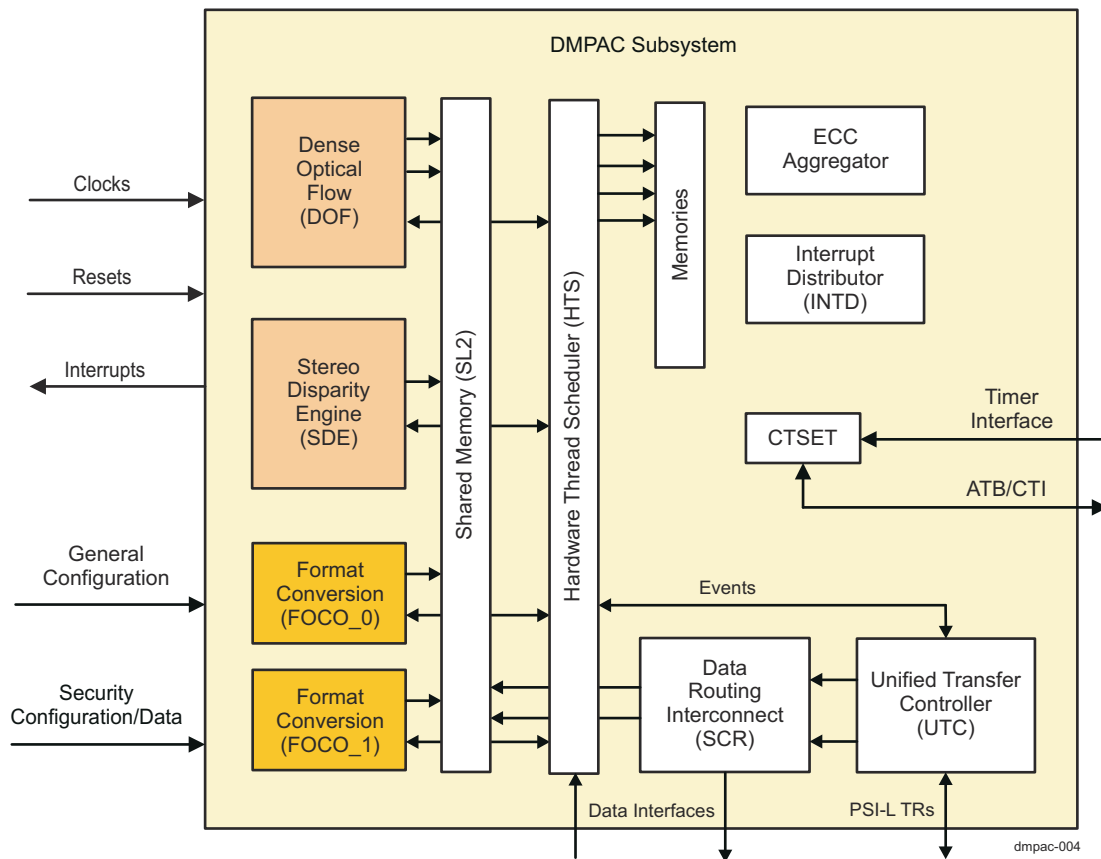
**Table 6-157. DMPAC Hardware Requests (continued)**

DMPAC0_CFG_EXP_INTR	SEC_IN_40	WKUP_DMSC0	DMPAC0 firewall exception pending interrupt for the internal configuration interconnect	N/A
DMPAC0_SL2_EXP_INTR	SEC_IN_41		DMPAC0 firewall exception pending interrupt for the SL2 interconnect	N/A

### 6.10.3 DMPAC Functional Description

#### 6.10.3.1 DMPAC Block Diagram

Figure 6-154 shows a simplified block diagram of the DMPAC subsystem. The Stereo and Optical flow processing is partitioned into two top level blocks (SDE and DOF) which share a common L2 memory subsystem, DMA, external messaging and control infrastructure.



**Figure 6-154. DMPAC Block Diagram**

#### 6.10.3.2 DMPAC Data Formats and Image Resolution

##### 6.10.3.2.1 Resolution and Frame Rate

The stereo and optical flow processing in DMPAC is designed to support scalability with regards to input image resolution and achievable frame rate.

This section contains examples for 520MHz operating clock. For lower clock frequencies, the frame rate should be scaled accordingly.

The input resolution and frame rate support for stereo processing is shown in Table 6-158.

**Table 6-158. DMPAC Input Resolution and Frame Rate Support for Stereo Processing**

Imager Configuration	Resolution			Mpix	Fps	Comments
	Width	Height	Restrictions			
2MP Imager	2048	1024	See (2)	2.10	40	Baseline performance
2MP Imager with cropped ROI <sup>(1)</sup>	2048	512		1.05	80	Example of achievable scalability.
1MP Imager with cropped ROI	1024	512		0.52	160	
	1280	640		0.82	102	

(1) ROI = Region of Interest

- (2)
- Width must be  $128 + 16 \cdot n$ , where  $0 \leq n \leq 120$
  - Height must be  $64 + 16 \cdot m$ , where  $0 \leq m \leq 60$

The input resolution and frame rate support for optical flow processing is shown in [Table 6-159](#).

**Table 6-159. DMPAC Input Resolution and Frame Rate Support for Optical Flow Processing**

Imager Configuration	Resolution			Mpix	Fps	Comments
	Width	Height	Restrictions			
2MP Imager	2048	1024	Width and Height must be multiple of 32	2.10	100	Baseline performance
2MP Imager with cropped ROI	2048	512		1.05	200	Example of achievable scalability.
1MP Imager with cropped ROI	1024	512		0.52	400	
	1280	640		0.82	256	

#### 6.10.3.2.2 Input Data Formats

The DMPAC supports both 8-bit and fully packed 12-bit luma data formats. The DOF and SDE blocks only accept fully packed 12-bit luma data format. When processing of 8-bit luma data input formats is required, the format conversion (FOCO) blocks need to be enabled, as they can convert 8-bit data into 12-bit fully packed format for both stereo and optical flow processing inside DMPAC sub-system. The fully packed 12-bit data organization goes on till the end of the line. Every line always starts at a 64-byte aligned boundary in SL2 memory. In the system DDR, it is advised that every line starts at a new DDR burst (for example, 128 bytes for 64-bit LPDDR4).

**Table 6-160. DMPAC Input Pixel Data Format**

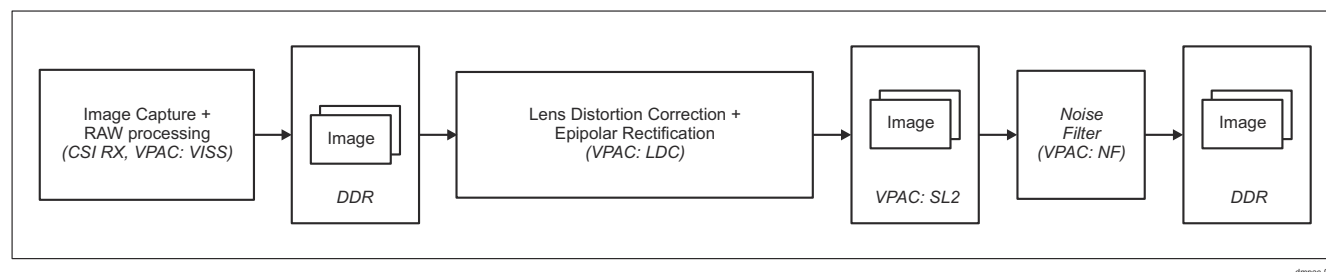
MSB																												LSB			
127																												0			
4b	4b	4b	4b	4b	4b	4b	4b	4b	4b	4b	4b	4b	4b	4b	4b	4b	4b	4b	4b	4b	4b	4b	4b	4b	4b	4b	4b	4b	4b		
Byte 15		Byte 14		Byte 13		Byte 12		Byte 11		Byte 10		Byte 9		Byte 8		Byte 7		Byte 6		Byte 5		Byte 4		Byte 3		Byte 2		Byte 1		Byte 0	
		Pixel 9		Pixel 8		Pixel 7		Pixel 6		Pixel 5		Pixel 4		Pixel 3		Pixel 2		Pixel 1		Pixel 0											

The image luminance data can be placed anywhere in the SoC memory system that is accessible by the DMA engine attached to the DMPAC.

#### 6.10.3.3 DMPAC Top Level Data Flow

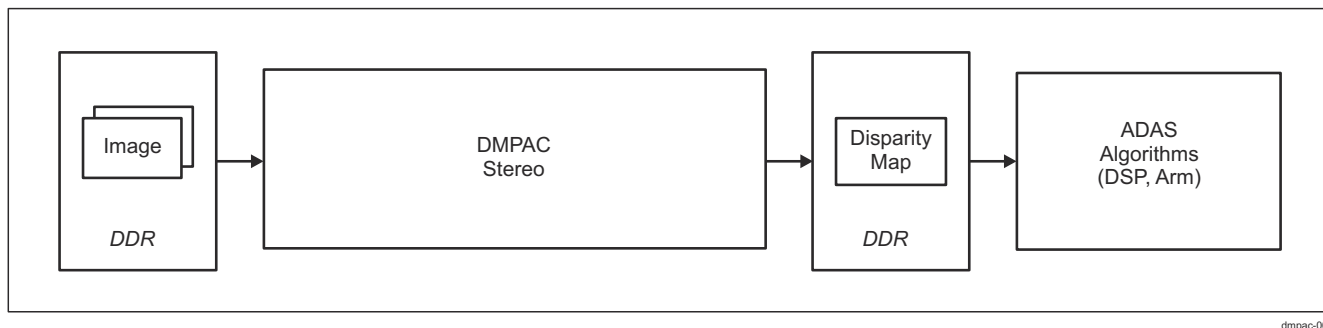
[Figure 6-155](#) through [Figure 6-158](#) illustrate an example of the SoC level dataflow for Stereo and Optical flow processing in a typical system implementation, where DMPAC has been integrated along with a Vision Imaging Sub-system (VISS) and a Vision Preprocessing Accelerator (VPAC).

The processing pipeline between these blocks can be synchronized at frame level or at a lower granularity of slice level (multiple full image rows) depending of SoC level control and data bandwidth capabilities.



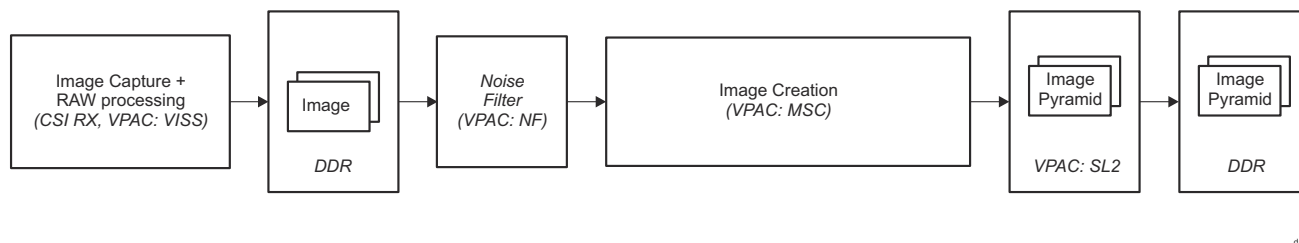
dmpac-007

**Figure 6-155. DMPAC SoC Level Dataflow for Stereo Pre-Processing**



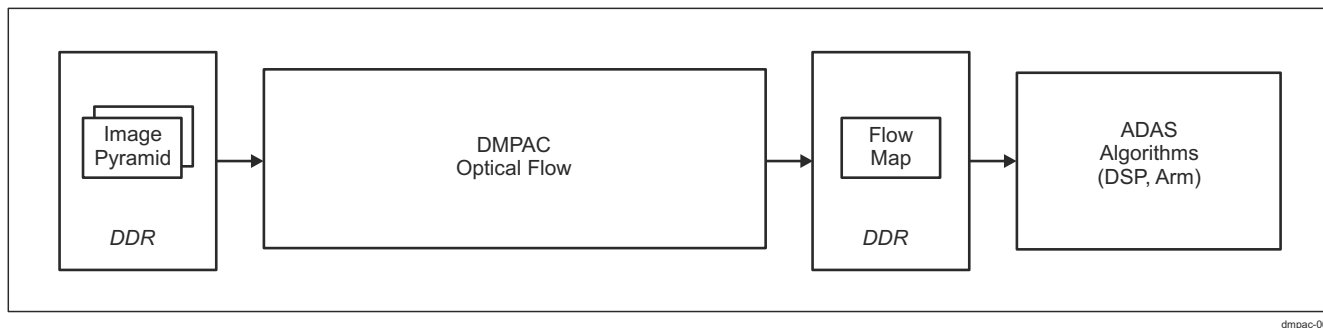
dmpac-008

**Figure 6-156. DMPAC SoC Level Dataflow for Stereo Processing**



dmpac-005

**Figure 6-157. DMPAC SoC Level Dataflow for Optical Flow Pre-Processing**



dmpac-006

**Figure 6-158. DMPAC SoC Level Dataflow for Optical Flow Processing**

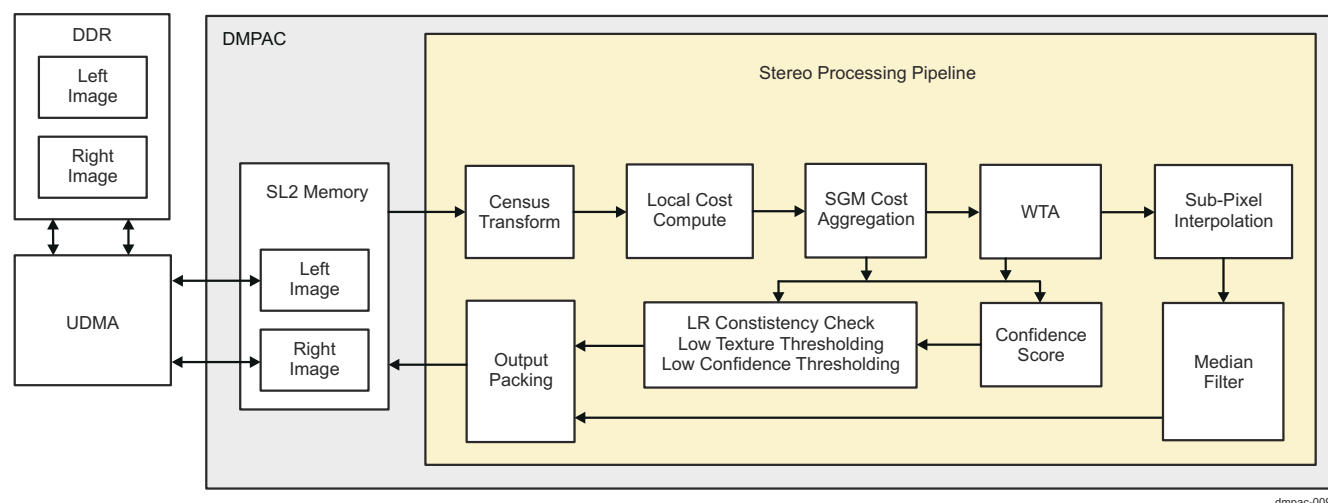
#### 6.10.3.4 DMPAC Stereo Functional Overview

The DMPAC implements Texas Instruments' proprietary algorithm in order to find the disparity map between the input image pair. The algorithm is based on Semi Global Matching method that is optimized and adapted to fit into real time, low power processing requirements without any significant loss of quality.

This section outlines the overall functionality of the Stereo Disparity Engine (SDE).

##### 6.10.3.4.1 Stereo Processing Dataflow

Figure 6-159 illustrates the functional view of dataflow for stereo processing within the DMPAC. The input image (with required image pre-processing including epipolar rectification done) is stored in external SDRAM (DDR). The DMPAC UTC fetches the input image pair luminance data in its level 2 memory sub-system and then processes through its pipeline. The final output from stereo module is written back to shared L2 memory which is transferred back to external SDRAM (DDR).



**Figure 6-159. DMPAC Functional View of Stereo Processing Dataflow**

#### 6.10.3.4.2 Disparity Range

The disparity ranges supported by the DMPAC SDE module are as follows:

- Disparity search range (SR) of 64/128/192. It can support either “0 to SR-1” or “-3 to SR-4”.
- Negative disparity -3 is supported for correcting camera calibration errors
- Disparity output is provided for each input pixels, that is, at 100% density (except for the border pixels)

#### 6.10.3.4.3 Epipolar Rectification

The DMPAC SDE does not implement support for epipolar rectification on input images. Epipolar rectification must be done in external modules (such as VPAC LDC) and the rectified input image must be fed as input to the DMPAC.

#### 6.10.3.4.4 Disparity Search Method

The following search strategy is employed by the DMPAC SDE module to establish pixel correspondence between the input image pairs:

- 1D search along the image scan line is applied, that is, the correspondence of a pixel in the right image is searched in the same image row in the left image.
- Disparity search is done for every pixel position in the right image without skipping any pixel in the search range.

#### 6.10.3.4.5 Cost Computation Method

The DMPAC SDE aggregates local cost in a custom semi global method. It follows the following steps:

- Input image intensity data is first transformed into Census domain by Census Transform . Census transformation is performed over an 9x9 pixel neighborhood window.
- The local matching cost between two image pixel positions are measured by the Hamming Distance of the corresponding census signature.
- Local costs are then aggregated using a custom semi global matching (SGM) method.
- Similar to standard SGM methods, penalty terms (P1, P2) for disparity changes in the local neighborhood is used.
- The penalty terms P1, P2 are used in a non-adaptive way, that is, they remain constant for the entire frame processing.

#### 6.10.3.4.6 Cost Plane Compression Method

In order to reduce on-chip storage of intermediate aggregated path costs, the DMPAC SDE employs a compression-decompression scheme.

#### 6.10.3.4.7 Sub-Pixel Interpolation Method

For increase precision or 'depth resolution' of the output disparity map, integer pixel disparity map from SGM processing step is further refined to a sub-pixel accurate disparity estimate.

- Sub-pixel accuracy of 1/16th pixel is implemented, encoded in 4 bits of fractional precision in the final output.

#### 6.10.3.4.8 Raw Disparity Output Cleaning Method

The raw output disparity results out of SGM contains several inconsistencies, errors and noise which is cleaned by the following post-processing steps:

- "Left-Right consistency check" is applied to find out exceptional pixel position where correspondence from Left-to-Right search does not match the correspondence from Right-to-Left search. A improvised 'fast method' or approximation is used to avoid full left to right and right to left disparity search.
- A "Low Texture Thresholding" method is applied to filter out pixels with low texture in the neighborhood.

#### Note

The above cleaning methods update the confidence values associated with the pixel positions and do not update the actual disparity values.

#### 6.10.3.4.9 Confidence Score Computation Method

All stereo algorithms, by virtue of the underlying assumption they make, have an inherent limitation on accuracy of the determined disparity map. Confidence score is a metric indicating the inherent correctness/inaccuracy of the each disparity value in the disparity map. The DMPAC SDE generates confidence map as follows:

- Confidence score value shall be 3 bits, 8 levels.

#### 6.10.3.4.10 Disparity Map Post Filtering Method

In order to remove impulsive estimation noise, the DMPAC SDE implements a final filter to remove noise from the generated disparity map:

- 2D 5x5 Median Filter (non-weighted) is used as post-processing filter.
- All samples within the filter kernel shall be considered for median filtering.

#### Note

The raw disparity map from the output of the SGM process is filtered, while the Confidence map is not filtered.

#### 6.10.3.4.11 Disparity Output Data Packing Format

The final disparity value and corresponding confidence score is packed as follows, before being sent out of DMPAC:

- Disparity output is 16-bit packed format.
- Disparity output format is: (MSB)[IntegerBits.FractionalBits.ConfidenceBits](LSB), where:
  - IntegerBits is 9 (support disparity range -3 to +255)
  - FractionalBits is 4 (support 1/16th pixel precision)
  - ConfidenceBits is 3 (8 levels of confidence value)

**Table 6-161. DMPAC Stereo Output Data Format**

Table 1. New Delta-16 Stereo Output Data Format					
15	7	6	3	2	0
IntegerBits		FractionalBits		ConfidenceBits	
9.4b signed number 1b sign, 8b int mag, 4b fraction				3b int	

#### 6.10.3.5 DMPAC Optical Flow Functional Overview

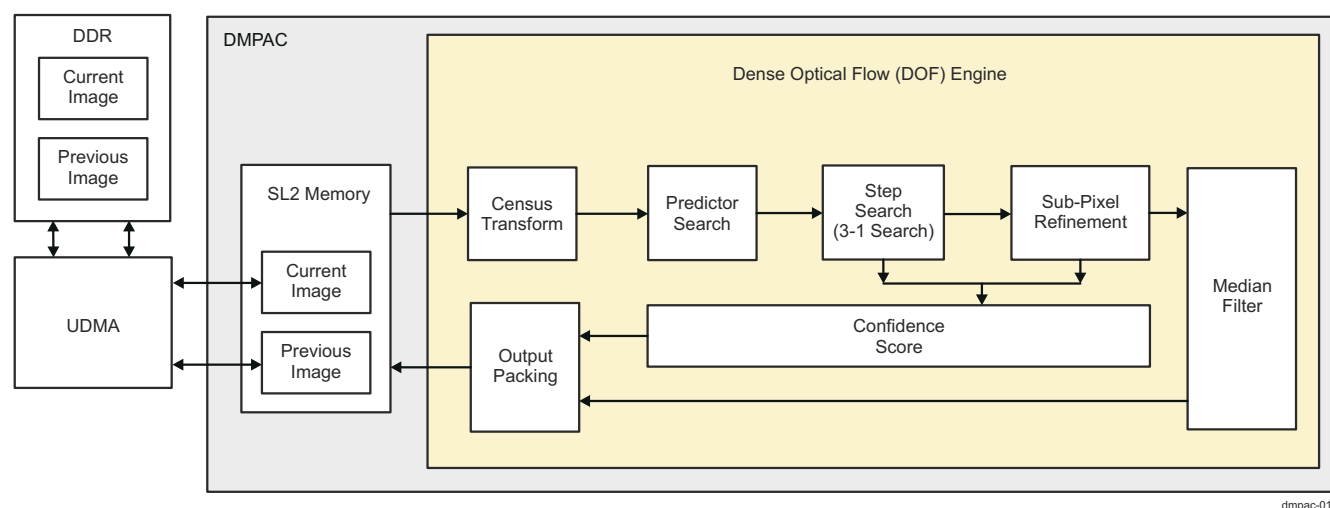
The DMPAC implements Texas Instruments' proprietary algorithm to find the optical flow vector map between the input image pair. The optical flow estimation process implemented in the DMPAC has been formulated

by combining pyramidal block matching (PBM) method with differential technique based optical flow method proposed by Lucas-Kanade(LK). In this formulation the PBM method performs optical flow estimation with integer pixel resolution and followed by an iteration of the LK method used to refine the obtained estimate and obtain optical flow estimate with fractional pixel precision. The PBM method starts the optical flow estimation at the highest pyramid level using spatial predictors and step search method to obtain accurate motion estimation. These optical flow estimates are then appropriately scaled up and refined (again using spatial predictors and step search method) at each of the lower pyramid levels sequentially. At base pyramid level the predictor configuration can be altered to include temporal predictors or auxiliary predictors, however step search remains the same. At base pyramid level, once PBM process completes, an iteration of LK step is used to refine the obtained flow estimates. During predictor evaluation and step search the PBM method uses binary census transform for pixel description and hamming distance as the optimization function. The 2D post filtering of the optical flow estimates can also be performed at output of each of the pyramid levels including base level. The algorithm has been optimized to fit into real time, low power processing requirements with highly precise and accurate estimates.

This section outlines the overall functionality of the Dense Optical Flow (DOF) core.

#### 6.10.3.5.1 Optical Flow Processing Dataflow

Figure 6-160 illustrates the functional view of dataflow for optical flow processing within the DMPAC. The input image pyramids (with required image pre-processing) are stored in external SDRAM (DDR). The DMPAC UTC/UDMA fetches the image pyramid luminance data in its shared L2 memory sub-system and then processes through its pipeline. The final output from the DOF module is written back to the shared DOF L2 memory, which is transferred back to the external SDRAM (DDR).



**Figure 6-160. DMPAC Functional View of Optical Flow Processing Dataflow**

#### 6.10.3.5.2 Flow Vector Range

The DMPAC optical flow core supports motion search region large enough for:

- Max flow vector range of -191 to +191 (horizontal) and -62 to +62 (vertical).
- Supports asymmetric vertical search range - different positive (which is downwards) than negative (which is upwards) vertical search range.
- Flow Vector output shall be of 100% density (flow vector for every pixel).



### Note

There is a further restriction on the combined maximum range for HSR and VSR. For example:

$|HSR| = 191$ , then  $|VSR| \leq 56$

$|VSR| = 62$ , then  $|HSR| \leq 170$

#### 6.10.3.5.3 Block Matching Process

The block matching process is based on motion predictors and two step coarse-to-fine refinement. Where:

- Predictor search stage evaluates configured optical flow predictors and selects the best predictor that minimizes the block matching cost.
- DMPAC defines a list 7 motion predictors that are supported and can be configured at pyramid level. These are referred to as TopLeft, Top, TopRight, Left, PyramidalLeft, PyramidalCurrent and Temporal predictors.
  - DMPAC allows use of upto 5 motion predictors for a pixel and 2 of these can be configured at each pyramid level and other three are fixed.
  - Upper pyramid levels (L5-L1) can use any 2 of Left, PyramidalLeft and PyramidalCurrent predictors.
  - Base pyramid level (L0) can use any 2 of Left, PyramidalLeft PyramidalCurrent and Temporal predictors.
- Two step coarse to fine step search method refines optical flow estimates around winner predictor. A configurable motion smoothness factor value allows the block matching process to control the smoothness of the flow field.

#### 6.10.3.5.4 Image Pyramid Generation Method

The DMPAC optical flow processing requires multi-level Gaussian Image Pyramids for implementing coarse-to-fine refinement method during pixel correspondence search. However, the DMPAC does not implement the logic required to build this image pyramid. It requires some SoC level external logic to build and store the image pyramid in external SDRAM (DDR).

The DMPAC optical flow processing requires/uses the following features for the image pyramid:

- Up to 6 levels of Pyramid (base + 5 octaves) shall be supported.
- Each higher octave is  $\frac{1}{2}$  in width and  $\frac{1}{2}$  in height with regard to its predecessor.
- Gaussian Smoothing followed by  $/2$  down sampling is required to generate next octave.

As shown in [Figure 6-157](#), *DMPAC SoC Level Dataflow for Optical Flow Pre-Processing*, the sclare module (MSC) in VPAC is responsible to generate the image pyramid.

#### 6.10.3.5.5 Cost Computation Method

The DMPAC uses the following methods to derive an effective and robust cost function for predictor search and hierarchical step search stages:

- Cost function used is 'Hamming Distance over Census Transform'.
- Motion smoothness factor (MSF) is fixed for the entire frame processing.

#### 6.10.3.5.6 Sub-Pixel Refinement Method

The DMPAC optical flow uses differential technique to refine flow vector estimates to sub-pixel precision:

- Lucas-Kanade (LK) method is used for computing sub-pixel accurate flow vectors.

#### 6.10.3.5.7 Confidence Score Computation Method

The DMPAC optical flow processing core uses sophisticated methods to compute an accurate and robust measure of the 'estimation confidence' of the computed flow vector and assigns a confidence score to each flow vector based on that.

- Both image and flow field features are used to compute confidence score.
- Following features are used as the basis of confidence measure calculation:
  1. Winner Cost
  2. Gradient in the feature descriptor space calculated using hamming distance

3. Flow gradient of horizontal flow
  4. Flow gradient of vertical flow
  5. Filtered Cost
  6. Filtered gradient
  7. Filtered gradient of horizontal flow
  8. Filtered gradient of vertical flow
- Fixed 16 independent decision trees are supported for adaptive weight computation.
  - Weights are combined linearly and scaled to the supported confidence value range.
  - Confidence score value is quantized to 4 bits, 16 levels.

#### 6.10.3.5.8 Flow Vector Post Filtering Method

The DMPAC implements robust and iterative outlier rejection technique after processing each level of image pyramid during optical flow processing. The following methods are used:

- Estimated flow vectors are filtered after each pyramid level.
- Horizontal and vertical component of the flow vector are filtered separately.
- 2D 5x5 Median Filter shall be supported as post-processing filter.
- All samples within the filter kernel shall be considered for median filtering.

#### 6.10.3.5.9 Flow Vector Output Data Packing Format

The DMPAC optical flow output is saved off to DDR (via the DMPAC SL2 using UTC/UDMA) in a packed format. The parameters are as follows:

- For Base Layer, the flow vector output is 32-bit packed format.
- The flow vector output format is: (MSB) [IntegerBitsU, FractionalBitsU, IntegerBitsV, FractionalBitsV, ConfidenceBits] (LSB), where:
  - IntegerBitsU is 9 bits, and IntegerBitsV is 7 bits (support for +/- 191 H flow vectors and +/- 63 V flow vectors).
  - FractionalBitsU, FractionalBitsV is 4 bits each (support for 1/16th pixel of precision).
  - ConfidenceBits is 4 bits (16 levels)

**Table 6-162. DMPAC Optical Flow Output Data Format for Base Layer**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Horizontal Flow Vector																Vertical Flow Vector															
Sign		Integer (with sign)										Fractional				Sign	Integer (with sign)						Fractional				Confidence				
9,4b signed number 9b integer, 4b fraction																7,4b signed number 7b integer, 4b fraction															

**Table 6-163. Flow Vector Map Output Packing Format for Non Base Layers**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Horizontal Flow Vector										Vertical Flow Vector					
Integer (with sign)										Integer (with sign)					
9b signed number										7b signed number					

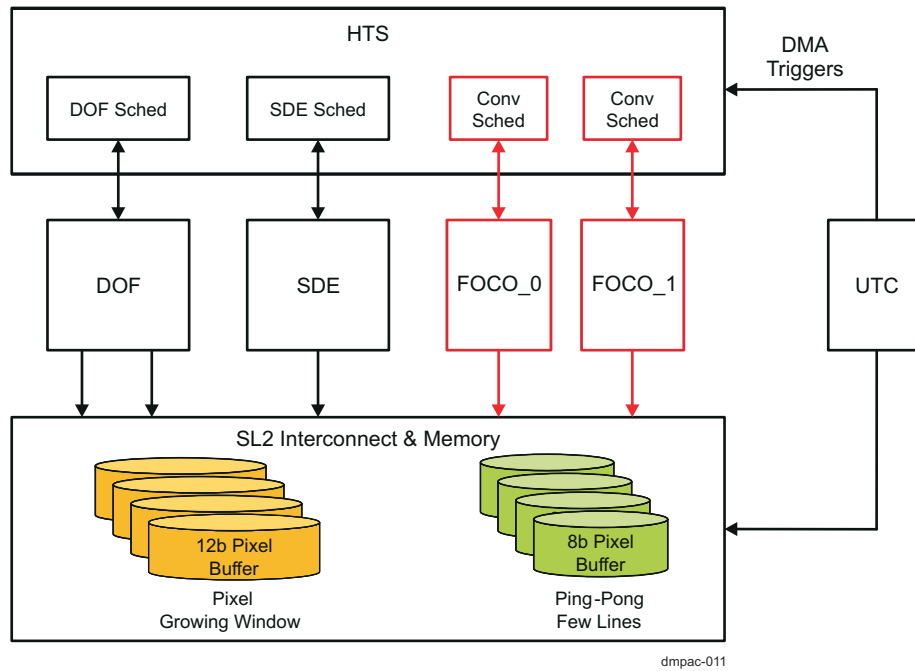
#### 6.10.3.5.10 Sparse Optical Flow Support

The DMPAC produces dense optical flow output by default. In order to facilitate smaller amount of output data that is easier to process in an external DSP/Arm during a Sparse Optical Flow and/or Tracking algorithm, the DMPAC supports a sparse flow vector output generation based on an input binary image map.

#### 6.10.3.6 DMPAC Format Conversion (FOCO) Module Operation

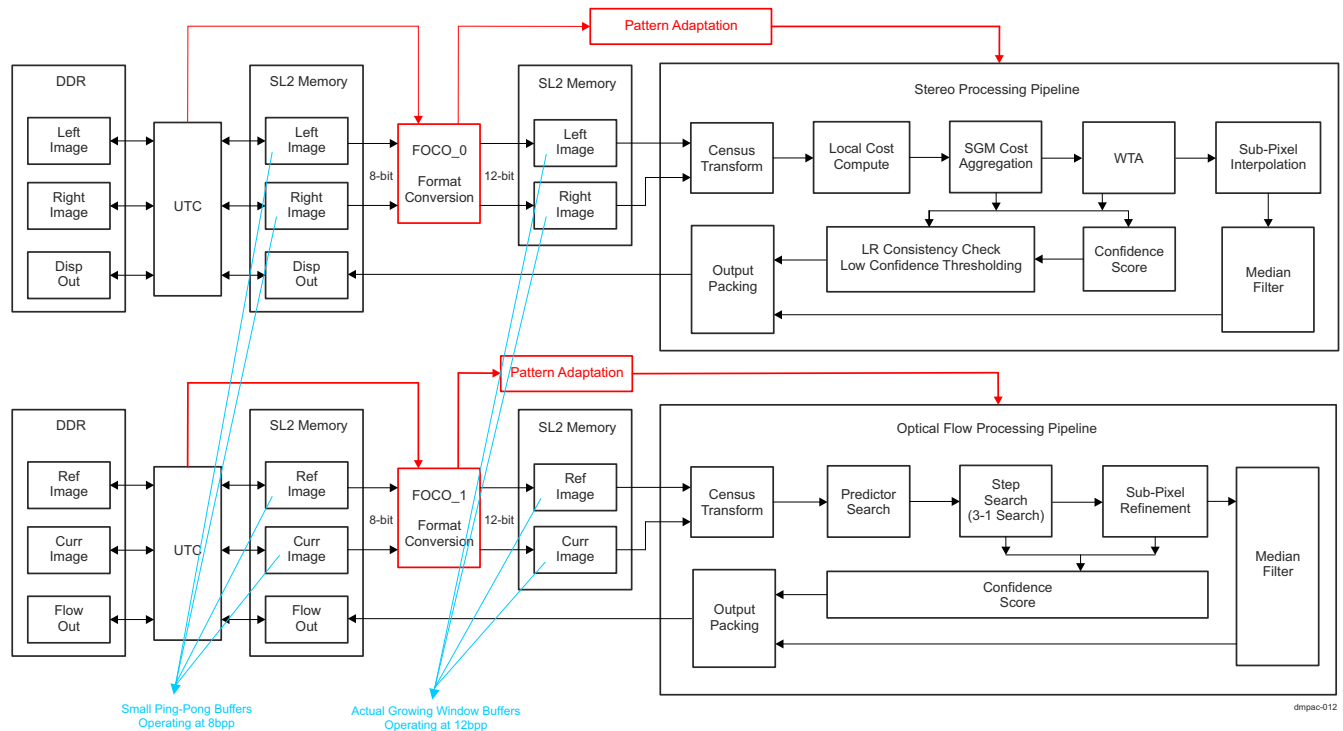
The SDE and DOF engines always work only on pixel data that is organized in a fully packed 12-bit format. These accelerators do not support any other input pixel data formats. To support other data formats, two format conversion modules (FOCO\_0 and FOCO\_1) are included inside the DMPAC sub-system. The FOCO converts

other data formats into the data format supported by the SDE and DOF engines. Figure 6-161 shows how the format conversion engine sits inside the DMPAC sub-system.



**Figure 6-161. DMPAC with Format Conversion Engine (FOCO)**

The FOCO operates in a SL2 memory-to-memory processing fashion. The DMA engine brings the data into the SL2 which is then read by the FOCO, converted and written back into SL2 to be consumed by the SDE and DOF engines. Figure 6-162 shows the data flow with the format conversion engines.



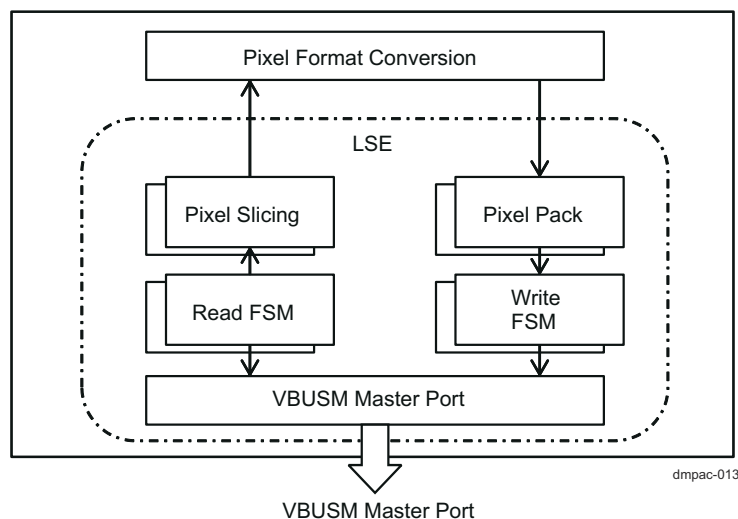
**Figure 6-162. DMPAC Data Flow with FOCO**

### Note

There is only one SL2 memory instance. The two SL2 Memory blocks share correspond to two logic partitions.

#### 6.10.3.6.1 FOCO Implementation Details

Figure 6-163 shows a logical block diagram of a format conversion engine.



**Figure 6-163. DMPAC FOCO Logical Block Diagram**

All the FOCO sub-components mentioned above, except the Pixel Format Conversion sub-component, are implemented using a Load Store Engine (LSE). The operating granularity of FOCO is always one pixel line. Since the input operation granularity of the SDE and DOF engines is different from one pixel line, there is a pattern adapter inside the HTS to enable connection between the FOCO and these engines.

Only 2 input and 2 output channels in each FOCO (one channel for base and reference image) are needed for the DOF and SDE operation. However, there are totally 4 input and 4 output channels in each FOCO instance to enable newer processing flows in the future.

The LSE inside FOCO has the following characteristics:

- 4 Read Channels supporting 8-bit packed, 12-bit packed + MSB/LSB aligned, 16-bit.
- 4 Write Channels supporting 8-bit packed, 12-bit packed + MSB/LSB aligned, 16-bit.
- Single threaded operation.

For more details on LSE operation, see *Load Store Engine (LSE)*.

#### 6.10.3.6.2 FOCO Core Details

The format conversion core implements a generic round-shift-mask pipelined structure. The idea is to get various different types of format conversion by configuring different values for round, shift and mask registers dedicated for each input-output channel pair.

There are two 32-bit configuration registers, DMPAC\_FOCO\_0\_CH\_CTRL\_j/DMPAC\_FOCO\_1\_CH\_CTRL\_j and DMPAC\_FOCO\_0\_CH\_COUNT\_j/DMPAC\_FOCO\_1\_CH\_COUNT\_j, for each input-output channel pair inside the FOCO core. [Section 6.10.4, DMPAC Programming Guide](#), contains two example FOCO configurations for converting different types of pixel format.

#### 6.10.3.7 DMPAC Clocks

The DMPAC receives the following clocking signals. For more information on the source of the clocks at SoC level, see *DMPAC Integration*.

- The MAIN\_CLK is the clock for the DMPAC top level infrastructure blocks.

- The DOF\_CLK is the clock for the DOF block.
- The SDE\_CLK is the clock for the SDE block.

#### 6.10.3.8 DMPAC Resets

The DMPAC receives the following hardware reset signals. For more information which is their source at SoC level, see *DMPAC Integration*.

- The MAIN\_DOF\_RST is the reset signal for the DMPAC top level infrastructure blocks, as well as for the DOF block.
- The SDE\_RST is the reset signal for the SDE block.

#### 6.10.3.9 DMPAC Interrupts

The DMPAC outputs several interrupt lines. For more information how they are connected at SoC level, see *DMPAC Integration*.

- DMPAC0\_INTD\_0\_SYSTEM\_INTR\_LEVEL\_0 and DMPAC0\_INTD\_0\_SYSTEM\_INTR\_LEVEL\_1 interrupt lines

The CP\_INTD block inside DMAPC is used to aggregate several internal interrupt event sources into 2 output interrupt lines: DMPAC0\_INTD\_0\_SYSTEM\_INTR\_LEVEL\_0 and DMPAC0\_INTD\_0\_SYSTEM\_INTR\_LEVEL\_1. The mapping of internal events to each line is controlled via a separate set of CP\_INTD registers.

Table 6-164 shows how the internal interrupt event sources are mapped on the CP\_INTD registers for the DMPAC0\_INTD\_0\_SYSTEM\_INTR\_LEVEL\_0 line.

**Table 6-164. DMPAC Interrupts Mapping for DMPAC0\_INTD\_0\_SYSTEM\_INTR\_LEVEL\_0**

Event Source Name	Event Source Type	Enable/Enable Clear Registers	Status/Status Clear Registers
DOF_ROW_DONE_INTR	Pulse	DMPAC_INTD_ENABLE_REG_PULSE_DM	DMPAC_INTD_STATUS_REG_PULSE_DMP
DOF_FRAME_DONE_INTR	Pulse	PAC_OUT_0_0/	AC_OUT_0_0/
DOF_READ_ERROR_INTR	Pulse	DMPAC_INTD_ENABLE_CLR_REG_PULSE	DMPAC_INTD_STATUS_CLR_REG_PULSE
DOF_WRITE_ERROR_INTR	Pulse	_DMPAC_OUT_0_0	_DMPAC_OUT_0_0
DOF_MP0_RD_STATUS_ERROR	Pulse		
SDE_BLK_DONE_INTR	Pulse		
SDE_FRAME_DONE_INTR	Pulse		
SDE_READ_ERROR_INTR	Pulse		
SDE_WRITE_ERROR_INTR	Pulse		
FOCO_0_FR_DONE_EVT	Pulse		
FOCO_0_SL2_RD_ERR	Pulse		
FOCO_0_SL2_WR_ERR	Pulse		
FOCO_1_FR_DONE_EVT	Pulse		
FOCO_1_SL2_RD_ERR	Pulse		
FOCO_1_SL2_WR_ERR	Pulse		
PIPE_DONE_[0:3]	Pulse	DMPAC_INTD_ENABLE_REG_PULSE_DM	DMPAC_INTD_STATUS_REG_PULSE_DMP
SPARE_DEC_0	Pulse	PAC_OUT_0_1/	AC_OUT_0_1/
SPARE_DEC_1	Pulse	DMPAC_INTD_ENABLE_CLR_REG_PULSE	DMPAC_INTD_STATUS_CLR_REG_PULSE
SPARE_PEND_0_P	Level	_DMPAC_OUT_0_1	_DMPAC_OUT_0_1
		DMPAC_INTD_ENABLE_REG_PULSE_DM	DMPAC_INTD_STATUS_REG_PULSE_DMP
		PAC_OUT_0_1/	AC_OUT_0_1/
		DMPAC_INTD_ENABLE_CLR_REG_PULSE	DMPAC_INTD_STATUS_CLR_REG_PULSE
		_DMPAC_OUT_0_1	_DMPAC_OUT_0_1

**Table 6-164. DMPAC Interrupts Mapping for DMPAC0\_INTD\_0\_SYSTEM\_INTR\_LEVEL\_0 (continued)**

Event Source Name	Event Source Type	Enable/Enable Clear Registers	Status/Status Clear Registers
SPARE_PEND_0_L	Level	DMPAC_INTD_ENABLE_REG_LEVEL_DMPAC_OUT_0_1/ DMPAC_INTD_ENABLE_CLR_REG_LEVEL_DMPAC_OUT_0_1	DMPAC_INTD_STATUS_REG_LEVEL_DMPAC_OUT_0_1/ DMPAC_INTD_STATUS_CLR_REG_LEVEL_DMPAC_OUT_0_1
SPARE_PEND_1_P	Level	DMPAC_INTD_ENABLE_REG_PULSE_DMPAC_OUT_0_1/ DMPAC_INTD_ENABLE_CLR_REG_PULSE_DMPAC_OUT_0_1	DMPAC_INTD_STATUS_REG_PULSE_DMPAC_OUT_0_1/ DMPAC_INTD_STATUS_CLR_REG_PULSE_DMPAC_OUT_0_1
SPARE_PEND_1_L	Level	DMPAC_INTD_ENABLE_REG_LEVEL_DMPAC_OUT_0_1/ DMPAC_INTD_ENABLE_CLR_REG_LEVEL_DMPAC_OUT_0_1	DMPAC_INTD_STATUS_REG_LEVEL_DMPAC_OUT_0_1/ DMPAC_INTD_STATUS_CLR_REG_LEVEL_DMPAC_OUT_0_1
TDONE_0	Pulse	DMPAC_INTD_ENABLE_REG_PULSE_DMPAC_OUT_0_1/ DMPAC_INTD_ENABLE_CLR_REG_PULSE_DMPAC_OUT_0_1	DMPAC_INTD_STATUS_REG_PULSE_DMPAC_OUT_0_1/ DMPAC_INTD_STATUS_CLR_REG_PULSE_DMPAC_OUT_0_1
TDONE_1	Pulse		
TDONE_7	Pulse		
TDONE_8	Pulse		
WATCHDOGTIMER_ERR_0	Pulse	DMPAC_INTD_ENABLE_REG_PULSE_DMPAC_OUT_0_2/ DMPAC_INTD_ENABLE_CLR_REG_PULSE_DMPAC_OUT_0_2	DMPAC_INTD_STATUS_REG_PULSE_DMPAC_OUT_0_2/ DMPAC_INTD_STATUS_CLR_REG_PULSE_DMPAC_OUT_0_2
WATCHDOGTIMER_ERR_1	Pulse		
WATCHDOGTIMER_ERR_7	Pulse		
WATCHDOGTIMER_ERR_8	Pulse		
DRU_ERROR_[0:31]	Pulse	DMPAC_INTD_ENABLE_REG_PULSE_DMPAC_OUT_0_3/ DMPAC_INTD_ENABLE_CLR_REG_PULSE_DMPAC_OUT_0_3	DMPAC_INTD_STATUS_REG_PULSE_DMPAC_OUT_0_3/ DMPAC_INTD_STATUS_CLR_REG_PULSE_DMPAC_OUT_0_3
DRU_COMPLETE_[0:31]	Pulse		
DRU_LOCAL_OUT_EVENT_[0:31]	Pulse	DMPAC_INTD_ENABLE_REG_PULSE_DMPAC_OUT_0_5/ DMPAC_INTD_ENABLE_CLR_REG_PULSE_DMPAC_OUT_0_5	DMPAC_INTD_STATUS_REG_PULSE_DMPAC_OUT_0_5/ DMPAC_INTD_STATUS_CLR_REG_PULSE_DMPAC_OUT_0_5
DRU_PROT_ERROR	Pulse		
CTM_PULSE	Pulse		

Table 6-165 shows how the internal interrupt event sources are mapped on the CP\_INTD registers for the DMPAC0\_INTD\_0\_SYSTEM\_INTR\_LEVEL\_1 line.

**Table 6-165. DMPAC Interrupts Mapping for DMPAC0\_INTD\_0\_SYSTEM\_INTR\_LEVEL\_1**

Event Source Name	Event Source Type	Enable/Enable Clear Registers	Status/Status Clear Registers
DOF_ROW_DONE_INTR	Pulse	DMPAC_INTD_ENABLE_REG_PULSE_DM PAC_OUT_1_0/ DMPAC_INTD_ENABLE_CLR_REG_PULSE _DMPAC_OUT_1_0	DMPAC_INTD_STATUS_REG_PULSE_DMP AC_OUT_1_0/ DMPAC_INTD_STATUS_CLR_REG_PULSE _DMPAC_OUT_1_0
DOF_FRAME_DONE_INTR	Pulse		
DOF_READ_ERROR_INTR	Pulse		
DOF_WRITE_ERROR_INTR	Pulse		
DOF_MP0_RD_STATUS_ER ROR	Pulse		
SDE_BLK_DONE_INTR	Pulse		
SDE_FRAME_DONE_INTR	Pulse		
SDE_READ_ERROR_INTR	Pulse		
SDE_WRITE_ERROR_INTR	Pulse		
FOCO_0_FR_DONE_EVT	Pulse		
FOCO_0_SL2_RD_ERR	Pulse		
FOCO_0_SL2_WR_ERR	Pulse		
FOCO_1_FR_DONE_EVT	Pulse		
FOCO_1_SL2_RD_ERR	Pulse		
FOCO_1_SL2_WR_ERR	Pulse		
PIPE_DONE_[0:3]	Pulse	DMPAC_INTD_ENABLE_REG_PULSE_DM PAC_OUT_1_1/ DMPAC_INTD_ENABLE_CLR_REG_PULSE _DMPAC_OUT_1_1	DMPAC_INTD_STATUS_REG_PULSE_DMP AC_OUT_1_1/ DMPAC_INTD_STATUS_CLR_REG_PULSE _DMPAC_OUT_1_1
SPARE_DEC_0	Pulse		
SPARE_DEC_1	Pulse		
SPARE_PEND_0_P	Level	DMPAC_INTD_ENABLE_REG_PULSE_DM PAC_OUT_1_1/ DMPAC_INTD_ENABLE_CLR_REG_PULSE _DMPAC_OUT_1_1	DMPAC_INTD_STATUS_REG_PULSE_DMP AC_OUT_1_1/ DMPAC_INTD_STATUS_CLR_REG_PULSE _DMPAC_OUT_1_1
SPARE_PEND_0_L	Level		
SPARE_PEND_1_P	Level	DMPAC_INTD_ENABLE_REG_PULSE_DM PAC_OUT_1_1/ DMPAC_INTD_ENABLE_CLR_REG_PULSE _DMPAC_OUT_1_1	DMPAC_INTD_STATUS_REG_PULSE_DMP AC_OUT_1_1/ DMPAC_INTD_STATUS_CLR_REG_PULSE _DMPAC_OUT_1_1
SPARE_PEND_1_L	Level		
TDONE_0	Pulse	DMPAC_INTD_ENABLE_REG_PULSE_DM PAC_OUT_1_1/ DMPAC_INTD_ENABLE_CLR_REG_PULSE _DMPAC_OUT_1_1	DMPAC_INTD_STATUS_REG_PULSE_DMP AC_OUT_1_1/ DMPAC_INTD_STATUS_CLR_REG_PULSE _DMPAC_OUT_1_1
TDONE_1	Pulse		
TDONE_7	Pulse		
TDONE_8	Pulse		
WATCHDOGTIMER_ERR_0	Pulse	DMPAC_INTD_ENABLE_REG_PULSE_DM PAC_OUT_1_2/ DMPAC_INTD_ENABLE_CLR_REG_PULSE _DMPAC_OUT_1_2	DMPAC_INTD_STATUS_REG_PULSE_DMP AC_OUT_1_2/ DMPAC_INTD_STATUS_CLR_REG_PULSE _DMPAC_OUT_1_2
WATCHDOGTIMER_ERR_1	Pulse		
WATCHDOGTIMER_ERR_7	Pulse		
WATCHDOGTIMER_ERR_8	Pulse		
DRU_ERROR_[0:31]	Pulse	DMPAC_INTD_ENABLE_REG_PULSE_DM PAC_OUT_1_3/ DMPAC_INTD_ENABLE_CLR_REG_PULSE _DMPAC_OUT_1_3	DMPAC_INTD_STATUS_REG_PULSE_DMP AC_OUT_1_3/ DMPAC_INTD_STATUS_CLR_REG_PULSE _DMPAC_OUT_1_3



**Table 6-165. DMPAC Interrupts Mapping for DMPAC0\_INTD\_0\_SYSTEM\_INTR\_LEVEL\_1 (continued)**

Event Source Name	Event Source Type	Enable/Enable Clear Registers	Status/Status Clear Registers
DRU_COMPLETE_[0:31]	Pulse	DMPAC_INTD_ENABLE_REG_PULSE_DM PAC_OUT_1_4/ DMPAC_INTD_ENABLE_CLR_REG_PULSE _DMPAC_OUT_1_4	DMPAC_INTD_STATUS_REG_PULSE_DMP AC_OUT_1_4/ DMPAC_INTD_STATUS_CLR_REG_PULSE _DMPAC_OUT_1_4
DRU_LOCAL_OUT_EVENT _[0:31]	Pulse	DMPAC_INTD_ENABLE_REG_PULSE_DM PAC_OUT_1_5/ DMPAC_INTD_ENABLE_CLR_REG_PULSE _DMPAC_OUT_1_5	DMPAC_INTD_STATUS_REG_PULSE_DMP AC_OUT_1_5/ DMPAC_INTD_STATUS_CLR_REG_PULSE _DMPAC_OUT_1_5
DRU_PROT_ERROR	Pulse	DMPAC_INTD_ENABLE_REG_PULSE_DM PAC_OUT_1_6/ DMPAC_INTD_ENABLE_CLR_REG_PULSE _DMPAC_OUT_1_6	DMPAC_INTD_STATUS_REG_PULSE_DMP AC_OUT_1_6/ DMPAC_INTD_STATUS_CLR_REG_PULSE _DMPAC_OUT_1_6

**Note**

In [Table 6-164](#) and [Table 6-165](#), the bit positions corresponding to SPARE\_PEND\_0\_L and SPARE\_PEND\_1\_L should not be used in registers controlling pulse outputs. Instead, the bit positions corresponding to SPARE\_PEND\_0\_P and SPARE\_PEND\_1\_P should be used to control SPARE\_PEND\_0 and SPARE\_PEND\_1 interrupts, respectively. These registers are:

- \*\_ENABLE\_REG\_PULSE\_DMPAC\_OUT\_\*
- \*\_ENABLE\_CLR\_REG\_PULSE\_DMPAC\_OUT\_\*
- \*\_STATUS\_REG\_PULSE\_DMPAC\_OUT\_\*
- \*\_STATUS\_CLR\_REG\_PULSE\_DMPAC\_OUT\_\*

Similarly, the bit positions corresponding to SPARE\_PEND\_0\_P and SPARE\_PEND\_1\_P should not be used in registers controlling level outputs. Instead, the bit positions corresponding to SPARE\_PEND\_0\_L and SPARE\_PEND\_1\_L should be used to control SPARE\_PEND\_0 and SPARE\_PEND\_1 interrupts, respectively. These registers are:

- \*\_ENABLE\_REG\_LEVEL\_DMPAC\_OUT\_\*
- \*\_ENABLE\_CLR\_REG\_LEVEL\_DMPAC\_OUT\_\*
- \*\_STATUS\_REG\_LEVEL\_DMPAC\_OUT\_\*
- \*\_STATUS\_CLR\_REG\_LEVEL\_DMPAC\_OUT\_\*

- DMPAC0\_ECC\_CORRECTED\_ERR\_LEVEL\_0 and DMPAC0\_ECC\_UNCORRECTED\_ERR\_LEVEL\_0 interrupt lines

The ECC Aggregator block within the DMPAC aggregates the pending status from the ECC protected RAMs into two output interrupt lines: DMPAC0\_ECC\_CORRECTED\_ERR\_LEVEL\_0 for correctable errors, and DMPAC0\_ECC\_UNCORRECTED\_ERR\_LEVEL\_0 for uncorrectable errors. For more information on the ECC memory protection, see [Section 6.10.3.16, DMPAC Memory Error Protection](#).

**6.10.3.10 DMPAC SL2 Memory Subsystem**

A common shared level 2 (SL2) memory sub-system serves the input output data for both SDE and DOF cores. Same memory is also used by the FOCO module, when required, and it also mapped to SoC address map for external usage. This SL2 memory sub-system has the following configuration:

- 512KB SRAM; No ECC implemented

SL2 size allocation restrictions:

SDE SL2 memory Calculation (M1+M2+M3+M4)

- Internal Memory

M1= (Image Width/8u)\*2u\*8u\*8u;



```

if(sl2AllocPrms->searchRange == 192)
{
M1 = M1 * 3u;
}
else if(sl2AllocPrms->searchRange == 128)
{
M1 = M1 * 2u;
}
- Reference Frame (M2)/Current Frame (M3)
pitch = ((Image Width)*3/2) + 63 & 0xFFFFF0; // 64 byte aligned
if(0u == ((pitch)%256u))
{
pitch += 64u;
}
M3 = M2 = pitch*24;
- Disparity Buffer (M4)
M4 = (Image Width)*2*10*2;
DOF SI2 memoeru Calculation (M1+M2+M3+M4+M5)
- Reference Frame (M1)/Current Frame (M2)
pitch = ((Image Width)*3/2) + 63 & 0xFFFFF0; // 64 byte aligned
if(0u == ((pitch)%256))
{
pitch += 64u;
}
M1 = pitch * (((topSR + 7) & 0xFFFFF0) + 2
+ ((botSR + 10) & 0xFFFFF0)
+ 2);
M2 = pitch * (6 + 2 + 8 + 2);
- Temporal predictor (M3)
M3 = (((Image Width) * 4) + 63) & 0xFFFFF0 * 2;
- Pyramidal Predictor (M4)
M4 = (((Image Width) + 63) & 0xFFFFF0) * 2;
- Output Flow Vector (M5)
M5 = (((Image Width) * 4) + 63) & 0xFFFFF0 * 4;

```

### 6.10.3.11 DMPAC Common DMA

The UTC module is used as the DMA for DMPAC. The UTC implements a Channel Controller (CC) as the programming interface and Transfer Controller (TC) for managing DMA transfers. The UTC used for DMPAC is optimized for transfer types such as 2D/3D/4D that are typical of DMPAC access patterns. The following features/parameters are implemented for the UTC module instantiated for DMPAC:

- 32 events; no QDMA requirement
- 32 TR entries

The DMPAC UTC channel controller supports DMA transfer events to read or write following data:

1. Stereo Base Input
2. Stereo Reference Input
3. Stereo Disparity Output
4. Optical Flow Current Input
5. Optical Flow Reference Input
6. Optical Flow Top Pyramid Flow Vector Input
7. Optical Flow Temporal Flow Vector Input
8. Optical Flow Sparse Processing Binary Map Input
9. Optical Flow Vector Output

These data transfer events would be mapped to one or more TR entries within UTC through its programming. All DMPAC data transfers are deterministic in nature and hence it does not require any interaction from the software in between a frame processing. The TR entries should be programmed in such a way that they operate on a circular buffer in SL2 and a full frame level buffer in DDR. For more details about the number and programming of TR entries, see [Section 6.10.4, DMPAC Programming Guide](#).

All data movement transactions at DMPAC boundary are in multiples of aligned 128-byte transfers.

*Chanid* signal on VBUSM Data Master interface carries the UTC channel number which initiated the transfer.

### 6.10.3.12 DMPAC Messaging and Control

The DMPAC SDE and DOF cores run independent hardware processing and data transfer (via UTC/UDMA) threads in order to process stereo and optical flow simultaneously. The thread management is flexible and supports different top level settings of resolution, frame rate, DDR data buffer address etc.

The HTS module for DMPAC is used to implement the thread management and control triggering of processing threads within the DMPAC. It is also used to manage message transfer and control between DMPAC and external SoC level components like VPAC and DSP. For more details on the HTS for DMPAC, see *Hardware Accelerator (HWA) Thread Scheduler (HTS)*.

#### 6.10.3.12.1 DOF Node Scheduler

The DOF engine uses the HWA0 node schedulers in the DMPAC HTS.

The scheduler implements 5 consumer sockets:

1. Reference Frame Read (UTC)
2. Current Frame Read (UTC)
3. Temporal Flow Vector Read (UTC)
4. Pyramidal Flow Vector Read (UTC)
5. Sparse Flow Binary Map Read (UTC)

Some of these consumer sockets will be disabled during different data processing flows. For example, in case of dense processing, the socket #5 would be disabled, as there would be no sparse map read.

The scheduler implements 1 producer socket:

1. Flow Vector Write (UTC)

Before the start of the optical flow processing, the first 71 image lines of reference frame must be aggregated to start the processing of the first paxel row and this progressively increases to 142 image lines in steady state. Similarly, for the current frame, 9 image lines must be aggregated before the start of the processing of the first

paxel row and then progressively increases to 15 lines in steady state. To enable this, the reference frame and current frame read DMA producer scheduler implements transaction aggregator to aggregate enough number of image lines to start the optical flow operation.

#### **6.10.3.12.2 SDE Node Scheduler**

The SDE core uses the HWA1 node schedulers in the DMPAC HTS.

The scheduler implements 2 consumer sockets:

1. Reference Frame Read (UTC)
2. Base Frame Read (UTC)

The scheduler implements 1 producer socket:

1. Stereo Disparity Write (UTC)

Before the start of the stereo disparity processing, the first 16 image lines of both reference and base frame must be aggregated to start the processing of the first row of blocks and then 8 additional image lines in steady state for the subsequent row of blocks. To enable this, the reference and base frame read DMA producer scheduler implements transaction aggregator to aggregate enough number of image lines to start the optical flow operation.

The SDE inherently operates on blocks and reads the inputs and writes the outputs in terms of blocks. The HTS manages the stereo triggers at block level. While the output DMA for stereo disparity happens at block level, the inputs from DDR are read at line level. To handle the difference in pattern between DMA and compute on the input side, there is a pattern adapter implemented inside the reference and base frame read DMA producer scheduler which adapts the lines based growing window fetched from UTC into several blocks to be computed by the SDE. This pattern adaptation is just a way to handle the start of the stereo disparity engine, it does not alter the data organization of these input buffers in SL2.

#### **6.10.3.13 DMPAC Hardware Security**

The DMPAC implements firewalls inside its configuration interconnect and SL2 interconnect to prevent any rogue master to control and corrupt the data and operation of the DMPAC. The firewalls only allows accesses for commands with appropriate previledges (defined by the VBUS privID). [Figure 6-164](#) gives a comprehensive picture of all the hardware security components inside the DMPAC.



#### 6.10.3.13.1 Configuration Interconnect

Copyright © 2024 Texas Instruments Incorporated

**Table 6-166. DMPAC Configuration Interconnect FW Details**

Interconnect	Security component	Parameter	Endpoints	DMSC VBUSP Offset	Regions	Num privIDs	FW ID	privID Reset Value (dec)
Config VBUSP CBASS	Region FW	1 region	DMPAC_TOP	0x400	1	3	5985	0
		4 region	INTD	0x800	4	3	5986	0
		4 region	HTS	0xC00	4	3	5987	0
		1 region	ECC Aggr	0x0	1	3	5984	0
		1 region	CTSET	0x1000	1	3	5988	0
		1 region	FOCO_0	0x1400	1	3	5989	0
		1 region	FOCO_1	0x1800	1	3	5990	0
		1 region	DOF	0x1C00	1	3	5991	0
		1 region	SDE	0x2000	1	3	5992	0
		4 region	UTC	0x2400	4	3	5992	0

### 6.10.3.13.2 SL2 Interconnect

The DMPAC SL2 interconnect has 4 end-points, each one corresponding to one of the 4 banks of SL2. Each of these end-points have a 4-region region-based firewall instance. Each of the regions in all of the SL2 interconnect firewalls have capability to allow the access from a maximum of 3 privIDs.

Of all the master ports on SL2, only the ones originating from the UTC (via the DMA interconnect) associate a privID with the commands. The master ports originating from the functional compute blocks like DOF, SDE, and FOCO engines do not associate a privID with the commands that they issue. For that purpose, an ISC is instantiated inside SL2 for each of these functional compute masters. These ISCs operate in a channelized mode with just one channel. There is no differentiation between various commands on a master port. A LUT inside ISC would associate a privID to all the commands based on what is configured by the DMSC at SoC level.

[Table 6-167](#) captures the details about all these instances and their IDs and offsets.

**Table 6-167. DMPAC SL2 Interconnect FW and ISC Details**

Interconnect	Security component	Parameter	Endpoints	DMSC VBUSP Offset	Regions	Num privIDs	FW ID	privID Reset Value (dec)
SL2 VBUSM CBASS	ISC	Single Master	SDE	0x1000	ch			208
		Single Master	DOF RD	0x1400	ch			209
		Single Master	DOF	0x1800	ch			210
		Single Master	FOCO_0	0x1C00	ch			211
		Single Master	FOCO_1	0x2000	ch			212
	Region FW	4-region	SL2 Bank0	0x0	4	3	6016	0
		4-region	SL2 Bank1	0x400	4	3	6017	0
		4-region	SL2 Bank2	0x800	4	3	6018	0
		4-region	SL2 Bank3	0xC00	4	3	6019	0

### 6.10.3.14 DMPAC Debug

The DMPAC HTS implements the host debug interfaces required for the SoC level debug manager. This provides the hardware thread level execution control and debug visibility capability to the host debugger.

The DMPAC also implements extensive benchmarking and real time event trace support via the Counter, Timer and System Event Trace (CTSET) module. An extensive set of event of interest within the stereo and optical flow processing cores and DMA completion events are mapped to the CTSET module, thus providing both benchmarking and event trace capabilities to DMPAC.

[Table 6-168](#) captures how the DMPAC internal events are hooked up with the CTSET module.

**Table 6-168. DMPAC CTSET Events List**

Index	Event Name	Event Type	Event Description
0	dof_hts_init	Pulse	DOF initialization request
1	dof_hts_init_tdone	Pulse	Acknowledgement of DOF init req
2	dof_hts_tstart	Pulse	DOF Thread start
3	dof_hts_tdone	Pulse	DOF Thread done
4	dof_hts_tdone_mask	Pulse	DOF output data mask, inline with tdone
5	dof_hts_eop	Pulse	DOF End of Frame processing
6	dof_hts_debug_rdy	Level	DOF Halt ready Status. 1 - Halted state, 0 - Functional state
7	dof_clkgate	Level	MPB all clk gate
8	dof_se_prefetch	Level	DOF SE is pre-fetching
9	dof_se_stall	Level	DOF SE is stalled
10	dof_stall	Level	MPB clk gate due to empty buffers
63:11	Reserved	-	-
64	sde_hts_init	Pulse	SDE initialization request
65	sde_hts_init_tdone	Pulse	Acknowledgement of SDE init req
66	sde_hts_tstart	Pulse	SDE Thread start
67	sde_hts_tdone	Pulse	SDE Thread done
68	sde_hts_tdone_mask	Pulse	SDE output data mask, inline with tdone
69	sde_hts_eop	Pulse	SDE End of Frame processing
70	sde_hts_debug_rdy	Level	SDE Halt ready Status. 1 - Halted state, 0 - Functional state
71	sde_stall	Level	SDE active high back pressure -or- stall
199:108	Reserved	-	-
159:128	dma_channel_start[31:0]	Pulse	DMPAC DMA Start triggers from HTS to UTC
191:160	dma_channel_done[31:0]	Pulse	DMPAC DMA Done events from UTC to HTS
223:192	dru_ctset_intr	Pulse	
254:218	Reserved	-	-

### 6.10.3.15 DMPAC Internal Diagnostic Features

The DMPAC implements several internal diagnostic features:

- Fault coverage on standard logic
  - Parallel Signature Analyzer (PSA) based signature generation for all critical address and data bus interfaces (internal and external) to enable faster "functional BIST" on logic gates and registers.
    - PSA are 32-bit CRC based
    - PSA is implemented within DOF, SDE, and FOCO sub-modules
- Error detection, diagnostics and recovery
  - Dedicated Watchdog Timers compute processing thread to detect hang condition; upon detection the DMPAC generates interrupt to external host
  - Support for external host access to registers while suspended due to error

#### Note

The external host (software) driven reset to DMPAC is the only guaranteed way of recovering from an error (due to any source).

If the input data and configuration to DMPAC operation is pre-determined (used for periodic testing), then output data also gets known. In such a situation, complete output buffer comparison can be avoided by enabling PSA

on output buffer. The PSA generates a 32-bit CRC value which can be compared to a pre-calculated value, thus avoiding complete output buffer comparison. The DOF and SDE cores implement PSA internal to them.

#### 6.10.3.16 DMPAC Memory Error Protection

The ECC Aggregator module integrated within the DMPAC provides a mechanism to control and monitor certain internal ECC RAMs via Single Error Correction (SEC) and Double Error Detection (DED) functions. The ECC Aggregator supports software readable status of ECC single/double-bit errors and associated information such as RAM address and data bit(s) that are in error.

The memories within the DMPAC that are ECC protected are:

- UTC block memories:
  - dru\_queue\_buffer0 RAM
  - dru\_queue\_buffer1 RAM
  - dru\_queue\_buffer2 RAM
  - dru\_state\_buffer0 RAM
  - dru\_response\_buffer0 RAM

For more information on ECC Aggregator operation, see *ECC Aggregator*.

#### 6.10.4 DMPAC Programming Guide

The DMPAC implements two pipelines within it. A pipeline is combination of several input DMA, compute and output DMA. These two pipelines are:

1. Optical Flow Pipeline
2. Stereo Disparity Pipeline

The two pipelines can operate independent of each other.

In the following sub-sections, the convention for describing register and its bit-fields is as follows:

module\_name -> register\_name -> bit\_field

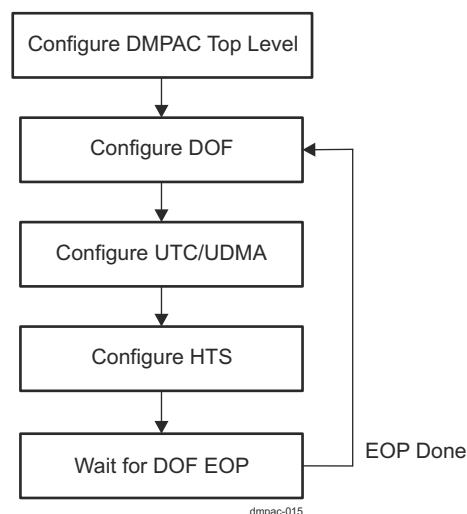
#### CAUTION

No access of any type, either read or write, should be made to the DOF and SDE internal memory space when the respective modules are operationally active (that is, between the initialization and end of pipeline). This would result in an unknown behavior. If the contents of the internal memory space are to be read, for debug purpose, then the respective module needs to be suspended first and only upon full suspension, this memory space should be read out. No writes should ever be done to the internal memory space as it would result in an unknown behavior.

To simplify the programming sequences in this section, the variables in [DMPAC 12bpp Optical Flow Processing Initialization Sequence](#) are defined.

##### 6.10.4.1 DMPAC Optical Flow Initialization Sequence - 12-bit Packed Input Pixel Data

The sequence in [Figure 6-165](#) describes a high level initialization sequence for Optical Flow processing when input images (current and reference) are organized in a fully packed 12-bit format. A pyramid of images is created by the MSC inside VPAC for every frame before it can be fed to the DMPAC Optical Flow processing engine. The sequence below needs to be repeated for every level of such a pyramid. The parameters pertaining to frame resolution, input and output dependencies, data movement and algorithm needs to be updated for every level.


**Figure 6-165. DMPAC 12bpp Optical Flow Processing Initialization Sequence**
**Table 6-169. DMPAC Programming Guide Terminology**

Variable	Equation	Comments
RefTop	$\text{multiple\_of\_2}(\text{vsr\_n}+6)$	/* 6=census tr + hamming dist */
RefCurr	2	
RefBott	$\text{multiple\_of\_2}(\text{vsr\_p}+6+2+1)$	/* 6=census tr + hamming dist; 2=prefetch; 1=adv-pred */
RefThreshold	$\text{RefTop} + \text{RefCurr} + \text{RefBott}$	
CurTop	6	/* 6=census tr + hamming dist */
CurCurr	2	
CurBott	8	/* 6=census tr + hamming dist; 1=adv-pred; 1= round */
CurThreshold	$\text{CurTop} + \text{CurCurr} + \text{CurBott}$	
mf_en	Median Filter Enable	

#### 6.10.4.1.1 Optical Flow 12bb - DMPAC Top Level Configuration

```

Set ENABLE->DMPAC_EN to '1'. // Enable DMPAC Module
Set ENABLE->DOF_EN to '1'. // Enable Optical Flow Processing Module
  
```

#### 6.10.4.1.2 Optical Flow 12bb - UTC Configuration

The Optical Flow engine needs a maximum of 5 input buffers to be fetched by UTC before it can start its processing. It produces one output which need to be sent out into DDR by the UTC.

##### 6.10.4.1.2.1 Reference Frame Growing Window Fetch

The reference frame growing window fetches happen in the granularity of two lines. The UTC needs to be configured to bring two consecutive lines on every event trigger. Only after several DMA event triggers, a sufficient amount of data would have been accumulated, which would trigger DOF operation. The SL2 buffer is sized to hold a maximum 146 lines (assuming  $\text{vsr\_p} = \text{vsp\_n} = 63$ ), so it must wrap around at that granularity. While setting up the destination side transfer parameters, care must be taken to achieve staggering. In other words, two vertically adjacent pixels should come from different SL2 banks. For example, in case the image width is 2048 pixels, this would mean 3KB of data per line. Without any staggering, two vertically adjacent pixels would fall in the same memory bank of DMPAC SL2. This can cause SL2 access problems considering DOF reference frame growing window access pattern. To avoid this situation, a dummy SL2 word (64 bytes) need to be inserted at the end of every line. This will make vertically adjacent pixels to come from different banks. If the



image width is such that staggering happens naturally, the addition of dummy SL2 word does not need to be done.

#### 6.10.4.1.2.2 Current Frame Growing Window Fetch

The current frame growing window fetches happen in the granularity of two lines. The UTC needs to be configured to bring two consecutive lines on every event trigger. Only after several DMA event triggers, a sufficient amount of data would have been accumulated, which would trigger DOF operation. The SL2 buffer is sized to hold a maximum 18 lines, so it must wrap around at that granularity. While setting up the destination side transfer parameters, care must be taken to achieve staggering. In other words, two vertically adjacent pixels should come from different SL2 banks. For example, in case the image width is 2048 pixels, this would mean 3KB of data per line. Without any staggering, two vertically adjacent pixels would fall in the same memory bank (or bank pair) of DMPAC SL2. This can cause SL2 access problems considering DOF current frame growing window access pattern. To avoid this situation, a dummy SL2 word (64 bytes) need to be inserted at the end of every line. This will make vertically adjacent pixels to come from different banks. If the image width is such that staggering happens naturally, the addition of dummy SL2 word does not need to be done.

#### 6.10.4.1.2.3 Temporal Predictor Fetch

There needs to be at least a ping-pong (2 deep) buffer in SL2 for temporal predictor fetch. This fetch is only needed while processing the base layer of the pyramid. This fetch needs to be disabled for other layer processing. Since a paxel level predictor evaluation is performed, only alternate (even) lines needs to be fetched for this buffer from the external memory. The UTC source side fetch parameters need to be programmed appropriately. Every line of this buffer needs to start from a new SL2 word. There cannot be any overlap with the previous word.

#### 6.10.4.1.2.4 Pyramidal Predictor Fetch

There needs to be at least a ping-pong (2 deep) buffer in SL2 for pyramidal predictor fetch. This fetch is not needed while processing the top most layer of the pyramid. This fetch needs to be enabled for other layer processing. Every line of this buffer needs to start from a new SL2 word. There cannot be any overlap with the previous word.

#### 6.10.4.1.2.5 Sparse Optical Flow Binary Map Fetch

There needs to be at least a ping-pong (2 deep) buffer in SL2 for sparse optical flow binary map fetch. This fetch is not needed when sparse processing is turned off. The UTC needs to be configured to bring two consecutive lines of this map on every event trigger. Every line of this buffer needs to start from a new SL2 word. There cannot be any overlap with the previous word.

#### 6.10.4.1.2.6 Flow Vector Output

There needs to be at least a ping-pong (2 deep) buffer in SL2 for flow vector output. The size of the buffer can vary between the sparse and dense optical flow processing. The UTC needs to be configured to send out two consecutive lines of this map on every event trigger. Every line of this buffer will start from a new SL2 word.

#### 6.10.4.1.3 Optical Flow 12bb - HTS Configuration

The HWA0 schedule in HTS is used for the DOF. The HTS programming is as follows:

```
// RFGW Fetch
HTS->DMA0_SCHEDULER_CONTROL->sch_en = 1; // scheduler enable
HTS->DMA0_SCHEDULER_CONTROL->pipeline_num = 0; // Belongs to pipeline 0
HTS->DMA0_SCHEDULER_CONTROL->dma_channel_no = 0; // Assign appropriate DMA channel
HTS->DMA0_HOP->hop = 1;
HTS->DMA0_HOP->hop_thread_count = ImgHeight/2;
HTS->DMA0_PROD0_CONTROL->prod_en = 1;
HTS->DMA0_PROD0_CONTROL->cons_select = 0;
HTS->DMA0_PROD0_BUF_CONTROL->depth = (RefThreshold + 2)/2;
HTS->DMA0_PROD0_BUF_CONTROL->count_dec = 1;

if(RefThreshold > ImgHeight)
{
    HTS->DMA0_PROD0_BUF_CONTROL->threshold = ImgHeight/2;
}
```

```

    if((refBot + 2u) < dofConfig->height)
    {
        HTS->DMA0_PROD0_COUNT->count_preload = (ImgHeight - RefBot - 2)/2;
    }
    else
    {
        HTS->DMA0_PROD0_COUNT->count_preload = 0;
    }
    if(refBot < dofConfig->height)
    {
        HTS->DMA0_PROD0_COUNT->count_postload = RefBot/2 + mf_en
    }
    else
    {
        HTS->DMA0_PROD0_COUNT->count_postload = (ImgHeight - RefCurr)/2 + mf_en
    }
}
else
{
    HTS->DMA0_PROD0_BUF_CONTROL->threshold = (RefThreshold)/2;
    HTS->DMA0_PROD0_COUNT->count_preload = RefTop/2
    HTS->DMA0_PROD0_COUNT->count_postload = RefBot/2 + mf_en
}

// CFGW Fetch
HTS->DMA1_SCHEDULER_CONTROL->sch_en = 1; // scheduler enable
HTS->DMA1_SCHEDULER_CONTROL->pipeline_num = 0; // Belongs to pipeline 0
HTS->DMA1_SCHEDULER_CONTROL->dma_channel_no = 1; // Assign appropriate DMA channel
HTS->DMA1_HOP->hop = 1;
HTS->DMA1_HOP->hop_thread_count = ImgHeight/2;
HTS->DMA1_PROD0_CONTROL->prod_en = 1;
HTS->DMA1_PROD0_CONTROL->cons_select = 0;
HTS->DMA1_PROD0_BUF_CONTROL->depth = (CurThreshold + 2)/2;
HTS->DMA1_PROD0_BUF_CONTROL->count_dec = 1;
HTS->DMA1_PROD0_BUF_CONTROL->threshold = (CurThreshold)/2;
HTS->DMA1_PROD0_COUNT->count_preload = CurTop/2;
HTS->DMA1_PROD0_COUNT->count_postload = CurBot/2 + mf_en;

// Temporal predictor fetch
HTS->DMA2_SCHEDULER_CONTROL->sch_en = 1; // Only for Base pyramid
HTS->DMA2_SCHEDULER_CONTROL->pipeline_num = 0; // Belongs to pipeline 0
HTS->DMA2_SCHEDULER_CONTROL->dma_channel_no = 2; // Assign appropriate DMA channel
HTS->DMA2_HOP->hop = 1;
HTS->DMA2_HOP->hop_thread_count = ImgHeight/2;
HTS->DMA2_PROD0_CONTROL->prod_en = 1;
HTS->DMA2_PROD0_CONTROL->cons_select = 0;
HTS->DMA2_PROD0_BUF_CONTROL->depth = 2;
HTS->DMA2_PROD0_BUF_CONTROL->count_dec = 1;
HTS->DMA2_PROD0_BUF_CONTROL->threshold = 1;
HTS->DMA2_PROD0_COUNT->count_preload = 0;
HTS->DMA2_PROD0_COUNT->count_postload = mf_en;

// Pyramidal predictor fetch
HTS->DMA3_SCHEDULER_CONTROL->sch_en = 1; // Only if Pyramid predictor enabled
HTS->DMA3_SCHEDULER_CONTROL->pipeline_num = 0; // Belongs to pipeline 0
HTS->DMA3_SCHEDULER_CONTROL->dma_channel_no = 3; // Assign appropriate DMA channel
HTS->DMA3_HOP->hop = 1;
HTS->DMA3_HOP->hop_thread_count = ImgHeight/2;
HTS->DMA3_PROD0_CONTROL->prod_en = 1;
HTS->DMA3_PROD0_CONTROL->cons_select = 0;
HTS->DMA3_PROD0_BUF_CONTROL->depth = 2;
HTS->DMA3_PROD0_BUF_CONTROL->count_dec = 1;
HTS->DMA3_PROD0_BUF_CONTROL->threshold = 1;
HTS->DMA3_PROD0_COUNT->count_preload = 0;
HTS->DMA3_PROD0_COUNT->count_postload = mf_en;

// Sparse binary map fetch
HTS->DMA4_SCHEDULER_CONTROL->sch_en = 1; // Only if SOF enabled
HTS->DMA4_SCHEDULER_CONTROL->pipeline_num = 0; // Belongs to pipeline 0
HTS->DMA4_SCHEDULER_CONTROL->dma_channel_no = 4; // Assign appropriate DMA channel
HTS->DMA4_HOP->hop = 1;
HTS->DMA4_HOP->hop_thread_count = ImgHeight/2;
HTS->DMA4_PROD0_CONTROL->prod_en = 1;
HTS->DMA4_PROD0_CONTROL->cons_select = 0;
if(mf_en)

```

```

{
    HTS->DMA4_PROD0_BUF_CONTROL->depth = 2;
}
else
{
    HTS->DMA4_PROD0_BUF_CONTROL->depth = 3;
}
HTS->DMA4_PROD0_BUF_CONTROL->count_dec = 1;
HTS->DMA4_PROD0_BUF_CONTROL->threshold = 1;
HTS->DMA4_PROD0_COUNT->count_preload = 0;
HTS->DMA4_PROD0_COUNT->count_postload = mf_en;

// HWA0 Scheduler Programming
HTS->HWA0_SCHEDULER_CONTROL->sch_en = 1; // scheduler enable
HTS->HWA0_SCHEDULER_CONTROL->pipeline_num = 0; // Belongs to pipeline 0
HTS->HWA0_WDTIMER->watchdog_timer_en = 0; //Activate WD
HTS->HWA0_HOP->hop = 0;
HTS->HWA0_BW_LIMITER->BW_limiter_en = 0;

// HWA0 consumer and producer control
HTS->HWA0_CONS0_CONTROL->cons_en = 1; // Enable RFGW Fetch
HTS->HWA0_CONS1_CONTROL->cons_en = 1; // Enable CFGW Fetch
HTS->HWA0_CONS2_CONTROL->cons_en = 0/1; // 1: Only for Base layer if Temporal predictor enabled
HTS->HWA0_CONS3_CONTROL->cons_en = 0/1; // 1: Only if Pyramidal predictor enabled
HTS->HWA0_CONS4_CONTROL->cons_en = 0/1; // 1: Only if SOF enabled

HTS->HWA0_CONS0_CONTROL->prod_select = 0;
HTS->HWA0_CONS1_CONTROL->prod_select = 0;
HTS->HWA0_CONS2_CONTROL->prod_select = 0;
HTS->HWA0_CONS3_CONTROL->prod_select = 0;
HTS->HWA0_CONS4_CONTROL->prod_select = 0;

HTS->HWA0_PROD0_CONTROL->prod_en = 1; // Enable Producer socket
HTS->HWA0_PROD0_CONTROL->cons_select = 0; // Fixed to DMA
HTS->HWA0_PROD0_BUF_CONTROL->depth = 2;
HTS->HWA0_PROD0_BUF_CONTROL->threshold = 1;
HTS->HWA0_PROD0_BUF_CONTROL->count_dec = 1;
HTS->HWA0_PROD0_COUNT->count_preload = 0;
HTS->HWA0_PROD0_COUNT->count_postload = 0;

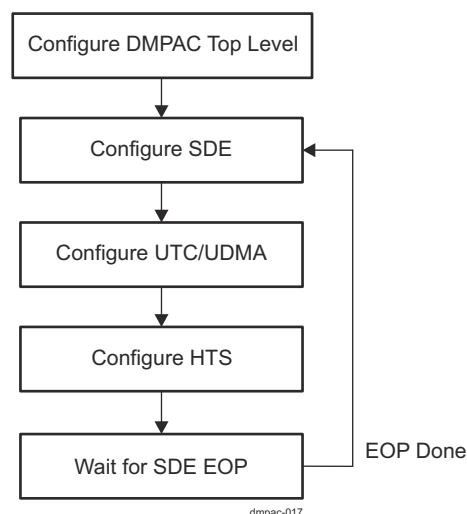
// Flow vector output
HTS->DMA240_SCHEDULER_CONTROL->sch_en = 1; // scheduler enable
HTS->DMA240_SCHEDULER_CONTROL->pipeline_num = 0; // Belongs to pipeline 0
HTS->DMA240_SCHEDULER_CONTROL->dma_channel_no = 5; // Assign appropriate DMA channel
HTS->DMA240_CONS0_CONTROL->cons_en = 1;
HTS->DMA240_CONS0_CONTROL->prod_select = 0;

// Enable Pipeline
HTS->PIPELINE_CONTROL_0->pipe_en = 1; // Enable DOF pipeline# 0

```

#### 6.10.4.2 DMPAC Stereo Disparity Initialization Sequence - 12-bit Packed Input Pixel Data

Figure 6-166 describes a high level initialization sequence for Stereo Disparity processing when input images (both left and right view) are organized in a fully packed 12-bit format. Rectified left and right images are created by the LDC module inside VPAC for every frame before it can be fed to the Stereo Disparity processing engine. The sequence below needs to be repeated for every frame. The parameters pertaining to frame resolution, data movement and algorithm needs to be updated for every frame as needed.



**Figure 6-166. DMPAC 12bpp Stereo Disparity Processing Initialization Sequence**

#### 6.10.4.2.1 Stereo Disparity 12bpp - DMPAC Top Configuration

```

Set ENABLE->DMPAC_EN to '1'. // Enable DMPAC Module
Set ENABLE->SDE_EN to '1'. // Enable Stereo Disparity Processing Module
  
```

If the input data and configuration to DMPAC operation is pre-determined (used for periodic testing), then output data also gets known. In such a situation, complete output buffer comparison can be avoided by enabling PSA on output buffer. The PSA generates a 32-bit CRC value which can be compared to a pre-calculated value, thus avoiding complete output buffer comparison. The SDE module implements a PSA function.

Upon end of pipe, *SDE->PSA\_SIGNATURE->crc\_sts* would reflect the 32-bit CRC value.

#### 6.10.4.2.2 Stereo Disparity 12bpp - DMA Configuration

The Stereo Disparity engine needs 2 input buffers to be fetched by the UTC before it can start its processing. It produces one output which needs to be sent out into DDR by the UTC.

##### 6.10.4.2.2.1 Reference Frame Growing Window Fetch

The reference frame growing window fetches happen in the granularity of 8 lines as the height of the SDE processing block is 8. The UTC needs to be configured to bring 8 consecutive lines on every event trigger. Only after 2 DMA event triggers, a sufficient amount of data (16 lines) would have been accumulated which would trigger the SDE operation. The SL2 buffer is sized to hold a maximum 24 lines, so it must wrap around at that granularity. Although it is not quite mandatory, while setting up the destination side transfer parameters, care must be taken to achieve staggering, only if SL2 size permits. In other words, two vertically adjacent pixels should come from different SL2 banks. For example, in case the image width is 2048 pixels, this would mean 3KB of data per line. Without any staggering, two vertically adjacent pixels would fall in the same memory bank of DMPAC SL2. This can cause SL2 access problems considering SDE reference frame growing window access pattern. To avoid this situation, a dummy SL2 word (64 bytes) needs to be inserted at the end of every line. This will make vertically adjacent pixels to come from different banks. If the image width is such that staggering happens naturally, the addition of a dummy SL2 word does not need to be done.

##### 6.10.4.2.2.2 Base Frame Growing Window Fetch

The base frame growing window fetches happen in the granularity of 8 lines. The UTC needs to be configured to bring 8 consecutive lines on every event trigger. Only after 2 DMA event triggers, a sufficient amount of data would have been accumulated, which would trigger the SDE operation. The SL2 buffer is sized to hold a maximum 24 lines, so it must wrap around at that granularity. Although it is not quite mandatory, while setting up the destination side transfer parameters, care must be taken to achieve staggering, only if SL2 size permits. In other words, two vertically adjacent pixels should come from different SL2 banks. For example, in case the

image width is 2048 pixels, this would mean 3KB of data per line. Without any staggering, two vertically adjacent pixels would fall in the same memory bank (or bank pair) of DMPAC SL2. This can cause SL2 access problems considering SDE base frame growing window access pattern. To avoid this situation, a dummy SL2 word (64 bytes) needs to be inserted at the end of every line. This will make vertically adjacent pixels to come from different banks. If the image width is such that staggering happens naturally, the addition of a dummy SL2 word does not need to be done.

#### 6.10.4.2.2.3 Stereo Disparity Output

There needs to be at least a ping-pong (2 deep) buffer in SL2 for the disparity output. The size of the output can vary based on the location of the processed block in the frame. The height of the block will have one particular value for the first row of blocks in the image. Then, in steady state, the height would have a different value for the central part of the image rows. It would finally have a different height for the last row of blocks. The UTC needs to be configured appropriately to handle this. One possible way is to use 3 linked TR entries in UTC for these three image regions. The UTC needs to send out a block on every event trigger. Every block of this buffer will start from a new SL2 word.

#### 6.10.4.2.3 Stereo Disparity 12bpp - HTS Configuration

The HWA1 scheduler in HTS is used for the SDE. The HTS programming is as follows:

```
// RFGW Fetch
HTS->DMA8_SCHEDULER_CONTROL->sch_en = 1; // scheduler enable
HTS->DMA8_SCHEDULER_CONTROL->pipeline_num = 1; // Belongs to pipeline 1
HTS->DMA8_SCHEDULER_CONTROL->dma_channel_no = 6; // Assign appropriate DMA channel
HTS->DMA8_HOP->hop = 1;
HTS->DMA8_HOP->hop_thread_count = ImgHeight/8;
HTS->DMA8_PROD0_CONTROL->prod_en = 1;
HTS->DMA8_PROD0_CONTROL->cons_select = 0;
HTS->DMA8_PROD0_BUF_CONTROL->depth = 3;
HTS->DMA8_PROD0_BUF_CONTROL->count_dec = 1;
HTS->DMA8_PROD0_BUF_CONTROL->threshold = 2;
HTS->DMA8_PROD0_COUNT->count_preload = 0;
HTS->DMA8_PROD0_COUNT->count_postload = 0;
HTS->DMA8_PA0_CONTROL->pa_enable = 1; // Enable Pattern Adapter
HTS->DMA8_PA0_CONTROL->pa_dec_cntl = 1; // post pattern adaptation, decrement ps count by count_dec
HTS->DMA8_PA0_CONTROL->pa_buf_cntl = 0; // Apply threshold, count_preload and count_postload on
prodcount

psMaxCnt = Imgwidth/64;
if((64 * psMaxCnt) != Imgwidth)
{
    psMaxCnt++;
}

HTS->DMA8_PA0_CONTROL->pa_ps_maxcount = Imgwidth; // Assuming frame width of 2048 and block width
of 64
HTS->DMA8_PA0_CONTROL->pa_cs_maxcount = 2; // Pattern Adapter starts when we have 16 lines in GW
HTS->DMA8_PROD0_BUF_CONTROL->count_dec = 1; // Overwrite 8 line in GW once the entire

// BFGW Fetch
HTS->DMA9_SCHEDULER_CONTROL->sch_en = 1; // scheduler enable
HTS->DMA9_SCHEDULER_CONTROL->pipeline_num = 1; // Belongs to pipeline 1
HTS->DMA9_SCHEDULER_CONTROL->dma_channel_no = 6; // Assign appropriate DMA channel
HTS->DMA9_HOP->hop = 1;
HTS->DMA9_HOP->hop_thread_count = ImgHeight/8;
HTS->DMA9_PROD0_CONTROL->prod_en = 1;
HTS->DMA9_PROD0_CONTROL->cons_select = 0;
HTS->DMA9_PROD0_BUF_CONTROL->depth = 3;
HTS->DMA9_PROD0_BUF_CONTROL->count_dec = 1;
HTS->DMA9_PROD0_BUF_CONTROL->threshold = 2;
HTS->DMA9_PROD0_COUNT->count_preload = 0;
HTS->DMA9_PROD0_COUNT->count_postload = 0;

psMaxCnt = Imgwidth/64;
if((64 * psMaxCnt) != Imgwidth)
{
    psMaxCnt++;
}
```

```

HTS->DMA9_PA0_CONTROL->pa_ps_maxcount = Imgwidth; // Assuming frame width of 2048 and block width
of 64
HTS->DMA9_PA0_CONTROL->pa_cs_maxcount = 2; // Pattern Adapter starts when we have 16 lines in GW
HTS->DMA9_PROD0_BUF_CONTROL->count_dec = 1; // Overwrite 8 line in GW once the entire

// HWA1 Scheduler Programming
HTS->HWA1_SCHEDULER_CONTROL->sch_en = 1; // scheduler enable
HTS->HWA1_SCHEDULER_CONTROL->pipeline_num = 1; // Belongs to pipeline
HTS->HWA1_WDTIMER->watchdog_timer_en = 0; //Activate WD
HTS->HWA1_HOP->hop = 0;
HTS->HWA1_BW_LIMITER->BW_limiter_en = 0;

// HWA0 consumer and producer control
HTS->HWA1_CONS0_CONTROL->cons_en = 1; // Enable RFGW Fetch
HTS->HWA1_CONS1_CONTROL->cons_en = 1; // Enable CFGW Fetch

HTS->HWA1_CONS0_CONTROL->prod_select = 0;
HTS->HWA1_CONS1_CONTROL->prod_select = 0;

HTS->HWA1_PROD0_CONTROL->prod_en = 1; // Enable Producer socket
HTS->HWA1_PROD0_CONTROL->cons_select = 0; // Fixed to DMA
HTS->HWA1_PROD0_BUF_CONTROL->depth = 2;
HTS->HWA1_PROD0_BUF_CONTROL->threshold = 1;
HTS->HWA1_PROD0_BUF_CONTROL->count_dec = 1;
HTS->HWA1_PROD0_COUNT->count_preload = 0;
HTS->HWA1_PROD0_COUNT->count_postload = 0;

// Flow vector output
HTS->DMA256_SCHEDULER_CONTROL->sch_en = 1; // scheduler enable
HTS->DMA256_SCHEDULER_CONTROL->pipeline_num = 0; // Belongs to pipeline 0
HTS->DMA256_SCHEDULER_CONTROL->dma_channel_no = 8; // Assign appropriate DMA channel
HTS->DMA256_CONS0_CONTROL->cons_en = 1;
HTS->DMA256_CONS0_CONTROL->prod_select = 0;

// Enable Pipeline
HTS->PIPELINE_CONTROL_1->pipe_en = 1; // Enable DOF pipeline# 1

```

#### 6.10.4.3 DMPAC End of Pipeline Processing

The DMPAC can have a maximum of 2 active pipelines at any given time. For more details on the end of pipeline processing for both of these pipelines, see *Hardware Accelerator (HWA) Thread Scheduler (HTS)*.

#### 6.10.4.4 DMPAC Debug Restrictions

During Halt request based debug of the SDE module, not all internal memories can be read, if the requirement is to resume or single-step to completion with correct output. Accessing the SDE BPCC memory would corrupt the processing of SDE, once it is read.

This SDE memory can still be accessed to probe the contents and debug. If a single step is required, the test must be run again without probing the memory.

No such restrictions exist for the DOF memories.

Chapter 7

Interprocessor Communication

---



This chapter describes the interprocessor communication modules in the device.

7.1 Mailbox.....	814
7.2 Spinlock.....	825



## 7.1 Mailbox

This chapter describes the mailbox module in the device.

### 7.1.1 Mailbox Overview

Mailbox module serves to facilitate the communication between the various on-chip processors of the device by providing a queued mailbox-interrupt mechanism.

The queued mailbox-interrupt mechanism allows the software to establish a communication channel between two processors (users) through a set of registers and associated interrupt signals.

The device implements the following:

- One system (NAVSS) mailbox module:
  - Number of clusters: 12
  - Used for communication between: Compute Cluster Arm (CC\_ARMSS), MCU\_ARMSS, and PRU\_ICSSG subsystems
  - Reference names: MAILBOX0

Table 7-1 shows the Mailbox allocation across device domains.

**Table 7-1. Mailbox Allocation Across Device Domains**

Instance	Domain		
	WKUP	MCU	MAIN
MAILBOX0	-	-	✓ (NAVSS)

#### 7.1.1.1 Mailbox Features

Each mailbox module supports the following features:

- Parameters configured at design time (see Table 7-2):
  - Number of mailbox clusters
    - Each mailbox cluster has the same configuration
    - Clusters are accessed via separate VBUS regions. Clusters can be treated as mailbox module instances.
  - Number of users
  - Number of mailbox message queues
  - Number of messages (FIFO depth) for each message queue
- 32-bit message width
- Partial writes do not advance the FIFO pointers
- Message reception and queue-not-full notification using interrupts
- Non-intrusive emulation

#### 7.1.1.2 Mailbox Parameters

Table 7-2 shows the configuration of the mailbox module in the device.

**Table 7-2. Mailbox Configuration Parameters**

Module Instance	Parameters			
	Users (Processors)	Clusters <sup>(1)</sup>	Message Queues <sup>(2)</sup>	Number of Messages <sup>(3)</sup>
MAILBOX0	4	12	16	4

(1) Number of mailbox clusters

(2) Number of mailbox message queues (mailboxes per cluster)

(3) Number of messages (FIFO depth) for each message queue

#### 7.1.1.3 Mailbox Not Supported Features

The following features are not supported by the module:

- Hardware protection to prevent users from reading FIFO mailboxes that it doesn't own as receiver or writing to FIFO mailboxes that it doesn't own as sender.

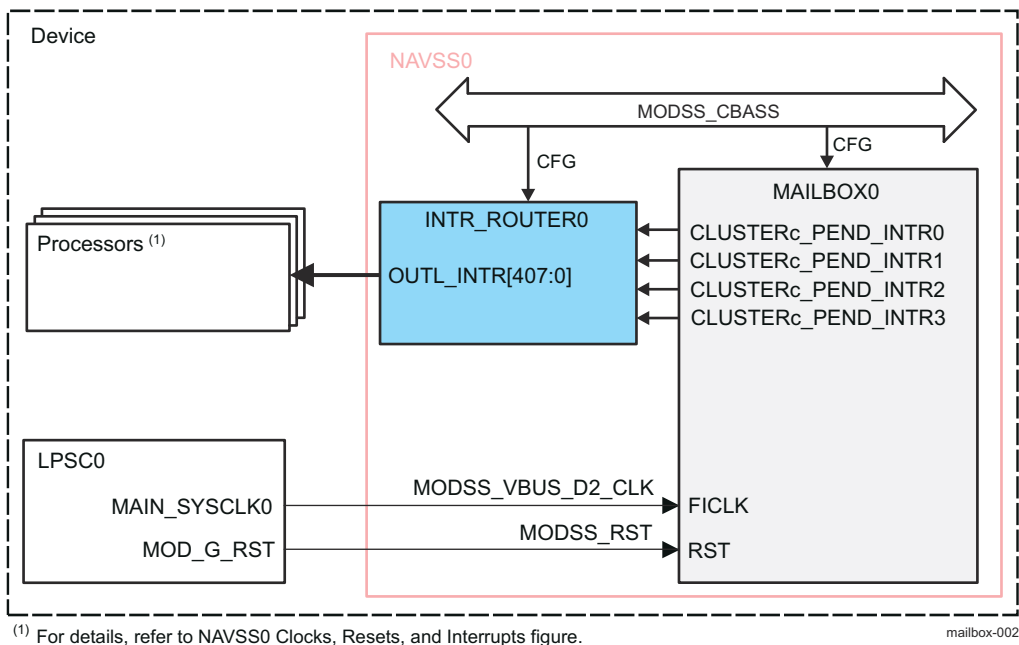


## 7.1.2 Mailbox Integration

This section describes the mailbox integration in the device, including information about clocks, resets, and hardware requests.

### 7.1.2.1 System Mailbox Integration

Figure 7-1 shows the MAILBOX integration in the NAVSS. MAILBOX module contains a number of mailbox clusters. Each cluster generates one interrupt per user (processor).



**Figure 7-1. Mailbox Integration**

c = 0 to 11

Table 7-3 through Table 7-5 summarize the MAILBOX integration in the device.

**Table 7-3. Mailbox Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
MAILBOX0	PSC0	GP	LPSC0	MODSS_CBASS

**Table 7-4. Mailbox Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
MAILBOX0	MAILBOX0_FICLK	MODSS_VBUS_D2_CLK	MAIN_SYSCLK0	MAILBOX0 clock. This clock is used for all interface and functional operations.

Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
MAILBOX0	MAILBOX0_RST	MODSS_RST	LPSC0	MAILBOX0 hardware reset

**Table 7-5. Mailbox Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type

**Table 7-5. Mailbox Hardware Requests (continued)**

MAILBOX0	CLUSTER0_PEND_INT R3	IN_INTR[439]	INTR_ROUTER0	Cluster 0 user 3 interrupt request.	Level
	CLUSTER0_PEND_INT R2	IN_INTR[438]	INTR_ROUTER0	Cluster 0 user 2 interrupt request.	Level
	CLUSTER0_PEND_INT R1	IN_INTR[437]	INTR_ROUTER0	Cluster 0 user 1 interrupt request.	Level
	CLUSTER0_PEND_INT R0	IN_INTR[436]	INTR_ROUTER0	Cluster 0 user 0 interrupt request.	Level
	CLUSTER1_PEND_INT R3	IN_INTR[435]	INTR_ROUTER0	Cluster 1 user 3 interrupt request.	Level
	CLUSTER1_PEND_INT R2	IN_INTR[434]	INTR_ROUTER0	Cluster 1 user 2 interrupt request.	Level
	CLUSTER1_PEND_INT R1	IN_INTR[433]	INTR_ROUTER0	Cluster 1 user 1 interrupt request.	Level
	CLUSTER1_PEND_INT R0	IN_INTR[432]	INTR_ROUTER0	Cluster 1 user 0 interrupt request.	Level
	CLUSTER2_PEND_INT R3	IN_INTR[431]	INTR_ROUTER0	Cluster 2 user 3 interrupt request.	Level
	CLUSTER2_PEND_INT R2	IN_INTR[430]	INTR_ROUTER0	Cluster 2 user 2 interrupt request.	Level
	CLUSTER2_PEND_INT R1	IN_INTR[429]	INTR_ROUTER0	Cluster 2 user 1 interrupt request.	Level
	CLUSTER2_PEND_INT R0	IN_INTR[428]	INTR_ROUTER0	Cluster 2 user 0 interrupt request.	Level
	CLUSTER3_PEND_INT R3	IN_INTR[427]	INTR_ROUTER0	Cluster 3 user 3 interrupt request.	Level
	CLUSTER3_PEND_INT R2	IN_INTR[426]	INTR_ROUTER0	Cluster 3 user 2 interrupt request.	Level
	CLUSTER3_PEND_INT R1	IN_INTR[425]	INTR_ROUTER0	Cluster 3 user 1 interrupt request.	Level
	CLUSTER3_PEND_INT R0	IN_INTR[424]	INTR_ROUTER0	Cluster 3 user 0 interrupt request.	Level
	CLUSTER4_PEND_INT R3	IN_INTR[423]	INTR_ROUTER0	Cluster 4 user 3 interrupt request.	Level
	CLUSTER4_PEND_INT R2	IN_INTR[422]	INTR_ROUTER0	Cluster 4 user 2 interrupt request.	Level
	CLUSTER4_PEND_INT R1	IN_INTR[421]	INTR_ROUTER0	Cluster 4 user 1 interrupt request.	Level
	CLUSTER4_PEND_INT R0	IN_INTR[420]	INTR_ROUTER0	Cluster 4 user 0 interrupt request.	Level
	CLUSTER5_PEND_INT R3	IN_INTR[419]	INTR_ROUTER0	Cluster 5 user 3 interrupt request.	Level
	CLUSTER5_PEND_INT R2	IN_INTR[418]	INTR_ROUTER0	Cluster 5 user 2 interrupt request.	Level
	CLUSTER5_PEND_INT R1	IN_INTR[417]	INTR_ROUTER0	Cluster 5 user 1 interrupt request.	Level
	CLUSTER5_PEND_INT R0	IN_INTR[416]	INTR_ROUTER0	Cluster 5 user 0 interrupt request.	Level
	CLUSTER6_PEND_INT R3	IN_INTR[415]	INTR_ROUTER0	Cluster 6 user 3 interrupt request.	Level
	CLUSTER6_PEND_INT R2	IN_INTR[414]	INTR_ROUTER0	Cluster 6 user 2 interrupt request.	Level
	CLUSTER6_PEND_INT R1	IN_INTR[413]	INTR_ROUTER0	Cluster 6 user 1 interrupt request.	Level

**Table 7-5. Mailbox Hardware Requests (continued)**

CLUSTER6_PEND_INT R0	IN_INTR[412]	INTR_ROUTER0	Cluster 6 user 0 interrupt request.	Level
CLUSTER7_PEND_INT R3	IN_INTR[411]	INTR_ROUTER0	Cluster 7 user 3 interrupt request.	Level
CLUSTER7_PEND_INT R2	IN_INTR[410]	INTR_ROUTER0	Cluster 7 user 2 interrupt request.	Level
CLUSTER7_PEND_INT R1	IN_INTR[409]	INTR_ROUTER0	Cluster 7 user 1 interrupt request.	Level
CLUSTER7_PEND_INT R0	IN_INTR[408]	INTR_ROUTER0	Cluster 7 user 0 interrupt request.	Level
CLUSTER8_PEND_INT R3	IN_INTR[407]	INTR_ROUTER0	Cluster 8 user 3 interrupt request.	Level
CLUSTER8_PEND_INT R2	IN_INTR[406]	INTR_ROUTER0	Cluster 8 user 2 interrupt request.	Level
CLUSTER8_PEND_INT R1	IN_INTR[405]	INTR_ROUTER0	Cluster 8 user 1 interrupt request.	Level
CLUSTER8_PEND_INT R0	IN_INTR[404]	INTR_ROUTER0	Cluster 8 user 0 interrupt request.	Level
CLUSTER9_PEND_INT R3	IN_INTR[403]	INTR_ROUTER0	Cluster 9 user 3 interrupt request.	Level
CLUSTER9_PEND_INT R2	IN_INTR[402]	INTR_ROUTER0	Cluster 9 user 2 interrupt request.	Level
CLUSTER9_PEND_INT R1	IN_INTR[401]	INTR_ROUTER0	Cluster 9 user 1 interrupt request.	Level
CLUSTER9_PEND_INT R0	IN_INTR[400]	INTR_ROUTER0	Cluster 9 user 0 interrupt request.	Level
CLUSTER10_PEND_IN TR3	IN_INTR[399]	INTR_ROUTER0	Cluster 10 user 3 interrupt request.	Level
CLUSTER10_PEND_IN TR2	IN_INTR[398]	INTR_ROUTER0	Cluster 10 user 2 interrupt request.	Level
CLUSTER10_PEND_IN TR1	IN_INTR[397]	INTR_ROUTER0	Cluster 10 user 1 interrupt request.	Level
CLUSTER10_PEND_IN TR0	IN_INTR[396]	INTR_ROUTER0	Cluster 10 user 0 interrupt request.	Level
CLUSTER11_PEND_IN TR3	IN_INTR[395]	INTR_ROUTER0	Cluster 11 user 3 interrupt request.	Level
CLUSTER11_PEND_IN TR2	IN_INTR[394]	INTR_ROUTER0	Cluster 11 user 2 interrupt request.	Level
CLUSTER11_PEND_IN TR1	IN_INTR[393]	INTR_ROUTER0	Cluster 11 user 1 interrupt request.	Level
CLUSTER11_PEND_IN TR0	IN_INTR[392]	INTR_ROUTER0	Cluster 11 user 0 interrupt request.	Level

**DMA Events**

Module Instance	Module DMA Event	Destination DMA Event Input	Destination	Description	Type
MAILBOX0	-	-	-	No PDMA channels to external DMA engines	-

---

**Note**

For more information on the interconnects, see *System Interconnect*.

For more information on the power, reset, and clock management, see the corresponding sections within *Device Configuration*.

---

---

**Note**

For information about interrupt source description, see *Mailbox Interrupt Requests*.

---

### 7.1.3 Mailbox Functional Description

The mailbox module provides a means of communication through message queues among the users. The individual mailbox modules, or FIFOs, can associate (or de-associate) with any of the processors using the MAILBOX\_IRQ\_ENABLE\_SET\_j (or MAILBOX\_IRQ\_ENABLE\_CLR\_j) register.

#### Note

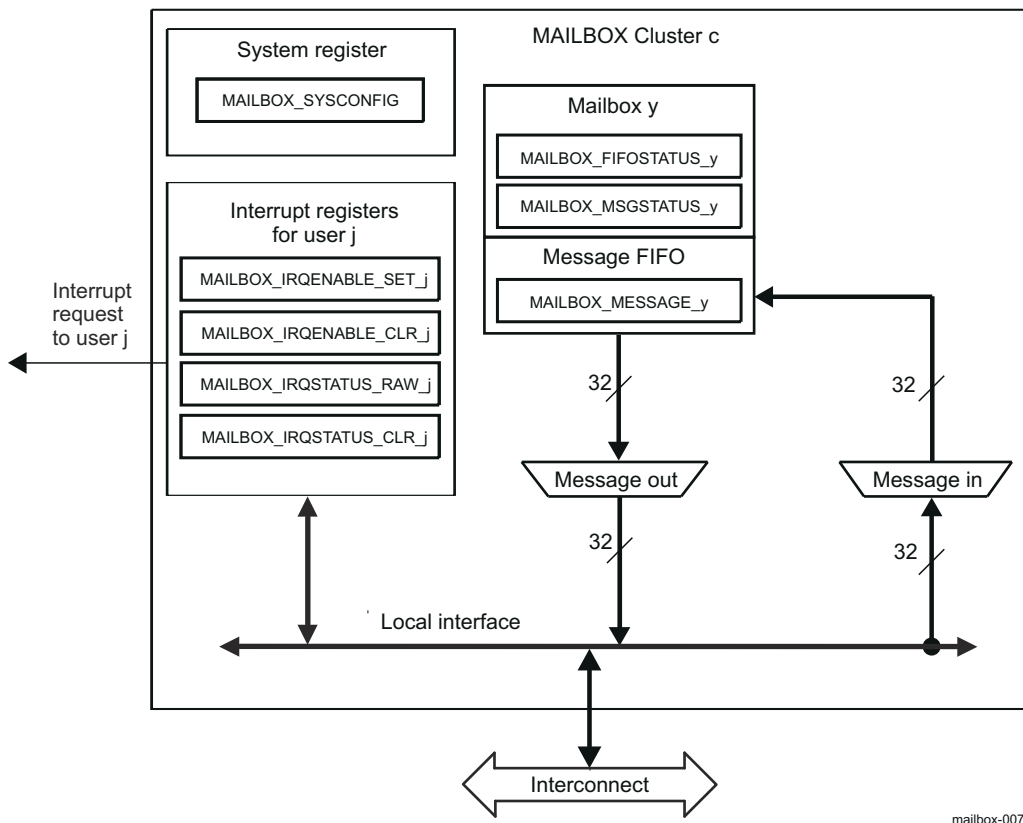
It is software responsibility to select a user by mapping (via INTR\_ROUTER0) the corresponding mailbox interrupt to the interrupt controller of the appropriate processor subsystem.

Each user has a dedicated interrupt signal from the corresponding mailbox module instance and dedicated interrupt enabling and status registers.

Each MAILBOX\_IRQ\_STATUS\_RAW\_j/MAILBOX\_IRQ\_STATUS\_CLR\_j interrupt status register corresponds to a particular user.

#### 7.1.3.1 Mailbox Block Diagram

Figure 7-2 shows the mailbox internal block diagram.



**Figure 7-2. Mailbox Block Diagram**

c = 0 to 11

j = 0 to 3

y = 0 to 15

### 7.1.3.2 Mailbox Software Reset

The mailbox module supports a software reset through the MAILBOX\_SYSCONFIG[0] SOFTRESET bit. Setting this bit to 1 enables an active software reset that is functionally equivalent to a hardware reset. Reading the MAILBOX\_SYSCONFIG[0] SOFTRESET bit gives the status of the software reset:

- Read 1: the software reset is on-going.
- Read 0: the software reset is complete.

The software must ensure that the software reset completes before doing mailbox operations.

### 7.1.3.3 Mailbox Power Management

The clkstop\_idle will be asserted if all of the following are true:

- The VBUSP interface is inactive
- All mailboxes are empty
- There are no enabled events or outstanding interrupts

An enabled event means there is pending action required from one of the users, and it means one or more mailboxes still expect data.

### 7.1.3.4 Mailbox Interrupt Requests

An interrupt request allows the user of the mailbox to be notified when a message is received or when the message queue is not full. There is one interrupt per user.

Table 7-6 lists the event flags, and their mask, that can cause module interrupts.

**Table 7-6. Interrupt Events**

Non-Maskable Event Flag <sup>(1)</sup>	Maskable Event Flag	Event Mask Bit	Event Unmask Bit	Description
MAILBOX_IRQ_STATUS_RAW_j[0+y*2]	MAILBOX_IRQ_STATUS_CLR_j[0+y*2]	MAILBOX_IRQ_ENABLE_CLR_j[0+y*2]	MAILBOX_IRQ_ENABLE_SET_j[0+y*2]	Mailbox y receives a new message.
NEWMSGSTATUSMBY	NEWMSG_STATUSMBY	NEWMSG_STATUSMBY	NEWMSG_STATUSMBY	
MAILBOX_IRQ_STATUS_RAW_j[1+y*2]	MAILBOX_IRQ_STATUS_CLR_j[1+y*2]	MAILBOX_IRQ_ENABLE_CLR_j[1+y*2]	MAILBOX_IRQ_ENABLE_SET_j[1+y*2]	Mailbox y message queue is not full.
NOTFULLSTATUSMBY	NOTFULLSTATUSMBY	NOTFULLSTATUSMBY	NOTFULLSTATUSMBY	

(1) MAILBOX\_IRQ\_STATUS\_RAW\_j register is mostly used for debug purposes.

#### CAUTION

Once an event generating the interrupt request has been processed by the software, it must be cleared by writing a logical 1 in the corresponding bit of the MAILBOX\_IRQ\_STATUS\_CLR\_j register.

Writing a logical 1 in a bit of the MAILBOX\_IRQ\_STATUS\_CLR\_j register will also clear to 0 the corresponding bit in the appropriate MAILBOX\_IRQ\_STATUS\_RAW\_j register.

An event can generate an interrupt request when a logical 1 is written to the corresponding unmask bit in the MAILBOX\_IRQ\_ENABLE\_SET\_j register. Events are reported in the appropriate MAILBOX\_IRQ\_STATUS\_CLR\_j and MAILBOX\_IRQ\_STATUS\_RAW\_j registers.

An event stops generating interrupt requests when a logical 1 is written to the corresponding mask bit in the MAILBOX\_IRQ\_ENABLE\_CLR\_j register. Events are only reported in the appropriate MAILBOX\_IRQ\_STATUS\_RAW\_j register.

In case of the MAILBOX\_IRQ\_STATUS\_RAW\_j register, the event is reported in the corresponding bit even if the interrupt request generation is disabled for this event.

### 7.1.3.5 Mailbox Assignment

#### 7.1.3.5.1 Description

To assign a receiver to a mailbox, set the new message interrupt enable bit corresponding to the desired mailbox in the MAILBOX\_IRQ\_ENABLE\_SET\_j register. The receiver reads the MAILBOX\_MESSAGE\_y register to retrieve a message from the mailbox.

An alternate method for the receiver that does not use the interrupts is to poll the MAILBOX\_FIFO\_STATUS\_y and/or MAILBOX\_MSG\_STATUS\_y registers to know when to send or retrieve a message to or from the mailbox. This method does not require assigning a receiver to a mailbox. Because this method does not include the explicit assignment of the mailbox, the software must avoid having multiple receivers use the same mailbox, which can result in incoherency.

To assign a sender to a mailbox, set the queue-not-full interrupt enable bit of the desired mailbox in the MAILBOX\_IRQ\_ENABLE\_SET\_j register, where *u* is the number of the sending user. However, direct allocation of a mailbox to a sender is not recommended because it can cause the sending processor to be constantly interrupted.

It is recommended that register polling be used to:

- Check the status of either the MAILBOX\_FIFO\_STATUS\_y or MAILBOX\_MSG\_STATUS\_y registers
- Write the message to the corresponding MAILBOX\_MESSAGE\_y register, if space is available.

The sender might use the queue-not-full interrupt when the initial mailbox status check indicates the mailbox is full. In this case, the sender can enable the queue-not-full interrupt for its mailbox in the appropriate MAILBOX\_IRQ\_ENABLE\_SET\_j register. This allows the sender to be notified by interrupt only when a FIFO queue has at least one available entry.

Reading the MAILBOX\_IRQ\_STATUS\_CLR\_j register determines the status of the new message and the queue-not-full interrupts for a particular user. Writing 1 to the corresponding bit in the MAILBOX\_IRQ\_STATUS\_CLR\_j register acknowledges, and subsequently clears, an interrupt.

#### CAUTION

Assigning multiple senders or multiple receivers to the same mailbox is not recommended.

### 7.1.3.6 Sending and Receiving Messages

#### 7.1.3.6.1 Description

When a 32-bit message is written to the MAILBOX\_MESSAGE\_y register, the message is appended into the FIFO queue. This queue holds four messages. If the queue is full, the message is discarded.

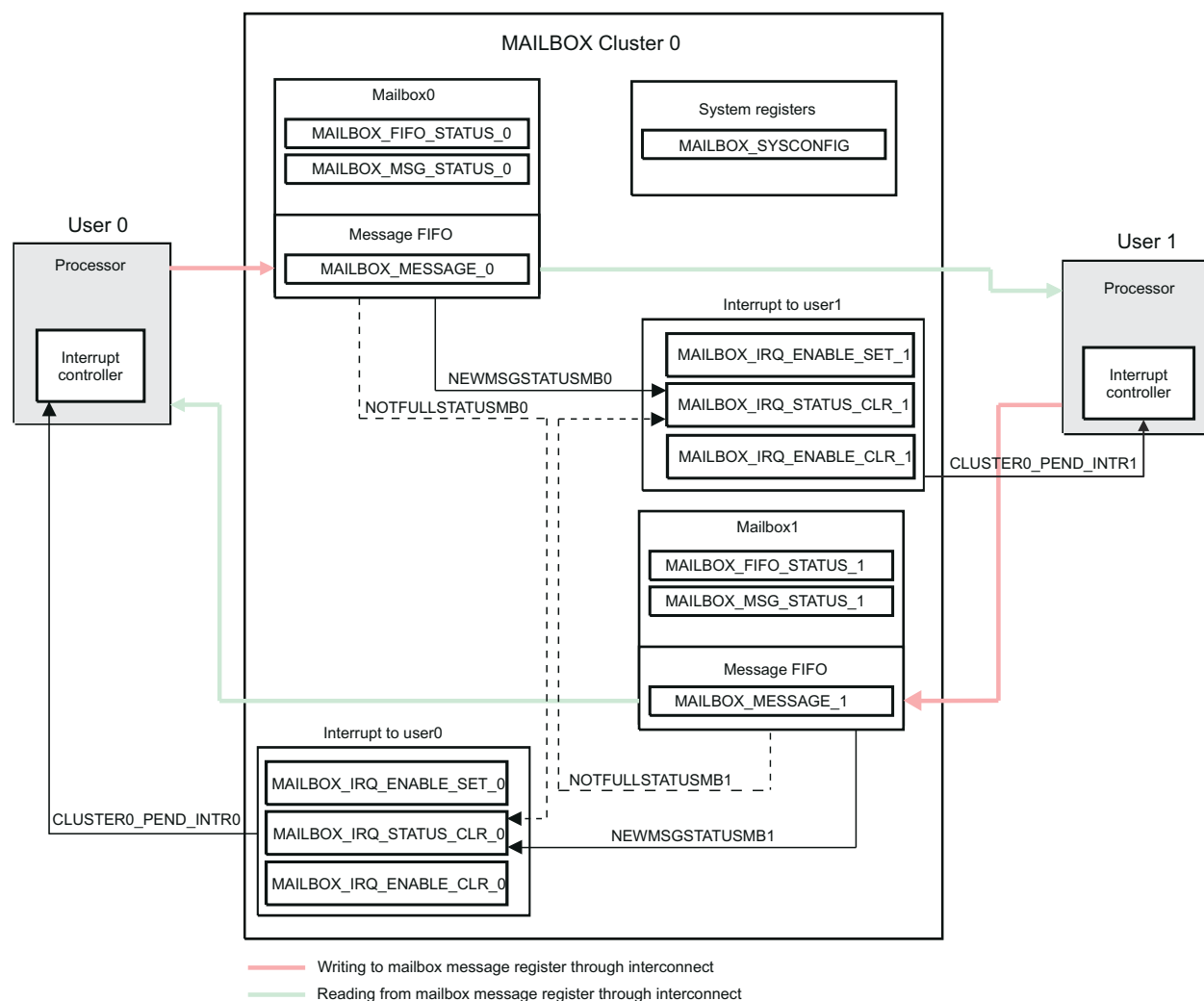
Queue overflow can be avoided by first reading the MAILBOX\_FIFO\_STATUS\_y register to check that the mailbox message queue is not full before writing a new message to it.

Reading the MAILBOX\_MESSAGE\_y register returns the message at the beginning of the FIFO queue and removes it from the queue. If the FIFO queue is empty when the MAILBOX\_MESSAGE\_y register is read, the value 0 is returned.

The new message interrupt is asserted when at least one message is in the mailbox message FIFO queue. To determine the number of messages in the mailbox message FIFO queue, read the MAILBOX\_MSG\_STATUS\_y register.

### 7.1.3.7 Example of Communication

Figure 7-3 shows an example of communication between two processors.



mailbox-008

**Figure 7-3. Example of Communication**



## 7.1.4 Mailbox Programming Guide

### 7.1.4.1 Mailbox Low-level Programming Models

This section covers the low-level hardware programming sequences for configuration and usage of the mailbox module.

#### 7.1.4.1.1 Global Initialization

##### 7.1.4.1.1.1 Surrounding Modules Global Initialization

This section identifies the requirements of initializing the surrounding modules when the mailbox module is to be used for the first time after a device reset. This initialization of surrounding modules is based on the integration of the mailbox.

See [Section 7.1.2, Mailbox Integration](#), for further information.

**Table 7-7. Global Initialization of Surrounding Modules for MAILBOX**

Surrounding Modules	Comments
LPSC	The MAILBOX functional and interface clock must be enabled.
Interrupt Controllers	Processor's (user's) interrupt controller must be configured to enable the interrupt request generation.
INTR_ROUTER0	INTR_ROUTER0 configuration must be done to allow module IRQs to be mapped to required INTC input. For more information see <i>Interrupt Routers</i> .

#### 7.1.4.1.1.2 Mailbox Global Initialization

##### 7.1.4.1.1.2.1 Main Sequence - Mailbox Global Initialization

This procedure initializes the mailbox module after a power-on or software reset.

**Table 7-8. Mailbox Global Initialization**

Step	Register/Bit Field/Programming Model	Value
Perform a software reset	MAILBOX_SYSCONFIG[0] SOFT_RESET	0x1
Wait until reset is complete	MAILBOX_SYSCONFIG[0] SOFT_RESET	=0x0

#### 7.1.4.1.2 Mailbox Operational Modes Configuration

##### 7.1.4.1.2.1 Mailbox Processing modes

##### 7.1.4.1.2.1.1 Main Sequence - Sending a Message (Polling Method)

**Table 7-9. Sending a Message (Polling Method)**

Step	Register/Bit Field/Programming Model	Value
IF: Is FIFO full?	MAILBOX_FIFO_STATUS_y[0] FULL	=0x1
Wait until at least one message slot is available	MAILBOX_FIFO_STATUS_y[0] FULL	=0x0
<b>ELSE</b>		
Write message	MAILBOX_MESSAGE_y[31:0] VALUE	0x-
<b>ENDIF</b>		

##### 7.1.4.1.2.1.2 Main Sequence - Sending a Message (Interrupt Method)

**Table 7-10. Sending a Message (Interrupt Method)**

Step	Register/Bit Field/Programming Model	Value
IF: Is FIFO full?	MAILBOX_FIFO_STATUS_y[0] FULL	=0x1
Enable interrupt event	MAILBOX_IRQ_ENABLE_SET_j[1+ y*2]	0x1
User (processor) can perform another task until interrupt occurs See <a href="#">Section 7.1.4.1.3.1</a> for interrupt handling in sending mode		
<b>ELSE</b>		

**Table 7-10. Sending a Message (Interrupt Method) (continued)**

Step	Register/Bit Field/Programming Model	Value
Write message	MAILBOX_MESSAGE_y[31:0] VALUE	0x-
<b>ENDIF</b>		

**7.1.4.1.2.1.3 Main Sequence - Receiving a Message (Polling Method)****Table 7-11. Receiving a Message (Polling Method)**

Step	Register/Bit Field/Programming Model	Value
IF: Number of messages is not equal to 0	MAILBOX_MSG_STATUS_y[2:0] NUM_MESSAGES	≠0x0
Read message	MAILBOX_MESSAGE_y[31:0] VALUE	0x-
<b>ENDIF</b>		

**7.1.4.1.2.1.4 Main Sequence - Receiving a Message (Interrupt Method)****Table 7-12. Receiving a Message (Interrupt Method)**

Step	Register/Bit Field/Programming Model	Value
Enable interrupt event	MAILBOX_IRQ_ENABLE_SET_j[0 + y*2]	0x1
User (processor) can perform another task until interrupt occurs See <a href="#">Section 7.1.4.1.3.2</a> for interrupt handling in receiving mode		

**7.1.4.1.3 Mailbox Events Servicing****7.1.4.1.3.1 Events Servicing in Sending Mode**

[Table 7-13](#) describes the events servicing in sending mode.

**Table 7-13. Events Servicing in Sending Mode**

Step	Register/Bit Field/Programming Model	Value
Read interrupt status bit	MAILBOX_IRQ_STATUS_CLR_j[1 + y*2]	0x1
Write message	MAILBOX_MESSAGE_y[31:0] VALUE	0x--
Write 1 to acknowledge interrupt	MAILBOX_IRQ_STATUS_CLR_j[1 + y*2]	0x1

**7.1.4.1.3.2 Events Servicing in Receiving Mode**

[Table 7-14](#) describes the events servicing in receiving mode.

**Table 7-14. Events Servicing in Receiving Mode**

Step	Register/Bit Field/Programming Model	Value
Read interrupt status bit	MAILBOX_IRQ_STATUS_CLR_j[0 + y*2]	0x1
IF: Number of messages is not equal to 0?	MAILBOX_MSG_STATUS_y[2:0] NUM_MESSAGES	≠0x0
Read message	MAILBOX_MESSAGE_y[31:0] VALUE	0x--
<b>ELSE</b>		
Write 1 to acknowledge interrupt	MAILBOX_IRQ_STATUS_CLR_j[0 + y*2]	0x1
<b>ENDIF</b>		

## 7.2 Spinlock

This chapter describes the Spinlock module of the device.

### 7.2.1 Spinlock Overview

The Spinlock module provides hardware assistance for synchronizing the processes running on multiple processors in the device.

The Spinlock module implements 256 spinlocks (or hardware semaphores), which provide an efficient way to perform a lock operation of a device resource using a single read-access, avoiding the need of a read-modify-write bus transfer that the programmable cores are not capable of.

Figure 7-4 shows an overview of the Spinlock module.

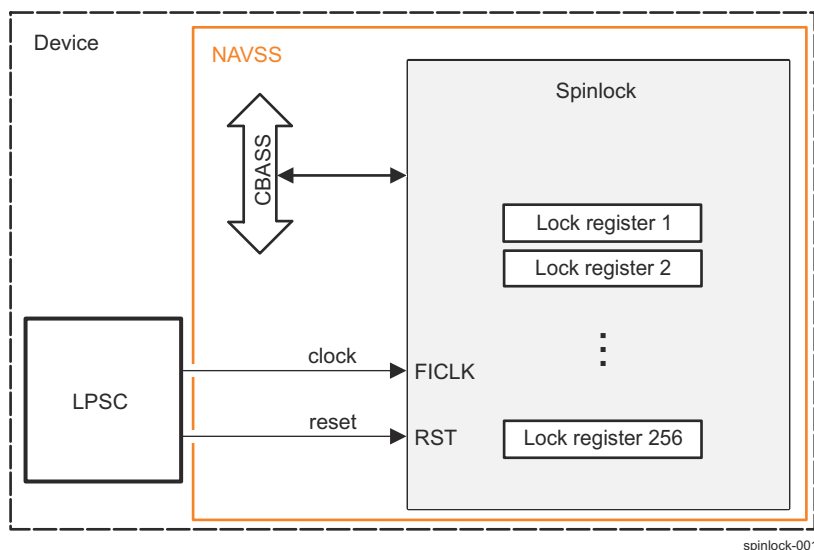


Figure 7-4. Spinlock Overview

Table 7-15. Spinlock Allocation Across Device Domains

Instance	Domain		
	WKUP	MCU	MAIN
SPINLOCK0	-	-	✓ (NAVSS)

#### 7.2.1.1 Spinlock Not Supported Features

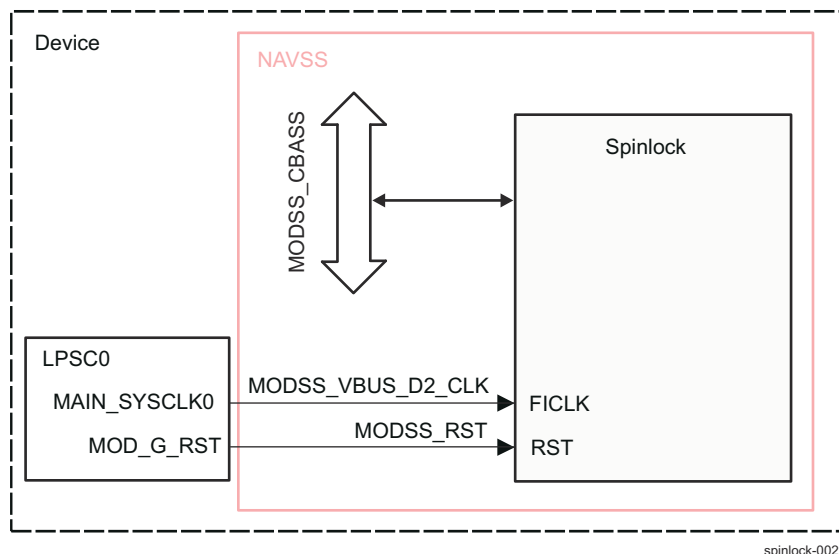
The following features are not supported by the module:

- Ownership enforcement. There is no support to ensure that a lock register is locked and unlocked by the same process
- There is no support for checking that the same VBUS initiator that acquired the lock is the one that is freeing the lock
- There is no support for fairness or congestion control.

## 7.2.2 Spinlock Integration

This section describes module integration in the device, including information about clocks, resets, and hardware requests.

Figure 7-5 shows the Spinlock integration.



**Figure 7-5. Spinlock Integration**

Table 7-16 and Table 7-17 summarize the integration of the module in the device.

**Table 7-16. Spinlock Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
SPINLOCK0	PSC0	GP	LPSC0	MODSS_CBASS

**Table 7-17. Spinlock Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
SPINLOCK0	SPINLOCK0_FICLK	MODSS_VBUS_D2_CLK	MAIN_SYSCLK0	Spinlock clock. This clock is used for all interface and functional operations.
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
SPINLOCK0	SPINLOCK0_RST	MODSS_RST	LPSC0	Spinlock hardware reset

**Table 7-18. Spinlock Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
SPINLOCK0	-	-	-	No interrupts to external processors	-
DMA Events					
Module Instance	Module DMA Event	Destination DMA Event Input	Destination	Description	Type
SPINLOCK0	-	-	-	No PDMA channels to external DMA engines	-

### 7.2.3 Spinlock Functional Description

#### 7.2.3.1 Spinlock Software Reset

The Spinlock module can be reset by software through the SPINLOCK\_SYSCONFIG[1] SOFTRESET bit. Setting this bit to 1 enables an active software reset that is functionally equivalent to a hardware reset. Note that reading back the value of SPINLOCK\_SYSCONFIG[1] SOFTRESET bit will always return 0.

#### 7.2.3.2 Spinlock Power Management

The Spinlock module supports the Power Idle interface. Software must arrange to only power off the Spinlock module when no locks would be lost. In general, the steps to powering down the Spinlock module are:

1. Software must check that all controllers that may be using the Spinlock module are either:
  - a. Already powered off (all in-use flags are 0)
  - b. Notified that Spinlock is not available and the notification is acknowledged
2. If desired, check that no locks are currently held in the Spinlock module. The status of each bank of 32 locks can be read from the SPINLOCK\_SYSTATUS register. If any locks are held, they are orphaned because they are not held by any controller that is still active. Alternatively, you may decide to wait a timeout period to allow any active controller to clean up its locks before powering down.

The Spinlock module may now be powered off.

#### 7.2.3.3 About Spinlocks

Spinlocks are present to solve the need for synchronization and mutual exclusion between heterogeneous processors and those not operating under a single, shared operating system. There is no alternative mechanism to accomplish these operations between processors in separate subsystems.

Spinlocks are not the best way to synchronize between tasks or threads on one CPU. Instead, spinlocks are for use in synchronization between different subsystems in the device that don't have any other means of hardware-based synchronization.

Spinlocks do not solve all system synchronization issues. They have limited applicability and should be used with care to implement higher level synchronization protocols.

A spinlock is appropriate for mutual exclusion for access to a shared data structure. It should be used only when:

1. The time to hold the lock is predictable and small (for example, a maximum hold time of less than 200 CPU cycles may be acceptable).
2. The locking task cannot be preempted, suspended, or interrupted while holding the lock (this would make the hold time large and unpredictable).
3. The lock is lightly contended, that is the chance of any other process (or processor) trying to acquire the lock while it is held is small.

If these conditions are met, then the locking code can retry a failed attempt to acquire the lock until success.

If the conditions are not met, then a spinlock is not a good candidate. One alternative is to use a spinlock for critical section control (engineered to meet the conditions) to implement a higher level semaphore that can support preemption, notification, timeout or other higher level properties.

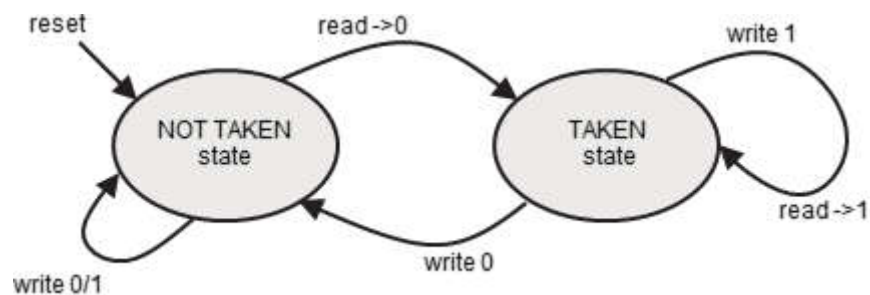
#### 7.2.3.4 Spinlock Functional Operation

The Spinlock module supports 256 spinlocks. It accepts only a single command at a time and processes the command fully before accepting the next command. A lock is requested by reading the SPINLOCK\_LOCK\_REG\_i[0] TAKEN bit. There are two states: Taken (SPINLOCK\_LOCK\_REG\_y[0] TAKEN = 1) or Not Taken (SPINLOCK\_LOCK\_REG\_y[0] TAKEN = 0), where y = 0h to FFh.

When the status of lock y (where y = 0 to 255) is Not Taken (free), a read from the SPINLOCK\_LOCK\_REG\_y register returns 0 and sets the lock to Taken (locked). When the status of lock y is Taken, a read returns 1 and does not change the state of the lock.

A write to the SPINLOCK\_LOCK\_REG\_y register does not change the state of lock, unless when writing 0 when the lock is in Taken state. By doing this, the requester frees the lock.

Figure 7-6 shows the SPINLOCK\_LOCK\_REG\_y register state diagram.



**Figure 7-6. Lock Register State Diagram**

#### Note

- There is no support to ensure that a lock register is locked and unlocked by the same process. This must be ensured in software.
- There is no support to check that the same initiator that acquired the lock is the one that is freeing the lock.

## 7.2.4 Spinlock Programming Guide

### 7.2.4.1 Spinlock Low-level Programming Models

This section covers the low-level hardware programming sequences for configuration and usage of the module.

#### 7.2.4.1.1 Surrounding Modules Global Initialization

This procedure initializes the surrounding modules when the Spinlock module is used for the first time after a device reset.

**Table 7-19. Global Initialization of Surrounding Modules**

Surrounding Modules	Comments
LPSC	Spinlock functional and interface clock must be enabled. For more information, see <i>Clocking</i>

#### 7.2.4.1.2 Basic Spinlock Operations

The main Spinlock operations are:

- Clear all the Taken spinlocks by writing 0 to SPINLOCK\_LOCK\_REG\_y (only after a system bug recovery)
- Take a spinlock by reading the SPINLOCK\_LOCK\_REG\_i[0] TAKEN bit
- Release spinlock by writing 0 to SPINLOCK\_LOCK\_REG\_i[0] TAKEN bit

##### 7.2.4.1.2.1 Spinlocks Clearing After a System Bug Recovery

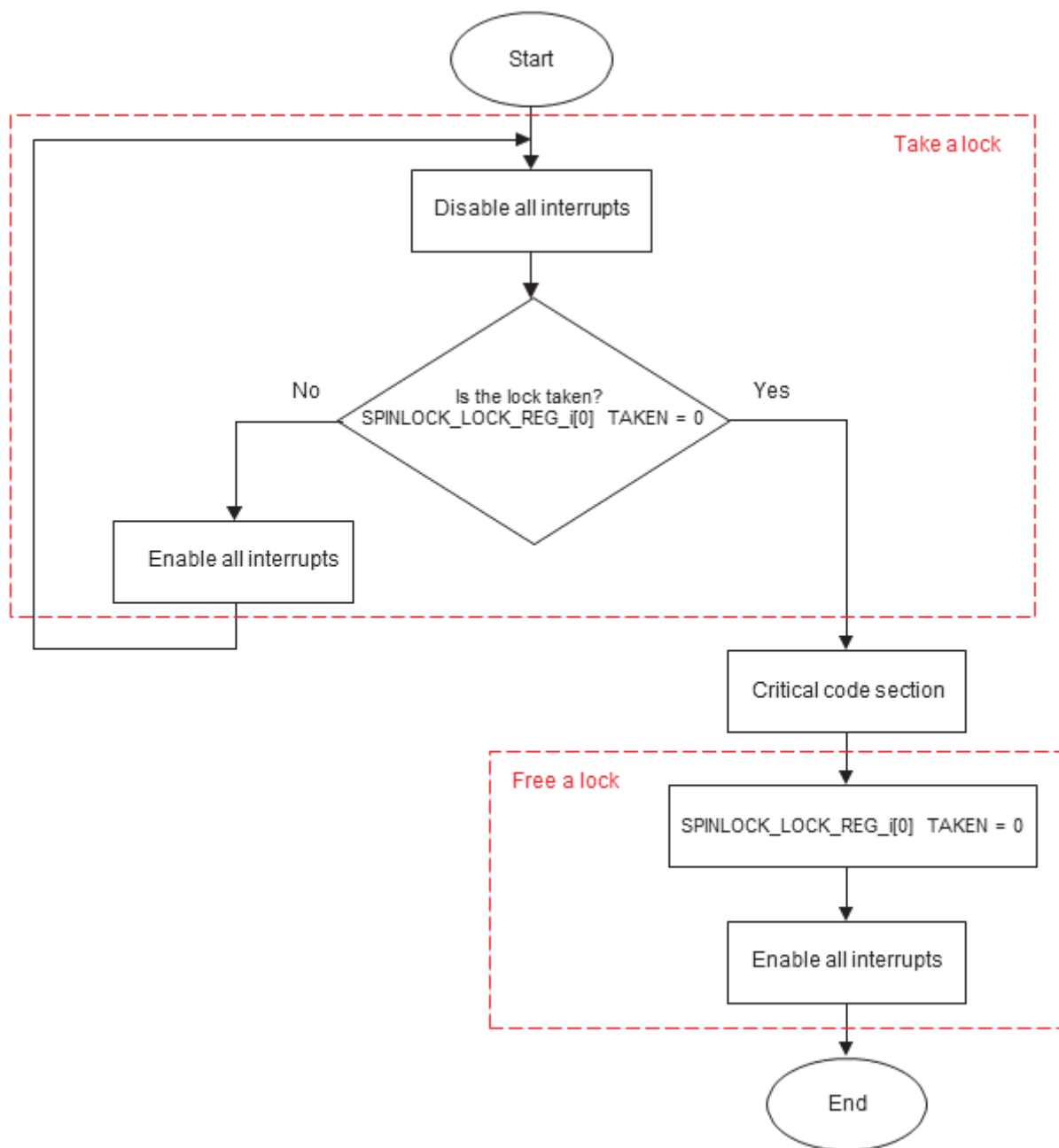
Module initialization (after reset) is not needed, except after system bug recovery. The following table presents the Spinlock initialization after a system bug recovery. Software should store 0 into each of the SPINLOCK\_LOCK\_REG\_y registers at system startup to insure that all locks are initialized to Not Taken.

**Table 7-20. Spinlock System Bug Recovery**

Step	Register	Value
IF: SPINLOCK_SYSTATUS[0] IU0 = 1?	SPINLOCK_SYSTATUS[0] IU0	=1
Free the 256 locks	SPINLOCK_LOCK_REG_y[0] TAKEN (y = 0 to 255)	0x0
END		

##### 7.2.4.1.2.2 Take and Release Spinlock

This procedure configures the take and release (free) operations for the Spinlock module. A spinlock should only be held with interrupts disabled. So, before attempting to obtain the spinlock, software must disable interrupts. Then it must read the SPINLOCK\_LOCK\_REG\_y[0] TAKEN bit to attempt to obtain the lock. If it succeeds, it must proceed directly through the critical section then unlock and re-enable interrupts. If the acquisition attempt fails, the acquisition must be reattempted. To prevent unknown interrupt disabled time, interrupts must be re-enabled and then disabled before reattempting to acquire the lock. [Figure 7-7](#) shows the described above procedure.


**Figure 7-7. Take and Release Spinlock**
**Table 7-21. Register Call Summary**

Register Name
SPINLOCK_LOCK_REG_y[0] TAKEN

**Table 7-22. Subprocess Call Summary**

Subprocess Name	Description
Disable (Mask) All Interrupts	For information about disabling/enabling all interrupts in an Arm® processor, refer to Arm <i>Technical Reference Manual</i> , available at <a href="http://infocenter.arm.com/help/index.jsp">infocenter.arm.com/help/index.jsp</a> .
Enable (Unmask) All Interrupts	For information about disabling/enabling all interrupts in other processors, refer to the corresponding processor chapter.





This chapter describes the memory controllers in the device.

<b>8.1 Multicore Shared Memory Controller (MSMC)</b>	<b>832</b>
<b>8.2 DDR Subsystem (DDRSS)</b>	<b>848</b>
<b>8.3 Virtualization Subsystem (VirtSS)</b>	<b>902</b>
<b>8.4 Region-based Address Translation (RAT) Module</b>	<b>930</b>

## 8.1 Multicore Shared Memory Controller (MSMC)

This section describes the Multicore Shared Memory Controller (MSMC) for the device.

### 8.1.1 MSMC Overview

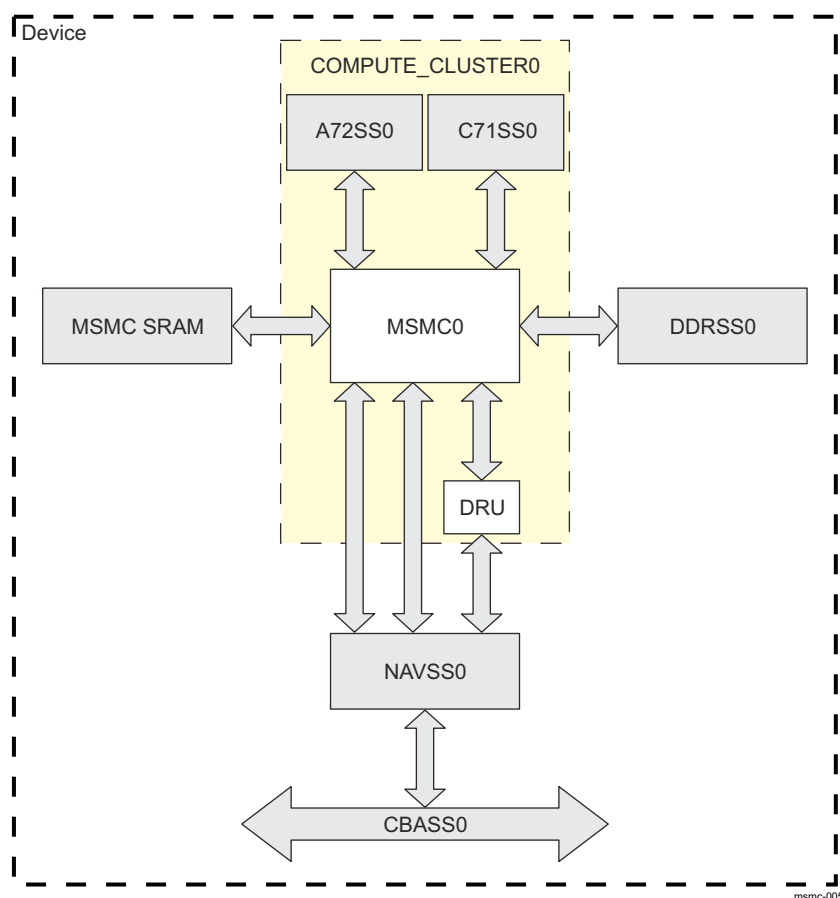
The Multicore Shared Memory Controller (MSMC) forms the heart of the compute cluster (COMPUTE\_CLUSTER0) providing high-bandwidth resource access both to and from all of the connected processing elements and the rest of the system. MSMC serves as the data-movement backbone of the compute cluster.

Table 8-1 shows MSMC modules allocation within device domains.

**Table 8-1. MSMC Modules Allocation within Device Domains**

Module Instance	Domain		
	WKUP	MCU	MAIN
MSMC	-	-	✓

Figure 8-1 shows an overview of the MSMC and its surrounding modules.



**Figure 8-1. MSMC Overview**

MSMC supports the following features:

- 8MB (4 banks x 2MB) SRAM with ECC:
  - Shared coherent level 2/level 3 memory-mapped SRAM
  - Shared coherent level 3 cache
- 512-bit processor port bus and 40-bit physical address bus

- Coherent unified bi-directional interfaces to connect to processors or device masters
- One infrastructure master interface
- Single external memory master interface
- Supports distributed virtual system
- Supports internal DMA engine – DRU (Data Routing Unit)
  - DMA in/out L2 SRAM, MSMC, DDR and system
  - L2, L3 cache pre-warming and post flushing
- Bandwidth management with starvation bound
- Two-level QoS support for real-time/nonreal-time split
- Security firewall flush support for SRAM/cache and external memory
- Functional reliability:
  - SEC/DED protection on all data and tag memories with hardware scrubbing
  - SEC/DED protection on all data pipelines
  - Data memory address hamming protection
  - Coherent interconnect transaction metadata parity protection
- One interconnect messaging interface that supports DMA/prefetch requests to DRU
- Trace and debugging support
- Supports dynamic clock gating on all logic units
- MSMC is always on when VD\_CORE is on

#### 8.1.1.1 MSMC Not Supported Features

MSMC does not support the following:

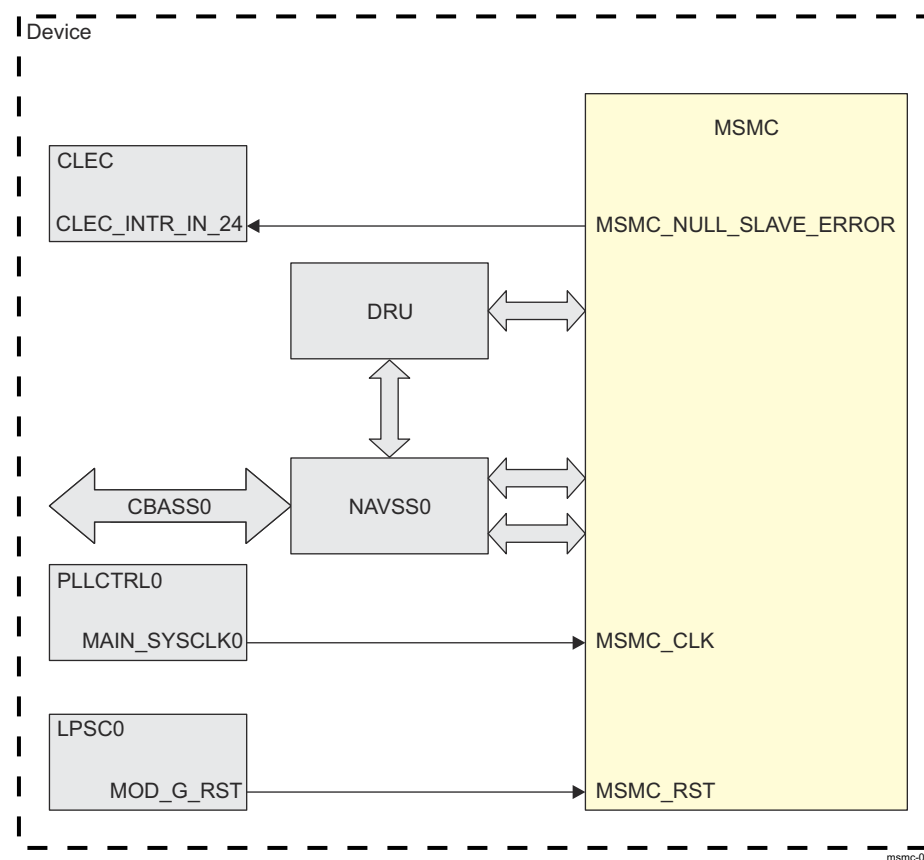
- RAM address decode protection
- Direct cache resize changes from one non-zero cache size configuration to another non-zero cache size configuration. In this case software is required to manually transition down to zero cache size configuration first, followed by a second transition from zero cache size configuration to the new non-zero cache size configuration.
- MSMC SRAM or SDRAM traffic during an MSMC cache resize transition. The traffic during this transition has undefined behavior.

## 8.1.2 MSMC Integration

This section describes MSMC integration in the device, including information about clocks, resets, and hardware requests.

### 8.1.2.1 MSMC Integration in MAIN Domain

A single MSMC module is integrated in the device MAIN domain. [Figure 8-2](#) shows the integration of MSMC.



**Figure 8-2. MSMC Integration**

Table 8-2 through Table 8-4 summarize the integration of MSMC in device MAIN domain.

**Table 8-2. MSMC Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
MSMC	PSC0	PD0	LPSC0	CBASS0 (Accessed through NAVSS0 NB0 <sup>(1)</sup> and NB1 <sup>(1)</sup> )

(1) NB = North Bridge

**Table 8-3. MSMC Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
MSMC	MSMC_CLK	MAIN_SYSCCLK0	PLLCTRL0	MSMC clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
MSMC	MSMC_RST	MOD_G_RST	LPSC0	MSMC module level main reset

**Table 8-4. MSMC Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MSMC	MSMC_NULL_SLAVE_ERROR	CLEC_INTR_IN_24	CLEC	Null slave error interrupt	Level
DMA Events					
Module Instance	Module DMA Event	Destination DMA Event Input	Destination	Description	Type
MSMC	-	-	-	-	-

## 8.1.2.2

---

**Note**

For more information about the MSMC interrupts, see [Section 8.1.3.8](#).

For more information on the interconnects, see [Chapter 3](#), *System Interconnect*.

For more information on the power, reset and clock management, see the corresponding sections within [Chapter 5](#), *Device Configuration*.

For more information on the device interrupt controllers, see [Section 9.2](#), *Interrupt Controllers*.

---

### 8.1.3 MSMC Functional Description

#### 8.1.3.1 MSMC Block Diagram

Figure 8-3 shows a high-level view of the MSMC module that includes the main interfaces, memory, and subunits.

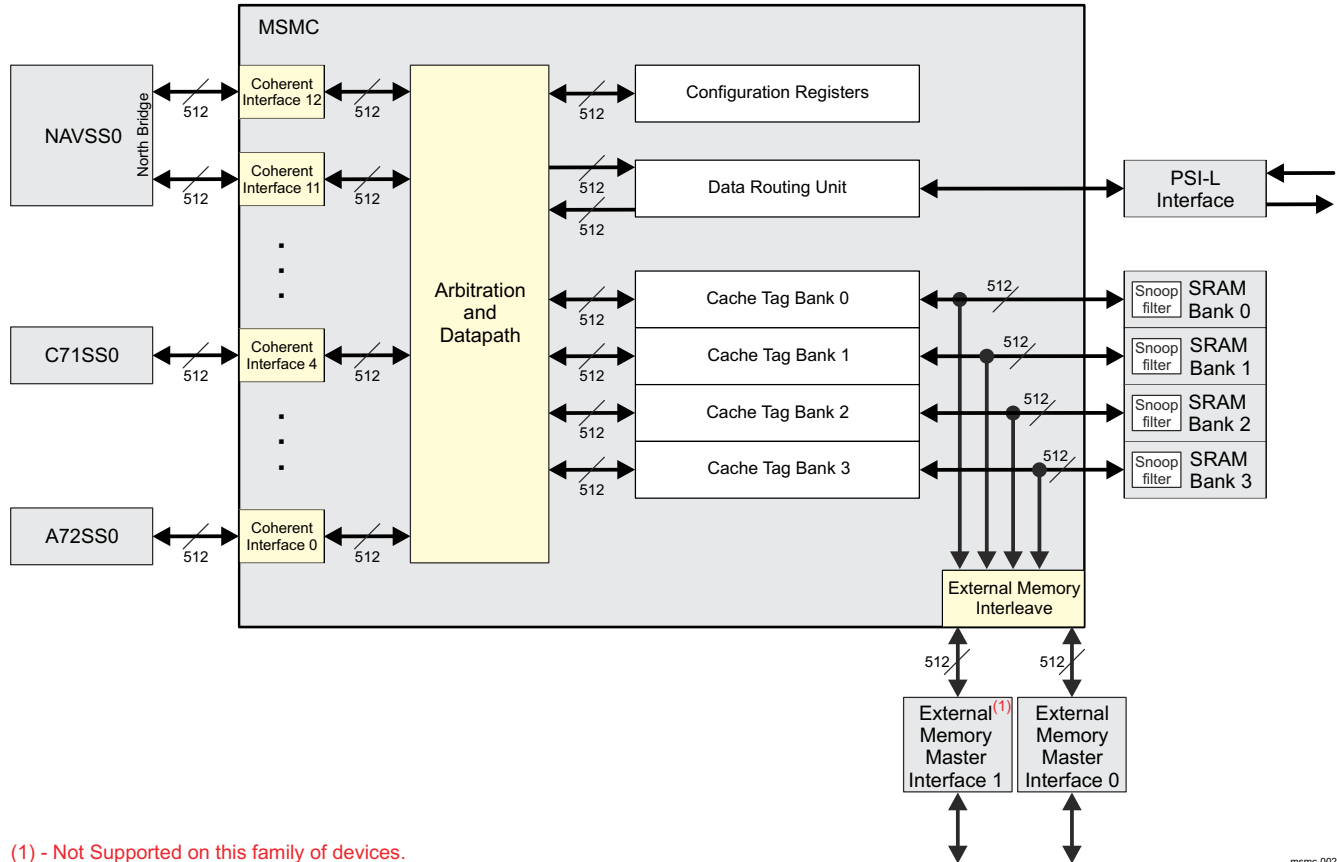


Figure 8-3. MSMC Functional Block Diagram

MSMC directly incorporates on-chip SRAM controllers to provide the compute cluster with low-latency memory. In addition, the individual memory banks can maintain coherence with the connected caching masters as well as the system slave ports.

#### 8.1.3.2 MSMC On-Chip Memory Banking

MSMC on-chip SRAM supports physical and virtual banking to maximize the available bandwidth to all masters. The total SRAM space is divided into four physical banks, and each physical bank contains two virtual sub-banks. These virtual banks provide access to that physical memory bank every MSMC\_CLK cycle.

Figure 8-4 shows the MSMC memory organization. Each SRAM bank provides 64 bytes per MSMC\_CLK cycle.

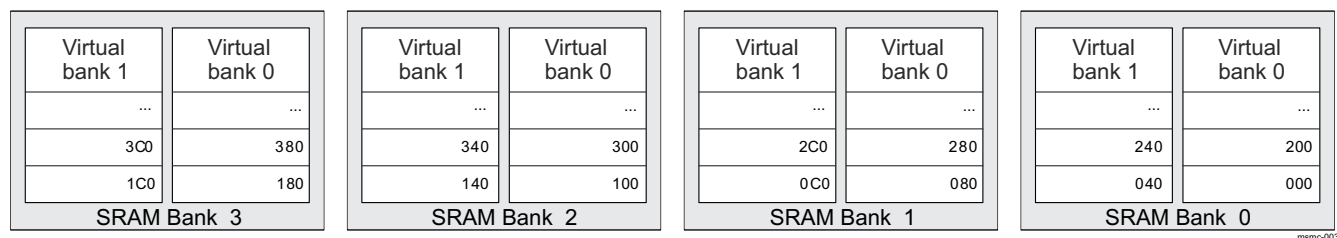


Figure 8-4. MSMC Memory Organization

### 8.1.3.3 MSMC Snoop Filter and Data Cache

Each MSMC SRAM bank implements the memory-mapped SRAM snoop filter in its own memory structure similar to a cache tag memory. The snoop filter tracks atomic coherent blocks at the same granularity as the level 3 cache for external memory: 128 bytes.

MSMC also implements a combined snoop filter and level 3 cache for the connected external memory space. As with the memory-mapped SRAM, the snoop filter exists to optimize performance of coherent external memory data cached in the attached level 1 and level 2 coherent caches. The external memory snoop filter is a 32-way set-associative cache structure which tracks coherent data at the same granularity as the SRAM snoop filter: 128 bytes.

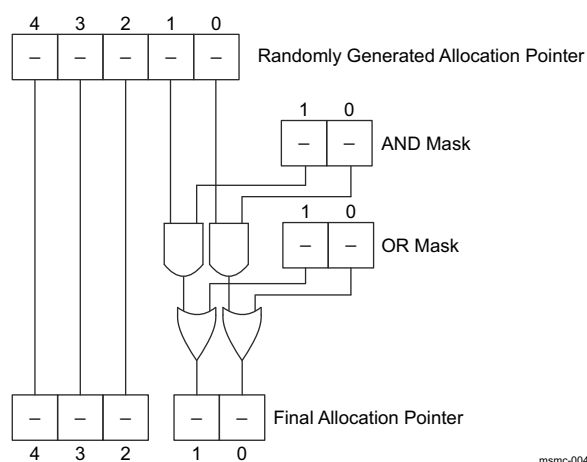
SoC and DRU transactions use snoop filter allocation policy which directs allocations strictly into the data-backed portion (ways) of the L3 cache. Random way selection only occurs when the targeted data-backed or non-data-backed portion (ways) of the L3 cache set are fully occupied. This applies also to CPU transactions, if the `MSMC_CACHE_CTRL[10] ALLOCATION_POLICY` bit is set to 0x0. When the `MSMC_CACHE_CTRL[10] ALLOCATION_POLICY` bit is set to 0x1, all CPU transactions use a randomized allocation policy which can select between both data-backed and non-data-backed ways.

The level 3 snoop filter/cache randomly chooses a way to replace when allocating a new cache line. The foundation of this random replacement is a 24-bit maximal period Linear Feedback Shift Register (LFSR) tuned to minimize any bias. The state of the LFSR supplies two to five bit allocation pointers depending on the configured cache size/number of ways. Each bank houses its own separately initialized LFSR which updates upon each allocation. For debugging purposes, MSMC supplies a linear mode replacement policy which initializes to the configured number of ways, eventually decrement to zero with allocations, and wrap back around to the total number of ways. Software can enable the linear replacement policy by writing 0x1 to the `MSMC_CACHE_CTRL[8] REPLACEMENT_POLICY` bit.

#### 8.1.3.3.1 Way Partitioning

The random replacement implementation attempts to minimize any allocation bias throughout the ways for a large number of allocations. However, it can be useful to introduce purposeful bias to help keep the state of certain tasks resident in the cache. To serve this end, MSMC supports programmable way-partitioning based on real-time versus nonreal-time traffic.

To accomplish this without introducing other unpredictable bias MSMC provides groups of AND-OR masks which can be attached to a group of IDs. The AND-OR masks consist of two fields: a two-bit AND field and a two-bit OR field. MSMC applies these two fields to the lower two bits of the randomly chosen allocation way. [Figure 8-5](#) shows how MSMC calculates the final allocation pointer using the initial random state and the user-defined AND-OR mask.



**Figure 8-5. Way Partition Allocation Pointer Generation**



The two-bit masks allow rough partitioning for quadrants of the cache. Depending on the configured cache size, the cache consists of 0-7 groups of 4 ways each, for a maximum of 32 ways. The upper 3 bits in the allocation pointer determine which group of 4 ways. The final two bits determine selection between the 4 ways within that group. [Table 8-5](#) describes the possible values for the AND-OR mask and the resulting available ways.

**Table 8-5. AND-OR Mask Way Selection**

Way 3	Way 2	Way 1	Way 0	AND Mask	OR Mask
X	X	X	X	11	00
X	X			-1	10
		X	X	01	00
X		X		1-	01
	X		X	10	00
X				--	11
	X			-0	10
		X		0-	01
			X	00	00

The AND-OR masks are programmed via the following bit fields:

- MSMC\_RT\_WAY\_SELECT[6-5] OR\_MASK
- MSMC\_RT\_WAY\_SELECT[1-0] AND\_MASK
- MSMC\_NRT\_WAY\_SELECT[6-5] OR\_MASK
- MSMC\_NRT\_WAY\_SELECT[1-0] AND\_MASK

#### 8.1.3.3.2 Cache Size Configuration and Associativity

MSMC has registers to provide software control to determine which portion of the on-chip SRAM will be used as cache data store. To maximize this flexibility, the associativity of the level 3 data cache scales with the desired cache size. The 32 ways in each snoop filter set are separated into 8 groups of 4 ways each as shown in [Table 8-6](#).

**Table 8-6. Cache Way Grouping**

	Group 0	Group 1	Group 2	Group 3
Way 0	0	4	8	12
Way 1	1	5	9	13
Way 2	2	6	10	14
Way 3	3	7	11	15
	Group 4	Group 5	Group 6	Group 7
Way 0	16	20	24	28
Way 1	17	21	25	29
Way 2	18	22	26	30
Way 3	19	23	27	31

The MSMC\_CACHE\_CTRL[3-0] CACHE\_SIZE bit field controls the cache size by selecting how many way groups are backed with data. [Table 8-7](#) describes the encodings of the CACHE\_SIZE field.

**Table 8-7. Cache Size Field Encodings**

CACHE_SIZE	Number of Active Way Groups	Total Active Data Ways
0x0	0	0 ways
0x1	1	4 ways
0x2	2	8 ways
0x3	3	12 ways
0x4	4	16 ways

**Table 8-7. Cache Size Field Encodings (continued)**

CACHE_SIZE	Number of Active Way Groups	Total Active Data Ways
0x5	5	20 ways
0x6	6	24 ways
0x7	7	28 ways
0x8	8	32 ways

Writing a new value to the CACHE\_SIZE field begins the cache size transition process. The MSMC hardware sets the MSMC\_CACHE\_CTRL[4] SZ\_TRANSITION bit to 0x1 during this transition process. When the transition process completes hardware sets SZ\_TRANSITION to 0x0. During this transition time window:

- Further writes to the CACHE\_SIZE field are ignored
- Reads to the CACHE\_SIZE field reflect the previous field value.

After setting a new CACHE\_SIZE value, software can poll the SZ\_TRANSITION bit to know when the transition is complete. At that point the new CACHE\_SIZE value can be read in the MSMC\_CACHE\_CTRL[3-0] CACHE\_SIZE bit field.

As the cache scales up in size, the memory-mapped space shrinks by the same amount, appearing as though the cache grows from the top of the memory mapped region down. This provides a single contiguous memory-mapped window starting from the SRAM base address. The memory space occupied by the cache is no longer accessible as memory-mapped SRAM and any accesses to this space return an addressing error.

#### 8.1.3.4 MSMC Access Protection Checks

To support potential software polling/reconfiguration for functional reliability, MSMC does not trigger error status for configuration accesses to unimplemented addresses. All reads of unimplemented addresses return data as zero and writes are ignored. This model should support a simple DMA "read, compare, and write-back" model for checking the MSMC configuration state.

#### 8.1.3.5 MSMC Null Slave

MSMC implements a null slave endpoint to handle accesses which cannot be sent to their intended destination. This can happen for several reasons listed in [Table 8-8](#). Commands routed to null slave return to the originating master with the appropriate response status listed in [Table 8-8](#). MSMC prioritizes protection error status over addressing error status when both can be applied to a single command.

**Table 8-8. Null Slave Response Status**

Description	Response Status
Firewall Flush	Protection Error
Invalid Memory Range	Addressing Error
Coherent Command to DMA	Success
Disabled Endpoint	Addressing Error
Real-time to DMA	Addressing Error

MSMC captures command information and triggers an event when routing unexpected command to null slave. Any of the conditions in [Table 8-8](#) which respond with a faulted response status (except firewall failures) triggers null slave error interrupt and logs appropriate command information to the MSMC\_NULL\_SLV\_STAT0 and MSMC\_NULL\_SLV\_STAT1 registers. The following bit fields capture that command information:

- ADDR
- BYTECNT
- ROUTEID
- PRIVID
- OPCODE
- MEMTYPE
- EMU

- SEC
- PRIV

For more information about the MSMC interrupts, see [Section 8.1.3.8](#).

MSMC captures a single null slave error event into the MSMC\_NULL\_SLV\_STAT0 and MSMC\_NULL\_SLV\_STAT1 registers.

#### 8.1.3.6 MSMC Resource Arbitration

When access requests arrive at the MSMC slave ports, the requests may target the same endpoint and hence need arbitration. These endpoint arbiters implement a multi-layer fairness arbitration scheme with starvation bounds to allow the user control to partition system bandwidth.

In each arbitration cycle, each arbiter prioritizes all active requests using the following prioritized layers until it selects a unique winner:

1. Transaction-tagged priority - Each active requestor pushes out a priority based on all of the currently active transactions in its pipeline. The highest priority requestors are chosen.
2. Fair-Share State - The arbiter selects the requestor(s) with the highest internal Fairshare state.
3. Static Priority - Finally, the arbiter uses a static priority to break any remaining ties after levels 1 and 2.

In addition to the three levels the arbiter tracks a starvation count for each requestor to each arbiter. Every time a request wins arbitration, all loser requests have their starvation count decremented. Every time a requestor wins arbitration for an active request, the starvation count resets to the initial state. If the starvation count reaches zero, that requestor priority level elevates to the highest priority level. Once the starved active request wins arbitration the priority level reverts back to the normal value. These starvation bounds can be programmed via the MSMC\_SBNDCOH0 through MSMC\_SBNDCOH12 registers.

#### Note

Not all MSMC\_SBNDCOHx registers are used on this device.

The three-layer arbitration scheme is not sufficient to ensure a bound on the minimum bandwidth experienced by lower priority requests relative to higher priority requests. Consequently, a starvation bound can be defined to limit that minimum bandwidth experienced by the lower priority requests relative to higher priority requests.

Since MSMC introduces multiple virtual channels feeding the same physical arbiter, the starvation mechanism is expanded to specify bounds for these channels.

**Table 8-9. Starvation Bound Definitions**

Port	Channel	Resource Target	Register Name and Field
Cache coherent 0	Real-time	External memory	MSMC_SBNDCOH0[55-48] SBNDE_RT
		On-chip resources	MSMC_SBNDCOH0[39-32] SBNDM_RT
	Non-real-time	External memory	MSMC_SBNDCOH0[23-16] SBNDE_NRT
		On-chip resources	MSMC_SBNDCOH0[7-0] SBNDM_NRT
Cache coherent 4	Real-time	External memory	MSMC_SBNDCOH4[55-48] SBNDE_RT
		On-chip resources	MSMC_SBNDCOH4[39-32] SBNDM_RT
	Non-real-time	External memory	MSMC_SBNDCOH4[23-16] SBNDE_NRT
		On-chip resources	MSMC_SBNDCOH4[7-0] SBNDM_NRT
Cache coherent 11	Real-time	External memory	MSMC_SBNDCOH11[55-48] SBNDE_RT
		On-chip resources	MSMC_SBNDCOH11[39-32] SBNDM_RT
	Non-real-time	External memory	MSMC_SBNDCOH11[23-16] SBNDE_NRT
		On-chip resources	MSMC_SBNDCOH11[7-0] SBNDM_NRT

**Table 8-9. Starvation Bound Definitions (continued)**

Port	Channel	Resource Target	Register Name and Field
Cache coherent 12	Real-time	External memory	MSMC_SBNDCOH12[55-48] SBNDE_RT
		On-chip resources	MSMC_SBNDCOH12[39-32] SBNDM_RT
	Non-real-time	External memory	MSMC_SBNDCOH12[23-16] SBNDE_NRT
		On-chip resources	MSMC_SBNDCOH12[7-0] SBNDM_NRT
DRU	Real-time	External memory	MSMC_SBNDDRU[55-48] SBNDE_RT
		On-chip resources	MSMC_SBNDDRU[39-32] SBNDM_RT
	Non-real-time	External memory	MSMC_SBNDDRU[23-16] SBNDE_NRT
		On-chip resources	MSMC_SBNDDRU[7-0] SBNDM_NRT
Responses Snoops	Real-time	External memory	MSMC_SBNDRRESP[55-48] SBNDE_RT
		On-chip resources	MSMC_SBNDRRESP[39-32] SBNDM_RT
	Non-real-time	External memory	MSMC_SBNDRRESP[23-16] SBNDE_NRT
		On-chip resources	MSMC_SBNDRRESP[7-0] SBNDM_NRT

The starvation bounds specified through the registers in [Table 8-9](#) apply to each trip an access requires through an arbitration point. The number of potential arbitrations should be considered when programming the starvation bounds. The following are few common examples of round-trip command arbitration for reads:

- Non-coherent SRAM read: Command central arbitration > Data SRAM arbitration > Response central arbitration
- Coherent SRAM read: Command central arbitration > Snoop central arbitration > Snoop response central arbitration > Read response central arbitration

Depending on whether caching and coherence are enabled and the state of the system, the number of potential arbitrations for a given access may vary and starvation should be considered accordingly.

#### 8.1.3.7 MSMC Error Detection and Correction

MSMC protects both on-chip memory resources and data busses throughout the MSMC pipelines. A mixture of SEC/DED (single-error correct/double-error detect) hamming code and simple parity protect the on-chip memory resources.

#### Note

MSMC aligns to the device architecture for system reliability using ECC aggregators to report detected parity and EDC faults both from scrubbing and functional access. The MSMC associated ECC aggregators are part of COMPUTE\_CLUSTER0. For more information, see [Section 6.1.2.3 Compute Cluster ECC Aggregators](#) in [Section 6.1 Compute Cluster](#).

For information about the ECC aggregator functionality, see [Section 12.11.4 ECC Aggregator](#).

##### 8.1.3.7.1 On-chip SRAM and Pipeline Data Protection

The on-chip data memory and data bus pipelines share the same Hamming EDC strategy to maximize protection across MSMC. Two EDC data/hamming pairs combine to form each 64 bytes of data. When functional accesses read data from the memory the EDC code travels through the MSMC pipeline with the data protecting all of those data pipelines in addition to the memory. Consumers of the protected data pairs should perform the EDC check and correction as needed before utilizing the data.

MSMC does not protect stored memory data all of the time. This requires memory initialization after reset to avoid spurious errors. In addition, read-modify-write combine cycle is also needed in case of any write access to the memory that does not commit to a full EDC quanta.

MSMC stores each 532-bit data + hamming code separately in the on-chip SRAM in the following way:

- SRAM[255:0] -> 256-bit Data Word 0
- SRAM[265:256] -> 10-bit Hamming Word 0
- SRAM[266:521] -> 256-bit Data Word 1
- SRAM[531:522] -> 10-bit Hamming Word 1

#### 8.1.3.7.2 On-chip SRAM L3 Cache Tag and Snoop Filter Protection

The level 3 cache/snoop filter tags stored in on-chip SRAM are also SEC/DED protected similarly to the data storage. Due to timing closure including all of the tag bits for a given set into one hamming data + code word is not possible. Instead, the 32 ways are broken into 8 groups of 4 ways each comprising hamming data (100 bits) + code (9 bits). The MSMC core logic outputs 8 logical SRAM interfaces for the tag memories based on the EDC protection quanta sizes. The hamming protection bits are stored at the most significant bit positions of the total data + hamming word inside the logical memory instance.

#### 8.1.3.7.3 On-chip SRAM Memory Mapped SRAM Snoop Filter Protection

MSMC stores the memory mapped SRAM snoop filter state in a separate memory from the tags and protects this information with simple parity. Each snoop filter entry requires five data bits and eight coherent blocks are stored together as a parity data word, resulting in one parity bit to every 40 data bits. The parity bit resides at the most significant bit position in the total parity + data word.

#### 8.1.3.7.4 Background Parity Refresh (Scrubbing)

The MSMC has scrubbing engine that periodically cycles through each location of each memory bank in the MSMC, reading and correcting the data, recalculating the parity bits for the data and storing the data and parity information.

The frequency with which each scrubbing cycle is initiated and the delay between each burst by the scrubbing engine is programmed using the MSMC\_SMEDCC register. The MSMC\_SMEDCC[7-0] REFDEL bitfield controls the number of MSMC\_CLK cycles between each scrub. To prevent the bursts from the scrubbing engine from posing a significant performance impact, the value in the REFDEL field is pre-scaled by 1024 in functional mode. A value of 0 is interpreted to be the same as a value of 1. At reset, REFDEL = 0 and a new scrub burst is issued 1024 cycles after the previous one ends. The operation of the scrubbing engine is enabled by default at reset but may be disabled by setting the MSMC\_SMEDCC[31] SEN bit to 0x0.

#### 8.1.3.8 MSMC Interrupts

The MSMC has the following registers for interrupt control:

- MSMC\_SMIRSTAT - Raw interrupt status
- MSMC\_SMIRWS - Raw interrupt software set
- MSMC\_SMIRC - Raw interrupt software clear
- MSMC\_SMIESTAT - Enable interrupt status
- MSMC\_SMIIEWS - Enable interrupt software set
- MSMC\_SMIIEC - Enable interrupt software clear
- MSMC\_SMESTAT - Raw and Enabled Interrupt Status

Each bit in these registers represents a single event or interrupt supported by MSMC, which generates only one interrupt. This is the null slave error interrupt (MSMC\_NULL\_SLAVE\_ERROR). MSMC sets this event when routing a command through the MSMC null slave port which results in an unsuccessful response status. Commands failing firewall checks do not trigger the null slave error interrupt event in MSMC.

##### 8.1.3.8.1 Raw Interrupt Registers

MSMC sets the associated bit in the MSMC\_SMIRSTAT register when a software or hardware event occurs. Software can trigger the event by setting the corresponding bit in the MSMC\_SMIRWS register. MSMC keeps the MSMC\_SMIRSTAT bit asserted until software clears it by setting the corresponding bit in MSMC\_SMIRC. Both MSMC\_SMIRWS and MSMC\_SMIRC are write-only registers and do not implement any actual state.

### 8.1.3.8.2 Interrupt Enable Registers

Software can enable an event by setting the associated bit in the MSMC\_SMIEWS register, and MSMC will assert the same bit in the MSMC\_SMIESTAT register in response. The bit of MSMC\_SMIESTAT remains asserted until software disables the event by setting the corresponding bit in the MSMC\_SMIEC register. Both MSMC\_SMIEWS and MSMC\_SMIEC are write-only registers and do not implement any actual state.

### 8.1.3.8.3 Triggered and Enabled Interrupts

The MSMC\_SMIESTAT register is a read-only reflection of events that are both triggered and enabled. It is a combination of MSMC\_SMIRSTAT and MSMC\_SMIESTAT.

### 8.1.3.9 MSMC Memory Regions

Device masters access MSMC controlled resources through two memory map windows - internal and external.

[Table 8-10](#) shows all MSMC associated memory regions including detailed breakdown of the MSMC internal memory mapped window.

**Table 8-10. MSMC Memory Regions**

Region	Start Address	End Address	Region Size
Internal memory mapped window			
MSMC Coherent Interface for A72SS0	0x00 6000 0000	0x00 60FF FFFF	16 MB
MSMC Coherent Interface for C71SS0	0x00 6400 0000	0x00 64FF FFFF	16 MB
MSMC Coherent Interface for the NB0 Port of North Bridge in NAVSS0	0x00 6B00 0000	0x00 6BFF FFFF	16 MB
MSMC Coherent Interface for the NB1 Port of North Bridge in NAVSS0	0x00 6C00 0000	0x00 6CFF FFFF	16 MB
DRU Configuration Registers	0x00 6D00 0000	0x00 6DFF FFFF	16 MB
MSMC Configuration Registers <sup>(1)</sup>	0x00 6E00 0000	0x00 6EFF FFFF	16 MB
MSMC SRAM	0x00 7000 0000	0x00 707F FFFF	8 MB
External memory mapped window			
2 GB External SDRAM Space	0x00 8000 0000	0x00 FFFF FFFF	2 GB
8 GB External SDRAM Space <sup>(2)</sup>	0x08 0000 0000	0x09 FFFF FFFF	8 GB
Other MSMC associated memory regions			
Compute Cluster ECC Aggregator 0 Configuration Registers	0x4D 2000 0000	0x4D 2000 03FF	1 KB
Compute Cluster ECC Aggregator 1 Configuration Registers	0x4D 2000 0400	0x4D 2000 07FF	1 KB
Compute Cluster ECC Aggregator 2 Configuration Registers	0x4D 2000 0800	0x4D 2000 0BFF	1 KB

(1) The MSMC configuration space supports 1-, 2-, 4- and 8-byte aligned reads and 4- and 8-byte aligned writes.

(2) The first 2 GB of this region are inaccessible.

As shown in [Table 8-10](#) software accesses the memory-mapped on-chip SRAM as part of the MSMC memory mapped space.

### 8.1.3.10 MSMC Hardware Coherence

MSMC supports hardware cache coherence between CPU cache-coherent masters and peripherals accessing MSMC through the system slave ports for shared SRAM and DDR data range spaces. Hardware coherence support abstracts away the cache memory systems of supported components from software, simplifying software implementation.

MSMC does not support memory coherence for the following spaces:

- DDRSS configuration space
- MSMC configuration space



- System-on-chip master port peripherals/memory
- CPU local resources such as memory-mapped level 2 SRAM
- Any other memory not directly connected to MSMC

To stay coherent with each other, caches inside coherent masters typically track more states than the traditional valid/dirty combinations. MSMC uses the ACE coherence protocol which supports these five MOESI states:

- Modified - Cache line is unique (no other caches currently have it) and dirty
- Owned - Cache line is shared (other caches may have it) and dirty
- Exclusive - Cache line is unique (no other caches have it) and clean
- Shared - Cache line is shared (other caches may have it) and clean
- Invalid - No allocated line

In order to maintain coherence between cached masters, MSMC can initiate requests to coherent masters for shared regions of memory. These requests are referred to as "snoop requests".

The following example demonstrates how coherence is maintained between DMA and CPU, when DMA writes to shareable memory space.

1. DMA (noncaching master) issues a write to shareable space.
2. MSMC coherence controller issues Invalidate and Writeback snoop to all CPUs.
3. All CPUs invalidate the line from their cache and respond with dirty data if necessary
4. MSMC merges the DMA write data with the victim and commits to memory.

The following example demonstrates how coherence is maintained between DMA and CPU, when DMA reads from shareable memory space.

1. DMA issues a read to shareable space.
2. MSMC coherence controller issues a ReadOnce snoop to the CPU.
3. CPU responds with cached data.
4. MSMC returns the read data to DMA.

#### 8.1.3.10.1 Snoop Filter Broadcast Mode

As described in [Section 8.1.3.3](#) MSMC implements inclusive snoop filters for both memory-mapped SRAM and external memory shared space to limit the amount of required snoop traffic. Both the SRAM and external snoop filters encode the following states for each coherent memory block in their respective filtering structures:

- INVALID - This memory block must be in Invalid state in all coherent CPUs. Since the external snoop filter is inclusive of all cached coherent data, a miss in the snoop filter is functionally equivalent to this state.
- CPU\_SHARED - This memory block may be cached in Shared or Owned state in the associated coherent CPU.
- CPU\_UNIQUE - This memory block may be cached in Shared, Owned, Exclusive, or Modified state in the associated coherent CPU.
- BROADCAST\_SHARED - This memory block may be cached in Shared or Owned states in multiple coherent CPUs.
- BROADCAST\_UNIQUE - This memory block may be cached in Shared, Owned, Exclusive, or Modified states in multiple coherent CPUs. Typically lines only reach this state when MSMC encounters a snoop filter parity error or broadcast mode is enabled.

As a safeguard MSMC provides a "Broadcast Mode" configuration that forces MSMC to treat all snoop filter entries as BROADCAST\_UNIQUE. This results in snoops generated to all applicable coherent masters for snoop filter hits. This mode can be toggled on and off, but the user should be aware that the snoop filter becomes no longer strictly inclusive once enabled since snoop filter entries accessed during this mode are no longer accurately tracking line ownership. Coherency is still maintained in this mode, though additional potentially unnecessary snoop activity may occur. To enable the broadcast mode feature the MSMC\_COHCTRL[0] BCM bit should be set to 0x1.

### 8.1.3.11 MSMC Quality-of-Service

As an orthogonal axis to arbitration priority MSMC provides two classes of traffic, real-time (RT) and nonreal-time (NRT), to maintain quality-of-service (QoS) guarantees and alleviate head-of-line blocking issues. The currently implemented dataflows supported by the QoS hardware are:

- SoC coherent slave - MSMC SRAM
- SoC coherent slave - External memory
- DRU coherent slave - MSMC SRAM
- DRU coherent slave - External memory

At a high-level priority and QoS represent orthogonal axes controlling the latency (priority) and bandwidth protection (QoS) for multiple data flows routed through the MSMC. To accomplish this MSMC provides a dedicated buffering at each arbitration point that can only be consumed by RT traffic. This way NRT traffic cannot completely starve out RT requests.

The SoC coherent, DRU coherent, and external memory interfaces need dedicated credits to support QoS. Each of these interfaces implements additional threading to exchange these dedicated credits.

**Table 8-11. QoS Threading**

Thread Index	[0]	[2]
SoC Coherent Slave	CPU	RT_CPU
DRU Coherent Slave	CPU	RT_CPU
External Memory Master	CPU	RT_CPU

The RT\_CPU thread provides a completely separate pool of credits, where necessary, to implement the dedicated buffers for RT traffic.

#### Note

There is no software control over the MSMC QoS hardware.

#### Note

Interfaces which do not support QoS features denote all traffic as nonreal-time.

### 8.1.3.12 MSMC Memory Regions Protection

There is no built-in firewall to protect the on-chip and external memory regions. MSMC relies on the system distributed firewall structure to protect these memory windows. MSMC receives a "pass/fail" attribute on each of its entry points including the DRU interfaces which it uses to route the transaction to either the addressed endpoint (pass) or to the internal null slave (fail). Transactions carrying the "fail" return Protection Error on read or write status. Since these failed transactions route through the null slave endpoint no snoops, allocations, or other hardware side-effects occur as a result of these transactions.

### 8.1.3.13 MSMC Cache Tag View

MSMC implements debug hardware to allow software to query the state of the SRAM snoop filter and L3 cache tag/snoop filter. Software can query the tag structures using the MSMC\_DBGTAGCTL and MSMC\_DBGTAGVIEW registers in the following sequence:

1. Write the desired tag or snoop filter information to the MSMC\_DBGTAGCTL register and wait for status return
2. Read the MSMC\_DBGTAGVIEW register to collect the values populated from step 1

The write in step 1 triggers population of the MSMC\_DBGTAGVIEW register with the appropriate values from the tag or snoop filter memory. The following are important notes about this feature:



- Writing invalid values into the MSMC\_DBGTAGCTL register results in undefined values in the MSMC\_DBGTAGVIEW register.
- All writes to MSMC\_DBGTAGCTL triggers re-population of MSMC\_DBGTAGVIEW, regardless of whether MSMC\_DBGTAGVIEW has been read since the last write to MSMC\_DBGTAGCTL
- This feature is intended for debug use when the system is halted or quiet. If software writes MSMC\_DBGTAGCTL during active operation MSMC will find an empty cycle on the tag/snoop filter memory to read the contents and return to MSMC\_DBGTAGVIEW. However, if there are accesses in flight which operate on the desired tag or snoop filter block the contents of MSMC\_DBGTAGVIEW will reflect whatever was in the memory at the time of read.
- Security considerations for cache tag view must be comprehended in the firewalls present before all MSMC request interfaces. MSMC does not perform additional security checks.

## 8.2 DDR Subsystem (DDRSS)

This section describes the DDR Subsystem (DDRSS) for the device.

### 8.2.1 DDRSS Overview

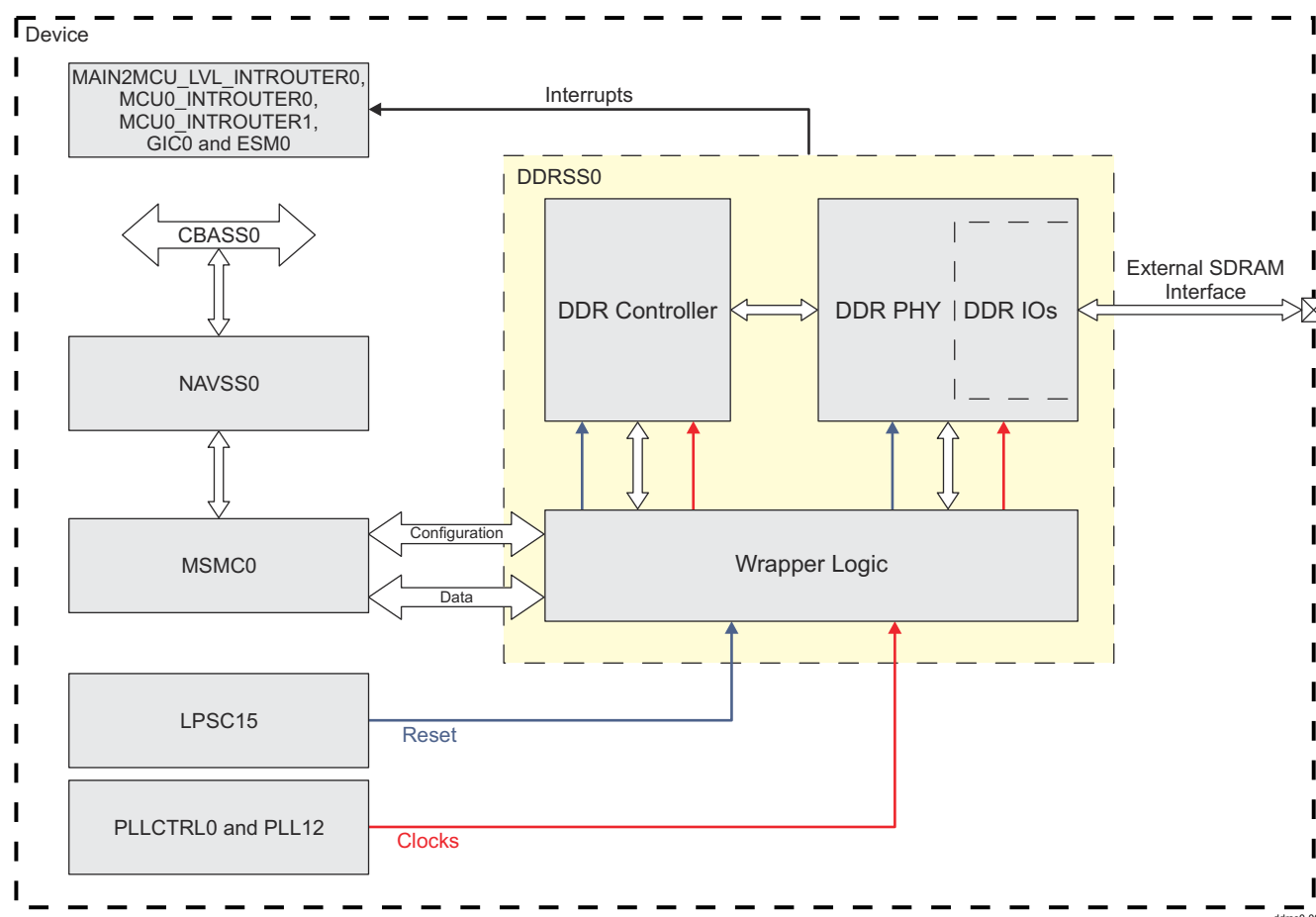
The DDR subsystem in this device comprises DDR controller, DDR PHY and wrapper logic to integrate these blocks in the device. The DDR subsystem is referred to as DDRSS0 and is used to provide an interface to external SDRAM devices which can be utilized for storing program or data. DDRSS0 is accessed via MSMC, and not directly through the system interconnect.

Table 8-12 shows DDRSS modules allocation within device domains.

**Table 8-12. DDRSS Modules Allocation within Device Domains**

Module Instance	Domain		
	WKUP	MCU	MAIN
DDRSS0	-	-	✓

Figure 8-6 shows an overview of the DDRSS0 and its surrounding modules.



**Figure 8-6. DDRSS Overview**

The DDRSS0 supports:

- Memory Types:
  - LPDDR4
- Memory Bus Features:

- 32-bit width with in-line ECC
- Up to 2 ranks (in one package)
- SDRAM address range up to 8 GB
- System Bus Interface:
  - 512-bit data width
  - Little endian only
  - Address aliasing prevention to block accesses to unpopulated SDRAM region
  - Clock asynchronous to DDR clock
- Configuration Bus Interface:
  - 32-bit data width
  - Linear incrementing addressing mode
  - 32-bit aligned accesses only
  - Little endian only
  - Clock asynchronous to DDR clock
- Key Features:
  - Full coherency across all commands
  - Bank interleaving
  - Priority based scheduling
  - Scheduling based on bank openness
  - Class of Service (CoS) - Three latency classes supported
  - Read/write scheduling to avoid turn-around time
  - Prioritized refresh scheduling
  - Dynamic change of refresh rate via software for extended temperatures
  - Statistical counters for performance management
- SDRAM ECC Features:
  - In-line ECC
  - Read-modify-write ECC for sub-word writes
  - ECC address error logging
  - Statistical counters for counting ECC errors
  - Injecting ECC errors during normal operation for validation
- Low Power Features:
  - Self-refresh entry and exit via software or hardware clock stop request
  - System bus clock stop via hardware clock stop request when controller is idle
  - Automatic idle power saving mode when no or low activity is detected
  - DDR and system bus clock frequency change using self-refresh via software or hardware clock stop request
  - Turning off SoC power after DDR is put into self-refresh (DDR reset and CKE I/O retention)
  - Tri-stating of all DDR I/O cells via software while driving CKE and RSTN pins during self-refresh
  - LPDDR4 Frequency Set Point (FSP)
  - I/O retention managed by an external device through dedicated pin
- Functional Reliability Features:
  - ECC on data pipe
  - Parity on address and command pipes
  - Data EDC on the master port interface
  - Command parity on the master port interface
  - Data EDC on the configuration port interface
  - Command parity on the configuration port interface
  - MSMC2DDR bridge AXI bus timeout
- DDR PHY Features:
  - Automatic and software controllable initialization and calibration (ZQ) for the DDR PHY and I/O cells
  - Automatic and software controllable delay line calibrations with Voltage and Temperature (VT) compensation

- Automatic and software controllable write levelling with VT compensation
- Automatic read DQS gate training per rank with VT compensation
- Automatic and software controllable DQ/DQS eye training per rank
- Automatic and software controllable read and write data bit deskew
- Automatic and software controllable Command/Address (CA) levelling with VT compensation for LPDDR4
- Automatic and software controllable CA bit deskew for LPDDR4
- Refreshes to SDRAM during leveling and training
- No seeding requirement based on board topology for any of the leveling and training algorithms
- Dynamic/automatic I/O Receiver disable when read transfer is not on going
- Capability of disabling data macros and I/O cells when not in use

#### 8.2.1.1 DDRSS Not Supported Features

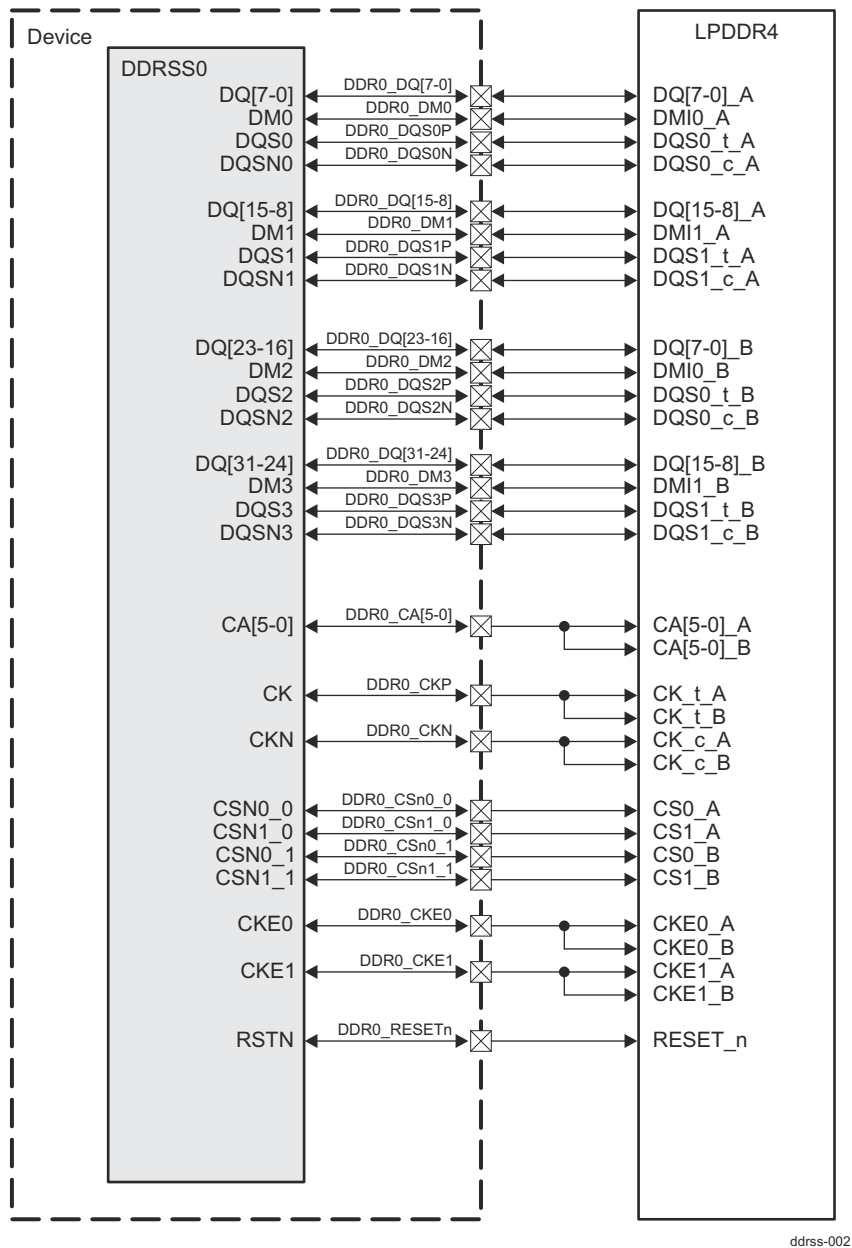
DDRSS0 does not support the following:

- DDR3 SDRAMs
- DDR3L SDRAMs
- DDR3U SDRAMs
- DDR4 SDRAMs
- DIMM
- 1/4 width (8-bit) mode via software configuration
- Data bus obfuscation or any other kind of encryption
- Fail-safe reset I/O to maintain reset state during SoC power OFF
- Independent, 16-bit, two-channel operation for LPDDR4
- The ECC engine of the DDR controller
- No support for byte mode, or memories with more than 17 row address bits

## 8.2.2 DDRSS Environment

This section describes the DDRSS0 external connections (environment).

Figure 8-7 shows an example connection to dual rank LPDDR4 x32 device.



ddrss-002

Figure 8-7. LPDDR4 Connection

Table 8-13 describes the DDRSS0 I/O signals used for connection to SDRAM devices.

**Table 8-13. DDRSS0 I/O signals**

Module Pin	Device Level Signal	I/O <sup>(1)</sup>	Description	Module Pin Reset Value <sup>(2)</sup>
RSTN	DDR0 RSTN	I/O	SDRAM reset	0x0
CKE[1-0]	DDR0 CKE[1-0]	I/O	SDRAM CKE[1-0] signals	0x0
CK	DDR0 CKP	I/O	SDRAM differential clock pair	0x0
CKN	DDR0 CKN			0x1

**Table 8-13. DDRSS0 I/O signals (continued)**

Module Pin	D e v i c e L e v e l S i g n a l	I/O <sup>(1)</sup>	Description	Module Pin Reset Value <sup>(2)</sup>
CSN0_0	D D R 0   C S n 0   0	I/O	SDRAM chip select 0 (two copies of CS0) <sup>(3)</sup>	0x0
CSN0_1	D D R 0   C S n 0   1			0x0
CSN1_0	D D R 0   C S n 1   0	I/O		0x0
CSN1_1	D D R 0   C S n 1   1			0x0

**Table 8-13. DDRSS0 I/O signals (continued)**

Module Pin	Device Level Signal	I/O <sup>(1)</sup>	Description	Module Pin Reset Value <sup>(2)</sup>
CA[5-0]	DDR0 [CA[5-0]	I/O	SDRAM address and command bus	HiZ
DQS[3-0]	DDR0 [DQS[3-0]P	I/O	SDRAM data strobe	HiZ
DQSN[3-0]	DDR0 [DQS[3-0]N	I/O	SDRAM data strobe invert	HiZ
DQ[31-0]	DDR0 [DQ[31-0]	I/O	SDRAM data bus	HiZ



**Table 8-13. DDRSS0 I/O signals (continued)**

Module Pin	D e v i c e L e v e l S i g n a l	I/O <sup>(1)</sup>	Description	Module Pin Reset Value <sup>(2)</sup>
DM[3-0]	D D R 0 [ D M [ 3 - 0 ]	I/O	SDRAM data mask/DBI	HiZ
RET	D D R [ R E T	I/O	External I/O retention enable	-

(1) I = Input; O = Output

(2) HiZ = High Impedance

(3) Chip select for each channel is controlled independently during CA training, but they run in locked-step during normal operation.

#### Note

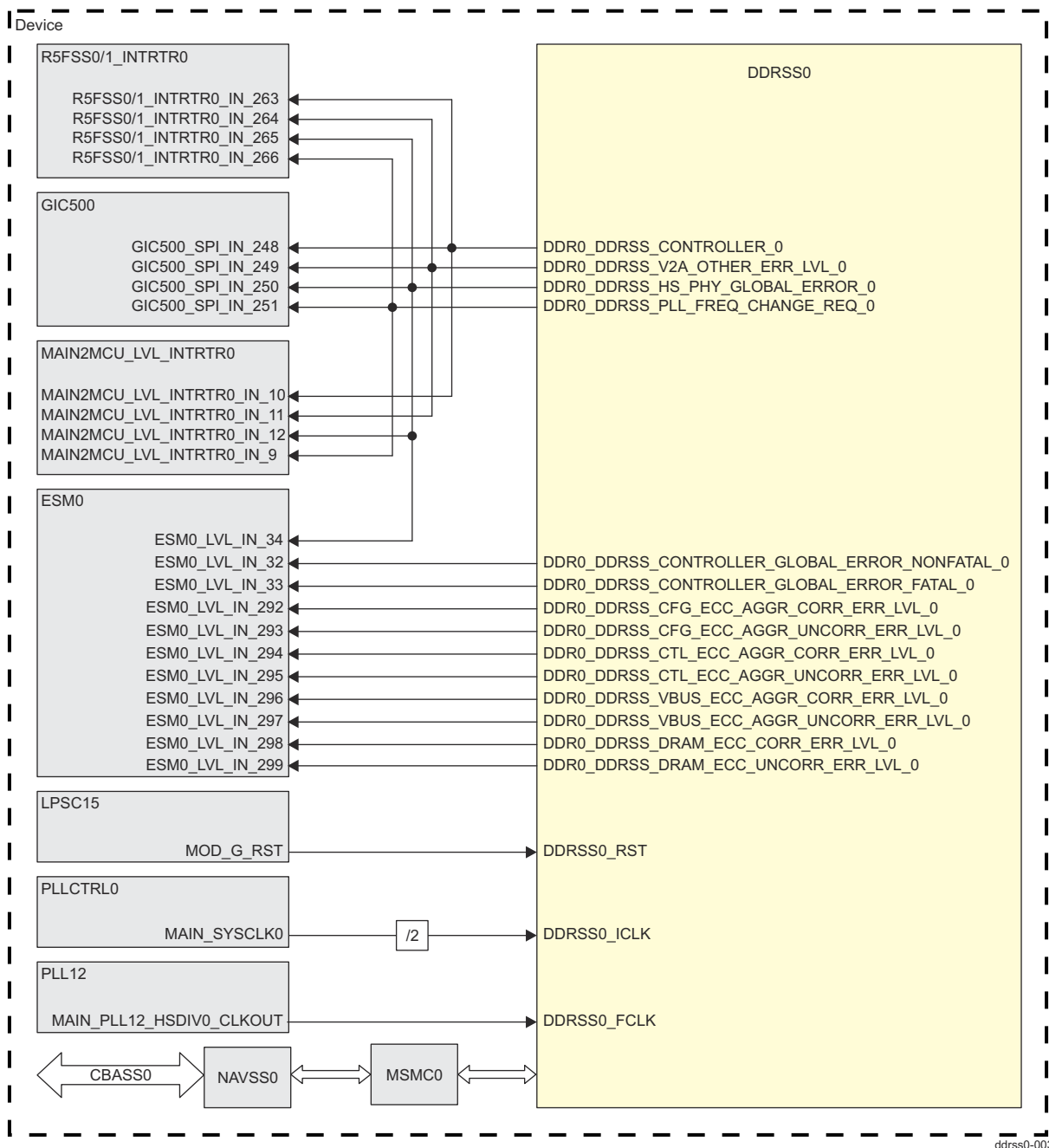
For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

### 8.2.3 DDRSS Integration

This section describes DDRSS0 integration in the device, including information about clocks, resets, and hardware requests.

#### 8.2.3.1 DDRSS Integration in MAIN Domain

A single DDRSS0 module is integrated in the device MAIN domain. [Figure 8-8](#) shows the integration of DDRSS0.



**Figure 8-8. DDRSS0 Integration**

Table 8-14 through Table 8-16 summarize the integration of DDRSS0 in device MAIN domain.

**Table 8-14. DDRSS0 Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
DDRSS0	PSC0	PD0	LPSC15	CBASS0 (Accessed through MSMC)

**Table 8-15. DDRSS0 Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
DDRSS0	DDRSS0_ICLK	MAIN_SYSCCLK0/2	PLLCTRL0	DDRSS0 configuration interface clock
	DDRSS0_FCLK	MAIN_PLL12_HSDIV0_CLKO UT	PLL12	DDRSS0 functional clock. It supplies the SDRAM clock.
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
DDRSS0	DDRSS0_RST	MOD_G_RST	LPSC15	DDRSS0 reset

**Table 8-16. DDRSS0 Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DDRSS0	DDR0_DDRSS_CONTROLLER_GLOBAL_ERROR_NONFATAL_0	ESM0_LVL_IN_32	ESM0	DDR controller interrupt associated with bit [0] of the DDRSS_CTL_437[31-24] GLOBAL_ERROR_INFO field	Level
	DDR0_DDRSS_CONTROLLER_GLOBAL_ERROR_FATAL_0	ESM0_LVL_IN_33	ESM0	DDR controller interrupt associated with bits [7-1] of the DDRSS_CTL_437[31-24] GLOBAL_ERROR_INFO field	Level
	DDR0_DDRSS_CFG_ECC_AGGR_CORRECTABLE_ERROR_LVL_0	ESM0_LVL_IN_292	ESM0	DDRSS0_ECC_AGGR_CFG correctable error interrupt	Level
	DDR0_DDRSS_CFG_ECC_AGGR_UNCORRECTABLE_ERROR_LVL_0	ESM0_LVL_IN_293	ESM0	DDRSS0_ECC_AGGR_CFG non-correctable error interrupt	Level
	DDR0_DDRSS_CTL_ECC_AGGR_CORRECTABLE_ERROR_LVL_0	ESM0_LVL_IN_294	ESM0	DDRSS0_ECC_AGGR_CTL correctable error interrupt	Level
	DDR0_DDRSS_CTL_ECC_AGGR_UNCORRECTABLE_ERROR_LVL_0	ESM0_LVL_IN_295	ESM0	DDRSS0_ECC_AGGR_CTL non-correctable error interrupt	Level
	DDR0_DDRSS_VBUS_ECC_AGGR_CORRECTABLE_ERROR_LVL_0	ESM0_LVL_IN_296	ESM0	DDRSS0_ECC_AGGR_VBUS correctable error interrupt	Level
	DDR0_DDRSS_VBUS_ECC_AGGR_UNCORRECTABLE_ERROR_LVL_0	ESM0_LVL_IN_297	ESM0	DDRSS0_ECC_AGGR_VBUS non-correctable error interrupt	Level
	DDR0_DDRSS_DRAM_ECC_CORRECTABLE_ERROR_LVL_0	ESM0_LVL_IN_298	ESM0	MSMC2DDR bridge 1-bit error interrupt. See <a href="#">Section 8.2.4.1.4.2</a> .	Level

**Table 8-16. DDRSS0 Hardware Requests (continued)**

DDR0_DDRSS_DRAM_ECC_UNCORR_ERR_LVL_0	ESM0_LVL_IN_299	ESM0	MSMC2DDR bridge 2-bit error interrupt. See <a href="#">Section 8.2.4.1.4.2</a> .	Level
DDR0_DDRSS_CONTROLLER_0	GIC500_SPI_IN_248	GIC500	DDR controller common interrupt	Level
	R5FSS0_INTRTR0_IN_263	R5FSS0_INTRTR0		
	R5FSS1_INTRTR0_IN_263	R5FSS1_INTRTR0		
	MAIN2MCU_LVL_INT_RTR0_IN_10	MAIN2MCU_LVL_I_NTRTR0		
DDR0_DDRSS_V2A_OTHER_ERR_LVL_0	GIC500_SPI_IN_249	GIC500	MSMC2DDR bridge interrupt indicating:	Level
	R5FSS0_INTRTR0_IN_264	R5FSS0_INTRTR0	• Unsupported	
	R5FSS1_INTRTR0_IN_264	R5FSS1_INTRTR0	VBUSM.C opcode See <a href="#">Section 8.2.4.1.5</a> .	
	MAIN2MCU_LVL_INT_RTR0_IN_11	MAIN2MCU_LVL_I_NTRTR0	• Address out of range. See <a href="#">Section 8.2.4.1.6</a> .	
			• Timeout counter expiry. See <a href="#">Section 8.2.4.1.8</a> .	
DDR0_DDRSS_HS_PHY_GLOBAL_ERROR_0	GIC500_SPI_IN_250	GIC500	DDR PHY global error interrupt	Level
	ESM0_LVL_IN_34	ESM0		
	R5FSS0_INTRTR0_IN_265	R5FSS0_INTRTR0		
	R5FSS1_INTRTR0_IN_265	R5FSS1_INTRTR0		
	MAIN2MCU_LVL_INT_RTR0_IN_12	MAIN2MCU_LVL_I_NTRTR0		
DDR0_DDRSS_PLL_FREQ_CHANGE_REQ_0	GIC500_SPI_IN_251	GIC500	DDRSS0 frequency change request interrupt. See <a href="#">Section 8.2.4.5</a> .	Level
	R5FSS0_INTRTR0_IN_266	R5FSS0_INTRTR0		
	R5FSS1_INTRTR0_IN_266	R5FSS1_INTRTR0		
	MAIN2MCU_LVL_INT_RTR0_IN_9	MAIN2MCU_LVL_I_NTRTR0		

DMA Events					
Module Instance	Module DMA Event	Destination DMA Event Input	Destination	Description	Type
DDRSS0	-	-	-	-	-

**Note**

For more information about the DDRSS0 interrupts, see [Section 8.2.4.2](#).

## 8.2.4 DDRSS Functional Description

### 8.2.4.1 DDRSS MSMC2DDR Bridge

The MSMC2DDR bridge is part of the DDRSS and connects the DDR controller and MSMC data paths.

#### 8.2.4.1.1 VBUSM.C Threads

##### Note

VBUS (VBUSM and VBUSP) is the device interconnect communication protocol. The letter "C" in "VBUSM.C" stands for Coherence Extension and denotes the interface is dedicated to COMPUTE\_CLUSTER0. The VBUS protocol is beyond the scope of the device documentation as it is purely hardware oriented. For details about the device interconnect and the associated software controls, see *System Interconnect*.

The MSMC2DDR bridge supports two threads on the VBUSM.C interface. They are as follows:

- High Priority Thread (HPT) - traffic received on VBUSM.C thread 2 belongs to HPT.
- Low Priority Thread (LPT) - traffic received on VBUSM.C thread 2 belongs to LPT.

HPT has priority over LPT. Therefore, the execution of commands from the command FIFO can be out-of-order. As a result, the read data can also be returned out-of-order. This ensures that the HPT will be guaranteed execution even when LPT is fully blocked.

The MSMC2DDR bridge maintains data coherency across threads. Therefore, any HPT transactions that depend on the LPT transactions due to address conflicts are blocked until execution of the corresponding LPT transactions. This might result in a scenario where HPT can be blocked because of blockage on LPT. Once the LPT blockage has cleared, the blockage on the dependent HPT commands is also cleared.

HPT transactions can use global credits as well as thread 2 credits. LPT transactions can only use global credits.

On the command interface, the MSMC2DDR bridge gives back global credits on the VBUSM.C interface only if the available credits on the command queue in the DDR controller are greater than the DDRSS\_V2A\_CTL\_REG[16-12] CRIT\_THRESH field.

On the read data return interface, HPT traffic is prioritized over LPT traffic. The MSMC2DDR bridge sends HPT read data if there is at least 1 global or 1 thread 2 credit available.

The MSMC2DDR bridge sends LPT read data if there is no HPT data to send, and:

- There are at least 2 global credits available, or
- There is at least 1 global credit available and it is the last data phase

#### 8.2.4.1.2 Class of Service (CoS)

Commands arriving on LPT and HPT to the DDRSS0 carry the VBUSM.C priority whereas the DDR controller uses another priority. The MSMC2DDR bridge has the following registers for flexible mapping of the VBUSM.C priority to DDR controllers's priority:

- Range match registers:
  - DDRSS\_V2A\_R1\_MAT\_REG
  - DDRSS\_V2A\_R2\_MAT\_REG
  - DDRSS\_V2A\_R3\_MAT\_REG
- Priority map registers:
  - DDRSS\_V2A\_LPT\_DEF\_PRI\_MAP\_REG
  - DDRSS\_V2A\_LPT\_R1\_PRI\_MAP\_REG
  - DDRSS\_V2A\_LPT\_R2\_PRI\_MAP\_REG
  - DDRSS\_V2A\_LPT\_R3\_PRI\_MAP\_REG
  - DDRSS\_V2A\_HPT\_DEF\_PRI\_MAP\_REG
  - DDRSS\_V2A\_HPT\_R1\_PRI\_MAP\_REG
  - DDRSS\_V2A\_HPT\_R2\_PRI\_MAP\_REG

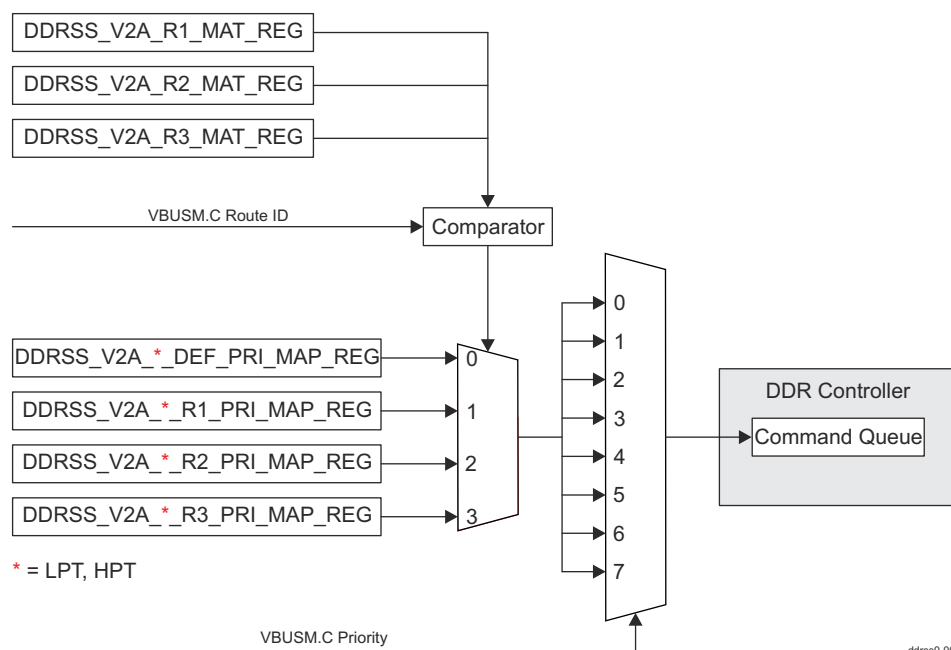
#### – DDRSS\_V2A\_HPT\_R3\_PRI\_MAP\_REG

This allows the system to essentially create different classes of service based on the initiator (Route ID) and the priority of the commands.

Each thread has a set of corresponding priority map registers to map the incoming priority on that thread to appropriate priority for the DDR controller as shown in Figure 8-9. The default values of these registers are such that inherently the HPT traffic has higher priority than the LPT traffic inside the DDR controller, but these settings can be changed via software per system needs.

#### Note

For information about Route ID, see *Route ID* in *System Interconnect*.



**Figure 8-9. DDRSS CoS Mapping Diagram**

#### 8.2.4.1.3 AXI Write Data All-Strobes

The DDR controller data path is connected to the MSMC2DDR bridge through AXI interface. To enable higher performance and lower latency, the bridge drives the AXI AWALLSTRB signal to a "1" when the AXI write data do not have any byte enable holes. This allows the DDR controller to accept and start processing the write command as soon as possible without waiting for all data. The DDRSS\_CTL\_377[16] AXI0\_ALL\_STROBES\_USED\_ENABLE bit must be set to a 0x1 to utilize this feature. The controller ignores the AWALLSTRB signal if AXI0\_ALL\_STROBES\_USED\_ENABLE is set to 0x0.

#### 8.2.4.1.4 Inline ECC for SDRAM Data

For SDRAM data integrity, the MSMC2DDR bridge supports inline ECC on the data written to or read from the SDRAM. ECC is enabled by programming the bits in the DDRSS\_ECC\_CTRL\_REG register. ECC is stored together with the data so that a dedicated SDRAM device for ECC is not required. When using this inline ECC feature the inline ECC inside the DDR controller must be disabled by setting to 0x0 the DDRSS\_CTL\_206[17-16] ECC\_ENABLE field.

8-bit single error correction double error detection (SECCDED) ECC is calculated over 64-bit data quanta. For every 512-byte data block 64 bytes of ECC is stored inline. Thus 1/9th of the total SDRAM space is used for ECC storage and the rest 8/9th is available for system use. From system point of view that 8/9th of the SDRAM

data space are seen as consecutive byte addresses. Even if there are non-ECC protected regions the previously described 1/9th-8/9th rule still applies and consecutive byte addresses are seen from system point of view.

The ECC is calculated for all accesses that are within the address ranges protected by ECC. The address ranges are specified through the following registers:

- DDRSS\_ECC\_R0\_STR\_ADDR\_REG
- DDRSS\_ECC\_R0\_END\_ADDR\_REG
- DDRSS\_ECC\_R1\_STR\_ADDR\_REG
- DDRSS\_ECC\_R1\_END\_ADDR\_REG
- DDRSS\_ECC\_R2\_STR\_ADDR\_REG
- DDRSS\_ECC\_R2\_END\_ADDR\_REG

The ECC is read and verified during reads if both the DDRSS\_ECC\_CTRL\_REG[0] ECC\_EN and DDRSS\_ECC\_CTRL\_REG[2] ECC\_CK bits are set to 0x1. For 1-bit ECC error, the MSMC2DDR bridge corrects the data and returns it to the requestor. Although the error is corrected on the returned data, the SDRAM is not corrected. It is responsibility of the system software to correct the ECC error at that location.

It is also responsibility of the system software to pre-load the ECC protected region with known data before functional reads and writes are performed. This can be done by writing to the SDRAM with ECC enabled (DDRSS\_ECC\_CTRL\_REG[0] ECC\_EN = 0x1 and DDRSS\_ECC\_CTRL\_REG[1] RMW\_EN = 0x1) and ECC check disabled (DDRSS\_ECC\_CTRL\_REG[2] ECC\_CK = 0x0). Once the data is loaded in the SDRAM, ECC check must be enabled (DDRSS\_ECC\_CTRL\_REG[2] ECC\_CK = 0x1) before using the DDR interface.

#### 8.2.4.1.4.1 ECC Cache

The MSMC2DDR bridge implements a 128-line deep ECC cache for improving inline ECC performance. Each cache line can be allocated to an initiator in the system based on its Route ID. The Route ID allocation to cache line can be done by writing to the DDRSS\_ECC\_RID\_INDX\_REG and DDRSS\_ECC\_RID\_VAL\_REG registers. Since the bridge uses unallocated cache lines for all read accesses without Route ID allocation, one or more locations in the cache should be kept unallocated for better performance. To ensure that at least one cache line remains unallocated, in the case all cache lines are allocated by software, the bridge automatically unallocates the 127th cache line. Write accesses without Route ID allocation result in ECC and data writes to the SDRAM, when the DDRSS\_ECC\_CTRL\_REG[4] WR\_ALLOC bit is set to 0x0. When DDRSS\_ECC\_CTRL\_REG[4] WR\_ALLOC is set to 0x1, an unassigned cache-line is allocated to write accesses without Route ID allocation.

#### 8.2.4.1.4.2 ECC Statistics

For 1-bit ECC error, the MSMC2DDR bridge logs the address location for the error in an internal 2-level deep FIFO. It stores the first two 1-bit ECC errors. The DDRSS\_ECC\_1B\_ERR\_ADR\_LOG\_REG register shows the address on top of the internal FIFO. Software should write 0x1 to the DDRSS\_ECC\_1B\_ERR\_ADR\_LOG\_REG[28-0] ECC\_1B\_ERR\_ADR field to pop the FIFO and display the next address stored. The FIFO is loaded with the address for the next 1-bit ECC error if it is not full. If a single address is associated with more than one ECC error, that address is logged twice.

The number of 1-bit ECC errors can be counted using the DDRSS\_ECC\_1B\_ERR\_CNT\_REG register. The MSMC2DDR bridge also supports programming a threshold in the DDRSS\_ECC\_1B\_ERR\_THRSH\_REG register. When the 1-bit error count is equal to or greater than the programmed threshold, the bridge sets the DDRSS\_V2A\_INT\_RAW\_REG[3] ECC1BERR bit and also triggers the DDR0\_DDRSS\_DRAM\_ECC\_CORR\_ERR\_LVL\_0 interrupt. When servicing the interrupt, software needs to clear the error count otherwise further interrupts will not be triggered. For 2-bit ECC errors, the bridge sets the DDRSS\_V2A\_INT\_RAW\_REG[4] ECC2BERR bit and also triggers the DDR0\_DDRSS\_DRAM\_ECC\_UNCORR\_ERR\_LVL\_0 interrupt. The bridge does not correct the data for these uncorrectable errors. Along with generating the interrupt, the bridge also reports an error to the requesting initiator and sends all zeros for the data. The bridge also logs the address location for the error in the DDRSS\_ECC\_2B\_ERR\_ADR\_LOG\_REG register.

The MSMC2DDR bridge sets the DDRSS\_V2A\_INT\_RAW\_REG[5] ECCM1BERR bit and triggers the DDR0\_DDRSS\_DRAM\_ECC\_UNCORR\_ERR\_LVL\_0 interrupt whenever it receives multiple 1-bit errors in



different data words of the same SDRAM burst. This is done because the probability of receiving multiple 1-bit errors in different data words of the same SDRAM burst is very low. If this occurs, the chances are that there were multi-bit errors in each data word. Therefore, for reliability, pessimistic approach is taken to report these as a fatal error. The threshold for the number of 1-bit errors that result in an uncorrected error, reporting can be set by writing to the DDRSS\_ECC\_CTRL\_REG[12-8] COR\_ECC\_THRESH field. For reliability, the threshold is always kept at 0/1 meaning 1-bit error in 2 or more data words is reported as a 2-bit error. For debug, the threshold can be changed to a higher value. Note that since these are multiple 1-bit errors, the statistic logging is done in the DDRSS\_ECC\_1B\_ERR\_CNT\_REG and DDRSS\_ECC\_1B\_ERR\_ADR\_LOG\_REG registers.

#### 8.2.4.1.5 Opcode Checking

The MSMC2DDR bridge checks the VBUSM.C opcode received for unsupported opcodes and flags error for debug purposes. Only opcode values for read and write operations are supported.

If an unsupported VBUSM.C opcode is received for an access, the bridge sets the DDRSS\_V2A\_INT\_RAW\_REG[0] OERR bit to 0x1 and triggers the DDR0\_DDRSS\_V2A\_OTHER\_ERR\_LVL\_0 interrupt. The opcode and the Route ID for the command caused error are logged in the DDRSS\_V2A\_OERR\_LOG\_REG[17-12] OERR\_OP\_CODE and DDRSS\_V2A\_OERR\_LOG\_REG[11-0] OERR\_ROUTE\_ID fields.

Since the bridge does not know how to process the command with an unsupported opcode, it discards the command and does not respond with any status on the write status or the read status buses.

#### 8.2.4.1.6 Address Alias Prevention

The MSMC2DDR bridge checks the VBUSM.C address received against the valid SDRAM address space programmed in the DDRSS\_V2A\_CTL\_REG register. If the VBUSM.C address for an access falls outside the programmed range the bridge sets to 0x1 the DDRSS\_V2A\_INT\_RAW\_REG[1] AERR bit and also triggers the DDR0\_DDRSS\_V2A\_OTHER\_ERR\_LVL\_0 interrupt. The address and the Route ID for the command caused error are logged in the DDRSS\_V2A\_AERR\_LOG1\_REG and DDRSS\_V2A\_AERR\_LOG2\_REG registers.

The valid address range can be programmed using the DDRSS\_V2A\_CTL\_REG[9-5] SDRAM\_IDX and DDRSS\_V2A\_CTL\_REG[4-0] REGION\_IDX fields. [Table 8-17](#) summarizes the scenarios for determining valid address range and interrupt generation.

**Table 8-17. REGION\_IDX and SDRAM\_IDX Scenarios**

Condition	Description
REGION_IDX = SDRAM_IDX	DDR region size is equal to the connected SDRAM size. It is SoC responsibility to ensure that the addresses received by the DDR subsystem do not fall outside the region size. No address error is generated if the address received falls outside the region size.
REGION_IDX < SDRAM_IDX	DDR region size is less than the connected SDRAM size. It is SoC responsibility to ensure that the addresses received by the DDR subsystem do not fall outside the region size. No address error is generated if the address received falls outside the region size.
REGION_IDX > SDRAM_IDX	DDR region size is greater than the connected SDRAM size. Address error is generated if the address received falls outside the connected SDRAM size.

A write access outside the programmed range is discarded. A write error associated status is sent back to the VBUSM.C interface.

A read access outside the programmed range is executed on the SDRAM interface as a normal read, but data will contain all zeroes before forwarding it to the VBUSM.C interface. A read error associated status is sent back to the VBUSM.C interface along with the read data containing all zeroes.

When inline ECC is enabled, the available SDRAM size is reduced by 1/9th of the size programmed in the SDRAM\_IDX field. Therefore, the bridge reports an error if an access falls outside the reduced SDRAM size when inline ECC is enabled. The reduced SDRAM size limit applies to all accesses, both to the protected and non-protected ECC regions.



#### 8.2.4.1.7 Data Error Detection and Correction

The MSMC2DDR bridge performs data error detection and correction (EDC) on write data received. The EDC is performed over 256-bit data quanta and uses a Hamming code algorithm supporting single error correction and double error detection.

If the write data has a single bit error the bridge indicates a single bit error through the ECC aggregator interrupts. The write data will be corrected and written to the SDRAM. The address, the error position, and the Route ID for the command caused error are logged in the DDRSS\_V2A\_1B\_ERR\_LOG1\_REG and DDRSS\_V2A\_1B\_ERR\_LOG2\_REG registers. The number of 1-bit EDC errors is kept in the DDRSS\_V2A\_1B\_ERR\_CNT\_REG register.

If the write data has a double bit error the bridge indicates a double bit error through the ECC aggregator interrupts. The write will be executed on the SDRAM interface as a normal write and the data inside the SDRAM will be corrupted. The address and the route ID for the command caused the error are logged in the DDRSS\_V2A\_2B\_ERR\_LOG1\_REG and DDRSS\_V2A\_2B\_ERR\_LOG2\_REG registers. The bridge does not respond with a write error associated status for 2-bit EDC errors.

The bridge implements four EDC checkers that check EDC on each 256-bit quanta of a 1024-bit data word, in a single clock cycle. Therefore, when ECC errors occur on multiple 256-bit quanta in the same 1024-bit word, the error reported and logged is for the least significant 256-bits. If different quanta in the same 1024-bit word have single and double bit errors simultaneously, both errors are reported via interrupts but the address and Route ID associated with only the double bit error are logged.

The bridge generates an EDC code for the read data and sends it along with its read response for checking by the receiving system master.

#### 8.2.4.1.8 AXI Bus Timeout

The MSMC2DDR bridge has a timeout counter that expires when no AXI transaction can be sent to the DDR controller or no AXI response is received from the controller for a programmed time interval. The counter only counts when there are pending commands in the bridge and the AXI bus is idle. Therefore, the bridge will not time out during low-power modes. The time interval can be programmed by writing to the DDRSS\_V2A\_BUS\_TO[23-0] BUS\_TIMER field.

Upon the expiry of the counter, the bridge terminates all pending commands in its internal FIFOs, returns error responses, sets the DDRSS\_V2A\_INT\_RAW\_REG[2] TOERR bit and triggers the DDR0\_DDRSS\_V2A\_OTHER\_ERR\_LVL\_0 interrupt.

After a timeout occurs, the bridge terminates any new commands received and returns error response. Writing a value of 0x0 to the DDRSS\_V2A\_BUS\_TO[23-0] BUS\_TIMER field exits the timeout mode. The other method to exit the timeout mode is to reset the DDRSS0, thus resetting the MSMC2DDR bridge, the DDR controller, and the DDR PHY.

#### 8.2.4.2 DDRSS Interrupts

The DDRSS0 generates the following interrupts:

- DDR0\_DDRSS\_PLL\_FREQ\_CHANGE\_REQ\_0
- DDR controller interrupts:
  - DDR0\_DDRSS\_CONTROLLER\_GLOBAL\_ERROR\_NONFATAL\_0
  - DDR0\_DDRSS\_CONTROLLER\_GLOBAL\_ERROR\_FATAL\_0
  - DDR0\_DDRSS\_CONTROLLER\_0
- DDR PHY interrupts:
  - DDR0\_DDRSS\_HS\_PHY\_GLOBAL\_ERROR\_0
- ECC aggregators interrupts:
  - DDR0\_DDRSS\_CTL\_ECC\_AGGR\_CORR\_ERR\_LVL\_0
  - DDR0\_DDRSS\_CTL\_ECC\_AGGR\_UNCORR\_ERR\_LVL\_0
  - DDR0\_DDRSS\_VBUS\_ECC\_AGGR\_CORR\_ERR\_LVL\_0

- DDR0\_DDRSS\_VBUS\_ECC\_AGGR\_UNCORR\_ERR\_LVL\_0
- DDR0\_DDRSS\_CFG\_ECC\_AGGR\_CORR\_ERR\_LVL\_0
- DDR0\_DDRSS\_CFG\_ECC\_AGGR\_UNCORR\_ERR\_LVL\_0
- MSMC2DDR bridge interrupts:
  - DDR0\_DDRSS\_DRAM\_ECC\_CORR\_ERR\_LVL\_0
  - DDR0\_DDRSS\_DRAM\_ECC\_UNCORR\_ERR\_LVL\_0
  - DDR0\_DDRSS\_V2A\_OTHER\_ERR\_LVL\_0

The MSMC2DDR bridge sets to 0x1 the DDRSS\_V2A\_INT\_RAW\_REG[0] OERR bit, if the VBUSM.C access has an unsupported opcode.

The MSMC2DDR bridge sets to 0x1 the DDRSS\_V2A\_INT\_RAW\_REG[1] AERR bit, if the VBUSM.C address for an access that falls outside the DDR space programmed in the DDRSS\_V2A\_CTL\_REG register.

The MSMC2DDR bridge sets to 0x1 the DDRSS\_V2A\_INT\_RAW\_REG[2] TOERR bit, if it detects a hang on its interface to the DDR controller.

The MSMC2DDR bridge sets to 0x1 the DDRSS\_V2A\_INT\_RAW\_REG[3] ECC1BERR bit, if the threshold for 1-bit ECC errors is met.

The MSMC2DDR bridge sets to 0x1 the DDRSS\_V2A\_INT\_RAW\_REG[4] ECC2BERR bit, in case of 2-bit errors for a read access performed within the SDRAM address range protected by ECC.

The DDRSS0 asserts particular interrupt line only if the interrupts are enabled by writing 0x1 to the corresponding bit in the DDRSS\_V2A\_INT\_SET\_REG register. The interrupts can be disabled by writing 0x1 to the corresponding bit in the DDRSS\_V2A\_INT\_CLR\_REG register.

When interrupts are enabled, the corresponding bits in the DDRSS\_V2A\_INT\_STAT\_REG register are also set if an interrupt condition occurs. The interrupts can be cleared once serviced by writing 0x1 to the corresponding bit in the DDRSS\_V2A\_INT\_STAT\_REG register as well as writing to the DDRSS\_V2A\_EOI\_REG register.

[Table 8-18](#) shows the events that are generated by the MSMC2DDR bridge.

**Table 8-18. MSMC2DDR Bridge Events**

Event Flag	Event Mask	Description
DDRSS_V2A_INT_RAW_REG[4] ECC2BERR DDRSS_V2A_INT_STAT_REG[4] ECC2BERR	DDRSS_V2A_INT_SET_REG[4] ECC2BERR_EN DDRSS_V2A_INT_CLR_REG[4] ECC2BERR_EN	Generated in case of 2-bit errors for a read access within the ECC protected SDRAM address range. For more information, see <a href="#">Section 8.2.4.1.4.2</a> .
DDRSS_V2A_INT_RAW_REG[3] ECC1BERR DDRSS_V2A_INT_STAT_REG[3] ECC1BERR	DDRSS_V2A_INT_SET_REG[3] ECC1BERR_EN DDRSS_V2A_INT_CLR_REG[3] ECC1BERR_EN	Generated in case of meeting the threshold for 1-bit ECC errors. For more information, see <a href="#">Section 8.2.4.1.4.2</a> .
DDRSS_V2A_INT_RAW_REG[2] TOERR DDRSS_V2A_INT_STAT_REG[2] TOERR	DDRSS_V2A_INT_SET_REG[2] TOERR_EN DDRSS_V2A_INT_CLR_REG[2] TOERR_EN	Generated in case of a hang of the interface between the MSMC2DDR bridge and the DDR controller. For more information, see <a href="#">Section 8.2.4.1.8</a> .
DDRSS_V2A_INT_RAW_REG[1] AERR DDRSS_V2A_INT_STAT_REG[1] AERR	DDRSS_V2A_INT_SET_REG[1] AERR_EN DDRSS_V2A_INT_CLR_REG[1] AERR_EN	Generated if the VBUSM.C address for an access falls outside the programmed range. For more information, see <a href="#">Section 8.2.4.1.6</a> .
DDRSS_V2A_INT_RAW_REG[0] OERR DDRSS_V2A_INT_STAT_REG[0] OERR	DDRSS_V2A_INT_SET_REG[0] OERR_EN DDRSS_V2A_INT_CLR_REG[0] OERR_EN	Generated if an unsupported VBUSM.C opcode is received for an access. For more information, see <a href="#">Section 8.2.4.1.5</a> .

### 8.2.4.3 DDRSS Memory Regions

Table 8-19 shows all memory regions associated with the DDRSS0.

**Table 8-19. DDRSS0 Memory Regions**

Region	Start Address	End Address	Region Size
DDRSS0 wrapper logic registers	0x00 0298 0000	0x00 0298 01FF	512 B
DDR controller registers	0x00 0299 0000	0x00 0299 1FFF	8 KB
DDR PHY independent module registers	0x00 0299 2000	0x00 0299 3FFF	8 KB
DDR PHY registers	0x00 0299 4000	0x00 0299 7FFF	16 KB
DDRSS0_ECC_AGGR_CTL registers	0x4D 200B 0000	0x4D 200B 03FF	1 KB
DDRSS0_ECC_AGGR_VBUS registers	0x4D 200B 0400	0x4D 200B 07FF	1 KB
DDRSS0_ECC_AGGR_CFG registers	0x4D 200B 0800	0x4D 200B 0BFF	1 KB
External SDRAM data space - region 0 (32-bit low memory space)	0x00 8000 0000	0x00 FFFF FFFF	2 GB
External SDRAM data space - region 1 (high memory space) <sup>(1)</sup>	0x08 0000 0000	0x09 FFFF FFFF	8 GB

(1) The first 2 GB of this region are inaccessible.

### 8.2.4.4 DDRSS ECC Support

The DDRSS0 aligns to the device architecture for system reliability and uses the following ECC aggregators for detection and reporting of ECC errors:

- DDRSS0\_ECC\_AGGR\_CTL
- DDRSS0\_ECC\_AGGR\_VBUS
- DDRSS0\_ECC\_AGGR\_CFG

For information about the ECC aggregator functionality, see *ECC Aggregator*.

### 8.2.4.5 DDRSS Dynamic Frequency Change Interface

This interface is used for handshaking between DDRSS0 and CTRL\_MMR0 to dynamically change DDR clock frequency to support LPDDR4 Frequency Set Point (FSP). The frequency change can be initiated either by a SoC processor or by the DDRSS0.

The associated CTRL\_MMR0 registers for processor initiated frequency change are:

- CTRLMMR\_CHNG\_DDR4\_FSP\_REQ
- CTRLMMR\_CHNG\_DDR4\_FSP\_ACK

The associated CTRL\_MMR0 registers for DDRSS0 initiated frequency change are:

- CTRLMMR\_DDR4\_FSP\_CLKCHNG\_REQ
- CTRLMMR\_DDR4\_FSP\_CLKCHNG\_ACK

The sequence for an SoC processor initiated frequency change is as follows:

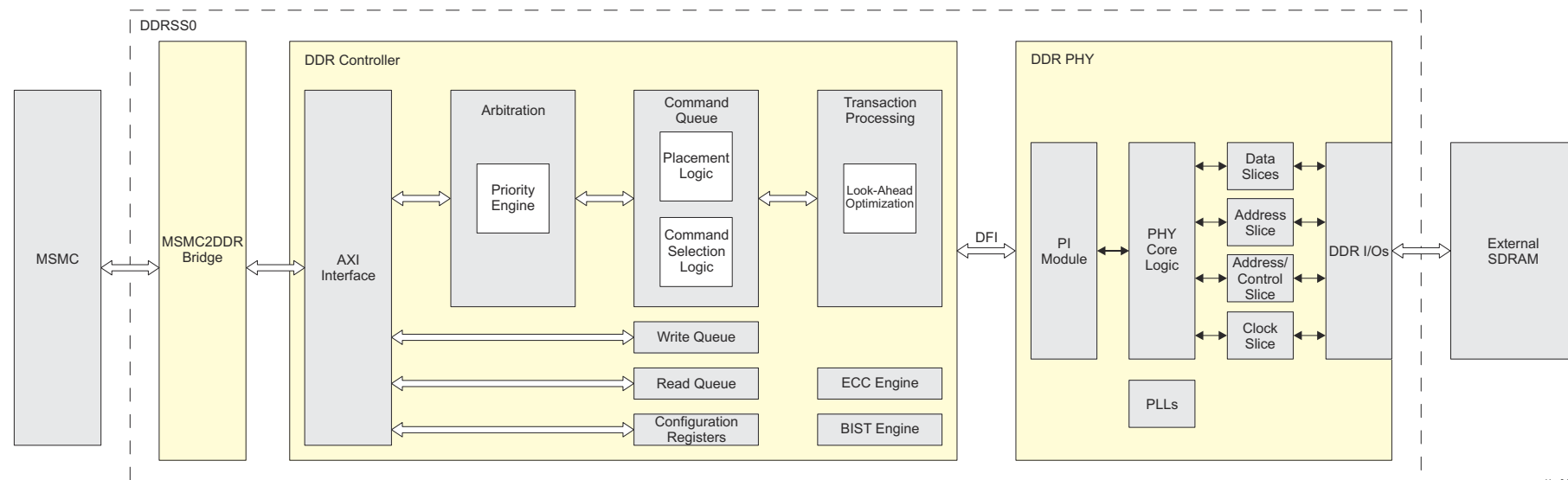
1. The processor writes the desired frequency to the CTRLMMR\_CHNG\_DDR4\_FSP\_REQ[1-0] REQ\_TYPE field.
2. The processor sets to 0x1 the CTRLMMR\_CHNG\_DDR4\_FSP\_REQ[8] REQ bit to initiate frequency change.
3. The processor programs PLL12 according to the CTRLMMR\_CHNG\_DDR4\_FSP\_REQ[1-0] REQ\_TYPE value.
4. The processor polls the CTRLMMR\_CHNG\_DDR4\_FSP\_ACK[7] ACK and CTRLMMR\_CHNG\_DDR4\_FSP\_ACK[0] ERROR bits to check if the frequency change has been completed successfully.

The sequence for DDRSS0 initiated frequency change is as follows:

1. DDRSS0 requests frequency change by asserting the CTRLMMR\_DDR4\_FSP\_CLKCHNG\_REQ[7] REQ bit and the CTRLMMR\_DDR4\_FSP\_CLKCHNG\_REQ[1-0] REQ\_TYPE field. The REQ bit is also connected to the DDR0\_DDRSS\_PLL\_FREQ\_CHANGE\_REQ\_0 interrupt.
2. Software reads the CTRLMMR\_DDR4\_FSP\_CLKCHNG\_REQ[1-0] REQ\_TYPE value and programs PLL12 accordingly.
3. After the DDRSS0\_FCLK has been changed software should set to 0x1 the CTRLMMR\_DDR4\_FSP\_CLKCHNG\_ACK[0] ACK bit.

#### 8.2.4.6 DDR Controller Functional Description

Figure 8-10 shows the DDR controller blocks along with the DDR PHY and MSMC2DDR bridge.



The ECC Engine block of the DDR controller is not supported.

BIST Engine is not supported.

**Figure 8-10. DDR Controller Functional Blocks**

#### 8.2.4.6.1 DDR PHY Interface (DFI)

The DDR controller and DDR PHY are connected via DDR PHY Interface (DFI) which is used for transferring control information and data between the PHY and the controller. There are DFI programmable parameters which software can interact with through some of the DDR controller (see *DDR Controller Registers*), PI (see *PI Registers*) and DDR PHY (see *DDR PHY Registers*) registers. For more information about DFI, see <http://www.ddr-phy.org/>.

#### 8.2.4.6.2 Command Queue

The DDR controller contains a command queue which uses a placement algorithm to determine the order of commands to be placed into the command queue. The placement logic follows many rules to determine where new commands should be inserted into the queue, relative to the contents of the command queue at a time. Placement is determined by considering address collisions, source collisions, data collisions, command types and priorities. The placement logic also attempts to maximize efficiency of the DDR controller through command grouping and bank splitting.

##### 8.2.4.6.2.1 Placement Logic

The placement logic is a 2-stage queue which determines the order of commands that run in the DDR controller. The placement logic follows rules for determining placement of new commands into the queue, relative to the contents of the command queue at that time. Placement is determined by considering coherency, address collisions, source collisions, data collisions, user assigned priority, latency, age, and command type to offer low latency for critical masters while optimizing bandwidth for all masters. A second reordering stage allows ready-to-run commands to start even if the head-of-queue command is not yet ready to run. Some of the rules used in the placement logic can be individually enabled or disabled via the DDRSS\_CTL\_274 through DDRSS\_CTL\_277 registers. In addition, the queue can be disabled completely, resulting in an in-line queue that services requests in the order that they arrive.

The DDR controller also has a full look-ahead facility that reduces the effect of page misses by pre-conditioning rows for upcoming requests by using “spare” cycles in preceding transactions.

##### 8.2.4.6.2.2 Command Selection Logic

After a command is in the command queue, command selection logic determines the method of pulling commands from the queue for running. On each clock cycle, the selection logic scans the entries of the command queue for determining the command to run. Commands for running are based on bank readiness, availability of at least 1 burst of data (writes), availability of storage for at least 1 burst of data (reads), bus turnaround timing, and conflicts. Similar to the placement rules, a command does not run before a command that was placed ahead of it in the command queue if it conflicts with address, source ID, or bank commands. Lower priority commands can run ahead of higher priority commands if the higher priority commands are not ready to run, as long as they do not conflict with commands that are ahead in the command queue.

##### 8.2.4.6.3 Low Power Control

The DDR controller contains an arbitration block that has software programmable interface for low power control. Placement of and removal from the various memory low power modes is controlled through programming of registers in the DDR controller. The user can also monitor the status of the memory devices through a programmable register. This interface supports a lock option allowing software to execute additional commands through this interface without worrying about state changes through other interfaces.

##### 8.2.4.6.4 Transaction Processing

The transaction command processing logic is used to process the commands in the command queue. The logic organizes the commands to the memories in such a way that data throughput is maximized. Bank opening and closing cycles are used for data transfers. The logic reviews the entire command queue for look-ahead of which banks have to be accessed in the future and ensures that the programmed memory timing conditions are met. This flexibility allows the controller to be tuned to extract the maximum performance out of memories. During processing, the controller examines the commands and issues the appropriate set of signals to the memory.

##### 8.2.4.6.5 BIST Engine

---

#### Note

BIST Engine is not supported.

---

The DDR controller has a BIST engine that supports MOV13N and limited MOV1 algorithms, a self-refresh retention test, an idle retention test and a memory initialization test for detecting the following faults:

- All address decoder faults
- All memory cell stuck-at faults
- All transition faults
- Most unlinked inversion coupling faults
- Most unlinked idem potent coupling faults
- Most idem potent coupling that are linked with inversion coupling faults

The DDR controller also supports BIST on ECC lanes for out-of-band ECC configuration. The following are the BIST related registers:

- DDRSS\_CTL\_194 through DDRSS\_CTL\_205
- DDRSS\_CTL\_302 through DDRSS\_CTL\_311

#### 8.2.4.6.6 ECC Engine

##### Note

The ECC Engine block of the DDR controller is not supported.

The ECC engine can confirm the data accuracy and remove or identify bit errors if they occur. It checks for errors in both the data and the check code on all read transactions. The DDR controller can detect single-bit and double-bit data errors and can correct single-bit errors. The ECC engine supports interrupts, register storage of ECC error signatures, signaling of ECC errors, ECC scrubbing and write-back, automatic ECC corruption and ECC error forcing. The DDR controller supports inline ECC, which means a portion of the memory device connected to the controller is reserved for storing the ECC check codes and is not available for user data storage. ECC can be enabled or disabled through the DDRSS\_CTL\_206[17-16] ECC\_ENABLE field. When enabled, all read data is checked for ECC (and optionally corrected) and ECC is computed and stored on all write data. Single-bit errors can be corrected and double-bit errors can be flagged. ECC information is not returned and ECC scrubbing is supported to maintain memory contents. The DDRSS\_CTL\_206 through DDRSS\_CTL\_227 registers contain the ECC related software controls.

#### 8.2.4.6.7 Address Mapping

The DDR controller automatically maps SoC addresses to the SDRAM memory in a contiguous block. Addressing starts at address 0x0 and ends at the highest available address according to the size and number of SDRAM memories present. This mapping depends on the programmed values in the DDR controller registers and is based on the actual size of the available SDRAM memories. The size is configured via the DDRSS\_CTL\_286[2-0] MEMDATA\_RATIO\_0 and DDRSS\_CTL\_287[10-8] MEMDATA\_RATIO\_1 fields and that must be initialized at power-up.

#### 8.2.4.6.8 Paging Policy

The DDR controller offers a flexible paging policy that allows open page operation, closed page operation, or an auto-precharge-per-command option that allows both modes simultaneously. Auto-precharge-per-command allows marking a particular master or transaction (for example, a CPU cache) as a closed-page transaction. This type of transaction reduces power and improves latency for the next transaction to a different row in the same SDRAM bank. Other transactions can be marked as open page transactions. This type of transaction reduces power and improves latency and bandwidth to the next transaction to the same row in the same SDRAM bank.

#### 8.2.4.6.9 DDR Controller Initialization

Once the power to the SDRAM and SoC is stable, the DDR controller must be initialized. It then automatically initializes the external memory. The general steps to initialize the DDR controller are as follows:

1. Reset the DDRSS0 through the DDRSS0\_RST signal. This resets all DDR controller registers.
2. Issue write register commands to configure the SDRAM protocols and the settings for the DDR controller. Keep the DDRSS\_CTL\_0[0] START bit to 0x0 during this initialization step.
3. Set to 0x1 the DDRSS\_CTL\_0[0] START bit. This triggers the controller to execute initialization sequence using the settings written to its registers. If the DDRSS\_CTL\_88[0] PWRUP\_SREFRESH\_EXIT bit is set to 0x0, the DDR controller issues MRWs and requests a ZQCL.



The DDR controller waits for the DDR PHY to assert the `dfi_init_complete` signal, which indicates that the PHY and SDRAM are ready to accept commands. When this signal asserts, the controller begins its initialization routine. The DDR controller sets to 0x1 bit [9] in the `DDRSS_CTL_293[31-0] INT_STATUS_0` field when initialization is complete and it is ready to accept commands.

For information about the DDR PHY initialization, see [Section 8.2.4.7.5](#).

#### **8.2.4.6.10 Programming LPDDR4 Memories**

LPDDR4 memories are specified to run at high frequency and include additional features such as CA training, write leveling, and ODT. At these high speeds, the DDR PHY also supports read data-eye training and gate training. Leveling and training operations are all performed through the PI module.

##### **8.2.4.6.10.1 Frequency Set Point (FSP)**

LPDDR4 memories use frequency set points to allow the memory to easily switch between two different frequencies without ever being in an un-trained state. There are two frequency set points - frequency set point 0 and frequency set point 1.

FSP is controlled through the following bits:

- `DDRSS_CTL_192[0] DFS_ALWAYS_WRITE_FSP`
- `DDRSS_CTL_163[16] DISABLE_UPDATE_TVRCG`
- `DDRSS_CTL_192[16] FSP_OP_CURRENT`
- `DDRSS_CTL_191[24] FSP_PHY_UPDATE_MRW`
- `DDRSS_CTL_192[8] FSP_STATUS`
- `DDRSS_CTL_192[24] FSP_WR_CURRENT`
- `DDRSS_CTL_193[17-16] FSP0_FRC`
- `DDRSS_CTL_193[25-24] FSP1_FRC`
- `DDRSS_CTL_190[8] MR_FSP_DATA_VALID_F0`
- `DDRSS_CTL_190[16] MR_FSP_DATA_VALID_F1`
- `DDRSS_CTL_190[24] MR_FSP_DATA_VALID_F2`
- `DDRSS_CTL_166[4-0] TCKFSPE_F0`
- `DDRSS_CTL_168[4-0] TCKFSPE_F1`
- `DDRSS_CTL_171[4-0] TCKFSPE_F2`
- `DDRSS_CTL_166[12-8] TCKFSPX_F0`
- `DDRSS_CTL_168[28-24] TCKFSPX_F1`
- `DDRSS_CTL_171[12-8] TCKFSPX_F2`
- `DDRSS_CTL_166[31-16] TVREF_LONG_F0`
- `DDRSS_CTL_169[15-0] TVREF_LONG_F1`
- `DDRSS_CTL_171[31-16] TVREF_LONG_FN`

##### **8.2.4.6.10.1.1 FSP Mode Register Programming During Initialization**

Before initialization, the user should program the FSP-related mode registers (MR1, MR2, MR3, MR11, MR12, MR13, MR14 and MR22) with the relevant values, and the `DDRSS_CTL_21[28-24] DFIBUS_FREQ_F2`, `DDRSS_CTL_21[20-16] DFIBUS_FREQ_F1` and `DDRSS_CTL_21[12-8] DFIBUS_FREQ_F0` fields with the frequency set to use. These mode registers are programmed through the following fields:

- `DDRSS_CTL_175[7-0] MR1_DATA_F0_0`
- `DDRSS_CTL_175[23-16] MR1_DATA_F1_0`
- `DDRSS_CTL_176[7-0] MR1_DATA_F2_0`
- `DDRSS_CTL_182[23-16] MR1_DATA_F0_1`
- `DDRSS_CTL_183[7-0] MR1_DATA_F1_1`
- `DDRSS_CTL_183[23-16] MR1_DATA_F2_1`
- `DDRSS_CTL_175[15-8] MR2_DATA_F0_0`
- `DDRSS_CTL_175[31-24] MR2_DATA_F1_0`
- `DDRSS_CTL_176[15-8] MR2_DATA_F2_0`

- DDRSS\_CTL\_182[31-24] MR2\_DATA\_F0\_1
- DDRSS\_CTL\_183[15-8] MR2\_DATA\_F1\_1
- DDRSS\_CTL\_183[31-24] MR2\_DATA\_F2\_1
- DDRSS\_CTL\_176[31-24] MR3\_DATA\_F0\_0
- DDRSS\_CTL\_177[7-0] MR3\_DATA\_F1\_0
- DDRSS\_CTL\_177[15-8] MR3\_DATA\_F2\_0
- DDRSS\_CTL\_184[15-8] MR3\_DATA\_F0\_1
- DDRSS\_CTL\_184[23-16] MR3\_DATA\_F1\_1
- DDRSS\_CTL\_184[31-24] MR3\_DATA\_F2\_1
- DDRSS\_CTL\_178[23-16] MR11\_DATA\_F0\_0
- DDRSS\_CTL\_178[31-24] MR11\_DATA\_F1\_0
- DDRSS\_CTL\_179[7-0] MR11\_DATA\_F2\_0
- DDRSS\_CTL\_186[7-0] MR11\_DATA\_F0\_1
- DDRSS\_CTL\_186[15-8] MR11\_DATA\_F1\_1
- DDRSS\_CTL\_186[23-16] MR11\_DATA\_F2\_1
- DDRSS\_CTL\_179[15-8] MR12\_DATA\_F0\_0
- DDRSS\_CTL\_179[23-16] MR12\_DATA\_F1\_0
- DDRSS\_CTL\_179[31-24] MR12\_DATA\_F2\_0
- DDRSS\_CTL\_186[31-24] MR12\_DATA\_F0\_1
- DDRSS\_CTL\_187[7-0] MR12\_DATA\_F1\_1
- DDRSS\_CTL\_187[15-8] MR12\_DATA\_F2\_1
- DDRSS\_CTL\_180[7-0] MR13\_DATA\_0
- DDRSS\_CTL\_180[15-8] MR14\_DATA\_F0\_0
- DDRSS\_CTL\_180[23-16] MR14\_DATA\_F1\_0
- DDRSS\_CTL\_180[31-24] MR14\_DATA\_F2\_0
- DDRSS\_CTL\_187[31-24] MR14\_DATA\_F0\_1
- DDRSS\_CTL\_188[7-0] MR14\_DATA\_F1\_1
- DDRSS\_CTL\_188[15-8] MR14\_DATA\_F2\_1
- DDRSS\_CTL\_181[31-24] MR22\_DATA\_F0\_0
- DDRSS\_CTL\_182[7-0] MR22\_DATA\_F1\_0
- DDRSS\_CTL\_182[15-8] MR22\_DATA\_F2\_0
- DDRSS\_CTL\_189[15-8] MR22\_DATA\_F0\_1
- DDRSS\_CTL\_189[23-16] MR22\_DATA\_F1\_1
- DDRSS\_CTL\_189[31-24] MR22\_DATA\_F2\_1

In addition, the following fields must also be programmed before asserting the DDRSS\_CTL\_0[0] START bit:

- DDRSS\_CTL\_165[25-16] TFC\_F0
- DDRSS\_CTL\_166[4-0] TCKFSPE\_F0
- DDRSS\_CTL\_166[12-8] TCKFSPX\_F0
- DDRSS\_CTL\_164[25-16] TVRCG\_ENABLE\_F0
- DDRSS\_CTL\_165[9-0] TVRCG\_DISABLE\_F0
- DDRSS\_CTL\_168[9-0] TFC\_F1
- DDRSS\_CTL\_168[20-16] TCKFSPE\_F1
- DDRSS\_CTL\_160[28-24] TCKFSPX\_F1
- DDRSS\_CTL\_167[9-0] TVRCG\_ENABLE\_F1
- DDRSS\_CTL\_167[25-16] TVRCG\_DISABLE\_F1
- DDRSS\_CTL\_170[25-16] TFC\_F2
- DDRSS\_CTL\_171[4-0] TCKFSPE\_F2
- DDRSS\_CTL\_171[12-8] TCKFSPX\_F2
- DDRSS\_CTL\_169[25-16] TVRCG\_ENABLE\_F2
- DDRSS\_CTL\_170[9-0] TVRCG\_DISABLE\_F2
- DDRSS\_CTL\_21[1-0] DFIBUS\_BOOT\_FREQ
- DDRSS\_CTL\_20[25-24] DFIBUS\_FREQ\_INIT

- DDRSS\_CTL\_149[24] DFS\_ENABLE
- DDRSS\_CTL\_191[24] FSP\_PHY\_UPDATE\_MRW

The following bits may or may not be modified, depending on what operations are needed after a frequency change occurs:

- DDRSS\_CTL\_192[0] DFS\_ALWAYS\_WRITE\_FSP
- DDRSS\_CTL\_89[16] DFS\_ZQ\_EN

Once the DDRSS\_CTL\_0[0] START bit is asserted, the DDR controller programs the mode registers in the memory devices with these values using the FSP-OP and FSP-WR values from the DDRSS\_CTL\_180[7-0] MR13\_DATA\_0 field. Once initialization is complete, either the PHY, PI, or controller will have updated the frequency set points with the needed values.

#### 8.2.4.6.10.1.2 FSP Mode Register Programming During Normal Operation

During normal operation, the DDR controller may modify the FSP mode register values in memory. To do this, the user should program the DDRSS\_CTL\_164[1-0] MRW\_DFS\_UPDATE\_FRC field with the frequency set that the DDR controller is using and program the appropriate new values in the FSP-related mode registers previously listed. Since there are two copies of each of these mode registers in the memory device (other than MR13), the user should also identify the appropriate frequency set points for operation and writing in the DDRSS\_CTL\_192[16] FSP\_OP\_CURRENT and DDRSS\_CTL\_192[24] FSP\_WR\_CURRENT bits.

Generally, the user should not modify the mode registers for the frequency set being used, and therefore should program the DDRSS\_CTL\_192[24] FSP\_WR\_CURRENT with the opposite value of the DDRSS\_CTL\_192[16] FSP\_OP\_CURRENT bit.

Once established the values to write, the frequency set in the DDR controller to use, and the frequency set points, the user should program the DDRSS\_CTL\_159[26-0] WRITE\_MODEREG field for the operation:

- Set bit [24] to 1h for all chips to be written with new values, or identify the target chip select in bits [15-8].
- Set bit [26] to 1h for a FSP-write operation. A write to this bit targets the mode registers MR1, MR2, MR3, MR11 and MR22 for writing. MR12 and MR14 are also written if the DDRSS\_CTL\_191[24] FSP\_PHY\_UPDATE\_MRW bit is set to 0h.
- Set bit [25] to 1h to trigger the operation.

All of these bits may be programmed simultaneously.

#### Note

When the DDRSS\_CTL\_159[26-0] WRITE\_MODEREG field is written with bit [26] set to 1h, the corresponding MR13 mode register is updated automatically with the value in the DDRSS\_CTL\_180[7-0] MR13\_DATA\_0 or DDRSS\_CTL\_187[23-16] MR13\_DATA\_1 fields, except that the value in the DDRSS\_CTL\_192[24] FSP\_WR\_CURRENT bit will define the value written in MR13 [6] and the value in the DDRSS\_CTL\_192[16] FSP\_OP\_CURRENT bit will define the value written in MR13 [7]. This write occurs before the other mode register writes, which means that the user may inadvertently write the wrong frequency set point mode or switch to the new frequency set point.

#### 8.2.4.6.10.1.3 FSP Mode Register Programming During Dynamic Frequency Scaling

During a hardware DFS operation, the FSP mode registers are used to switch from one frequency set to the other. Before a hardware frequency change can be requested, the following must be programmed:

- DDRSS\_CTL\_193[17-16] FSP0\_FRC
- DDRSS\_CTL\_193[25-24] FSP1\_FRC
- DDRSS\_CTL\_192[24] FSP\_WR\_CURRENT
- DDRSS\_CTL\_192[16] FSP\_OP\_CURRENT
- DDRSS\_CTL\_190[24] MR\_FSP\_DATA\_VALID\_F2
- DDRSS\_CTL\_190[16] MR\_FSP\_DATA\_VALID\_F1
- DDRSS\_CTL\_190[8] MR\_FSP\_DATA\_VALID\_F0

FSP mode register writes are required as part of the DFS operation. To accomplish this, three mode register writes are issued to MR13, and additional mode register write to each of the FSP mode registers. These commands are issued at three times:

1. After frequency change entry and before the memory is placed into self refresh the first mode register write to MR13 is issued. This writes the value from the DDRSS\_CTL\_180[7-0] MR13\_DATA\_0 and DDRSS\_CTL\_187[23-16] MR13\_DATA\_1 fields for all bits other than the FSP-WR bit which will be defined as opposite of the FSP-OP bit. Then mode register writes to all the other FSP mode registers are issued.
2. After the memory is in self-refresh, the MR13 is written again. This write causes the FSP-OP bit to toggle, and to set the VRCG bit. At this point, the frequency change should take place.
3. After the frequency change, one final mode register write to MR13 is performed to clear the VRCG bit. The DFS operation will update the FSP mode registers and the DDRSS\_CTL\_192[8] FSP\_STATUS bit once it is complete.

#### 8.2.4.6.10.2 Data Bus Inversion (DBI)

LPDDR4 memories support data bus inversion (DBI) for signal integrity and power savings. The DDR controller can support DBI on the read or write path, or on both. DBI functions by ensuring that in each data byte, there are no more than four '1' bits. This can be used to confirm data transmission, and also for power savings.

DBI is controlled through the following:

- The DDRSS\_CTL\_291[16] RD\_DBI\_EN bit
- The DDRSS\_CTL\_291[8] WR\_DBI\_EN bit

Before enabling controller-level data bus inversion, the DBI mode must be enabled in the memories through a write to bits [7-6] of MR3. The system behavior depends on the value of the memory mode register bits, the DDRSS\_CTL\_291[16] RD\_DBI\_EN or DDRSS\_CTL\_291[8] WR\_DBI\_EN bit, and the value of each byte of data.

If the DDRSS\_CTL\_291[8] WR\_DBI\_EN bit is set to 1h, the user should ensure that the memory mode register WR DBI bit is also set to 1h. When both of these bits are set, the DDR controller examines each byte of the incoming write data and identifies the number of '1' bits within each byte. If a byte of data has 5 or more '1' bits, the DDR controller automatically flips the data bits in that byte and drives the dfi\_wrddata\_mask/dfi\_wrddata\_mask\_p1 signal that corresponds to that byte to '1'. If the number of '1' bits within a byte is 4 or less or the DDRSS\_CTL\_291[8] WR\_DBI\_EN bit is set to 0h, then the data is sent to the DRAMs unchanged. On the DRAM, if the dfi\_wrddata\_mask/dfi\_wrddata\_mask\_p1 signal is driven to '1', the DRAM re-inverts that byte before storing the data and if the dfi\_wrddata\_mask/dfi\_wrddata\_mask\_p1 signal is set to '0', the DRAM stores the data as received.

For the read path, if the memory mode register RD DBI bit is set to 1h, the DDRSS\_CTL\_291[16] RD\_DBI\_EN bit should also be set to 1h. When both of these bits are set, the DDR controller examines the input signal dfi\_rddata\_dbi\_n/dfi\_rddata\_dbi\_n\_w1. If the signal is driven to '1' for a byte, the DDR controller inverts the associated byte before sending it to the requestor. If the dfi\_rddata\_dbi\_n/dfi\_rddata\_dbi\_n\_w1 signal is held to '0' for a byte, the data is sent out to the requestor unchanged.

#### 8.2.4.6.10.3 On-Die Termination

LPDDR4 memories provide two different types of on-die termination (ODT) - ODT for the DQ and ODT for the CA buses.

##### 8.2.4.6.10.3.1 LPDDR4 DQ ODT

LPDDR4 supports termination for the DQ and DQS input signals, but this termination does not utilize an ODT input pin. Instead, the ODT is controlled by the command. The functionality is enabled through bits [2-0] of mode register MR11, where a value of 0h disables the ODT functionality. The DQ ODT settings can be programmed through the following fields:

- DDRSS\_CTL\_178[23-16] MR11\_DATA\_F0\_0
- DDRSS\_CTL\_186[7-0] MR11\_DATA\_F0\_1
- DDRSS\_CTL\_178[31-24] MR11\_DATA\_F1\_0
- DDRSS\_CTL\_186[15-8] MR11\_DATA\_F1\_1

- DDRSS\_CTL\_179[7-0] MR11\_DATA\_F2\_0
- DDRSS\_CTL\_186[23-16] MR11\_DATA\_F2\_1

These field values are written to the associated mode registers when the DDR controller issues a MRW to the memory.

When DQ ODT is enabled, RTT is asserted with a write or masked write command.

#### **8.2.4.6.10.3 LPDDR4 CA ODT**

LPDDR4 can terminate the CA bus, including the CS and CLK signals. The value on the ODT input pin, ODT\_CA, on the DRAM and the settings in the mode registers determine if CA ODT is enabled in the memories and the RTT termination resistance. The functionality is enabled through bits [6-4] of mode register MR11, where the value of 0h disables ODT functionality. Bits [5-3] of MR22 control individual aspects of the CA ODT when ODT is not disabled. The CA ODT settings can be programmed through the following fields:

- DDRSS\_CTL\_178[23-16] MR11\_DATA\_F0\_0
- DDRSS\_CTL\_178[31-24] MR11\_DATA\_F1\_0
- DDRSS\_CTL\_179[7-0] MR11\_DATA\_F2\_0
- DDRSS\_CTL\_186[7-0] MR11\_DATA\_F0\_1
- DDRSS\_CTL\_186[15-8] MR11\_DATA\_F1\_1
- DDRSS\_CTL\_186[23-16] MR11\_DATA\_F2\_1
- DDRSS\_CTL\_181[31-24] MR22\_DATA\_F0\_0
- DDRSS\_CTL\_182[7-0] MR22\_DATA\_F1\_0
- DDRSS\_CTL\_182[15-8] MR22\_DATA\_F2\_0
- DDRSS\_CTL\_189[15-8] MR22\_DATA\_F0\_1
- DDRSS\_CTL\_189[23-16] MR22\_DATA\_F1\_1
- DDRSS\_CTL\_189[31-24] MR22\_DATA\_F2\_1

These field values are written to the associated mode registers when the DDR controller issues a MRW to the memory.

#### **8.2.4.6.10.4 Byte Lane Swapping**

The DDR controller supports byte lane swapping on the board or a DIMM for better timing or routing. In most cases, this swapping is invisible to the user as data byte lanes are swapped when data is being written, and then unswapped as data is read. However, mode register reads does not un-swap the data. Therefore, the user must be aware of the location of the 0<sup>th</sup> byte of the data in order to process the information from the mode registers correctly.

Whether byte lane swapping is used or not, the user must program the following fields:

- DDRSS\_CTL\_286[11-8] DEVICE0\_BYTE0\_CS0
- DDRSS\_CTL\_286[19-16] DEVICE1\_BYTE0\_CS0
- DDRSS\_CTL\_286[27-24] DEVICE2\_BYTE0\_CS0
- DDRSS\_CTL\_287[3-0] DEVICE3\_BYTE0\_CS0
- DDRSS\_CTL\_287[19-16] DEVICE0\_BYTE0\_CS1
- DDRSS\_CTL\_287[27-24] DEVICE1\_BYTE0\_CS1
- DDRSS\_CTL\_288[3-0] DEVICE2\_BYTE0\_CS1
- DDRSS\_CTL\_288[11-8] DEVICE3\_BYTE0\_CS1

Unique parameters are provided for each chip select to allow for different byte swapping on each chip select. Each field contains one bit for each byte of the memory data bus. If a device is not being used, the DEVICEx\_BYTE0\_CSy field should be programmed to 0h, but if the device is being used, then only one bit of the DEVICEx\_BYTE0\_CSy field should be set to 1h.

#### **8.2.4.6.10.5 DQS Interval Oscillator**

The DDR controller includes a feature to control the internal DQS clock tree oscillator in the LPDDR4 memories. This oscillator is used to track the delay variance of the DQS clock tree that will occur over time due to temperature and voltage. This feature allows the oscillator to be initiated and results read. If the result is outside

of the programmed allowable variance, the DDR controller informs the PHY. The DDR controller does not perform any training functions as a part of or a result of this oscillator measurement - the PHY is expected to handle all training requirements.

The DQS oscillator is controlled through the following fields:

- DDRSS\_CTL\_26[16] DQS\_OSC\_ENABLE
- DDRSS\_CTL\_28[31-24] DQS\_OSC\_HIGH\_THRESHOLD
- DDRSS\_CTL\_28[23-16] DQS\_OSC\_NORM\_THRESHOLD
- DDRSS\_CTL\_27[14-0] DQS\_OSC\_PERIOD
- DDRSS\_CTL\_29[15-8] DQS\_OSC\_PROMOTE\_THRESHOLD
- DDRSS\_CTL\_30[0] DQS\_OSC\_REQUEST
- DDRSS\_CTL\_29[7-0] DQS\_OSC\_TIMEOUT
- DDRSS\_CTL\_27[19-16] FUNC\_VALID\_CYCLES
- DDRSS\_CTL\_30[23-8] OSC\_BASE\_VALUE\_0\_CS0
- DDRSS\_CTL\_31[15-0] OSC\_BASE\_VALUE\_1\_CS0
- DDRSS\_CTL\_31[31-16] OSC\_BASE\_VALUE\_2\_CS0
- DDRSS\_CTL\_32[15-0] OSC\_BASE\_VALUE\_3\_CS0
- DDRSS\_CTL\_32[31-16] OSC\_BASE\_VALUE\_0\_CS1
- DDRSS\_CTL\_33[15-0] OSC\_BASE\_VALUE\_1\_CS1
- DDRSS\_CTL\_33[31-16] OSC\_BASE\_VALUE\_2\_CS1
- DDRSS\_CTL\_34[15-0] OSC\_BASE\_VALUE\_3\_CS1
- DDRSS\_CTL\_29[31-16] OSC\_VARIANCE\_LIMIT
- DDRSS\_CTL\_27[31-24] TOSCO\_F0
- DDRSS\_CTL\_28[7-0] TOSCO\_F1
- DDRSS\_CTL\_28[15-8] TOSCO\_F2

Once the DDR controller has been initialized, before the controller reaches its terminal power-on state, a request is generated to the oscillator module to set the initial base values. These values are read from the memories and stored in the OSC\_BASE\_VALUE\_x\_CSy fields.

Following a frequency change, the base values to be used for comparison must be updated. As part of the operation, the frequency change task generates a subtask request to the DQS oscillator. This request, like the initialization request, will run the oscillator and then load the resulting values into the OSC\_BASE\_VALUE\_x\_CSy fields.

Once the base values have been updated, the DDR controller sets to 1h bit [31] in the DDRSS\_CTL\_293[31-0] INT\_STATUS\_0 field indicating DQS oscillator base value update interrupt.

#### 8.2.4.6.10.5.1 Oscillator State Machine

A state machine inside the DDR controller tracks oscillator-related commands and wait times. The basic flow is as follows:

1. If the oscillator function is enabled by setting the DDRSS\_CTL\_26[16] DQS\_OSC\_ENABLE bit to 1h, the process begins when a request is received to run the oscillator function. This request can be issued automatically during the initialization process, during a frequency change process, as a software request by setting the DDRSS\_CTL\_30[0] DQS\_OSC\_REQUEST bit to 1h, or through a programmable periodic timer (the DDRSS\_CTL\_28[23-16] DQS\_OSC\_NORM\_THRESHOLD, DDRSS\_CTL\_28[31-24] DQS\_OSC\_HIGH\_THRESHOLD, or DDRSS\_CTL\_29[7-0] DQS\_OSC\_TIMEOUT fields). Once this process has started, the memory is not allowed to enter any low power modes until complete.
2. If the request is through initialization, MR23 is written with the value of the DDRSS\_CTL\_190[7-0] MR23\_DATA field defining the number of cycles to run the oscillator.
3. An MPC is issued to start the oscillator. If there are multiple DRAM devices on a rank, the MPC is issued to all devices on the rank at the same time, and then the state machine proceeds to the next rank and issues MPCs to that rank, and so on.



4. The state machine waits the cycles defined by the DDRSS\_CTL\_27[14-0] DQS\_OSC\_PERIOD field and the cycles specified in the DDRSS\_CTL\_27[31-24] TOSCO\_F0, DDRSS\_CTL\_28[7-0] TOSCO\_F1, and DDRSS\_CTL\_28[15-8] TOSCO\_F2 fields to allow the operation to complete and the results to be stored.
5. The state machine issues a read to MR18.
6. The state machine waits the number of cycles specified in the DDRSS\_CTL\_51[27-24] TMRR field, and then stores the result in a temporary location. If there are multiple DRAM devices on a rank, the MRR captures the mode register content for all devices on that rank.
7. The state machine issues a read to MR19.
8. The state machine waits the number of cycles specified in the DDRSS\_CTL\_51[27-24] TMRR field, and then stores the result in a temporary location. If there are multiple DRAM devices on a rank, the MRR captures the mode register content for all devices on that rank.
9. The values read in MR18 and MR19 represent the LSB and MSB of the DQS oscillator count. The DDR controller combines them and compares the result. If multiple devices exist on a rank, the comparison is made between each device and the base value.
  - a. If this is the initial read or a read related to a DFS operation, the values are stored in the OSC\_BASE\_VALUE\_x\_CSy fields. The DDR controller drives a 1h value on the dfi\_function signal and asserts the dfi\_function\_valid signal for the number of cycles defined by the DDRSS\_CTL\_27[19-16] FUNC\_VALID\_CYCLES field.
  - b. If the value is FFFFh, an overflow condition occurred during the measurement process. The DDR controller sets to 1h bit [0] in the DDRSS\_CTL\_294[12-0] INT\_STATUS\_1 field indicating DQS oscillator measurement overflow interrupt and ignores the value.
  - c. If the difference between the measured value and the base value is lower than or equal to the value programmed in the DDRSS\_CTL\_29[31-16] OSC\_VARIANCE\_LIMIT field, the DDR controller returns to idle state. If the request was issued through software, the DDR controller sets to 1h bit [30] in the DDRSS\_CTL\_293[31-0] INT\_STATUS\_0 field indicating DQS oscillator request complete interrupt.
  - d. If the difference between the measured value and the base value is greater than the value programmed in the DDRSS\_CTL\_29[31-16] OSC\_VARIANCE\_LIMIT field, the DDR controller sets to 1h bit [1] in the DDRSS\_CTL\_294[12-0] INT\_STATUS\_1 field indicating DQS oscillator out of variance interrupt. The DDR controller also drives a 2h value on the dfi\_function signal and asserts the dfi\_function\_valid signal for the number of cycles defined by the DDRSS\_CTL\_27[19-16] FUNC\_VALID\_CYCLES field. The measured value will replace the base value for any device for which the measured value is outside the variance. This base value update does not set to 1h bit [31] in the DDRSS\_CTL\_293[31-0] INT\_STATUS\_0 field.

#### 8.2.4.6.10.6 Per-Bank Refresh (PBR)

Refresh commands are important memory operations to preserve memory contents, but they are also disruptive to system transaction flow. To minimize the impact, the DDR controller implements refresh on a per-bank basis. This does increase the quantity of refresh commands, but allows banks that are not targeted by the refresh to be accessed for read and write commands.

PBR is controlled through the following fields:

- DDRSS\_CTL\_228[12-8] AREF\_MAX\_CREDIT
- DDRSS\_CTL\_228[4-0] AREF\_MAX\_DEFICIT
- DDRSS\_CTL\_227[20-16] AREF\_NORM\_THRESHOLD
- DDRSS\_CTL\_71[28-24] AREF\_PBR\_CONT\_DIS\_THRESHOLD
- DDRSS\_CTL\_71[20-16] AREF\_PBR\_CONT\_EN\_THRESHOLD
- DDRSS\_CTL\_71[3-0] PBR\_BANK\_SELECT\_DELAY
- DDRSS\_CTL\_71[8] PBR\_CONT\_REQ\_EN
- DDRSS\_CTL\_67[0] PBR\_EN
- DDRSS\_CTL\_70[31-16] PBR\_MAX\_BANK\_WAIT
- DDRSS\_CTL\_67[8] PBR\_NUMERIC\_ORDER
- DDRSS\_CTL\_59[16] TREF\_ENABLE
- DDRSS\_CTL\_68[15-0] TREFI\_PB\_F0

- DDRSS\_CTL\_69[15-0] TREFI\_PB\_F1
- DDRSS\_CTL\_70[15-0] TREFI\_PB\_F2

#### 8.2.4.6.10.6.1 Normal Operation

Per-bank refresh commands may be issued to all banks or to a single bank, and may be issued to each bank in any order if programmed for this behavior. However, all banks must be refreshed before the same bank can be issued a new per-bank refresh command. The DDR controller can issue the commands in any order, and subsequent per-bank refresh sequences can be a completely different order than the last command. The order of banks is specified via the DDRSS\_CTL\_67[8] PBR\_NUMERIC\_ORDER bit. It is potentially advantageous to execute out-of-numerical order to prevent an active bank from stalling refresh operations to any other banks.

The PBR logic is part of the auto-refresh logic. The auto-refresh logic is responsible for maintaining the debit / credit counts for refresh that ensure that refresh commands are occurring as needed. The DRAM allows the user to postpone a maximum of 8 refresh commands which means that the user can wait as much as 9 x TREFI\_PB\_Fx cycles before issuing a refresh command. The DDR controller keeps track of how many refreshes it has skipped (DDRSS\_CTL\_228[4-0] AREF\_MAX\_DEFICIT) or advanced (DDRSS\_CTL\_228[12-8] AREF\_MAX\_CREDIT) and ensures that the DRAM requirements are satisfied. The PBR logic issues refresh commands to single banks if enabled via the DDRSS\_CTL\_67[0] PBR\_EN bit and in accordance with the programmed values. The DDRSS\_CTL\_59[16] TREF\_ENABLE bit must also be set to 1h for the PBR logic to work. The auto-refresh logic issues refresh commands to all banks and high priority refresh commands if the refresh count falls below a certain threshold.

When a refresh is required, the PBR logic issues a per-bank refresh command to a specific bank. If that bank is currently being accessed by the DDR controller core, the command will be held off. If the command is held off beyond the number of cycles specified in the DDRSS\_CTL\_70[31-16] PBR\_MAX\_BANK\_WAIT field, the PBR logic inhibits the active process in the DDR controller core, closes the bank, executes the PBR command and then un-inhibits the core to allow the interrupted process to continue. Once the bank is free, the PBR command is issued. The bank will be blocked from accesses within the number of clocks programmed into the DDRSS\_CTL\_71[3-0] PBR\_BANK\_SELECT\_DELAY field and the refresh will occur and then release the bank.

#### 8.2.4.6.10.6.2 Continuous Refresh Request Mode

The PBR logic can be temporarily modified to continuously assert refresh requests. This option may be desirable when the auto-refresh logic is running at a deficit and the user wishes to run per-bank refreshes instead of auto-refreshes to restore the refresh count.

Setting the DDRSS\_CTL\_71[8] PBR\_CONT\_REQ\_EN bit to 1h enables this mode. Automatic mode falls between the normal priority and high priority commands in terms of priority. When enabled, if the number of all-bank refresh commands pending exceeds the value programmed into the DDRSS\_CTL\_71[20-16] AREF\_PBR\_CONT\_EN\_THRESHOLD field, then the DDR controller automatically begins issuing per-bank refresh commands. As refreshes occur, the pending command count drops. Once that value meets the value programmed in the DDRSS\_CTL\_71[28-24] AREF\_PBR\_CONT\_DIS\_THRESHOLD field, the DDR controller stops automatically issuing per-bank refresh commands.

If however the refresh command pending count exceeds the value programmed into the DDRSS\_CTL\_227[20-16] AREF\_NORM\_THRESHOLD field, the DDR controller temporarily suspends the PBR logic, and the auto-refresh logic issues all-bank refresh commands until the refresh logic catches up. Once caught up, the PBR logic is resumed.

#### 8.2.4.7 DDR PHY Functional Description

The DDR PHY provides functionality to interface the DDR controller to SDRAM devices. The PHY has a slice based and DQS-delay architecture. It contains data, address, address/control and clock slices and uses programmable clock delay lines to align write data, read data capture, and DQS gating from the I/O pads across the DFI interface to the DDR controller.



#### 8.2.4.7.1 Data Slice

The data slice transfers data between the DDR controller and the SDRAM devices. The data slice is a module that interfaces to the DQ, DM, and DQS signals of the SDRAM and is duplicated as many times as necessary to create the current SoC SDRAM data width.

The write data path logic (from DFI to pads) and the read data path logic (from pads to DFI) are contained within the data slice. Termination and directional controls for the data path related I/Os are also contained in this slice.

#### 8.2.4.7.2 Address Slice

The address slice transfers address information between the DDR controller and the SDRAM devices and contains 6 address bits.

The address write data path logic (from DFI to pads) and the read data path logic (loopback only) are contained within the address slice. Termination and directional controls for the address path related I/Os also exist in this slice.

##### 8.2.4.7.2.1 Address Swapping

The address slice has the ability to programmably change which DFI address bits that are connected to the I/O pads. Address signals can be swapped within an address slice without restrictions. The swapping is done via the DDRSS\_PHY\_1053[23-0] PHY\_ADR\_ADDDR\_SEL\_0 field.

#### 8.2.4.7.3 Address/Control Slice

The address/control slice is a module that interfaces to the control, command, and address connections of the SDRAM and is duplicated as many times as necessary to create the appropriate width for the SoC. The address/control signals within a slice have per bit delay line control.

#### 8.2.4.7.4 Clock Slice

The clock slice creates the clock signal used by the SDRAM. The slice also disables the clock when instructed by the DDR controller through the DFI bus. When the clock is disabled, the CK pin drives low and the CKN drives high. Duty cycle correction is also applied in the memory clock slice.

#### 8.2.4.7.5 DDR PHY Initialization

Once the power to the SDRAM and SoC is stable, the DDR PHY must be reset and initialized before normal traffic can commence. Once the initialization has completed, the DDR PHY asserts the dfi\_init\_complete signal to notify the DDR controller that the PHY and SDRAM are ready to accept commands.

The procedure to initialize the DDR PHY is as follows:

1. Reset the DDRSS0 through the DDRSS0\_RST signal. This resets all DDR PHY registers.
2. Program all needed PHY registers.
3. The dfi\_init\_start is then asserted by the DDR controller. The PHY slice master delay lines locks and then the PHY asserts the dfi\_init\_complete signal. This indicates that the PHY and SDRAM are ready to accept commands. The PHY is now ready for normal traffic.

For information about the DDR controller initialization, see [Section 8.2.4.6.9](#).

#### 8.2.4.7.6 DDR PHY Dynamic Frequency Scaling (DFS)

The DDR PHY supports Dynamic Frequency Scaling (DFS) as defined by the DFI specification. This feature allows the PHY to be configured for multiple frequency sets, and have the DFI bus dynamically specify which frequency set to use at a given time. Before using a frequency set in normal operation, leveling operation should be performed on that set. Failure to do so may result in invalid operation. Once leveling has been performed with each frequency set, the user can freely switch between frequencies without need to repeat the leveling, unless the leveling environment requires periodic updates for voltage and temperature compensation.

This DDR PHY contains registers for each frequency set. This feature creates a lookup table that may be accessed through the DDRSS\_PHY\_1281[17-16] PHY\_FREQ\_SEL\_INDEX and DDRSS\_PHY\_1281[8]

PHY\_FREQ\_SEL\_MULTICAST\_EN fields, which gives a flexibility in controlling these frequency set registers with a simple register interface. This fact should be taken into account when reading and writing these registers.

The DDRSS\_PHY\_1280[1-0] PHY\_FREQ\_SEL and DDRSS\_PHY\_1281[17-16] PHY\_FREQ\_SEL\_INDEX fields operate independently, but if the PHY\_FREQ\_SEL\_INDEX field specifies a frequency set different than the one currently being used, accidental write to or read from invalid registers may happen.

Reads from frequency based registers return only the value that correlates to the frequency set identified in the DDRSS\_PHY\_1281[17-16] PHY\_FREQ\_SEL\_INDEX field. For writes to the frequency based registers, the resulting write depends on the value of the DDRSS\_PHY\_1281[8] PHY\_FREQ\_SEL\_MULTICAST\_EN bit. If it is set to 0x1, all frequency sets are concurrently written. If the PHY\_FREQ\_SEL\_MULTICAST\_EN bit is set to 0x0, only the frequency set identified in the PHY\_FREQ\_SEL\_INDEX field is written.

For example, consider a write to DDRSS\_PHY\_130[9-0] PHY\_RDDQS\_DM\_FALL\_SLAVE\_DELAY\_0 with the following settings:

- DDRSS\_PHY\_1281[17-16] PHY\_FREQ\_SEL\_INDEX = 0x0001 (frequency set 1)
- DDRSS\_PHY\_1281[8] PHY\_FREQ\_SEL\_MULTICAST\_EN = 0x0.

In this case, only the frequency set 1 version of the DDRSS\_PHY\_130[9-0] PHY\_RDDQS\_DM\_FALL\_SLAVE\_DELAY\_0 field is written. If the same settings are used, but with PHY\_FREQ\_SEL\_MULTICAST\_EN = 0x1, then all frequency-indexed versions of PHY\_RDDQS\_DM\_FALL\_SLAVE\_DELAY\_0 are written.

#### 8.2.4.7.7 Chip Select and Frequency Based Register Settings

Some register settings within the PHY are chip select based. The DDRSS\_PHY\_6[0] PHY\_PER\_CS\_TRAINING\_INDEX\_0 bit allows access to a particular chip select copy. The PHY\_PER\_CS\_TRAINING\_INDEX\_0 bit is associated with data slice 0. The following bits are associated with the other data slices:

- DDRSS\_PHY\_262[0] PHY\_PER\_CS\_TRAINING\_INDEX\_1
- DDRSS\_PHY\_518[0] PHY\_PER\_CS\_TRAINING\_INDEX\_2
- DDRSS\_PHY\_774[0] PHY\_PER\_CS\_TRAINING\_INDEX\_3

When writing to chip select based registers, if the DDRSS\_PHY\_5[24] PHY\_PER\_CS\_TRAINING\_MULTICAST\_EN\_0 bit is set to 0x1, then all chip select copies receive the same data. The PHY\_PER\_CS\_TRAINING\_MULTICAST\_EN\_0 bit is associated with data slice 0. The following bits are associated with the other data slices:

- DDRSS\_PHY\_261[24] PHY\_PER\_CS\_TRAINING\_MULTICAST\_EN\_1
- DDRSS\_PHY\_517[24] PHY\_PER\_CS\_TRAINING\_MULTICAST\_EN\_2
- DDRSS\_PHY\_773[24] PHY\_PER\_CS\_TRAINING\_MULTICAST\_EN\_3

Some register settings in the PHY are frequency based. As already described, the DDRSS\_PHY\_1281[17-16] PHY\_FREQ\_SEL\_INDEX field allows access to a particular frequency copy. When writing to frequency based registers, if the DDRSS\_PHY\_1281[8] PHY\_FREQ\_SEL\_MULTICAST\_EN bit is set to 0x1, then all frequency copies receive the same data. [Table 8-20](#) shows the frequency based register settings.

**Table 8-20. Frequency Based Register Settings**

Register Settings	Category
Data Slice	

**Table 8-20. Frequency Based Register Settings (continued)**

Register Settings	Category
DDRSS_PHY_103[3-0] PHY_SW_MASTER_MODE_0 DDRSS_PHY_359[3-0] PHY_SW_MASTER_MODE_1 DDRSS_PHY_615[3-0] PHY_SW_MASTER_MODE_2 DDRSS_PHY_871[3-0] PHY_SW_MASTER_MODE_3	Master delay line
DDRSS_PHY_103[18-8] PHY_MASTER_DELAY_START_0 DDRSS_PHY_359[18-8] PHY_MASTER_DELAY_START_0 DDRSS_PHY_615[18-8] PHY_MASTER_DELAY_START_0 DDRSS_PHY_871[18-8] PHY_MASTER_DELAY_START_0	
DDRSS_PHY_103[29-24] PHY_MASTER_DELAY_STEP_0 DDRSS_PHY_359[29-24] PHY_MASTER_DELAY_STEP_0 DDRSS_PHY_615[29-24] PHY_MASTER_DELAY_STEP_0 DDRSS_PHY_871[29-24] PHY_MASTER_DELAY_STEP_0	
DDRSS_PHY_104[7-0] PHY_MASTER_DELAY_WAIT_0 DDRSS_PHY_360[7-0] PHY_MASTER_DELAY_WAIT_1 DDRSS_PHY_616[7-0] PHY_MASTER_DELAY_WAIT_2 DDRSS_PHY_872[7-0] PHY_MASTER_DELAY_WAIT_3	
DDRSS_PHY_104[15-8] PHY_MASTER_DELAY_HALF_MEASURE_0 DDRSS_PHY_360[15-8] PHY_MASTER_DELAY_HALF_MEASURE_1 DDRSS_PHY_616[15-8] PHY_MASTER_DELAY_HALF_MEASURE_2 DDRSS_PHY_872[15-8] PHY_MASTER_DELAY_HALF_MEASURE_3	
DDRSS_PHY_110[1-0] PHY_WRPATH_GATE_DISABLE_0 DDRSS_PHY_366[1-0] PHY_WRPATH_GATE_DISABLE_1 DDRSS_PHY_622[1-0] PHY_WRPATH_GATE_DISABLE_2 DDRSS_PHY_878[1-0] PHY_WRPATH_GATE_DISABLE_3	Write path
DDRSS_PHY_110[10-8] PHY_WRPATH_GATE_TIMING_0 DDRSS_PHY_366[10-8] PHY_WRPATH_GATE_TIMING_1 DDRSS_PHY_622[10-8] PHY_WRPATH_GATE_TIMING_2 DDRSS_PHY_878[10-8] PHY_WRPATH_GATE_TIMING_3	
DDRSS_PHY_139[9-8] PHY_DQ_FFE_0 DDRSS_PHY_395[9-8] PHY_DQ_FFE_1 DDRSS_PHY_651[9-8] PHY_DQ_FFE_2 DDRSS_PHY_907[9-8] PHY_DQ_FFE_3	
DDRSS_PHY_139[17-16] PHY_DQS_FFE_0 DDRSS_PHY_395[17-16] PHY_DQS_FFE_1 DDRSS_PHY_651[17-16] PHY_DQS_FFE_2 DDRSS_PHY_907[17-16] PHY_DQS_FFE_3	
DDRSS_PHY_104[31-24] PHY_WRLVL_DLY_STEP_0 DDRSS_PHY_360[31-24] PHY_WRLVL_DLY_STEP_1 DDRSS_PHY_616[31-24] PHY_WRLVL_DLY_STEP_2 DDRSS_PHY_872[31-24] PHY_WRLVL_DLY_STEP_3	Write leveling
DDRSS_PHY_105[3-0] PHY_WRLVL_DLY_FINE_STEP_0 DDRSS_PHY_361[3-0] PHY_WRLVL_DLY_FINE_STEP_1 DDRSS_PHY_617[3-0] PHY_WRLVL_DLY_FINE_STEP_2 DDRSS_PHY_873[3-0] PHY_WRLVL_DLY_FINE_STEP_3	
DDRSS_PHY_105[13-8] PHY_WRLVL_RESP_WAIT_CNT_0 DDRSS_PHY_361[13-8] PHY_WRLVL_RESP_WAIT_CNT_1 DDRSS_PHY_617[13-8] PHY_WRLVL_RESP_WAIT_CNT_2 DDRSS_PHY_873[13-8] PHY_WRLVL_RESP_WAIT_CNT_3	

**Table 8-20. Frequency Based Register Settings (continued)**

Register Settings	Category
DDRSS_PHY_107[7-0] PHY_WDQLVL_DLY_STEP_0 DDRSS_PHY_363[7-0] PHY_WDQLVL_DLY_STEP_1 DDRSS_PHY_619[7-0] PHY_WDQLVL_DLY_STEP_2 DDRSS_PHY_875[7-0] PHY_WDQLVL_DLY_STEP_3	Write DQ training
DDRSS_PHY_107[11-8] PHY_WDQLVL_QTR_DLY_STEP_0 DDRSS_PHY_363[11-8] PHY_WDQLVL_QTR_DLY_STEP_1 DDRSS_PHY_619[11-8] PHY_WDQLVL_QTR_DLY_STEP_2 DDRSS_PHY_875[11-8] PHY_WDQLVL_QTR_DLY_STEP_3	
DDRSS_PHY_86[16] PHY_NTP_TRAIN_EN_0 DDRSS_PHY_342[16] PHY_NTP_TRAIN_EN_1 DDRSS_PHY_598[16] PHY_NTP_TRAIN_EN_2 DDRSS_PHY_854[16] PHY_NTP_TRAIN_EN_3	
DDRSS_PHY_86[31-24] PHY_NTP_WDQ_STEP_SIZE_0 DDRSS_PHY_342[31-24] PHY_NTP_WDQ_STEP_SIZE_1 DDRSS_PHY_598[31-24] PHY_NTP_WDQ_STEP_SIZE_2 DDRSS_PHY_854[31-24] PHY_NTP_WDQ_STEP_SIZE_3	
DDRSS_PHY_87[10-0] PHY_NTP_WDQ_START_0 DDRSS_PHY_343[10-0] PHY_NTP_WDQ_START_1 DDRSS_PHY_599[10-0] PHY_NTP_WDQ_START_2 DDRSS_PHY_855[10-0] PHY_NTP_WDQ_START_3	
DDRSS_PHY_87[26-16] PHY_NTP_WDQ_STOP_0 DDRSS_PHY_343[26-16] PHY_NTP_WDQ_STOP_1 DDRSS_PHY_599[26-16] PHY_NTP_WDQ_STOP_2 DDRSS_PHY_855[26-16] PHY_NTP_WDQ_STOP_3	
DDRSS_PHY_88[17-8] PHY_WDQLVL_DVW_MIN_0 DDRSS_PHY_344[17-8] PHY_WDQLVL_DVW_MIN_1 DDRSS_PHY_600[17-8] PHY_WDQLVL_DVW_MIN_2 DDRSS_PHY_856[17-8] PHY_WDQLVL_DVW_MIN_3	
DDRSS_PHY_88[24] PHY_SW_WDQLVL_DVW_MIN_EN_0 DDRSS_PHY_344[24] PHY_SW_WDQLVL_DVW_MIN_EN_1 DDRSS_PHY_600[24] PHY_SW_WDQLVL_DVW_MIN_EN_2 DDRSS_PHY_856[24] PHY_SW_WDQLVL_DVW_MIN_EN_3	
DDRSS_PHY_89[5-0] PHY_WDQLVL_PER_START_OFFSET_0 DDRSS_PHY_345[5-0] PHY_WDQLVL_PER_START_OFFSET_1 DDRSS_PHY_601[5-0] PHY_WDQLVL_PER_START_OFFSET_2 DDRSS_PHY_857[5-0] PHY_WDQLVL_PER_START_OFFSET_3	
DDRSS_PHY_113[6-0] PHY_WDQ_OSC_DELTA_0 DDRSS_PHY_369[6-0] PHY_WDQ_OSC_DELTA_1 DDRSS_PHY_625[6-0] PHY_WDQ_OSC_DELTA_2 DDRSS_PHY_881[6-0] PHY_WDQ_OSC_DELTA_3	

**Table 8-20. Frequency Based Register Settings (continued)**

Register Settings	Category
DDRSS_PHY_136[15-8] PHY_DATA_DC_DQ0_CLK_ADJUST_0 DDRSS_PHY_136[23-16] PHY_DATA_DC_DQ1_CLK_ADJUST_0 DDRSS_PHY_136[31-24] PHY_DATA_DC_DQ2_CLK_ADJUST_0 DDRSS_PHY_137[7-0] PHY_DATA_DC_DQ3_CLK_ADJUST_0 DDRSS_PHY_137[15-8] PHY_DATA_DC_DQ4_CLK_ADJUST_0 DDRSS_PHY_137[23-16] PHY_DATA_DC_DQ5_CLK_ADJUST_0 DDRSS_PHY_137[31-24] PHY_DATA_DC_DQ6_CLK_ADJUST_0 DDRSS_PHY_138[7-0] PHY_DATA_DC_DQ7_CLK_ADJUST_0 DDRSS_PHY_392[15-8] PHY_DATA_DC_DQ0_CLK_ADJUST_1 DDRSS_PHY_392[23-16] PHY_DATA_DC_DQ1_CLK_ADJUST_1 DDRSS_PHY_392[31-24] PHY_DATA_DC_DQ2_CLK_ADJUST_1 DDRSS_PHY_393[7-0] PHY_DATA_DC_DQ3_CLK_ADJUST_1 DDRSS_PHY_393[15-8] PHY_DATA_DC_DQ4_CLK_ADJUST_1 DDRSS_PHY_393[23-16] PHY_DATA_DC_DQ5_CLK_ADJUST_1 DDRSS_PHY_393[31-24] PHY_DATA_DC_DQ6_CLK_ADJUST_1 DDRSS_PHY_394[7-0] PHY_DATA_DC_DQ7_CLK_ADJUST_1 DDRSS_PHY_648[15-8] PHY_DATA_DC_DQ0_CLK_ADJUST_2 DDRSS_PHY_648[23-16] PHY_DATA_DC_DQ1_CLK_ADJUST_2 DDRSS_PHY_648[31-24] PHY_DATA_DC_DQ2_CLK_ADJUST_2 DDRSS_PHY_649[7-0] PHY_DATA_DC_DQ3_CLK_ADJUST_2 DDRSS_PHY_649[15-8] PHY_DATA_DC_DQ4_CLK_ADJUST_2 DDRSS_PHY_649[23-16] PHY_DATA_DC_DQ5_CLK_ADJUST_2 DDRSS_PHY_649[31-24] PHY_DATA_DC_DQ6_CLK_ADJUST_2 DDRSS_PHY_650[7-0] PHY_DATA_DC_DQ7_CLK_ADJUST_2 DDRSS_PHY_904[15-8] PHY_DATA_DC_DQ0_CLK_ADJUST_3 DDRSS_PHY_904[23-16] PHY_DATA_DC_DQ1_CLK_ADJUST_3 DDRSS_PHY_904[31-24] PHY_DATA_DC_DQ2_CLK_ADJUST_3 DDRSS_PHY_905[7-0] PHY_DATA_DC_DQ3_CLK_ADJUST_3 DDRSS_PHY_905[15-8] PHY_DATA_DC_DQ4_CLK_ADJUST_3 DDRSS_PHY_905[23-16] PHY_DATA_DC_DQ5_CLK_ADJUST_3 DDRSS_PHY_905[31-24] PHY_DATA_DC_DQ6_CLK_ADJUST_3 DDRSS_PHY_906[7-0] PHY_DATA_DC_DQ7_CLK_ADJUST_3	Duty cycle correction

**Table 8-20. Frequency Based Register Settings (continued)**

Register Settings	Category
DDRSS_PHY_138[15-8] PHY_DATA_DC_DM_CLK_ADJUST_0 DDRSS_PHY_394[15-8] PHY_DATA_DC_DM_CLK_ADJUST_1 DDRSS_PHY_650[15-8] PHY_DATA_DC_DM_CLK_ADJUST_2 DDRSS_PHY_906[15-8] PHY_DATA_DC_DM_CLK_ADJUST_3	Duty cycle correction
DDRSS_PHY_136[7-0] PHY_DATA_DC_DQS_CLK_ADJUST_0 DDRSS_PHY_392[7-0] PHY_DATA_DC_DQS_CLK_ADJUST_1 DDRSS_PHY_648[7-0] PHY_DATA_DC_DQS_CLK_ADJUST_2 DDRSS_PHY_904[7-0] PHY_DATA_DC_DQS_CLK_ADJUST_3	
DDRSS_PHY_110[17-16] PHY_DATA_DC_INIT_DISABLE_0 DDRSS_PHY_366[17-16] PHY_DATA_DC_INIT_DISABLE_1 DDRSS_PHY_622[17-16] PHY_DATA_DC_INIT_DISABLE_2 DDRSS_PHY_878[17-16] PHY_DATA_DC_INIT_DISABLE_3	
DDRSS_PHY_112[0] PHY_DATA_DC_WRLVL_ENABLE_0 DDRSS_PHY_368[0] PHY_DATA_DC_WRLVL_ENABLE_1 DDRSS_PHY_624[0] PHY_DATA_DC_WRLVL_ENABLE_2 DDRSS_PHY_880[0] PHY_DATA_DC_WRLVL_ENABLE_3	
DDRSS_PHY_112[8] PHY_DATA_DC_WDQLVL_ENABLE_0 DDRSS_PHY_368[8] PHY_DATA_DC_WDQLVL_ENABLE_1 DDRSS_PHY_624[8] PHY_DATA_DC_WDQLVL_ENABLE_2 DDRSS_PHY_880[8] PHY_DATA_DC_WDQLVL_ENABLE_3	
DDRSS_PHY_111[26-16] PHY_DATA_DC_DQ_INIT_SLV_DELAY_0 DDRSS_PHY_367[26-16] PHY_DATA_DC_DQ_INIT_SLV_DELAY_1 DDRSS_PHY_623[26-16] PHY_DATA_DC_DQ_INIT_SLV_DELAY_2 DDRSS_PHY_879[26-16] PHY_DATA_DC_DQ_INIT_SLV_DELAY_3	
DDRSS_PHY_111[9-0] PHY_DATA_DC_DQS_INIT_SLV_DELAY_0 DDRSS_PHY_367[9-0] PHY_DATA_DC_DQS_INIT_SLV_DELAY_1 DDRSS_PHY_623[9-0] PHY_DATA_DC_DQS_INIT_SLV_DELAY_2 DDRSS_PHY_879[9-0] PHY_DATA_DC_DQS_INIT_SLV_DELAY_3	
DDRSS_PHY_97[18-16] PHY_DATA_DC_CAL_CLK_SEL_0 DDRSS_PHY_353[18-16] PHY_DATA_DC_CAL_CLK_SEL_1 DDRSS_PHY_609[18-16] PHY_DATA_DC_CAL_CLK_SEL_2 DDRSS_PHY_865[18-16] PHY_DATA_DC_CAL_CLK_SEL_3	
DDRSS_PHY_112[23-16] PHY_DATA_DC_DM_CLK_SE_THRSHLD_0 DDRSS_PHY_368[23-16] PHY_DATA_DC_DM_CLK_SE_THRSHLD_1 DDRSS_PHY_624[23-16] PHY_DATA_DC_DM_CLK_SE_THRSHLD_2 DDRSS_PHY_880[23-16] PHY_DATA_DC_DM_CLK_SE_THRSHLD_3	
DDRSS_PHY_112[31-24] PHY_DATA_DC_DM_CLK_DIFF_THRSHLD_0 DDRSS_PHY_368[31-24] PHY_DATA_DC_DM_CLK_DIFF_THRSHLD_1 DDRSS_PHY_624[31-24] PHY_DATA_DC_DM_CLK_DIFF_THRSHLD_2 DDRSS_PHY_880[31-24] PHY_DATA_DC_DM_CLK_DIFF_THRSHLD_3	

**Table 8-20. Frequency Based Register Settings (continued)**

Register Settings	Category
DDRSS_PHY_93[9-0] PHY_RDDQ0_SLAVE_DELAY_0 DDRSS_PHY_93[25-16] PHY_RDDQ1_SLAVE_DELAY_0 DDRSS_PHY_94[9-0] PHY_RDDQ2_SLAVE_DELAY_0 DDRSS_PHY_94[25-16] PHY_RDDQ3_SLAVE_DELAY_0 DDRSS_PHY_95[9-0] PHY_RDDQ4_SLAVE_DELAY_0 DDRSS_PHY_95[25-16] PHY_RDDQ5_SLAVE_DELAY_0 DDRSS_PHY_96[9-0] PHY_RDDQ6_SLAVE_DELAY_0 DDRSS_PHY_96[25-16] PHY_RDDQ7_SLAVE_DELAY_0 DDRSS_PHY_349[9-0] PHY_RDDQ0_SLAVE_DELAY_1 DDRSS_PHY_349[25-16] PHY_RDDQ1_SLAVE_DELAY_1 DDRSS_PHY_350[9-0] PHY_RDDQ2_SLAVE_DELAY_1 DDRSS_PHY_350[25-16] PHY_RDDQ3_SLAVE_DELAY_1 DDRSS_PHY_351[9-0] PHY_RDDQ4_SLAVE_DELAY_1 DDRSS_PHY_351[25-16] PHY_RDDQ5_SLAVE_DELAY_1 DDRSS_PHY_352[9-0] PHY_RDDQ6_SLAVE_DELAY_1 DDRSS_PHY_352[25-16] PHY_RDDQ7_SLAVE_DELAY_1 DDRSS_PHY_605[9-0] PHY_RDDQ0_SLAVE_DELAY_2 DDRSS_PHY_605[25-16] PHY_RDDQ1_SLAVE_DELAY_2 DDRSS_PHY_606[9-0] PHY_RDDQ2_SLAVE_DELAY_2 DDRSS_PHY_606[25-16] PHY_RDDQ3_SLAVE_DELAY_2 DDRSS_PHY_607[9-0] PHY_RDDQ4_SLAVE_DELAY_2 DDRSS_PHY_607[25-16] PHY_RDDQ5_SLAVE_DELAY_2 DDRSS_PHY_608[9-0] PHY_RDDQ6_SLAVE_DELAY_2 DDRSS_PHY_608[25-16] PHY_RDDQ7_SLAVE_DELAY_2 DDRSS_PHY_861[9-0] PHY_RDDQ0_SLAVE_DELAY_3 DDRSS_PHY_861[25-16] PHY_RDDQ1_SLAVE_DELAY_3 DDRSS_PHY_862[9-0] PHY_RDDQ2_SLAVE_DELAY_3 DDRSS_PHY_862[25-16] PHY_RDDQ3_SLAVE_DELAY_3 DDRSS_PHY_863[9-0] PHY_RDDQ4_SLAVE_DELAY_3 DDRSS_PHY_863[25-16] PHY_RDDQ5_SLAVE_DELAY_3 DDRSS_PHY_864[9-0] PHY_RDDQ6_SLAVE_DELAY_3 DDRSS_PHY_864[25-16] PHY_RDDQ7_SLAVE_DELAY_3	Read path
DDRSS_PHY_97[9-0] PHY_RDDM_SLAVE_DELAY_0 DDRSS_PHY_353[9-0] PHY_RDDM_SLAVE_DELAY_1 DDRSS_PHY_609[9-0] PHY_RDDM_SLAVE_DELAY_2 DDRSS_PHY_865[9-0] PHY_RDDM_SLAVE_DELAY_3	Read path
DDRSS_PHY_101[25-24] PHY_RDDATA_EN_IE_DLY_0 DDRSS_PHY_357[25-24] PHY_RDDATA_EN_IE_DLY_1 DDRSS_PHY_613[25-24] PHY_RDDATA_EN_IE_DLY_2 DDRSS_PHY_869[25-24] PHY_RDDATA_EN_IE_DLY_3	
DDRSS_PHY_102[1-0] PHY_IE_MODE_0 DDRSS_PHY_358[1-0] PHY_IE_MODE_1 DDRSS_PHY_614[1-0] PHY_IE_MODE_2 DDRSS_PHY_870[1-0] PHY_IE_MODE_3	
DDRSS_PHY_102[20-16] PHY_RDDATA_EN_TSEL_DLY_0 DDRSS_PHY_358[20-16] PHY_RDDATA_EN_TSEL_DLY_1 DDRSS_PHY_614[20-16] PHY_RDDATA_EN_TSEL_DLY_2 DDRSS_PHY_870[20-16] PHY_RDDATA_EN_TSEL_DLY_3	
DDRSS_PHY_102[28-24] PHY_RDDATA_EN_OE_DLY_0 DDRSS_PHY_358[28-24] PHY_RDDATA_EN_OE_DLY_1 DDRSS_PHY_614[28-24] PHY_RDDATA_EN_OE_DLY_2 DDRSS_PHY_870[28-24] PHY_RDDATA_EN_OE_DLY_3	
DDRSS_PHY_113[20-16] PHY_RDDATA_EN_DLY_0 DDRSS_PHY_369[20-16] PHY_RDDATA_EN_DLY_1 DDRSS_PHY_625[20-16] PHY_RDDATA_EN_DLY_2 DDRSS_PHY_881[20-16] PHY_RDDATA_EN_DLY_3	
DDRSS_PHY_104[19-16] PHY_RPTR_UPDATE_0 DDRSS_PHY_360[19-16] PHY_RPTR_UPDATE_1 DDRSS_PHY_616[19-16] PHY_RPTR_UPDATE_2 DDRSS_PHY_872[19-16] PHY_RPTR_UPDATE_3	



**Table 8-20. Frequency Based Register Settings (continued)**

Register Settings	Category
DDRSS_PHY_105[19-16] PHY_GTLVL_DLY_STEP_0 DDRSS_PHY_361[19-16] PHY_GTLVL_DLY_STEP_1 DDRSS_PHY_617[19-16] PHY_GTLVL_DLY_STEP_2 DDRSS_PHY_873[19-16] PHY_GTLVL_DLY_STEP_3	Gate training
DDRSS_PHY_105[28-24] PHY_GTLVL_RESP_WAIT_CNT_0 DDRSS_PHY_361[28-24] PHY_GTLVL_RESP_WAIT_CNT_1 DDRSS_PHY_617[28-24] PHY_GTLVL_RESP_WAIT_CNT_2 DDRSS_PHY_873[28-24] PHY_GTLVL_RESP_WAIT_CNT_3	
DDRSS_PHY_106[9-0] PHY_GTLVL_BACK_STEP_0 DDRSS_PHY_362[9-0] PHY_GTLVL_BACK_STEP_1 DDRSS_PHY_618[9-0] PHY_GTLVL_BACK_STEP_2 DDRSS_PHY_874[9-0] PHY_GTLVL_BACK_STEP_3	
DDRSS_PHY_106[25-16] PHY_GTLVL_FINAL_STEP_0 DDRSS_PHY_362[25-16] PHY_GTLVL_FINAL_STEP_1 DDRSS_PHY_618[25-16] PHY_GTLVL_FINAL_STEP_2 DDRSS_PHY_874[25-16] PHY_GTLVL_FINAL_STEP_3	
DDRSS_PHY_107[27-24] PHY_RDLVL_DLY_STEP_0 DDRSS_PHY_363[27-24] PHY_RDLVL_DLY_STEP_1 DDRSS_PHY_619[27-24] PHY_RDLVL_DLY_STEP_2 DDRSS_PHY_875[27-24] PHY_RDLVL_DLY_STEP_3	Read data eye training
DDRSS_PHY_108[9-0] PHY_RDLVL_MAX_EDGE_0 DDRSS_PHY_364[9-0] PHY_RDLVL_MAX_EDGE_1 DDRSS_PHY_620[9-0] PHY_RDLVL_MAX_EDGE_2 DDRSS_PHY_876[9-0] PHY_RDLVL_MAX_EDGE_3	
DDRSS_PHY_109[9-0] PHY_RDLVL_DVW_MIN_0 DDRSS_PHY_365[9-0] PHY_RDLVL_DVW_MIN_1 DDRSS_PHY_621[9-0] PHY_RDLVL_DVW_MIN_2 DDRSS_PHY_877[9-0] PHY_RDLVL_DVW_MIN_3	
DDRSS_PHY_109[16] PHY_SW_RDLVL_DVW_MIN_EN_0 DDRSS_PHY_365[16] PHY_SW_RDLVL_DVW_MIN_EN_1 DDRSS_PHY_621[16] PHY_SW_RDLVL_DVW_MIN_EN_2 DDRSS_PHY_877[16] PHY_SW_RDLVL_DVW_MIN_EN_3	
DDRSS_PHY_109[29-24] PHY_RDLVL_PER_START_OFFSET_0 DDRSS_PHY_365[29-24] PHY_RDLVL_PER_START_OFFSET_1 DDRSS_PHY_621[29-24] PHY_RDLVL_PER_START_OFFSET_2 DDRSS_PHY_877[29-24] PHY_RDLVL_PER_START_OFFSET_3	
DDRSS_PHY_135[9-0] PHY_RDLVL_RDDQS_DQ_SLV_DLY_START_0 DDRSS_PHY_391[9-0] PHY_RDLVL_RDDQS_DQ_SLV_DLY_START_1 DDRSS_PHY_647[9-0] PHY_RDLVL_RDDQS_DQ_SLV_DLY_START_2 DDRSS_PHY_903[9-0] PHY_RDLVL_RDDQS_DQ_SLV_DLY_START_3	
DDRSS_PHY_85[30-24] PHY_VREF_INITIAL_START_POINT_0 DDRSS_PHY_341[30-24] PHY_VREF_INITIAL_START_POINT_1 DDRSS_PHY_597[30-24] PHY_VREF_INITIAL_START_POINT_2 DDRSS_PHY_853[30-24] PHY_VREF_INITIAL_START_POINT_3	
DDRSS_PHY_86[6-0] PHY_VREF_INITIAL_STOP_POINT_0 DDRSS_PHY_342[6-0] PHY_VREF_INITIAL_STOP_POINT_1 DDRSS_PHY_598[6-0] PHY_VREF_INITIAL_STOP_POINT_2 DDRSS_PHY_854[6-0] PHY_VREF_INITIAL_STOP_POINT_3	
DDRSS_PHY_100[27-16] PHY_PAD_VREF_CTRL_DQ_0 DDRSS_PHY_356[27-16] PHY_PAD_VREF_CTRL_DQ_1 DDRSS_PHY_612[27-16] PHY_PAD_VREF_CTRL_DQ_2 DDRSS_PHY_868[27-16] PHY_PAD_VREF_CTRL_DQ_3	



**Table 8-20. Frequency Based Register Settings (continued)**

Register Settings	Category
DDRSS_PHY_98[7-0] PHY_DQ_OE_TIMING_0 DDRSS_PHY_354[7-0] PHY_DQ_OE_TIMING_1 DDRSS_PHY_610[7-0] PHY_DQ_OE_TIMING_2 DDRSS_PHY_866[7-0] PHY_DQ_OE_TIMING_3	Timing
DDRSS_PHY_98[15-8] PHY_DQ_TSEL_RD_TIMING_0 DDRSS_PHY_354[15-8] PHY_DQ_TSEL_RD_TIMING_1 DDRSS_PHY_610[15-8] PHY_DQ_TSEL_RD_TIMING_2 DDRSS_PHY_866[15-8] PHY_DQ_TSEL_RD_TIMING_3	
DDRSS_PHY_98[23-16] PHY_DQ_TSEL_WR_TIMING_0 DDRSS_PHY_354[23-16] PHY_DQ_TSEL_WR_TIMING_1 DDRSS_PHY_610[23-16] PHY_DQ_TSEL_WR_TIMING_2 DDRSS_PHY_866[23-16] PHY_DQ_TSEL_WR_TIMING_3	
DDRSS_PHY_98[31-24] PHY_DQS_OE_TIMING_0 DDRSS_PHY_354[31-24] PHY_DQS_OE_TIMING_1 DDRSS_PHY_610[31-24] PHY_DQS_OE_TIMING_2 DDRSS_PHY_866[31-24] PHY_DQS_OE_TIMING_3	
DDRSS_PHY_99[15-8] PHY_DQS_TSEL_RD_TIMING_0 DDRSS_PHY_355[15-8] PHY_DQS_TSEL_RD_TIMING_1 DDRSS_PHY_611[15-8] PHY_DQS_TSEL_RD_TIMING_2 DDRSS_PHY_867[15-8] PHY_DQS_TSEL_RD_TIMING_3	
DDRSS_PHY_99[31-24] PHY_DQS_TSEL_WR_TIMING_0 DDRSS_PHY_355[31-24] PHY_DQS_TSEL_WR_TIMING_1 DDRSS_PHY_611[31-24] PHY_DQS_TSEL_WR_TIMING_2 DDRSS_PHY_867[31-24] PHY_DQS_TSEL_WR_TIMING_3	
DDRSS_PHY_84[2-0] PHY_DQ_TSEL_ENABLE_0 DDRSS_PHY_340[2-0] PHY_DQ_TSEL_ENABLE_1 DDRSS_PHY_596[2-0] PHY_DQ_TSEL_ENABLE_2 DDRSS_PHY_852[2-0] PHY_DQ_TSEL_ENABLE_3	
DDRSS_PHY_84[23-8] PHY_DQ_TSEL_SELECT_0 DDRSS_PHY_340[23-8] PHY_DQ_TSEL_SELECT_1 DDRSS_PHY_596[23-8] PHY_DQ_TSEL_SELECT_2 DDRSS_PHY_852[23-8] PHY_DQ_TSEL_SELECT_3	
DDRSS_PHY_84[26-24] PHY_DQS_TSEL_ENABLE_0 DDRSS_PHY_340[26-24] PHY_DQS_TSEL_ENABLE_1 DDRSS_PHY_596[26-24] PHY_DQS_TSEL_ENABLE_2 DDRSS_PHY_852[26-24] PHY_DQS_TSEL_ENABLE_3	
DDRSS_PHY_85[15-0] PHY_DQS_TSEL_SELECT_0 DDRSS_PHY_341[15-0] PHY_DQS_TSEL_SELECT_1 DDRSS_PHY_597[15-0] PHY_DQS_TSEL_SELECT_2 DDRSS_PHY_853[15-0] PHY_DQS_TSEL_SELECT_3	
DDRSS_PHY_99[23-16] PHY_DQS_OE_RD_TIMING_0 DDRSS_PHY_355[23-16] PHY_DQS_OE_RD_TIMING_1 DDRSS_PHY_611[23-16] PHY_DQS_OE_RD_TIMING_2 DDRSS_PHY_867[23-16] PHY_DQS_OE_RD_TIMING_3	
DDRSS_PHY_101[15-8] PHY_DQ_IE_TIMING_0 DDRSS_PHY_357[15-8] PHY_DQ_IE_TIMING_1 DDRSS_PHY_613[15-8] PHY_DQ_IE_TIMING_2 DDRSS_PHY_869[15-8] PHY_DQ_IE_TIMING_3	
DDRSS_PHY_101[23-16] PHY_DQS_IE_TIMING_0 DDRSS_PHY_357[23-16] PHY_DQS_IE_TIMING_1 DDRSS_PHY_613[23-16] PHY_DQS_IE_TIMING_2 DDRSS_PHY_869[23-16] PHY_DQS_IE_TIMING_3	
DDRSS_PHY_92[21-16] PHY_PAD_DSLICE_IO_CFG_0 DDRSS_PHY_348[21-16] PHY_PAD_DSLICE_IO_CFG_1 DDRSS_PHY_604[21-16] PHY_PAD_DSLICE_IO_CFG_2 DDRSS_PHY_860[21-16] PHY_PAD_DSLICE_IO_CFG_3	Pad Controls
Address Slice	

**Table 8-20. Frequency Based Register Settings (continued)**

Register Settings	Category
DDRSS_PHY_1065[4-0] PHY_ADR0_SW_WRADDR_SHIFT_0 DDRSS_PHY_1065[28-24] PHY_ADR1_SW_WRADDR_SHIFT_0 DDRSS_PHY_1066[20-16] PHY_ADR2_SW_WRADDR_SHIFT_0 DDRSS_PHY_1067[20-16] PHY_ADR3_SW_WRADDR_SHIFT_0 DDRSS_PHY_1068[20-16] PHY_ADR4_SW_WRADDR_SHIFT_0 DDRSS_PHY_1069[20-16] PHY_ADR5_SW_WRADDR_SHIFT_0	Overrides
DDRSS_PHY_1070[19-16] PHY_ADR_SW_MASTER_MODE_0	Master delay line
DDRSS_PHY_1071[10-0] PHY_ADR_MASTER_DELAY_START_0	
DDRSS_PHY_1071[21-16] PHY_ADR_MASTER_DELAY_STEP_0	
DDRSS_PHY_1071[31-24] PHY_ADR_MASTER_DELAY_WAIT_0	
DDRSS_PHY_1072[7-0] PHY_ADR_MASTER_DELAY_HALF_MEASURE_0	Termination settings
DDRSS_PHY_1064[7-0] PHY_ADR_TSEL_SELECT_0	
DDRSS_PHY_1073[3-0] PHY_ADR_CALVL_DLY_STEP_0	CA training
DDRSS_PHY_1074[3-0] PHY_ADR_CALVL_CAPTURE_CNT_0	
DDRSS_PHY_1072[17-8] PHY_ADR_SW_CALVL_DVW_MIN_0	
DDRSS_PHY_1072[24] PHY_ADR_SW_CALVL_DVW_MIN_EN_0	
DDRSS_PHY_1065[18-8] PHY_ADR0_CLK_WR_SLAVE_DELAY_0 DDRSS_PHY_1066[10-0] PHY_ADR1_CLK_WR_SLAVE_DELAY_0 DDRSS_PHY_1067[10-0] PHY_ADR2_CLK_WR_SLAVE_DELAY_0 DDRSS_PHY_1068[10-0] PHY_ADR3_CLK_WR_SLAVE_DELAY_0 DDRSS_PHY_1069[10-0] PHY_ADR4_CLK_WR_SLAVE_DELAY_0 DDRSS_PHY_1070[10-0] PHY_ADR5_CLK_WR_SLAVE_DELAY_0	Address delay lines
DDRSS_PHY_1064[10-8] PHY_ADR_DC_CAL_CLK_SEL_0	Duty cycle correction
DDRSS_PHY_1074[25-16] PHY_ADR_DC_INIT_SLV_DELAY_0	
DDRSS_PHY_1075[0] PHY_ADR_DC_CALVL_ENABLE_0	
DDRSS_PHY_1075[15-8] PHY_ADR_DC_DM_CLK_THRSHLD_0	
DDRSS_PHY_1064[26-16] PHY_PAD_ADR_IO_CFG_0	Miscellaneous
PHY Level	
DDRSS_PHY_1396[12-0] PHY_PLL_CTRL	PLL settings
DDRSS_PHY_1395[0] PHY_PLL_BYPASS	
DDRSS_PHY_1405[2-0] PHY_CLK_DC_CAL_CLK_SEL	Duty cycle correction
DDRSS_PHY_1308[7-0] PHY_CLK_DC_DM_THRSHLD	
DDRSS_PHY_1399[10-0] PHY_GRP0_SLAVE_DELAY_0 DDRSS_PHY_1399[26-16] PHY_GRP1_SLAVE_DELAY_0 DDRSS_PHY_1400[10-0] PHY_GRP2_SLAVE_DELAY_0 DDRSS_PHY_1400[26-16] PHY_GRP3_SLAVE_DELAY_0 DDRSS_PHY_1401[10-0] PHY_GRP0_SLAVE_DELAY_1 DDRSS_PHY_1402[10-0] PHY_GRP1_SLAVE_DELAY_1 DDRSS_PHY_1403[10-0] PHY_GRP2_SLAVE_DELAY_1 DDRSS_PHY_1404[10-0] PHY_GRP3_SLAVE_DELAY_1	Address/control delay lines
DDRSS_PHY_1397[27-24] PHY_CSLVL_DLY_STEP	CS training
DDRSS_PHY_1396[16] PHY_LOW_FREQ_SEL	Low frequency select
DDRSS_PHY_1422[2-0] PHY_CAL_CLK_SELECT_0	Calibration
DDRSS_PHY_1422[30-24] PHY_CAL_SETTLING_PRD_0	
DDRSS_PHY_1422[23-8] PHY_CAL_VREF_SWITCH_TIMER_0	
DDRSS_PHY_1393[17-0] PHY_PAD_CAL_IO_CFG_0	

**Table 8-20. Frequency Based Register Settings (continued)**

Register Settings	Category
DDRSS_PHY_1406[29-0] PHY_PAD_FDBK_DRIVE	Pad Controls
DDRSS_PHY_1407[17-0] PHY_PAD_FDBK_DRIVE2	
DDRSS_PHY_1408[30-0] PHY_PAD_DATA_DRIVE	
DDRSS_PHY_1409[31-0] PHY_PAD_DQS_DRIVE	
DDRSS_PHY_1410[29-0] PHY_PAD_ADDR_DRIVE	
DDRSS_PHY_1411[26-0] PHY_PAD_ADDR_DRIVE2	
DDRSS_PHY_1412[31-0] PHY_PAD_CLK_DRIVE	
DDRSS_PHY_1413[17-0] PHY_PAD_CLK_DRIVE2	

Some register settings in the PHY are both chip select based and frequency based. For them, the PHY\_PER\_CS\_TRAINING\_INDEX\_0/1/2/3 bits and the DDRSS\_PHY\_1281[17-16] PHY\_FREQ\_SEL\_INDEX field apply to the register access. The DDRSS\_PHY\_1281[8] PHY\_FREQ\_SEL\_MULTICAST\_EN bit applies as well. [Table 8-21](#) shows the chip select based and frequency based register settings.

**Table 8-21. Chip Select Based and Frequency Based Register Settings**

Register Settings	Category
Data Slice	

**Table 8-21. Chip Select Based and Frequency Based Register Settings (continued)**

Register Settings	Category
DDRSS_PHY_628[10-0] PHY_CLK_WRDQ0_SLAVE_DELAY_2 DDRSS_PHY_628[26-16] PHY_CLK_WRDQ1_SLAVE_DELAY_2 DDRSS_PHY_629[10-0] PHY_CLK_WRDQ2_SLAVE_DELAY_2 DDRSS_PHY_629[26-16] PHY_CLK_WRDQ3_SLAVE_DELAY_2 DDRSS_PHY_630[10-0] PHY_CLK_WRDQ4_SLAVE_DELAY_2 DDRSS_PHY_630[26-16] PHY_CLK_WRDQ5_SLAVE_DELAY_2 DDRSS_PHY_631[10-0] PHY_CLK_WRDQ6_SLAVE_DELAY_2 DDRSS_PHY_631[26-16] PHY_CLK_WRDQ7_SLAVE_DELAY_2 DDRSS_PHY_884[10-0] PHY_CLK_WRDQ0_SLAVE_DELAY_3 DDRSS_PHY_884[26-16] PHY_CLK_WRDQ1_SLAVE_DELAY_3 DDRSS_PHY_885[10-0] PHY_CLK_WRDQ2_SLAVE_DELAY_3 DDRSS_PHY_885[26-16] PHY_CLK_WRDQ3_SLAVE_DELAY_3 DDRSS_PHY_886[10-0] PHY_CLK_WRDQ4_SLAVE_DELAY_3 DDRSS_PHY_886[26-16] PHY_CLK_WRDQ5_SLAVE_DELAY_3 DDRSS_PHY_887[10-0] PHY_CLK_WRDQ6_SLAVE_DELAY_3 DDRSS_PHY_887[26-16] PHY_CLK_WRDQ7_SLAVE_DELAY_3 DDRSS_PHY_116[10-0] PHY_CLK_WRDQ0_SLAVE_DELAY_0 DDRSS_PHY_116[26-16] PHY_CLK_WRDQ1_SLAVE_DELAY_0 DDRSS_PHY_117[10-0] PHY_CLK_WRDQ2_SLAVE_DELAY_0 DDRSS_PHY_117[26-16] PHY_CLK_WRDQ3_SLAVE_DELAY_0 DDRSS_PHY_118[10-0] PHY_CLK_WRDQ4_SLAVE_DELAY_0 DDRSS_PHY_118[26-16] PHY_CLK_WRDQ5_SLAVE_DELAY_0 DDRSS_PHY_119[10-0] PHY_CLK_WRDQ6_SLAVE_DELAY_0 DDRSS_PHY_119[26-16] PHY_CLK_WRDQ7_SLAVE_DELAY_0 DDRSS_PHY_372[10-0] PHY_CLK_WRDQ0_SLAVE_DELAY_1 DDRSS_PHY_372[26-16] PHY_CLK_WRDQ1_SLAVE_DELAY_1 DDRSS_PHY_373[10-0] PHY_CLK_WRDQ2_SLAVE_DELAY_1 DDRSS_PHY_373[26-16] PHY_CLK_WRDQ3_SLAVE_DELAY_1 DDRSS_PHY_374[10-0] PHY_CLK_WRDQ4_SLAVE_DELAY_1 DDRSS_PHY_374[26-16] PHY_CLK_WRDQ5_SLAVE_DELAY_1 DDRSS_PHY_375[10-0] PHY_CLK_WRDQ6_SLAVE_DELAY_1 DDRSS_PHY_375[26-16] PHY_CLK_WRDQ7_SLAVE_DELAY_1	Write path
DDRSS_PHY_120[10-0] PHY_CLK_WRDM_SLAVE_DELAY_0 DDRSS_PHY_376[10-0] PHY_CLK_WRDM_SLAVE_DELAY_1 DDRSS_PHY_632[10-0] PHY_CLK_WRDM_SLAVE_DELAY_2 DDRSS_PHY_888[10-0] PHY_CLK_WRDM_SLAVE_DELAY_3	
DDRSS_PHY_120[25-16] PHY_CLK_WRDQS_SLAVE_DELAY_0 DDRSS_PHY_376[25-16] PHY_CLK_WRDQS_SLAVE_DELAY_1 DDRSS_PHY_632[25-16] PHY_CLK_WRDQS_SLAVE_DELAY_2 DDRSS_PHY_888[25-16] PHY_CLK_WRDQS_SLAVE_DELAY_3	
DDRSS_PHY_131[10-8] PHY_WRITE_PATH_LAT_ADD_0 DDRSS_PHY_387[10-8] PHY_WRITE_PATH_LAT_ADD_1 DDRSS_PHY_643[10-8] PHY_WRITE_PATH_LAT_ADD_2 DDRSS_PHY_899[10-8] PHY_WRITE_PATH_LAT_ADD_3	

**Table 8-21. Chip Select Based and Frequency Based Register Settings (continued)**

Register Settings	Category
DDRSS_PHY_136[15-8] PHY_DATA_DC_DQ0_CLK_ADJUST_0 DDRSS_PHY_136[23-16] PHY_DATA_DC_DQ1_CLK_ADJUST_0 DDRSS_PHY_136[31-24] PHY_DATA_DC_DQ2_CLK_ADJUST_0 DDRSS_PHY_137[7-0] PHY_DATA_DC_DQ3_CLK_ADJUST_0 DDRSS_PHY_137[15-8] PHY_DATA_DC_DQ4_CLK_ADJUST_0 DDRSS_PHY_137[23-16] PHY_DATA_DC_DQ5_CLK_ADJUST_0 DDRSS_PHY_137[31-24] PHY_DATA_DC_DQ6_CLK_ADJUST_0 DDRSS_PHY_138[7-0] PHY_DATA_DC_DQ7_CLK_ADJUST_0 DDRSS_PHY_392[15-8] PHY_DATA_DC_DQ0_CLK_ADJUST_1 DDRSS_PHY_392[23-16] PHY_DATA_DC_DQ1_CLK_ADJUST_1 DDRSS_PHY_392[31-24] PHY_DATA_DC_DQ2_CLK_ADJUST_1 DDRSS_PHY_393[7-0] PHY_DATA_DC_DQ3_CLK_ADJUST_1 DDRSS_PHY_393[15-8] PHY_DATA_DC_DQ4_CLK_ADJUST_1 DDRSS_PHY_393[23-16] PHY_DATA_DC_DQ5_CLK_ADJUST_1 DDRSS_PHY_393[31-24] PHY_DATA_DC_DQ6_CLK_ADJUST_1 DDRSS_PHY_394[7-0] PHY_DATA_DC_DQ7_CLK_ADJUST_1 DDRSS_PHY_648[15-8] PHY_DATA_DC_DQ0_CLK_ADJUST_2 DDRSS_PHY_648[23-16] PHY_DATA_DC_DQ1_CLK_ADJUST_2 DDRSS_PHY_648[31-24] PHY_DATA_DC_DQ2_CLK_ADJUST_2 DDRSS_PHY_649[7-0] PHY_DATA_DC_DQ3_CLK_ADJUST_2 DDRSS_PHY_649[15-8] PHY_DATA_DC_DQ4_CLK_ADJUST_2 DDRSS_PHY_649[23-16] PHY_DATA_DC_DQ5_CLK_ADJUST_2 DDRSS_PHY_649[31-24] PHY_DATA_DC_DQ6_CLK_ADJUST_2 DDRSS_PHY_650[7-0] PHY_DATA_DC_DQ7_CLK_ADJUST_2 DDRSS_PHY_904[15-8] PHY_DATA_DC_DQ0_CLK_ADJUST_3 DDRSS_PHY_904[23-16] PHY_DATA_DC_DQ1_CLK_ADJUST_3 DDRSS_PHY_904[31-24] PHY_DATA_DC_DQ2_CLK_ADJUST_3 DDRSS_PHY_905[7-0] PHY_DATA_DC_DQ3_CLK_ADJUST_3 DDRSS_PHY_905[15-8] PHY_DATA_DC_DQ4_CLK_ADJUST_3 DDRSS_PHY_905[23-16] PHY_DATA_DC_DQ5_CLK_ADJUST_3 DDRSS_PHY_905[31-24] PHY_DATA_DC_DQ6_CLK_ADJUST_3 DDRSS_PHY_906[7-0] PHY_DATA_DC_DQ7_CLK_ADJUST_3	Duty cycle correction
DDRSS_PHY_138[15-8] PHY_DATA_DC_DM_CLK_ADJUST_0 DDRSS_PHY_394[15-8] PHY_DATA_DC_DM_CLK_ADJUST_1 DDRSS_PHY_650[15-8] PHY_DATA_DC_DM_CLK_ADJUST_2 DDRSS_PHY_906[15-8] PHY_DATA_DC_DM_CLK_ADJUST_3	Duty cycle correction
DDRSS_PHY_136[7-0] PHY_DATA_DC_DQS_CLK_ADJUST_0 DDRSS_PHY_392[7-0] PHY_DATA_DC_DQS_CLK_ADJUST_1 DDRSS_PHY_648[7-0] PHY_DATA_DC_DQS_CLK_ADJUST_2 DDRSS_PHY_904[7-0] PHY_DATA_DC_DQS_CLK_ADJUST_3	Duty cycle correction
DDRSS_PHY_131[25-16] PHY_WRLVL_DELAY_EARLY_THRESHOLD_0 DDRSS_PHY_387[25-16] PHY_WRLVL_DELAY_EARLY_THRESHOLD_1 DDRSS_PHY_643[25-16] PHY_WRLVL_DELAY_EARLY_THRESHOLD_2 DDRSS_PHY_899[25-16] PHY_WRLVL_DELAY_EARLY_THRESHOLD_3	Write leveling
DDRSS_PHY_132[9-0] PHY_WRLVL_DELAY_PERIOD_THRESHOLD_0 DDRSS_PHY_388[9-0] PHY_WRLVL_DELAY_PERIOD_THRESHOLD_1 DDRSS_PHY_644[9-0] PHY_WRLVL_DELAY_PERIOD_THRESHOLD_2 DDRSS_PHY_900[9-0] PHY_WRLVL_DELAY_PERIOD_THRESHOLD_3	
DDRSS_PHY_132[16] PHY_WRLVL_EARLY_FORCE_ZERO_0 DDRSS_PHY_388[16] PHY_WRLVL_EARLY_FORCE_ZERO_1 DDRSS_PHY_644[16] PHY_WRLVL_EARLY_FORCE_ZERO_2 DDRSS_PHY_900[16] PHY_WRLVL_EARLY_FORCE_ZERO_3	
DDRSS_PHY_121[1-0] PHY_WRLVL_THRESHOLD_ADJUST_0 DDRSS_PHY_377[1-0] PHY_WRLVL_THRESHOLD_ADJUST_1 DDRSS_PHY_633[1-0] PHY_WRLVL_THRESHOLD_ADJUST_2 DDRSS_PHY_889[1-0] PHY_WRLVL_THRESHOLD_ADJUST_3	

**Table 8-21. Chip Select Based and Frequency Based Register Settings (continued)**

Register Settings	Category
DDRSS_PHY_134[10-0] PHY_WDQLVL_DQDM_SLV_DLY_START_0 DDRSS_PHY_390[10-0] PHY_WDQLVL_DQDM_SLV_DLY_START_1 DDRSS_PHY_646[10-0] PHY_WDQLVL_DQDM_SLV_DLY_START_2 DDRSS_PHY_902[10-0] PHY_WDQLVL_DQDM_SLV_DLY_START_3	Write DQ training
DDRSS_PHY_134[19-16] PHY_NTP_WRLAT_START_0 DDRSS_PHY_390[19-16] PHY_NTP_WRLAT_START_1 DDRSS_PHY_646[19-16] PHY_NTP_WRLAT_START_2 DDRSS_PHY_902[19-16] PHY_NTP_WRLAT_START_3	
DDRSS_PHY_134[24] PHY_NTP_PASS_0 DDRSS_PHY_390[24] PHY_NTP_PASS_1 DDRSS_PHY_646[24] PHY_NTP_PASS_2 DDRSS_PHY_902[24] PHY_NTP_PASS_3	
DDRSS_PHY_121[17-8] PHY_RDDQS_DQ0_RISE_SLAVE_DELAY_0 DDRSS_PHY_122[25-16] PHY_RDDQS_DQ1_RISE_SLAVE_DELAY_0 DDRSS_PHY_123[25-16] PHY_RDDQS_DQ2_RISE_SLAVE_DELAY_0 DDRSS_PHY_124[25-16] PHY_RDDQS_DQ3_RISE_SLAVE_DELAY_0 DDRSS_PHY_125[25-16] PHY_RDDQS_DQ4_RISE_SLAVE_DELAY_0 DDRSS_PHY_126[25-16] PHY_RDDQS_DQ5_RISE_SLAVE_DELAY_0 DDRSS_PHY_127[25-16] PHY_RDDQS_DQ6_RISE_SLAVE_DELAY_0 DDRSS_PHY_128[25-16] PHY_RDDQS_DQ7_RISE_SLAVE_DELAY_0 DDRSS_PHY_377[17-8] PHY_RDDQS_DQ0_RISE_SLAVE_DELAY_1 DDRSS_PHY_378[25-16] PHY_RDDQS_DQ1_RISE_SLAVE_DELAY_1 DDRSS_PHY_379[25-16] PHY_RDDQS_DQ2_RISE_SLAVE_DELAY_1 DDRSS_PHY_380[25-16] PHY_RDDQS_DQ3_RISE_SLAVE_DELAY_1 DDRSS_PHY_381[25-16] PHY_RDDQS_DQ4_RISE_SLAVE_DELAY_1 DDRSS_PHY_382[25-16] PHY_RDDQS_DQ5_RISE_SLAVE_DELAY_1 DDRSS_PHY_383[25-16] PHY_RDDQS_DQ6_RISE_SLAVE_DELAY_1 DDRSS_PHY_384[25-16] PHY_RDDQS_DQ7_RISE_SLAVE_DELAY_1 DDRSS_PHY_633[17-8] PHY_RDDQS_DQ0_RISE_SLAVE_DELAY_2 DDRSS_PHY_634[25-16] PHY_RDDQS_DQ1_RISE_SLAVE_DELAY_2 DDRSS_PHY_635[25-16] PHY_RDDQS_DQ2_RISE_SLAVE_DELAY_2 DDRSS_PHY_636[25-16] PHY_RDDQS_DQ3_RISE_SLAVE_DELAY_2 DDRSS_PHY_637[25-16] PHY_RDDQS_DQ4_RISE_SLAVE_DELAY_2 DDRSS_PHY_638[25-16] PHY_RDDQS_DQ5_RISE_SLAVE_DELAY_2 DDRSS_PHY_639[25-16] PHY_RDDQS_DQ6_RISE_SLAVE_DELAY_2 DDRSS_PHY_640[25-16] PHY_RDDQS_DQ7_RISE_SLAVE_DELAY_2 DDRSS_PHY_889[17-8] PHY_RDDQS_DQ0_RISE_SLAVE_DELAY_3 DDRSS_PHY_890[25-16] PHY_RDDQS_DQ1_RISE_SLAVE_DELAY_3 DDRSS_PHY_891[25-16] PHY_RDDQS_DQ2_RISE_SLAVE_DELAY_3 DDRSS_PHY_892[25-16] PHY_RDDQS_DQ3_RISE_SLAVE_DELAY_3 DDRSS_PHY_893[25-16] PHY_RDDQS_DQ4_RISE_SLAVE_DELAY_3 DDRSS_PHY_894[25-16] PHY_RDDQS_DQ5_RISE_SLAVE_DELAY_3 DDRSS_PHY_895[25-16] PHY_RDDQS_DQ6_RISE_SLAVE_DELAY_3 DDRSS_PHY_896[25-16] PHY_RDDQS_DQ7_RISE_SLAVE_DELAY_3	Read path

**Table 8-21. Chip Select Based and Frequency Based Register Settings (continued)**

Register Settings	Category
DDRSS_PHY_122[9-0] PHY_RDDQS_DQ0_FALL_SLAVE_DELAY_0 DDRSS_PHY_123[9-0] PHY_RDDQS_DQ1_FALL_SLAVE_DELAY_0 DDRSS_PHY_124[9-0] PHY_RDDQS_DQ2_FALL_SLAVE_DELAY_0 DDRSS_PHY_125[9-0] PHY_RDDQS_DQ3_FALL_SLAVE_DELAY_0 DDRSS_PHY_126[9-0] PHY_RDDQS_DQ4_FALL_SLAVE_DELAY_0 DDRSS_PHY_127[9-0] PHY_RDDQS_DQ5_FALL_SLAVE_DELAY_0 DDRSS_PHY_128[9-0] PHY_RDDQS_DQ6_FALL_SLAVE_DELAY_0 DDRSS_PHY_129[9-0] PHY_RDDQS_DQ7_FALL_SLAVE_DELAY_0 DDRSS_PHY_378[9-0] PHY_RDDQS_DQ0_FALL_SLAVE_DELAY_1 DDRSS_PHY_379[9-0] PHY_RDDQS_DQ1_FALL_SLAVE_DELAY_1 DDRSS_PHY_380[9-0] PHY_RDDQS_DQ2_FALL_SLAVE_DELAY_1 DDRSS_PHY_381[9-0] PHY_RDDQS_DQ3_FALL_SLAVE_DELAY_1 DDRSS_PHY_382[9-0] PHY_RDDQS_DQ4_FALL_SLAVE_DELAY_1 DDRSS_PHY_383[9-0] PHY_RDDQS_DQ5_FALL_SLAVE_DELAY_1 DDRSS_PHY_384[9-0] PHY_RDDQS_DQ6_FALL_SLAVE_DELAY_1 DDRSS_PHY_385[9-0] PHY_RDDQS_DQ7_FALL_SLAVE_DELAY_1 DDRSS_PHY_634[9-0] PHY_RDDQS_DQ0_FALL_SLAVE_DELAY_2 DDRSS_PHY_635[9-0] PHY_RDDQS_DQ1_FALL_SLAVE_DELAY_2 DDRSS_PHY_636[9-0] PHY_RDDQS_DQ2_FALL_SLAVE_DELAY_2 DDRSS_PHY_637[9-0] PHY_RDDQS_DQ3_FALL_SLAVE_DELAY_2 DDRSS_PHY_638[9-0] PHY_RDDQS_DQ4_FALL_SLAVE_DELAY_2 DDRSS_PHY_639[9-0] PHY_RDDQS_DQ5_FALL_SLAVE_DELAY_2 DDRSS_PHY_640[9-0] PHY_RDDQS_DQ6_FALL_SLAVE_DELAY_2 DDRSS_PHY_641[9-0] PHY_RDDQS_DQ7_FALL_SLAVE_DELAY_2 DDRSS_PHY_890[9-0] PHY_RDDQS_DQ0_FALL_SLAVE_DELAY_3 DDRSS_PHY_891[9-0] PHY_RDDQS_DQ1_FALL_SLAVE_DELAY_3 DDRSS_PHY_892[9-0] PHY_RDDQS_DQ2_FALL_SLAVE_DELAY_3 DDRSS_PHY_893[9-0] PHY_RDDQS_DQ3_FALL_SLAVE_DELAY_3 DDRSS_PHY_894[9-0] PHY_RDDQS_DQ4_FALL_SLAVE_DELAY_3 DDRSS_PHY_895[9-0] PHY_RDDQS_DQ5_FALL_SLAVE_DELAY_3 DDRSS_PHY_896[9-0] PHY_RDDQS_DQ6_FALL_SLAVE_DELAY_3 DDRSS_PHY_897[9-0] PHY_RDDQS_DQ7_FALL_SLAVE_DELAY_3	Read path
DDRSS_PHY_129[25-16] PHY_RDDQS_DM_RISE_SLAVE_DELAY_0 DDRSS_PHY_385[25-16] PHY_RDDQS_DM_RISE_SLAVE_DELAY_1 DDRSS_PHY_641[25-16] PHY_RDDQS_DM_RISE_SLAVE_DELAY_2 DDRSS_PHY_897[25-16] PHY_RDDQS_DM_RISE_SLAVE_DELAY_3	Read path
DDRSS_PHY_130[9-0] PHY_RDDQS_DM_FALL_SLAVE_DELAY_0 DDRSS_PHY_386[9-0] PHY_RDDQS_DM_FALL_SLAVE_DELAY_1 DDRSS_PHY_642[9-0] PHY_RDDQS_DM_FALL_SLAVE_DELAY_2 DDRSS_PHY_898[9-0] PHY_RDDQS_DM_FALL_SLAVE_DELAY_3	
DDRSS_PHY_130[25-16] PHY_RDDQS_GATE_SLAVE_DELAY_0 DDRSS_PHY_386[25-16] PHY_RDDQS_GATE_SLAVE_DELAY_1 DDRSS_PHY_642[25-16] PHY_RDDQS_GATE_SLAVE_DELAY_2 DDRSS_PHY_898[25-16] PHY_RDDQS_GATE_SLAVE_DELAY_3	
DDRSS_PHY_131[3-0] PHY_RDDQS_LATENCY_ADJUST_0 DDRSS_PHY_387[3-0] PHY_RDDQS_LATENCY_ADJUST_1 DDRSS_PHY_643[3-0] PHY_RDDQS_LATENCY_ADJUST_2 DDRSS_PHY_899[3-0] PHY_RDDQS_LATENCY_ADJUST_3	
DDRSS_PHY_133[9-0] PHY_GTLVL_RDDQS_SLV_DLY_START_0 DDRSS_PHY_389[9-0] PHY_GTLVL_RDDQS_SLV_DLY_START_1 DDRSS_PHY_645[9-0] PHY_GTLVL_RDDQS_SLV_DLY_START_2 DDRSS_PHY_901[9-0] PHY_GTLVL_RDDQS_SLV_DLY_START_3	Gate training
DDRSS_PHY_133[19-16] PHY_GTLVL_LAT_ADJ_START_0 DDRSS_PHY_389[19-16] PHY_GTLVL_LAT_ADJ_START_1 DDRSS_PHY_645[19-16] PHY_GTLVL_LAT_ADJ_START_2 DDRSS_PHY_901[19-16] PHY_GTLVL_LAT_ADJ_START_3	
DDRSS_PHY_135[9-0] PHY_RDLVL_RDDQS_DQ_SLV_DLY_START_0 DDRSS_PHY_391[9-0] PHY_RDLVL_RDDQS_DQ_SLV_DLY_START_1 DDRSS_PHY_647[9-0] PHY_RDLVL_RDDQS_DQ_SLV_DLY_START_2 DDRSS_PHY_903[9-0] PHY_RDLVL_RDDQS_DQ_SLV_DLY_START_3	Read data eye training



**Table 8-21. Chip Select Based and Frequency Based Register Settings (continued)**

Register Settings	Category
DDRSS_PHY_138[31-16] PHY_DSLICE_PAD_BOOSTPN_SETTING_0 DDRSS_PHY_394[31-16] PHY_DSLICE_PAD_BOOSTPN_SETTING_1 DDRSS_PHY_650[31-16] PHY_DSLICE_PAD_BOOSTPN_SETTING_2 DDRSS_PHY_906[31-16] PHY_DSLICE_PAD_BOOSTPN_SETTING_3	Miscellaneous
DDRSS_PHY_139[5-0] PHY_DSLICE_PAD_RX_CTL_E_SETTING_0 DDRSS_PHY_395[5-0] PHY_DSLICE_PAD_RX_CTL_E_SETTING_1 DDRSS_PHY_651[5-0] PHY_DSLICE_PAD_RX_CTL_E_SETTING_2 DDRSS_PHY_907[5-0] PHY_DSLICE_PAD_RX_CTL_E_SETTING_3	

#### 8.2.4.7.8 Low-Power Modes

The DDR PHY provides a means to reduce the operational power when it is not actively transferring read or write data for an extended period of time. This is done using the DFI low power interface. Using this interface, the DDR controller communicates to the PHY that it may enter its own low-power state and informs the PHY how quickly it requires the PHY to recover if a low-power state is entered. This DDR PHY has the following power modes:

- **Idle:** This is a period in which no read or write activity is being performed by the DDR controller. The PHY is not in any low power state, and is ready to accept commands at any time.
- **Read/Write:** This describes the time period when the PHY is actively performing a read or write operation triggered by the DDR controller.
- **Light Sleep:** This is a low-power state in which the master delay lines and their logic receive a clock, but all other slice logic is gated off. The PHY PLLs remain in an active state. Register reads and writes to data and address slices are not permitted. The PHY returns quickly to functional operation after an exit from light sleep. After exiting the PHY initializes the slave delay settings automatically. The master delay line macro maintains lock. This mode is entered if the dfi\_lp\_wakeup signal is driven with a value less than or equal to the programmed value in bits [3:0] of the the DDRSS\_PHY\_1318[15-8] PHY\_LP\_WAKEUP field. If low power idle state is enabled, the value must also be greater than the value of bits [7:4] in the DDRSS\_PHY\_1318[15-8] PHY\_LP\_WAKEUP field.
- **Deep Sleep:** This is an ultra low-power state in which clocking to the data, address, and address/control slices is disabled. The PLLs are in a power down state. Pad calibration functions are frozen. When in this mode, no DFI accesses or register reads and writes can be performed. This mode is entered if the dfi\_lp\_wakeup signal is driven with a value greater than the programmed value in bits [3:0] of the the DDRSS\_PHY\_1318[15-8] PHY\_LP\_WAKEUP field.
- **Low Power Idle:** This is a low power version of idle where the data slices are placed in a light sleep mode while still allowing accesses to the data slice register address space. The DDRSS\_PHY\_1318[16] PHY\_LS\_IDLE\_EN bit enables this low power idle state. The mode is entered if the dfi\_lp\_wakeup signal is driven with a value less than or equal to the programmed value of bits [7:4] in the DDRSS\_PHY\_1318[15-8] PHY\_LP\_WAKEUP field.

In general, if the SDRAM is placed into a power-down mode, the DDR PHY should enter light sleep. If the SDRAM is placed in a self-refresh mode, the DDR PHY could enter deep sleep.

The DDRSS\_PHY\_1319[9-0] PHY\_LP\_CTRLUPD\_CNTR\_CFG field controls the number of clock controller cycles between de-assertion of the dfi\_lp\_req signal and de-assertion of the dfi\_lp\_ack signal during exit of light sleep and low power idle modes.

Within each data slice, the DQ, DM and DQS I/O's are disabled when not being written to or read from. The gate feedback pads are constantly enabled to control the read DQS gate open logic. The following fields control disabling the gate feedback pad for the corresponding slice during deep sleep mode:

- DDRSS\_PHY\_79[26-24] PHY\_FDBK\_PWR\_CTRL\_0
- DDRSS\_PHY\_335[26-24] PHY\_FDBK\_PWR\_CTRL\_1
- DDRSS\_PHY\_591[26-24] PHY\_FDBK\_PWR\_CTRL\_2
- DDRSS\_PHY\_847[26-24] PHY\_FDBK\_PWR\_CTRL\_3



#### 8.2.4.7.9 Training Support

Training of the DDR PHY and SDRAM is necessary to support high-speed operation. The general training flow as follows:

1. I/O calibration
2. CA/CS training (VREF CA)
3. Write leveling
4. Read DQS gate training
5. Read data eye training
6. Write data eye training (VREF DQ)

##### 8.2.4.7.9.1 Write Leveling

When the write leveling algorithm has completed, the final write DQS delay settings can be found in the following fields:

- DDRSS\_PHY\_120[25-16] PHY\_CLK\_WRDQS\_SLAVE\_DELAY\_0
- DDRSS\_PHY\_376[25-16] PHY\_CLK\_WRDQS\_SLAVE\_DELAY\_1
- DDRSS\_PHY\_632[25-16] PHY\_CLK\_WRDQS\_SLAVE\_DELAY\_2
- DDRSS\_PHY\_888[25-16] PHY\_CLK\_WRDQS\_SLAVE\_DELAY\_3
- DDRSS\_PHY\_121[1-0] PHY\_WRLVL\_THRESHOLD\_ADJUST\_0
- DDRSS\_PHY\_377[1-0] PHY\_WRLVL\_THRESHOLD\_ADJUST\_1
- DDRSS\_PHY\_633[1-0] PHY\_WRLVL\_THRESHOLD\_ADJUST\_2
- DDRSS\_PHY\_889[1-0] PHY\_WRLVL\_THRESHOLD\_ADJUST\_3

After write leveling is complete, the following fields can be checked to obtain the leveling status:

- DDRSS\_PHY\_50[16-0] PHY\_WRLVL\_STATUS\_OBS\_0
- DDRSS\_PHY\_306[16-0] PHY\_WRLVL\_STATUS\_OBS\_1
- DDRSS\_PHY\_562[16-0] PHY\_WRLVL\_STATUS\_OBS\_2
- DDRSS\_PHY\_818[16-0] PHY\_WRLVL\_STATUS\_OBS\_3

After write leveling is complete, software can always override the results by writing directly to the following fields:

- DDRSS\_PHY\_120[25-16] PHY\_CLK\_WRDQS\_SLAVE\_DELAY\_0
- DDRSS\_PHY\_376[25-16] PHY\_CLK\_WRDQS\_SLAVE\_DELAY\_1
- DDRSS\_PHY\_632[25-16] PHY\_CLK\_WRDQS\_SLAVE\_DELAY\_2
- DDRSS\_PHY\_888[25-16] PHY\_CLK\_WRDQS\_SLAVE\_DELAY\_3

If the clock path to the SDRAM is longer than the DQS path to the SDRAM, the leveling result from the first SDRAM along the memory clock path always returns a small value greater than zero because the DQS signal needs to be moved slightly to the right to line up with the memory clock. If the clock path to the SDRAM is shorter than the DQS path to the SDRAM, the DQS signal may need to be moved close to a cycle before it lines up with the memory clock. The leveling result indicates to the slice that the DQS needs to be delayed almost a cycle. This would then result in the write data being a cycle later than expected by the SDRAM based on when the write command is received. In reality, the DQS needs to be negatively delayed to align to the memory clock in the proper cycle. To create this “negative” delay, the command bus needs to be delayed one cycle so that when the DQS is delayed almost one cycle, the DQS appears “early” with respect to the write command. For example, if the memory clock path for slice 0 is expected to be about 1/16th cycle shorter than the DQS path, a write leveling result of 0x1E0 would be expected. The DDRSS\_PHY\_131[25-16] PHY\_WRLVL\_DELAY\_EARLY\_THRESHOLD\_0 field enables the data slice 0 to understand if this is an expected delay or if this is a “negative” delay condition. When the result contained in DDRSS\_PHY\_120[25-16] PHY\_CLK\_WRDQS\_SLAVE\_DELAY\_0 field is greater than the DDRSS\_PHY\_131[25-16] PHY\_WRLVL\_DELAY\_EARLY\_THRESHOLD\_0 field, the command bus is delayed by one cycle. [Table 8-22](#) shows examples of programming this field for various expected delays. The following fields are associated with the other data slices:

- DDRSS\_PHY\_387[25-16] PHY\_WRLVL\_DELAY\_EARLY\_THRESHOLD\_1
- DDRSS\_PHY\_643[25-16] PHY\_WRLVL\_DELAY\_EARLY\_THRESHOLD\_2
- DDRSS\_PHY\_889[25-16] PHY\_WRLVL\_DELAY\_EARLY\_THRESHOLD\_3

**Table 8-22. PHY Write Level Delay Values**

Expected Delay	PHY_WRLVL_DELAY_EARLY_THRESHOLD_ <b>x</b> <b>x</b> = 0, 1, 2, 3	Command Bus Impact
0.25 cycle	0x200 (off)	None
1.00 cycle	0x200 (off)	None
0.00 cycles	0x1E0	1 cycle delay if result is > 0x1E0

The DDRSS\_PHY\_132[16] PHY\_WRLVL\_EARLY\_FORCE\_ZERO\_0 bit works with the DDRSS\_PHY\_131[25-16] PHY\_WRLVL\_DELAY\_EARLY\_THRESHOLD\_0 field for slice 0. The following fields are associated with the other slices:

- DDRSS\_PHY\_388[16] PHY\_WRLVL\_EARLY\_FORCE\_ZERO\_1 and DDRSS\_PHY\_387[25-16] PHY\_WRLVL\_DELAY\_EARLY\_THRESHOLD\_1
- DDRSS\_PHY\_644[16] PHY\_WRLVL\_EARLY\_FORCE\_ZERO\_2 and DDRSS\_PHY\_643[25-16] PHY\_WRLVL\_DELAY\_EARLY\_THRESHOLD\_2
- DDRSS\_PHY\_900[16] PHY\_WRLVL\_EARLY\_FORCE\_ZERO\_3 and DDRSS\_PHY\_899[25-16] PHY\_WRLVL\_DELAY\_EARLY\_THRESHOLD\_3

When the write leveling result satisfies the early threshold condition, the DQS is ideally aligned to clock but there is a cycle of latency added to the command path. Setting the PHY\_WRLVL\_EARLY\_FORCE\_ZERO\_ **x** bit to 0x1 forces the PHY\_CLK\_WRDQS\_SLAVE\_DELAY\_ **x** value to 0x0 so there is no command latency impact.

The PHY\_WRLVL\_THRESHOLD\_ADJUST\_ **x** fields contain the results of the PHY\_WRLVL\_DELAY\_PERIOD\_THRESHOLD\_ **x** and PHY\_WRLVL\_DELAY\_EARLY\_THRESHOLD\_ **x** calculations from the write leveling process. This allows for direct observation of the leveling results. It also allows to rewrite leveling results if desired. This can be useful in case leveling is performed once and the same results are used in the future. The PHY\_WRLVL\_DELAY\_EARLY\_THRESHOLD\_ **x** and PHY\_WRLVL\_DELAY\_PERIOD\_THRESHOLD\_ **x** results can be applied through the PHY\_WRLVL\_THRESHOLD\_ADJUST\_ **x** fields without having to go through the leveling process again.

The previously mentioned PHY\_WRLVL\_DELAY\_PERIOD\_THRESHOLD\_ **x** fields are accessed through the following registers:

- DDRSS\_PHY\_132[9-0] PHY\_WRLVL\_DELAY\_PERIOD\_THRESHOLD\_0
- DDRSS\_PHY\_388[9-0] PHY\_WRLVL\_DELAY\_PERIOD\_THRESHOLD\_1
- DDRSS\_PHY\_644[9-0] PHY\_WRLVL\_DELAY\_PERIOD\_THRESHOLD\_2
- DDRSS\_PHY\_900[9-0] PHY\_WRLVL\_DELAY\_PERIOD\_THRESHOLD\_3

#### 8.2.4.7.9.2 Read Gate Training

When the read gate training algorithm has completed, the final read DQS slave delay settings can be found in the following fields:

- DDRSS\_PHY\_130[25-16] PHY\_RDDQS\_GATE\_SLAVE\_DELAY\_0 (cycle fraction)
- DDRSS\_PHY\_386[25-16] PHY\_RDDQS\_GATE\_SLAVE\_DELAY\_1 (cycle fraction)
- DDRSS\_PHY\_642[25-16] PHY\_RDDQS\_GATE\_SLAVE\_DELAY\_2 (cycle fraction)
- DDRSS\_PHY\_898[25-16] PHY\_RDDQS\_GATE\_SLAVE\_DELAY\_3 (cycle fraction)
- DDRSS\_PHY\_131[3-0] PHY\_RDDQS\_LATENCY\_ADJUST\_0 (cycle offset)
- DDRSS\_PHY\_387[3-0] PHY\_RDDQS\_LATENCY\_ADJUST\_1 (cycle offset)
- DDRSS\_PHY\_643[3-0] PHY\_RDDQS\_LATENCY\_ADJUST\_2 (cycle offset)
- DDRSS\_PHY\_899[3-0] PHY\_RDDQS\_LATENCY\_ADJUST\_3 (cycle offset)

After read gate training is complete, the following fields can be checked to obtain the training status:

- DDRSS\_PHY\_54[17-0] PHY\_GTLVL\_STATUS\_OBS\_0
- DDRSS\_PHY\_310[17-0] PHY\_GTLVL\_STATUS\_OBS\_1
- DDRSS\_PHY\_566[17-0] PHY\_GTLVL\_STATUS\_OBS\_2
- DDRSS\_PHY\_822[17-0] PHY\_GTLVL\_STATUS\_OBS\_3

After read gate training is complete, software can always override the results by writing directly to the PHY\_RDDQS\_GATE\_SLAVE\_DELAY\_x and PHY\_RDDQS\_LATENCY\_ADJUST\_x fields.

#### 8.2.4.7.9.3 Read Data Eye Training

When the read gate training algorithm has completed, the final read DQS slave delay settings can be found in the following fields:

- DDRSS\_PHY\_121[17-8] PHY\_RDDQS\_DQ0\_RISE\_SLAVE\_DELAY\_0
- DDRSS\_PHY\_122[25-16] PHY\_RDDQS\_DQ1\_RISE\_SLAVE\_DELAY\_0
- DDRSS\_PHY\_123[25-16] PHY\_RDDQS\_DQ2\_RISE\_SLAVE\_DELAY\_0
- DDRSS\_PHY\_124[25-16] PHY\_RDDQS\_DQ3\_RISE\_SLAVE\_DELAY\_0
- DDRSS\_PHY\_125[25-16] PHY\_RDDQS\_DQ4\_RISE\_SLAVE\_DELAY\_0
- DDRSS\_PHY\_126[25-16] PHY\_RDDQS\_DQ5\_RISE\_SLAVE\_DELAY\_0
- DDRSS\_PHY\_127[25-16] PHY\_RDDQS\_DQ6\_RISE\_SLAVE\_DELAY\_0
- DDRSS\_PHY\_128[25-16] PHY\_RDDQS\_DQ7\_RISE\_SLAVE\_DELAY\_0
- DDRSS\_PHY\_377[17-8] PHY\_RDDQS\_DQ0\_RISE\_SLAVE\_DELAY\_1
- DDRSS\_PHY\_378[25-16] PHY\_RDDQS\_DQ1\_RISE\_SLAVE\_DELAY\_1
- DDRSS\_PHY\_379[25-16] PHY\_RDDQS\_DQ2\_RISE\_SLAVE\_DELAY\_1
- DDRSS\_PHY\_380[25-16] PHY\_RDDQS\_DQ3\_RISE\_SLAVE\_DELAY\_1
- DDRSS\_PHY\_381[25-16] PHY\_RDDQS\_DQ4\_RISE\_SLAVE\_DELAY\_1
- DDRSS\_PHY\_382[25-16] PHY\_RDDQS\_DQ5\_RISE\_SLAVE\_DELAY\_1
- DDRSS\_PHY\_383[25-16] PHY\_RDDQS\_DQ6\_RISE\_SLAVE\_DELAY\_1
- DDRSS\_PHY\_384[25-16] PHY\_RDDQS\_DQ7\_RISE\_SLAVE\_DELAY\_1
- DDRSS\_PHY\_633[17-8] PHY\_RDDQS\_DQ0\_RISE\_SLAVE\_DELAY\_2
- DDRSS\_PHY\_634[25-16] PHY\_RDDQS\_DQ1\_RISE\_SLAVE\_DELAY\_2
- DDRSS\_PHY\_635[25-16] PHY\_RDDQS\_DQ2\_RISE\_SLAVE\_DELAY\_2
- DDRSS\_PHY\_636[25-16] PHY\_RDDQS\_DQ3\_RISE\_SLAVE\_DELAY\_2
- DDRSS\_PHY\_637[25-16] PHY\_RDDQS\_DQ4\_RISE\_SLAVE\_DELAY\_2
- DDRSS\_PHY\_638[25-16] PHY\_RDDQS\_DQ5\_RISE\_SLAVE\_DELAY\_2
- DDRSS\_PHY\_639[25-16] PHY\_RDDQS\_DQ6\_RISE\_SLAVE\_DELAY\_2
- DDRSS\_PHY\_640[25-16] PHY\_RDDQS\_DQ7\_RISE\_SLAVE\_DELAY\_2
- DDRSS\_PHY\_889[17-8] PHY\_RDDQS\_DQ0\_RISE\_SLAVE\_DELAY\_3
- DDRSS\_PHY\_890[25-16] PHY\_RDDQS\_DQ1\_RISE\_SLAVE\_DELAY\_3
- DDRSS\_PHY\_891[25-16] PHY\_RDDQS\_DQ2\_RISE\_SLAVE\_DELAY\_3
- DDRSS\_PHY\_892[25-16] PHY\_RDDQS\_DQ3\_RISE\_SLAVE\_DELAY\_3
- DDRSS\_PHY\_893[25-16] PHY\_RDDQS\_DQ4\_RISE\_SLAVE\_DELAY\_3
- DDRSS\_PHY\_894[25-16] PHY\_RDDQS\_DQ5\_RISE\_SLAVE\_DELAY\_3
- DDRSS\_PHY\_895[25-16] PHY\_RDDQS\_DQ6\_RISE\_SLAVE\_DELAY\_3
- DDRSS\_PHY\_896[25-16] PHY\_RDDQS\_DQ7\_RISE\_SLAVE\_DELAY\_3
- DDRSS\_PHY\_122[9-0] PHY\_RDDQS\_DQ0\_FALL\_SLAVE\_DELAY\_0
- DDRSS\_PHY\_123[9-0] PHY\_RDDQS\_DQ1\_FALL\_SLAVE\_DELAY\_0
- DDRSS\_PHY\_124[9-0] PHY\_RDDQS\_DQ2\_FALL\_SLAVE\_DELAY\_0
- DDRSS\_PHY\_125[9-0] PHY\_RDDQS\_DQ3\_FALL\_SLAVE\_DELAY\_0
- DDRSS\_PHY\_126[9-0] PHY\_RDDQS\_DQ4\_FALL\_SLAVE\_DELAY\_0
- DDRSS\_PHY\_127[9-0] PHY\_RDDQS\_DQ5\_FALL\_SLAVE\_DELAY\_0
- DDRSS\_PHY\_128[9-0] PHY\_RDDQS\_DQ6\_FALL\_SLAVE\_DELAY\_0
- DDRSS\_PHY\_129[9-0] PHY\_RDDQS\_DQ7\_FALL\_SLAVE\_DELAY\_0
- DDRSS\_PHY\_378[9-0] PHY\_RDDQS\_DQ0\_FALL\_SLAVE\_DELAY\_1
- DDRSS\_PHY\_379[9-0] PHY\_RDDQS\_DQ1\_FALL\_SLAVE\_DELAY\_1
- DDRSS\_PHY\_380[9-0] PHY\_RDDQS\_DQ2\_FALL\_SLAVE\_DELAY\_1
- DDRSS\_PHY\_381[9-0] PHY\_RDDQS\_DQ3\_FALL\_SLAVE\_DELAY\_1
- DDRSS\_PHY\_382[9-0] PHY\_RDDQS\_DQ4\_FALL\_SLAVE\_DELAY\_1
- DDRSS\_PHY\_383[9-0] PHY\_RDDQS\_DQ5\_FALL\_SLAVE\_DELAY\_1
- DDRSS\_PHY\_384[9-0] PHY\_RDDQS\_DQ6\_FALL\_SLAVE\_DELAY\_1

- DDRSS\_PHY\_385[9-0] PHY\_RDDQS\_DQ7\_FALL\_SLAVE\_DELAY\_1
- DDRSS\_PHY\_634[9-0] PHY\_RDDQS\_DQ0\_FALL\_SLAVE\_DELAY\_2
- DDRSS\_PHY\_635[9-0] PHY\_RDDQS\_DQ1\_FALL\_SLAVE\_DELAY\_2
- DDRSS\_PHY\_636[9-0] PHY\_RDDQS\_DQ2\_FALL\_SLAVE\_DELAY\_2
- DDRSS\_PHY\_637[9-0] PHY\_RDDQS\_DQ3\_FALL\_SLAVE\_DELAY\_2
- DDRSS\_PHY\_638[9-0] PHY\_RDDQS\_DQ4\_FALL\_SLAVE\_DELAY\_2
- DDRSS\_PHY\_639[9-0] PHY\_RDDQS\_DQ5\_FALL\_SLAVE\_DELAY\_2
- DDRSS\_PHY\_640[9-0] PHY\_RDDQS\_DQ6\_FALL\_SLAVE\_DELAY\_2
- DDRSS\_PHY\_641[9-0] PHY\_RDDQS\_DQ7\_FALL\_SLAVE\_DELAY\_2
- DDRSS\_PHY\_890[9-0] PHY\_RDDQS\_DQ0\_FALL\_SLAVE\_DELAY\_3
- DDRSS\_PHY\_891[9-0] PHY\_RDDQS\_DQ1\_FALL\_SLAVE\_DELAY\_3
- DDRSS\_PHY\_892[9-0] PHY\_RDDQS\_DQ2\_FALL\_SLAVE\_DELAY\_3
- DDRSS\_PHY\_893[9-0] PHY\_RDDQS\_DQ3\_FALL\_SLAVE\_DELAY\_3
- DDRSS\_PHY\_894[9-0] PHY\_RDDQS\_DQ4\_FALL\_SLAVE\_DELAY\_3
- DDRSS\_PHY\_895[9-0] PHY\_RDDQS\_DQ5\_FALL\_SLAVE\_DELAY\_3
- DDRSS\_PHY\_896[9-0] PHY\_RDDQS\_DQ6\_FALL\_SLAVE\_DELAY\_3
- DDRSS\_PHY\_897[9-0] PHY\_RDDQS\_DQ7\_FALL\_SLAVE\_DELAY\_3
- DDRSS\_PHY\_129[25-16] PHY\_RDDQS\_DM\_RISE\_SLAVE\_DELAY\_0
- DDRSS\_PHY\_385[25-16] PHY\_RDDQS\_DM\_RISE\_SLAVE\_DELAY\_1
- DDRSS\_PHY\_641[25-16] PHY\_RDDQS\_DM\_RISE\_SLAVE\_DELAY\_2
- DDRSS\_PHY\_897[25-16] PHY\_RDDQS\_DM\_RISE\_SLAVE\_DELAY\_3
- DDRSS\_PHY\_130[9-0] PHY\_RDDQS\_DM\_FALL\_SLAVE\_DELAY\_0
- DDRSS\_PHY\_386[9-0] PHY\_RDDQS\_DM\_FALL\_SLAVE\_DELAY\_1
- DDRSS\_PHY\_642[9-0] PHY\_RDDQS\_DM\_FALL\_SLAVE\_DELAY\_2
- DDRSS\_PHY\_898[9-0] PHY\_RDDQS\_DM\_FALL\_SLAVE\_DELAY\_3

Read data eye training for LPDDR4 devices does return data on the DM pin so the DBI data is trained along with the other DQ pins. In this case, the PHY\_RDDQS\_DM\_RISE\_SLAVE\_DELAY\_x and PHY\_RDDQS\_DM\_FALL\_SLAVE\_DELAY\_x fields receive their own unique values.

After read data eye training is complete, no error condition is noted in the read leveling status fields. The RISE and FALL field results in the previously described registers can be checked to look for values that appear out of place (for example, near 0x000, near 0x100 or greater) or to look for a DQ that is much different than the rest. After read data eye training is complete, software can always override the results by writing directly to the PHY\_RDDQS\_DQy\_RISE\_SLAVE\_DELAY\_x, PHY\_RDDQS\_DQy\_FALL\_SLAVE\_DELAY\_x, PHY\_RDDQS\_DM\_RISE\_SLAVE\_DELAY\_x and PHY\_RDDQS\_DM\_FALL\_SLAVE\_DELAY\_x fields.

#### 8.2.4.7.9.4 Write DQ Training

The data that is written to the SDRAM during training is created within the data slice. The following data pattern options are available:

- Clock pattern - An alternating pattern of "1" and "0" is applied to each DQ/DM. The phase of the 1/0 pattern is identical on the DQ/DM pins.
- LFSR pattern - The same LFSR pattern that is used for data loopback can be applied to the DQ/DM pins. The LFSR is reset at the start of the pattern and each burst of data is different according to the progression of the LFSR.
- User defined pattern - A user defined pattern can also be applied to the DQ/DM pins. Independent data for each bit covering a full burst of 16 can be loaded into register fields to supply the training data. The same data pattern is repeated for every burst. The pattern is defined through the following fields:
  - DDRSS\_PHY\_36[31-0] PHY\_USER\_PATT0\_0
  - DDRSS\_PHY\_292[31-0] PHY\_USER\_PATT0\_1
  - DDRSS\_PHY\_548[31-0] PHY\_USER\_PATT0\_2
  - DDRSS\_PHY\_804[31-0] PHY\_USER\_PATT0\_3
  - DDRSS\_PHY\_37[31-0] PHY\_USER\_PATT1\_0

- DDRSS\_PHY\_293[31-0] PHY\_USER\_PATT1\_1
- DDRSS\_PHY\_549[31-0] PHY\_USER\_PATT1\_2
- DDRSS\_PHY\_805[31-0] PHY\_USER\_PATT1\_3
- DDRSS\_PHY\_38[31-0] PHY\_USER\_PATT2\_0
- DDRSS\_PHY\_294[31-0] PHY\_USER\_PATT2\_1
- DDRSS\_PHY\_550[31-0] PHY\_USER\_PATT2\_2
- DDRSS\_PHY\_806[31-0] PHY\_USER\_PATT2\_3
- DDRSS\_PHY\_39[31-0] PHY\_USER\_PATT3\_0
- DDRSS\_PHY\_295[31-0] PHY\_USER\_PATT3\_1
- DDRSS\_PHY\_551[31-0] PHY\_USER\_PATT3\_2
- DDRSS\_PHY\_807[31-0] PHY\_USER\_PATT3\_3
- DDRSS\_PHY\_40[15-0] PHY\_USER\_PATT4\_0
- DDRSS\_PHY\_296[15-0] PHY\_USER\_PATT4\_1
- DDRSS\_PHY\_552[15-0] PHY\_USER\_PATT4\_2
- DDRSS\_PHY\_808[15-0] PHY\_USER\_PATT4\_3

These patterns can be used in any combination and if multiple patterns are requested, the final results are based on the largest leading edge found and the smallest trailing edge found across the patterns selected.

#### 8.2.4.7.9.5 CA Training

CA bits can be swizzled between any bit positions via the DDRSS\_PHY\_1053[23-0] PHY\_ADR\_ADDR\_SEL\_0 field.

When the memory subsystem consists of more than one LPDDR device on the full bus width (two 16 bit devices on a 32 bit bus), all devices receive the same CA bits at some timing relative to each other depending on board placement and routing. In these cases, the DDR PHY can only supply a single CA training value for all devices to use together. Since each device may not receive ideal timing, the CA timing margin is reduced in the overall memory subsystem. Either a single one (any of them) or any combination of the devices participating in the training can be individually selected via the DDRSS\_PHY\_1357[20-16] PHY\_CALVL\_DEVICE\_MAP field.

In case of two channels, each one can be assigned to a specific chip select and data slices. The DDRSS\_PHY\_1293[31-24] PHY\_CALVL\_CS\_MAP field is used to map each CS rank to the appropriate DQ slice for training data return.

Bit 1 of the DDRSS\_PHY\_1038[25-24] PHY\_ADR\_CALVL\_RANK\_CTRL\_0 field enables current training results to be taken into account along with new training results, instead of discarding previous results. This can be used to train each rank or channel and construct an aggregate for all of them. Each rank or channel then receives the same CA training timing that is the best case possible for all of them. This may result in the setup/hold margin for the CA bits being narrower than any single rank or channel could achieve by itself.

#### Note

When multiple frequency sets are supported, then all ranks for a given frequency set must be trained before changing to a new frequency. Failure to do so results in rank aggregation not being performed completely with possible data corruption due to rank training being improper.

In case of multiple channels each rank of a channel must use the same CA swizzle. Each rank of a channel must also use the same DQ swizzle.

If the memory device DQ bits are not received in the same order at the PHY as the LPDDR device sent them, then the DQ data has been swizzled, and must be reordered back to its original form before CA training can interpret the results. The following fields specify the DQ swizzling in the corresponding data slice:

- DDRSS\_PHY\_114[31-0] PHY\_DQ\_DM\_SWIZZLE0\_0
- DDRSS\_PHY\_370[31-0] PHY\_DQ\_DM\_SWIZZLE0\_1
- DDRSS\_PHY\_626[31-0] PHY\_DQ\_DM\_SWIZZLE0\_2
- DDRSS\_PHY\_882[31-0] PHY\_DQ\_DM\_SWIZZLE0\_3



- DDRSS\_PHY\_115[3-0] PHY\_DQ\_DM\_SWIZZLE1\_0
- DDRSS\_PHY\_371[3-0] PHY\_DQ\_DM\_SWIZZLE1\_1
- DDRSS\_PHY\_627[3-0] PHY\_DQ\_DM\_SWIZZLE1\_2
- DDRSS\_PHY\_883[3-0] PHY\_DQ\_DM\_SWIZZLE1\_3

The PHY\_DQ\_DM\_SWIZZLE0\_x fields contain the bit position to map the received read DQ bits for proper comparison with the training pattern bits.

After CA training is complete the DDRSS\_PHY\_1043[31-0] PHY\_ADR\_CALVL\_OBS1\_0 field can be read to obtain training status information.

After CA training is complete, software can always override the results by writing to the following fields:

- DDRSS\_PHY\_1065[18-8] PHY\_ADR0\_CLK\_WR\_SLAVE\_DELAY\_0
- DDRSS\_PHY\_1066[10-0] PHY\_ADR1\_CLK\_WR\_SLAVE\_DELAY\_0
- DDRSS\_PHY\_1067[10-0] PHY\_ADR2\_CLK\_WR\_SLAVE\_DELAY\_0
- DDRSS\_PHY\_1068[10-0] PHY\_ADR3\_CLK\_WR\_SLAVE\_DELAY\_0
- DDRSS\_PHY\_1069[10-0] PHY\_ADR4\_CLK\_WR\_SLAVE\_DELAY\_0
- DDRSS\_PHY\_1070[10-0] PHY\_ADR5\_CLK\_WR\_SLAVE\_DELAY\_0

#### Note

The PHY\_ADRx\_CLK\_WR\_SLAVE\_DELAY\_0 fields can be programmed between 0x000 and 0x600. While training is permitted to set these fields below 0x0C0, the final result is always greater than 0x0C0. If software overrides training results, values less than 0x0C0 should never be programmed. The same restriction applies also to the PHY\_GRPy\_SLAVE\_DELAY\_x fields.

#### 8.2.4.7.9.6 CS Training

When the CS training algorithm has completed, the final CS slave delay settings can be found in the following fields:

- DDRSS\_PHY\_1399[10-0] PHY\_GRP0\_SLAVE\_DELAY\_0
- DDRSS\_PHY\_1399[26-16] PHY\_GRP1\_SLAVE\_DELAY\_0
- DDRSS\_PHY\_1400[10-0] PHY\_GRP2\_SLAVE\_DELAY\_0
- DDRSS\_PHY\_1400[26-16] PHY\_GRP3\_SLAVE\_DELAY\_0
- DDRSS\_PHY\_1401[10-0] PHY\_GRP0\_SLAVE\_DELAY\_1
- DDRSS\_PHY\_1402[10-0] PHY\_GRP1\_SLAVE\_DELAY\_1
- DDRSS\_PHY\_1403[10-0] PHY\_GRP2\_SLAVE\_DELAY\_1
- DDRSS\_PHY\_1404[10-0] PHY\_GRP3\_SLAVE\_DELAY\_1

After CS training is complete, the DDRSS\_PHY\_1289[31-0] PHY\_CSLVL\_OBS1 field can be read to obtain training status information.

After CS training is complete, software can always override the results by writing directly to the PHY\_GRPy\_SLAVE\_DELAY\_x fields.

#### 8.2.4.7.10 Data Bus Inversion (DBI)

Data bus inversion is supported in the DDR PHY. The PHY only passes the DBI information between the DDR controller and SDRAM devices. It does not perform DBI generation and decode. If the DDRSS\_PHY\_102[8] PHY\_DBI\_MODE\_0 bit is set to 0x1 the SDRAM DBI information for slice 0 is passed to the DDR controller. The following bits are associated with the other slices:

- DDRSS\_PHY\_358[8] PHY\_DBI\_MODE\_1
- DDRSS\_PHY\_614[8] PHY\_DBI\_MODE\_2
- DDRSS\_PHY\_870[8] PHY\_DBI\_MODE\_3

#### 8.2.4.7.11 I/O Pad Calibration

I/O pad calibration runs at initialization time by default but it can also be programmed to operate in a background mode after initialization. Bits [3:1] of the DDRSS\_PHY\_1330[12-0] PHY\_CAL\_MODE\_0 field

enable the periodic calibration mechanism and determine the base clock frequency of the interval counter. The DDRSS\_PHY\_1331[31-0] PHY\_CAL\_INTERVAL\_COUNT\_0 field then sets a timer to indicate how frequently calibration is performed. When the timer expires, the calibration runs in background, independent of other traffic running through the PHY. When the calibration is complete, the updated PVTP/PVTN/PVTR codes are transferred to the PHY I/O cells. Whether or not the update of the memory clock PVTP/PVTN/PVTR codes is applied at the same time as the remaining I/O cells is controlled through the DDRSS\_PHY\_1298[8] PHY\_CONTINUOUS\_CLK\_CAL\_UPDATE bit. If the automatic update of the memory clock codes is disabled through the PHY\_CONTINUOUS\_CLK\_CAL\_UPDATE bit, the DDRSS\_PHY\_1298[0] SC\_PHY\_UPDATE\_CLK\_CAL\_VALUES bit can be used to trigger a manual update.

#### 8.2.4.7.12 DQS Error

A DQS error results from a mismatch between the expected number of read DQS pulses and the actual number of read DQS pulses. Counters exist at DDR PHY level to track the number of DQS errors in each data slice. Counter information can be obtained by reading the following fields:

- DDRSS\_PHY\_1363[31-0] PHY\_DS0\_DQS\_ERR\_COUNTER
- DDRSS\_PHY\_1364[31-0] PHY\_DS1\_DQS\_ERR\_COUNTER
- DDRSS\_PHY\_1365[31-0] PHY\_DS2\_DQS\_ERR\_COUNTER
- DDRSS\_PHY\_1366[31-0] PHY\_DS3\_DQS\_ERR\_COUNTER

The DQS error signals from each data slice are accumulated at PHY level and can be used to generate a dfi\_error signal. Bit 2 of the DDRSS\_PHY\_1361[26-24] PHY\_ERR\_MASK\_EN field is used to mask the DQS error from generating a dfi\_error or being reported on dfi\_error\_info[2].

#### 8.2.4.8 PI Functional Description

The PHY Independent (PI) module is part of the DDR PHY and can perform PHY independent leveling without involvement of the DDR controller. The PI can also initialize the SDRAM independent of the DDR controller. The PI module provides an interface between the PHY core logic and the DDR controller.

##### 8.2.4.8.1 PI Initialization

The PI module can be used to perform SDRAM initialization before training instead of the DDR controller. The following steps represent the main actions of the initialization process:

1. Set to 0x1 the DDRSS\_PI\_139[8] PI\_DRAM\_INIT\_EN bit.
2. Enable the PI by setting to 0x1 the DDRSS\_PI\_0[0] PI\_START bit.
3. Enable the DDR controller.
4. The PI, DDR controller and PHY core are involved in a process of specific hardware communication between them that does not require software intervention.
5. The PI starts the power-up sequence which includes the SDRAM initialization timing, the SDRAM mode register programming and training.
6. After completion of the power-up sequence, the PI notifies the DDR controller through the dfi\_init\_complete signal.
7. The DDR controller can now access the SDRAM.

## 8.3 Virtualization Subsystem (VirtSS)

This chapter describes the features and functions of the Virtualization Subsystem (VirtSS) hardware modules in the device.



### 8.3.1 VirtSS Overview

The VirtSS is instantiated in the main NAVSS.

#### 8.3.1.1 VirtSS Features

The VirtSS provides all the translation components supported in K3 for virtualization as well as NAVSS connectivity between the AC, MODSS, UDMASS, and NB ports. The virtualization components include PATs, PVUs, and Arm® SMMUv3 components, TBU and TCU. The number of each component, as well as the configuration of each component, can be seen in [Section 8.3.1.3, VirtSS Configuration](#). VirtSS instances the components and any required bridges and provides the CBA busses, connected into the main NAVSS CBASS. All routing to the VirtSS is defined in the NAVSS CBASS.

There is one VirtSS in the device and it is part of the main Navigator Subsystem (NAVSS0).

The VirtSS supports the following functions and features:

- Implements all virtualization translation components in a subsystem
- Implements NAVSS connectivity between AC, MODSS, UDMASS, and NB ports
- Supports 5× PATs (Page-based Address Translator), see [Section 8.3.3](#)
- Supports 3× PVUs (Peripheral Virtualization Unit), see [Section 8.3.2](#)
- Supports a SMMU TBU (Arm SMMUv3 Translation look-aside Buffer Unit)
- Supports a SMMU TCU (Arm SMMUv3 Translation Controller Unit)
- Supports a SMMU DTI ( Arm® AMBA® Distributed Translation Interface) interconnect
- Supports external DTI connections
- Bridges all AXI or APB interfaces to CBA VBUSM and VBUSP busses.

For SMMU TBU and SMMU TCU functional description please refer to Arm documentation at <http://infocenter.arm.com>.

#### 8.3.1.2 Functional Description

The Arm® System Memory Management Unit (SMMU) provides facilities to translate addresses of transactions mastered by non-processor modules in the device. For embedded automotive applications, SMMU is not used to increase hardware utilization through datacenter-like virtualization, but for enhanced software reliability and safety by rigid partitioning of chip-level resources.

The block diagram of the VirtSS, Virtualization Subsystem, that implements SMMU is described in [Figure 8-11](#).

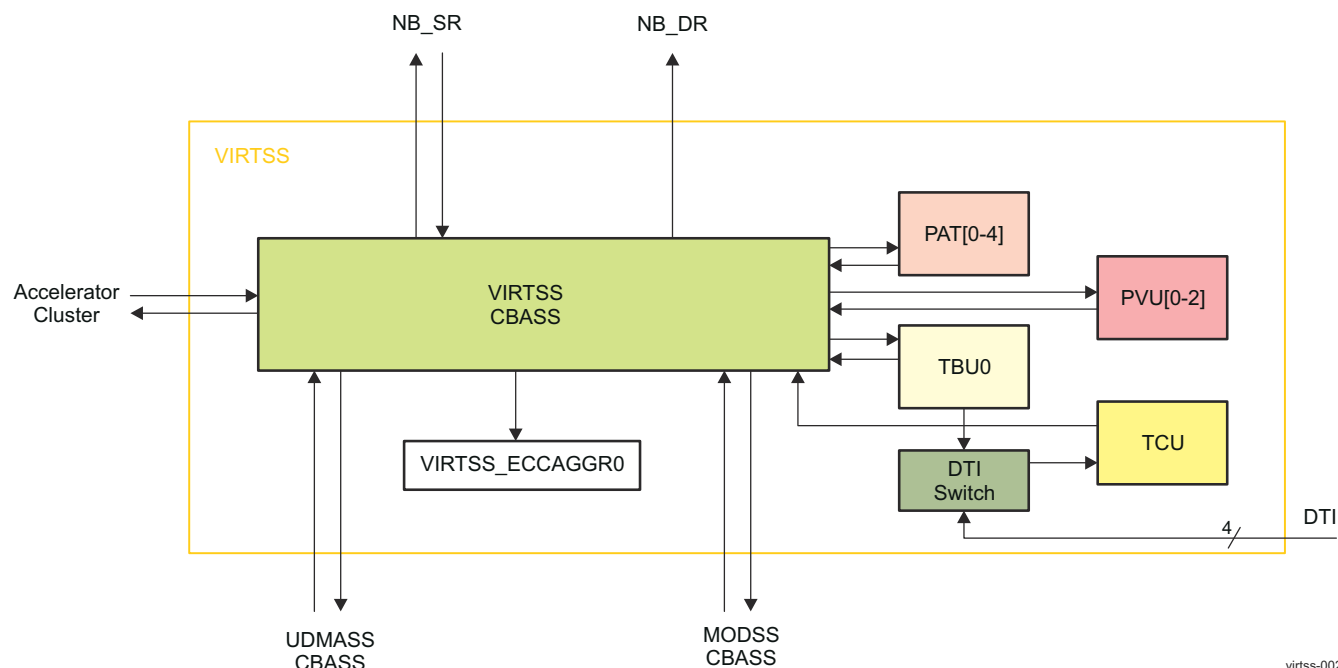


Figure 8-11. VirtSS Block Diagram

### 8.3.1.2.1 Ports

The VirtSS has 4 sets of ports, in and out, for the AC, MODSS, UDMASS, and NB. Each set includes a configurable number of possibly parallel ports where *orderid* defines the port to use for the transaction. If the port group matches a slave port group then those masters only connect to those slaves, otherwise all masters connect to the slaves without matching group names. The ports can carry PA, IPA, or VA addresses and route to the slave ports or the virtualization components. The NAVSS must define these as parallel ports using the same *orderid* mappings. It should also map the PAT physical addresses to these ports. These ports support the following table of accesses - [Table 8-23](#).

Table 8-23. Port Table of Accesses

Atype	Path
PA	Access to slave port
PA	Access to PAT
IPA	Access to PAT (must hit PAT range), then to PVU
IPA	Access to PVU
VA	Access to SMMU

### 8.3.1.2.2 CBASS

The VirtSS uses a CBASS to support decoding transactions to the slaves and PATs, which are address based, or to the PVUs which is *atype* and *orderid* based, or to the TBUs which are *atype* and *orderid* based. To support this all input transactions of IPA *atype* that hit the PAT address range are converted to PA so that the CBASS can decode the address to a PAT region, and then the *atype* is restored going into the PAT. Any other IPA transaction will go to a PVU. And any VA transaction will go to a TBU. PAT, PVU, and TBU output transactions go back to this CBASS so that they can be decoded again. This allows an IPA transaction to go first to a PAT and then to a PVU. The PVU and TBU output transactions will target the slave based on the resulting translated address. DMA transactions that are just to PVU can use private DMA PVUs if configured. DMA transactions that access the PAT will transition like they were non-DMA transactions and use the IO PVU components.

### 8.3.1.2.3 PAT

The VirtSS contains PATs, which are page based address translation units. The PAT is address based, so only translations addressing the PAT will reach the PAT input bus. The PAT is then programmed for the page size and page translations. The PAT will then map the input address to a page and translate the address of the output transaction to the programmed physical address. The PAT output bus goes back to the VirtSS CBASS to be routed to the new target.

#### 8.3.1.2.3.1 Bandwidth Splitting

If a single PAT cannot support the required bandwidth, then multiple PAT instances must be used. Since the PAT is physically addressed, any bandwidth splitting must be done by address. That means the peripherals accessing the PAT must use the correct address mapping so that they are routed to the correct PAT. There may be some gaps in the PAT address map if the page sizes of the PATs are not programmed to the largest size, since the fixed SoC address map must support the largest PAT sizes.

### 8.3.1.2.4 PVU

The VirtSS contains PVUs, which are peripheral virtualization units. The PVU is IPA, intermediate physically addressed, so only translations using that address type will reach the PVU input bus. The PVU is then programmed for each *virtid* table, per VM, and all the translations, which are similar to LPAE pages. The PVU will then map the input address to a table and page and translate the address of the output transaction to the programmed physical address. The PVU output bus goes right back to the VirtSS CBASS to be routed to the new target.

#### 8.3.1.2.4.1 Bandwidth Splitting

If a single PVU cannot support the required bandwidth, then multiple PVU instances must be used. Since the PVU does not have any address decode, using an IPA, then any splitting is done based on the bus *orderid* value. That means the peripherals accessing the PVU must be set to use the correct *orderid* so that they are routed to the correct PVU.

### 8.3.1.2.5 TBU

The VirtSS contains SMMU TBUs, which are TLBs, translation look aside buffer units. The TBU is virtually addressed, so only translations using that address type will reach the TBU input bus. The TBU normally uses the *virtid* CBA signal as the AXI *streamid* signal to identify the VM for the translation. There is a bridge from the VBUSM bus to the AXI bus of the TBU. The TBU will perform the virtual address lookup and on a hit, or on a miss it will request the translation from the TCU, described later, via the DTI bus, also described later. The TBU will then translate the address of the output transaction based on the SMMU programmed page tables to the programmed physical address. The TBU output bus goes through another bridge from AXI to VBUSM and then goes back to the NAVSS CBASS to be routed to the target.

#### 8.3.1.2.5.1 Bandwidth Splitting

If a single TBU cannot support the required bandwidth, then multiple TBU instances must be used. Since the TBU does not have any address decode, using a VA, then any splitting is done based on the bus *orderid* value. That means the peripherals accessing the TBU must be set to use the correct *orderid* so that they are routed to the correct TBU.

#### 8.3.1.2.5.2 Initialization Delay Requirement

In order to initialize the TBL RAMs for ECC codes, there must be no transactions to the TBU for at least 32 TBU clock cycles after reset while the RAM is busy. During this time the software must not enable any IOs for SMMU usage.

### 8.3.1.2.6 TCU

The VirtSS contains an SMMU TCU, translation controller unit. TCU takes translation requests from all the TBUs via the DTI bus and walks the SMMU page tables in memory, if there are any TBUs enabled. The TCU has an output AXI bus to access memory, and this is bridged to VBUSM which then goes to the NAVSS CBASS to access the memory. The TCU is programmed to map the VM to SMMU page tables in memory, as well as

other features, via a VBUSP port bridge to APB. Due to the bridge not supporting byte enables, all TCU register access must be full 32-bit words, and it does not support individual byte writes.

### 8.3.1.2.7 DTI Interconnect

The VirtSS contains an SMMU DTI, distributed translation interface, to interconnect the TBU requests to the TCU. The interconnect simply switches all the TBU DTI busses to the single TCU bus and delivers the TCU responses. These busses are then connected to TBU and TCU.

### 8.3.1.2.8 External DTI Ports

The VirtSS contains external SMMU DTI ports to connect external subsystems that have DTI support, such as PCIe. The number of ports can be seen in [Table 8-26](#), *VirtSS Global Parameters*.

### 8.3.1.2.9 DMA Split

The VirtSS supports reserving some of the components for DMA only usage, to keep the DMA and peripheral usage from interfering with each other. Only the DMA ports will be able to access the DMA private PVUs or TBUs. All the TBUs still connect inside the VirtSS to the single TCU via the DTI interconnect.

### 8.3.1.2.10 Port Routing Rules

Since the ports are all split based on *orderid*, there are normally fixed connections to components as well as the final output port that a transaction will follow. [Table 8-24](#) details the rules that are applied to a transaction from the various input ports, where X is the port matching the transaction *orderid*. The only less consistent path is when the DMA source accesses the PAT, and then the output route becomes the IO path (due to problems routing from PAT to either IO or DMA PVUs), or when the VirtSS is configured for less DMA PVUs and the transactions will route to the DMA PVU if it exists for that *orderid* and if not then the IO PVU for that *orderid*. Note that the PAT and PVU contain registers that can modify the *orderid* of a transaction, and if these features are enabled, then the output port for those transactions will be modified as well, so these rules would no longer apply.

**Table 8-24. Port Routing Rules**

Input Port	Components
<non-dma>_mstX	PAT and/or IO PVU
dma_mstX	IO PVU only (if not private DMA PVU for that orderid)
dma_mstX	DMA PVU only
dma_mstX	PAT and IO PVU
<non-dma>_mstX	IO TBU
dma_mstX	DMA TBU

There are also paths that are explicitly blocked, either because they could cause deadlock or because the path is not possible. The impossible paths are due to the PVU and SMMU translations always resulting in physical addresses, which cannot target a PVU or SMMU since those only receive intermediate or virtual transactions. [Table 8-25](#) shows all the blocked paths, and any transactions that attempt to target these paths inside the VirtSS will return errors.

**Table 8-25. Blocked Paths**

Source Port or Component	Target Port or Component that is Blocked
PAT output	PAT input (could deadlock)
PVU output	PVU or SMMU input (impossible)
SMMU output	PVU or SMMU input (impossible)

If the system is using PAT to PVU paths, then the PVU tables are not allowed to map to physical addresses which map to a PAT. This is because it can cause a loop for transactions going to PAT to PVU back to PAT, which

could cause a deadlock. Software must avoid this condition by not allowing the PVU to map any translations back to a PAT range.

### 8.3.1.3 VirtSS Configuration

This section describes the hardware configuration of VirtSS.

[Table 8-26](#) shows the global VirtSS settings that are enabled.

**Table 8-26. VirtSS Global Parameters**

Parameter	Setting	Description
data_width	256	Data bus width in bits.
fault_addr	0x0000310cf000	Fault address location to send faulting transactions, so that they route to a known location to return errors.
pat_small_start	0x004800000000	Base address for the first small PAT.
pat_large_start	0x004900000000	Base address for the first large PAT.
num_ext_dtis	4	Number of external DTI ports.
tbu_burst_size	256	Max burst size for TBU input transactions for AXI bridge sizing.
tbu_id_width	4	Width in bits for TBU AXI ID pins based on toggling orderid range for AXI bridge sizing.
tbu_buffers	enabled	TBU Buffers for transaction offload are enabled.
safe	enabled	Safety is enabled.

#### 8.3.1.3.1 PAT Parameters

[Table 8-27](#) shows the PAT instance configuration.

**Table 8-27. PAT Parameters**

PAT Instance	Number of Pages	Large	Source ID	Config FW	RouteID
NAVSS0_PAT0	16384	0	0	0x0000:6:3:4736	224
NAVSS0_PAT1	16384	0	0	0x0400:6:3:4737	225
NAVSS0_PAT2	16384	0	0	0x0800:6:3:4738	226
NAVSS0_PAT3	2048	1	1	0x0c00:6:3:4739	227
NAVSS0_PAT4	2048	1	1	0x1000:6:3:4740	228

#### 8.3.1.3.2 PVU Parameters

[Table 8-28](#) shows the PVU instance configuration.

**Table 8-28. PVU Parameters**

PVU Instance	TLB Channels/ VirtIDs	Entries per TLB Channel	Source ID	Config FW	RouteID	OrderID range
NAVSS0_IO_PV U0	64	8	16	0x2000::3:4744	208	0-7
NAVSS0_IO_PV U1	64	8	17	0x2400::3:4745	209	8-15
NAVSS0_DMA_ PVU1	64	8	19	0x2c00::3:4747	211	8-15

#### 8.3.1.3.3 TBU Parameters

[Table 8-29](#) shows the TBU instance configuration.

**Table 8-29. TBU Parameters**

TBU Instance	Write Buffer Depth	Latency FIFO Depth	Translate Slots	Read Outstanding Transaction Depth	Write Outstanding Transaction Depth	Micro TLB Depth	Main TLB Depth	RouteIDs	OrderID range
IO_TBU0	64	4	4	64	64	12	128	232-233	0-15

**8.3.1.3.4 TCU Parameters**

[Table 8-30](#) shows the TCU configuration.

**Table 8-30. TCU Parameters**

Walk Cache Stage 1 Depth (L0, L1, L2, L3)	Walk Cache Stage 2 Depth (L0, L1, L2, L3)	Configuration Cache Depth	Translation Slots	Page Table Walk Slots	Cache Table Walk Slots	Config FW	RouteIDs
128, 128, 1024, 1024	128, 128, 1024, 1024	16	32	32	8	0x4000::3:4752	248-249

**8.3.1.3.5 ECC Aggregator Parameters**

[Table 8-31](#) shows the ECC Aggregator configuration.

**Table 8-31. ECC Aggregator Parameters**

Config Base	Config FW
0x000031002000	0x4400::3:4753

### 8.3.1.4 Theory of Operation

For SOC components outside the NAVSS and Compute Cluster (CC), the device provides a pool of address mapping resources composed of Arm SMMUv3 TBU instances, TI PVU instances and TI PAT instances. At system configuration time, software programs the SOC infrastructure (including SOC CBASS and NAVSS CBASS) such that I/O modules and hardware accelerators are associated with specific address mapping resources. I/O modules may also be configured to master transactions directly through NAVSS CBASS with trusted physical addresses by setting the transaction's CBA ATYPE bits either in the specific module's configuration or with the SOC CBASS port logic.

UDMA has its own dedicated set of four PVU address mapping resource modules that are in series with the connections from UDMA.

Components in the compute cluster that master transactions implement the Armv8 two-level translation scheme or equivalent functionality using PAT and PVU modules.

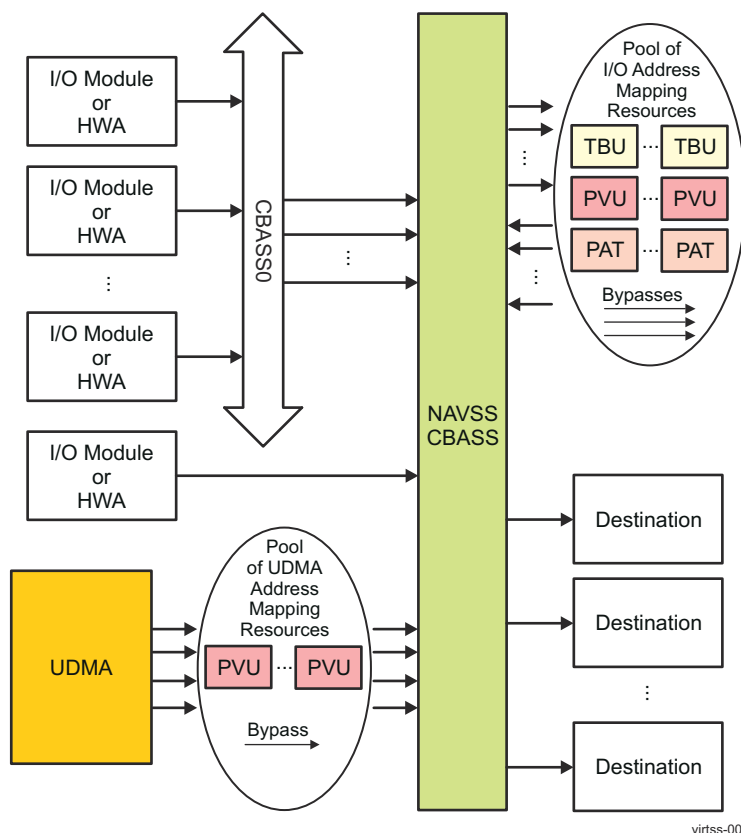
The options for transactions mastered by I/O, HWA or UDMA are summarized in [Table 8-32](#).

**Table 8-32. Transaction Options**

Source	Hop 1	Hop 2	Hop 3	Hop 4	Hop 5	Hop 6	Hop 7	Notes
I/O device	SOC switch	CBASS	Destination					Trusted physical address
I/O device	SOC switch	CBASS	TBU	CBASS	Destination			Non-realtime transaction
I/O device	SOC switch	CBASS	PAT	CBASS	Destination			Real-time, trusted transaction
I/O device	SOC switch	CBASS	PVU	CBASS	Destination			Real-time, trusted transaction
I/O device	SOC switch	CBASS	PAT	CBASS	PVU	CBASS	Destination	Real-time, trusted transaction
HWA	CBASS	Destination						Trusted physical address
HWA	CBASS	TBU	CBASS	Destination				Non-realtime transaction
HWA	CBASS	PAT	CBASS	Destination				Real-time, trusted transaction
HWA	CBASS	PVU	CBASS	Destination				Real-time, trusted transaction
HWA	CBASS	PAT	CBASS	PVU	CBASS	Destination		Real-time, trusted transaction
UDMA	CBASS	Destination						Trusted physical address
UDMA	CBASS	TBU	CBASS	Destination				Non-realtime transaction
UDMA	PVU	CBASS	Destination					Real-time, trusted transaction
UDMA	CBASS <sup>(1)</sup>	PAT	CBASS	Destination				Real-time, trusted transaction
UDMA	CBASS <sup>(1)</sup>	PAT	CBASS	PVU	CBASS	Destination		Real-time, trusted transaction

- (1) One connection into CBASS is provided for UDMA connections to bypass PVU translation and go through CBASS to a PAT instance. This path is a configuration-time option and only supports 25% of the UDMA to CBASS bandwidth.

[Figure 8-12](#) shows an overview of VirtSS operation.



**Figure 8-12. Functional Diagram**

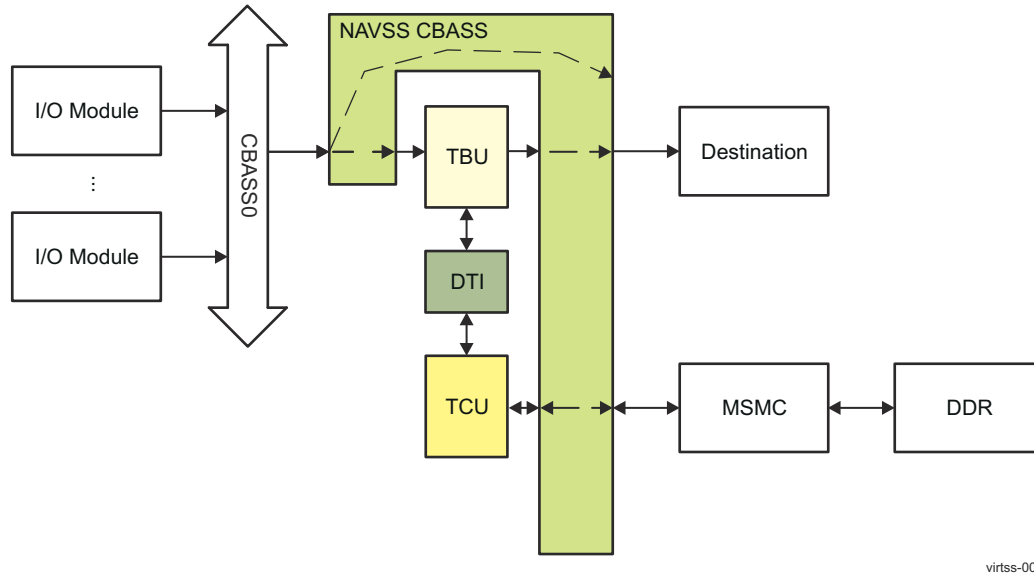
Aggregated requests from I/O or HWA modules are mapped directly to NAVSS CBASS ports. When using I/O address translation, normal switching in NAVSS CBASS passes transactions to dedicated CBASS ports connected to the address mapping resource ports. The logic on the path through each TBU, PVU or PAT module reads transactions, performs translations, optionally modifies transaction side-band information, and sends translated transactions back into the interconnect to be switched to the transaction's next destination.

The address mapping resource modules have a moderate number of cached or stored translations that can service translations with low latency. All of the TBU instances connect to a single TCU through the DTI protocol. When a TBU lookup miss occurs, a request is sent from the TBU using the DTI protocol over an AXI switch interconnect to the TCU. The requested translation may be cached in the TCU. If the translation is not present, table walking state machines load the translation, typically in multiple memory accesses, from tables via additional transactions through CBASS and MSMC with DDR memory.

Exception events can occur when PAT or PVU lookups are requested but there is not a matching memory range.

The architecture and microarchitecture of the TBU and TCU are specified by Arm documentation [<http://infocenter.arm.com>].





**Figure 8-13. Functional Example**

The address mapping resource modules have limited command bandwidth. The peak command rate of the TBU, PVU and PAT is one command per clock cycle, at the CBASS clock rate. There is some level of hit-under-miss support in the TBU allow commands to proceed through the TBU while misses are being serviced by the TCU. Depending on the TBU access pattern, the command bandwidth may be much less than the peak rate.

Software assignment of I/O modules or HWAs to address translation resource requires some estimation of the bandwidth or command rate of the specific I/O modules in the system, the characteristics of the address sequence per virtual machine (which may include multiple channels), the page sizes used for the address translation and the number of VMs expected to use the I/O modules. As a crude rule-of-thumb, a specific TBU should support near peak rate if  $3 \times \sum_i (VM_i \times Channels_i)$  for the virtual machines and each virtual machine's channels is less than 128.

Latencies grow roughly by orders of magnitude as translations proceed from the TBU through TCU. Accesses that hit the TBU uTLB or miss the TBU uTLB and hit the TBU MTLB have a throughput of one access per cycle. There is a < 100 cycle penalty on TBU uTLB miss that misses in TBU MTLB and hits in TCU. Subsequent accesses may not observe this penalty because of hit under miss processing. There is a < 1000 cycle penalty on TBU uTLB miss that misses in TBU MTLB and misses in TCU MTLB and require table walk. Hit-under-miss behavior is not beneficial in this case.

**Table 8-33. Address Translation Resource Summary**

Mapping Resource	Peak bandwidth supported (assuming 256b interconnect at 500MHz) over all instances
1 TBU for I/O, UDMA and HWA modules	16 GB/s
4 PAT (parallel small and large size instances) for I/O and HWA modules	64 GB/s
4 PVU for I/O and HWA modules	64 GB/s
4 PVU for UDMA	64 GB/s

This architecture depends on proper configuration to reach performance targets. The software configuration of the SOC and NAVSS components that control the mapping of I/O devices and hardware accelerators to address mapping resources must minimize contention for the limited TBU entries. Failure to load balance the TBU instances has several undesirable consequences. First, the latency of the initial hop through NAVSS CBASS to the TBU is modulated by the throughput of the TBU pair. Frequent TBU misses that require TCU lookups reduce the TBU throughput and can increase the congestion in CBASS. Second, long term severe congestion

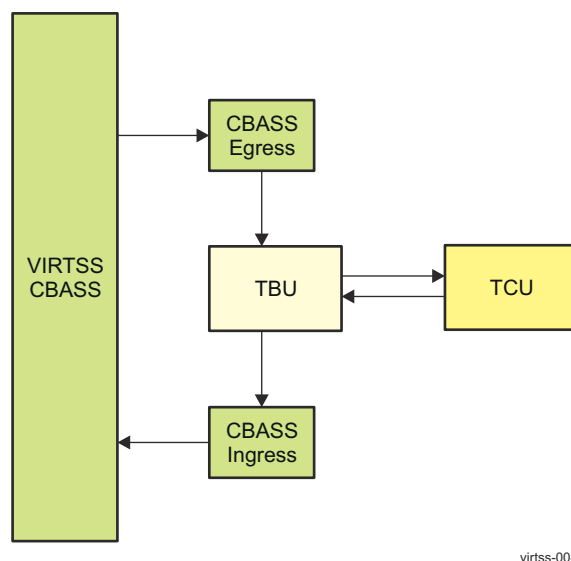
in links results in head-of-line (HOL) blocking for the other I/O devices or hardware accelerators that have been aggregated onto the same CBASS input port. This results in undesirable performance interference between I/O devices or hardware accelerators.

In addition to performance considerations, this architecture depends on proper configuration to avoid deadlocks. If all the available hit-under-miss capacity in the TBU and TCU are exhausted and there is a TBU miss that must be handled by the TCU and the TCU must access memory, that path to memory must not be obstructed by the transaction that is waiting on the TCU response to the TBU. If this obstruction occurs then there will be no forward progress on that TBU transaction and a deadlock will occur. For deadlock-free operation, the TCU path through CBASS to MSMC and DDR cannot share resources with any path from and I/O device or hardware accelerator to a TBU instance.

#### 8.3.1.4.1 TBU Address Translation Module

The TBU module reads transactions from CBASS, performs permission checks and address translations from virtual addresses then returns translated physical address transactions to CBASS. The architectural 2-stage translation and intermediate physical addresses (IPA) and the are not visible or exported to the rest of the system.

TBU is segregated from PAT and PVU paths to avoid short-term interference due to TBU blocking conditions on real-time traffic.

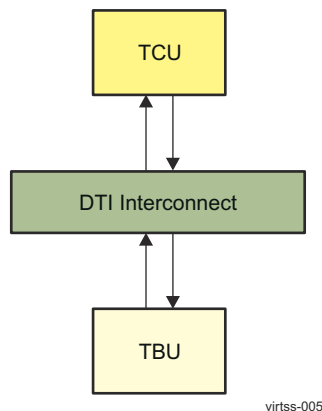


**Figure 8-14. TBU Functional Diagram**

On the path through the TBU, egress logic from CBASS converts the transactions to the format expected by the TBU. Internally, TBU provides payload-wide buffering and arbitration to support reordering as a result of TBU misses. Additionally, per Arm documentation, the TBU contains a bypass path that copies the input transaction to the output without translation or permission checking. Once the TBU has completed operating on a transaction, CBASS ingress logic converts the TBU output to format expected by CBASS and the transaction is transported to its final destination.

#### 8.3.1.4.2 DTI

Arm has defined the DTI (Distributed Translation Interface) message protocol to convey translation requests and responses between the TBUs and TCU. The protocol was defined to allow for a variety of transport mechanisms. The device uses AXI for DTI transport. Standard AXI bus components multiplex the TBU DTI requests onto a single command bus that the TCU reads. To maintain single-cycle TCU throughput, an AXI read channel returns DTI responses to the TBUs.



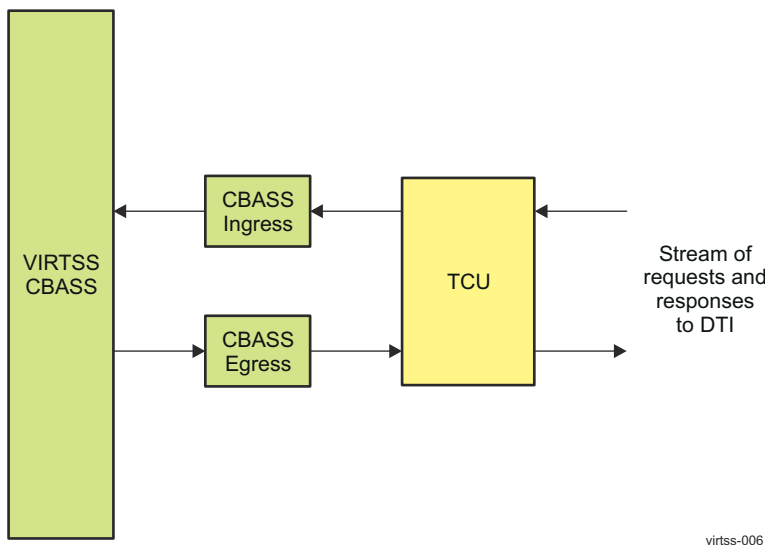
**Figure 8-15. DTI Functional Diagram**

#### 8.3.1.4.3 TCU

The Translation Control Unit (TCU) reads a stream of DTI-protocol translation requests from the distributed TBUs and returns translation information either from the TCU's local cache or from tables in memory. The TCU is wrapped in a module that includes the CBASS transaction push and pull interfaces. TCU operations that require table walking use these interfaces to access DDR memory where the in-memory page tables are stored.

Additionally, TCU handles Address Translation Service (ATS) for PCIe and the equivalent function for USB.

The K3 architecture does not support hardware coherence between any cached page table information in the TCU and operations on memory from the compute cluster. If an agent in the compute cluster modifies page tables, some mechanism must be used to initiate TCU invalidation. TCU invalidations automatically handle the subsequent TBU invalidations. This behavior is consistent with the Arm SMMU system architecture definition.

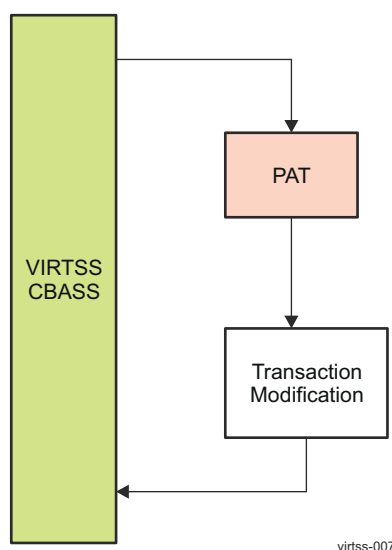


**Figure 8-16. TCU Functional Diagram**

#### 8.3.1.4.4

#### 8.3.1.4.5 PAT Address Translation Module

The PAT module reads transactions from CBASS, performs address translations from physical address or intermediate physical addresses using address lookup, then returns the same type (physical or intermediate physical) address transactions to CBASS. Details on PAT operation can be found in [Section 8.3.3, Page Based Address Translation Unit \(PAT\)](#).



**Figure 8-17. PAT Functional Diagram**

After address translation, the transaction passes through a transaction modification block that can change sideband information such as OrderID to divert the transaction to other modules for further translation (typically a PVU module). This transaction modification may use K3-specific data stored as page attributes in the PAT. Transaction modification can be globally disabled for all PAT instances with a single register write.

#### **8.3.1.4.5.1 Run-Time PAT Configuration**

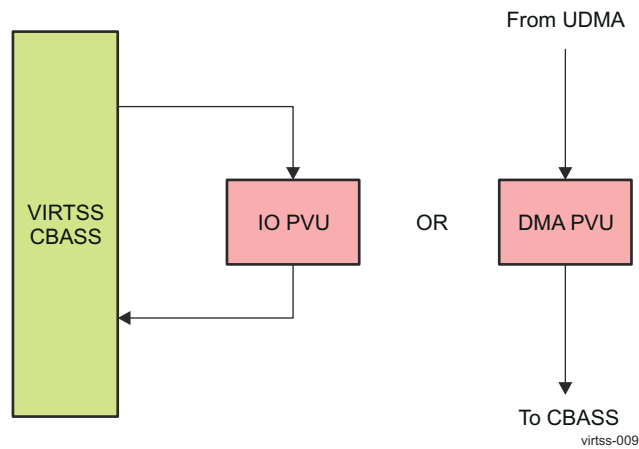
The registers in PAT that define the address spaces are mapped into the physical address space in contiguous regions. Software, probably a hypervisor, can use DMA to accelerate updates to PAT. However, these DMA writes are not indivisible. An address transaction to a region mapped by an incoming or over-written address range may result in a corrupted translation. The hypervisor must configure PAT before virtual machines using those mapping are allowed to perform memory translation through a PAT instance.

A subset of a PAT instance's registers are used for control and configuration. Each PAT instance has an independent copy of these registers. Except for the transaction modification control described above, software that wants to globally apply some configuration or operating mode to all PAT instances must iterate through all the PAT instances.

There is no special handshake mechanism in PAT can be used determine when the page updates through DMA are complete. The hypervisor must use the normal DMA completion indication to determine when the entries are updated and it is safe to allow traffic through the PAT using a new configuration.

#### **8.3.1.4.6 PVU Address Translation Module**

The PVU module reads transactions from CBASS or UDMA, and performs address translations on partitions of memory regions indexed by the transaction's VirtID. PVU checks page permissions, completes the translation and returns the translated transaction to CBASS for switching to the destination. Details of PVU operation can be found in [Section 8.3.2, Peripheral Virtualization Unit \(PVU\)](#).



**Figure 8-18. PVU Functional Diagram**

The PVU has a fixed number of contexts and pages per context. There are 64 contexts in each PVU instance, each containing 8 variably-sized pages.

The PVU does not implement table walking. An address sent to PVU that cannot be associated with a page is an error condition.

## 8.3.2 Peripheral Virtualization Unit (PVU)

This chapter describes the PVU module, part of the main NAVSS.

### 8.3.2.1 PVU Overview

The Peripheral Virtualization Unit (PVU) module provides TLBs (translation look-aside buffers) for a static virtual address translation on a CBA VBUSM bus.

Figure 8-19 shows a top-level overview of the PVU module.

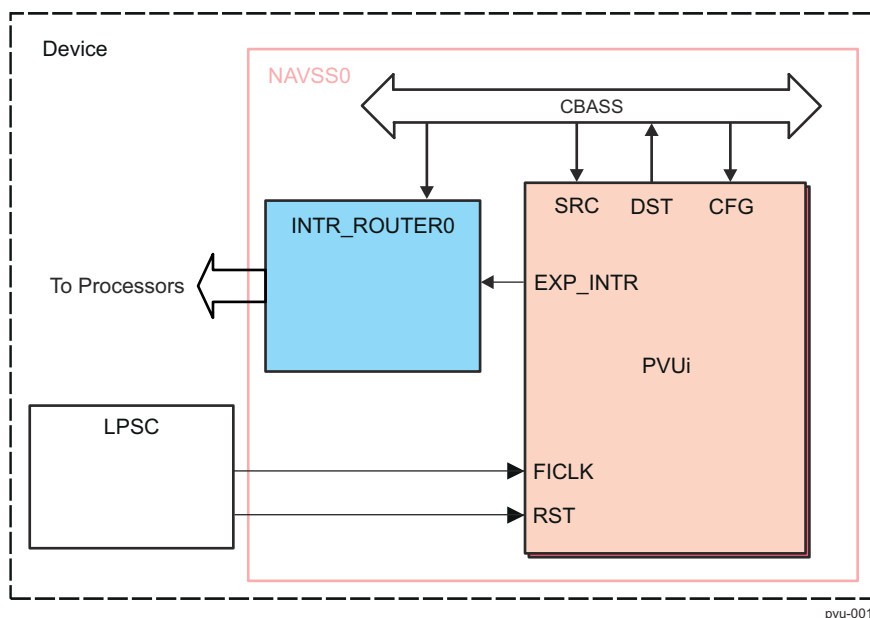


Figure 8-19. PVU Overview

#### 8.3.2.1.1 PVU Features

Each PVU supports the following features:

- Implements a channelized TLB for virtual address translation
- Supports a pre-configured number of virtIDs (channels). See [Section 8.3.2.1.2](#).
- Supports mapping of virtIDs to TLBs with a programmable number of DMA channel support
- Supports a pre-configured number of entries per channel of the TLB. See [Section 8.3.2.1.2](#)
- Supports TLB chaining to extend the search but at a latency penalty
- Supports a 48-bit address size
- Supports page sizes of 4 KB, 16 KB, 64 KB, 2 MB, 32 MB, 512 MB, 1 GB, and 16 GB
- Support TLBs in software mode, where they are maintained by software only
- Produces a fault interrupt when the TLB misses or upon a permission error.

#### 8.3.2.1.2 PVU Parameters

Table 8-34 shows the PVU configuration parameters set during SoC design.

Table 8-34. PVU Configuration Parameters

Module Instance	Parameters					
	TLB Channels <sup>(1)</sup>	Entries per TLB Channel <sup>(1)</sup>	Source ID	Config FW	RouteID	OrderID range <sup>(2)</sup>
NAVSS0_IO_PVU0	64	8	16	0x2000::3:4744	208	0-7 (IO and DMA)
NAVSS0_IO_PVU1	64	8	17	0x2400::3:4745	209	8-15 (IO)

**Table 8-34. PVU Configuration Parameters (continued)**

NAVSS0_DMA_PVU1	64	8	19	0x2c00::3:4747	211	8-15 (DMA)
-----------------	----	---	----	----------------	-----	------------

- (1) Can be read off from the PVU\_CONFIG read-only register.
- (2) Depending on the value of orderID of the transaction, and type of master, it will be routed to one of the PVU instances. Here, DMA indicates any transaction from NAVSS UDMA and IO indicates any other master excluding UDMA

#### 8.3.2.1.3 PVU Not Supported Features

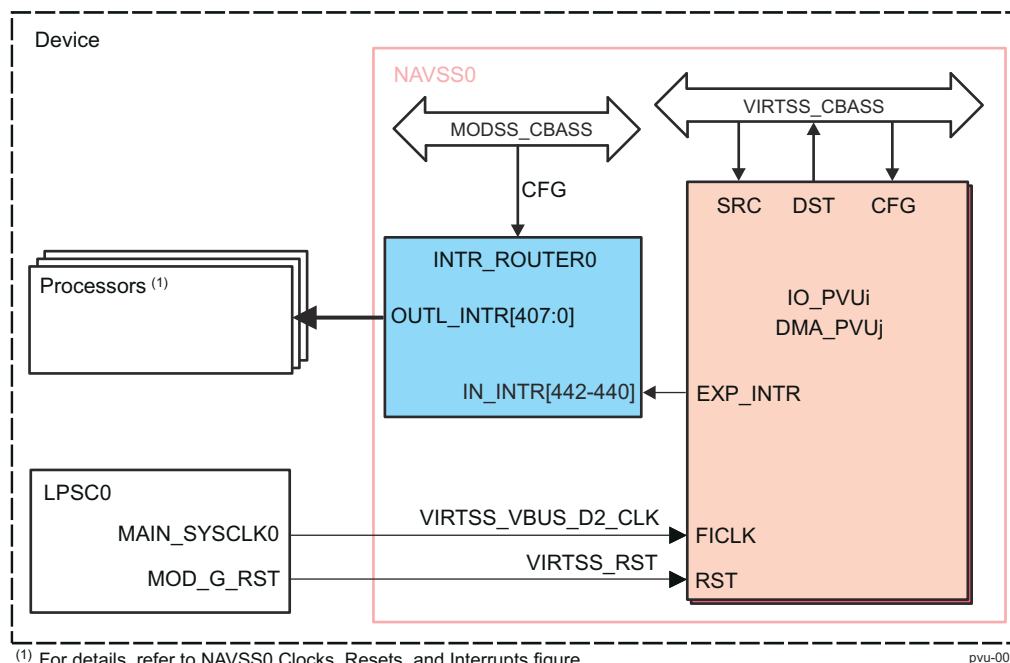
The following features are not supported by the module:

- Does not generate bus error responses
- Does not support transactions that cross a 4-KB boundary
- Does not support back fill from any MMU, hardware, or software.

### 8.3.2.2 PVU Integration

This section describes module integration in the device, including information about clocks, resets, and hardware requests.

Figure 8-20 shows the PVU integration in the device.



**Figure 8-20. PVU Integration**

i = 0 or 1

j = 1

Table 8-35 and Table 8-36 summarize the integration of the module in the device.

**Table 8-35. PVU Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
NAVSS0_IO_PVU0	PSC0	GP	LPSC0	VIRTSS_CBASS
NAVSS0_IO_PVU1	PSC0	GP	LPSC0	VIRTSS_CBASS
NAVSS0_DMA_PVU1	PSC0	GP	LPSC0	VIRTSS_CBASS

**Table 8-36. PVU Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
NAVSS0_IO_PVU0	FICLK	VIRTSS_VBUS_D2_CLK	MAIN_SYSCLK0	This clock is used for all interface and functional operations.
NAVSS0_IO_PVU1	FICLK	VIRTSS_VBUS_D2_CLK	MAIN_SYSCLK0	This clock is used for all interface and functional operations.
NAVSS0_DMA_PVU1	FICLK	VIRTSS_VBUS_D2_CLK	MAIN_SYSCLK0	This clock is used for all interface and functional operations.
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
NAVSS0_IO_PVU0	RST	VIRTSS_RST	LPSC0	PVU hardware reset
NAVSS0_IO_PVU1	RST	VIRTSS_RST	LPSC0	PVU hardware reset



**Table 8-36. PVU Clocks and Resets (continued)**

NAVSS0_DMA_PV U1	RST	VIRTSS_RST	LPSC0	PVU hardware reset
---------------------	-----	------------	-------	--------------------

**Table 8-37. PVU Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
NAVSS0_IO_PVU0	EXP_INTR	IN_INTR[442]	INTR_ROUTER0	Fault interrupt when the TLB misses or a permission error was found	Level
NAVSS0_IO_PVU1	EXP_INTR	IN_INTR[441]	INTR_ROUTER0	Fault interrupt when the TLB misses or a permission error was found	Level
NAVSS0_DMA_PV U1	EXP_INTR	IN_INTR[440]	INTR_ROUTER0	Fault interrupt when the TLB misses or a permission error was found	Level
DMA Events					
Module Instance	Module DMA Event	Destination DMA Event Input	Destination	Description	Type
NAVSS0_IO_PVU0	-	-	-	No PDMA channels to external DMA engines	-
NAVSS0_IO_PVU1	-	-	-	No PDMA channels to external DMA engines	-
NAVSS0_DMA_PV U1	-	-	-	No PDMA channels to external DMA engines	-

### 8.3.2.3 PVU Functional Description

#### 8.3.2.3.1 Functional Operation Overview

The Peripheral Virtualization Unit, PVU, module provides TLBs, translation look-aside buffers, for a static virtual address translation on a CBA VBUSM bus. This is in contrast to a dynamic translation TLB that works with an MMU, such as the Arm® processor SMMU. The module acts as a VBUSM retiming bridge that also performs the address translation. Each of the TLBs has its own set of private entries. It is a static translation because all the entries must be programmed by software and there is no hardware to fetch new translation entries when a transaction does not match any of the provided entries.

The entire PVU can be enabled or disabled via the PVU\_ENABLE register.

#### 8.3.2.3.2 PVU Channels

The module supports a number of channels (see [Section 8.3.2.1.2](#)), each having its own independent TLB and translation entries. This allows a single PVU to handle the translations for many masters, as well as keep their translations from interfering with each other.

The mapping of transactions to TLBs is made using mapping registers (PVU\_VIRTID\_MAP1 and PVU\_VIRTID\_MAP2). Each channel can be mapped to a virtID, or as a sub range of a DMA virtID. This allows the software to decide how the transactions are mapped to TLBs. It can be purely based on virtID, so that each virtual space has its own TLB. Or, it can be partly based on a highly channelized master, such as a DMA, can have 4 TLBs dedicated for channels where the sub-class on the upper chanid bits determine which of the 4 TLBs to use. This DMA mode allows for different classes of DMA traffic, such as descriptors or buffers, to be in their own TLBs and not interfere with each other. Since the TLB mapping is relatively simple, it is software's responsibility to control the definition of virtIDs in peripherals and DMAs so that they map to TLBs as required.

#### 8.3.2.3.3 TLB

Each PVU channel contains a single traditional TLB. The TLB contains a number of entries that store the translation recipes, and these entries are used only for this TLB. Any activity for other TLBs or PVU channels will not impact this TLB.

#### 8.3.2.3.4 TLB Entry

A TLB entry consists of the virtual address base and LPAE (Large Physical Address Extension) fields defined by Arm® architecture, stored in TLB Entry registers. This entry then includes the translated base address, size of the page, and other attributes of the page. When written by software, these fields must be written in the same way for correct operation.

The base addresses must be aligned to the defined page size.

#### 8.3.2.3.5 TLB Selection

The first step of the module is to map the transaction to a specific TLB by comparing the virtID, chanID upper bits, and those programmed in PVU\_VIRTID\_MAP1 and PVU\_VIRTID\_MAP2. This mapping determines which TLB to select from the virtID. That TLB is used for the remainder of the pipeline. If there is no match then the PVU automatically misses.

#### 8.3.2.3.6 DMA Classes

The first N virtIDs can be reserved for DMA class usage in the PVU\_VIRTID\_MAP1 register. These virtIDs map to TLBs along with classes that are defined in the chanID bits 10 to 11 on the bus. This creates four classes per virtID, that can select from four TLBs, allowing the transaction to restrict its translations to smaller sets from known distinct memory segments (such as the DMA accessing descriptors or buffers). A set of bitfields define the class mapping, which defines the final class value to use for an input class, and can be used to restrict the number of classes if all four are not used (by mapping 2 to 0 and 3 to 1 for example to limit to two classes). Then the final TLB selection is calculated as:

$$\text{virtID} \times 4 + \text{final class value}$$

The unused classes cause unused TLBs as entry points, but they can be used in chaining.

#### 8.3.2.3.7 General virtIDs

After the first N virtIDs for DMA class usage, the remaining virtIDs up to the maximum allowed (as set in PVU\_VIRTID\_MAP2) are for general usage. Each of these map one to one to a TLB, where the calculation is:

$$\text{dma\_cnt} \times 4 + (\text{virtID} - \text{dma\_cnt})$$

There are no classes used for this range of virtIDs. An input virtID that is beyond the MAX\_CNT bitfield value will produce an error.

#### 8.3.2.3.8 TLB Lookup

The next step of the module is to check the input address, which uses the bits just above the 4-kB size, bits 12 and up, as the virtual address base. This is compared against all the TLB entries for that TLB, given the size in each entry. The size affects which address bits need comparing for an entry. The minimum size of 4 kB, so address bits 12 and up are compared. But if the size is 64 kB, then bits 16 and up are compared, as the lower bits fall into the size of the page. In addition to the address, the psecure bit is used to match against the secure level of the transaction. Only secure transaction are allowed to match an entry with psecure set, while only non-secure transactions are allowed to match an entry with psecure clear. If the entry is valid and the base compare matches and the secure level matches then the entry is used for the translation and checks.

#### 8.3.2.3.9 TLB Miss

If the TLB lookup results in no matches and there is no chaining to another TLB (described later), then this is called a TLB miss. If the TLB misses, since there is no back-fill (such as from an MMU), the result is that the translation fails and will fault (a fault is when there is no physical address available). The transaction will be marked as in error with the flush signal, and a later slave will perform the flush that will return a bus error and read data as zeros. The PVU will record the fault as an interrupt and log the fault in the exception logging read-only registers.

#### 8.3.2.3.10 Multiple Matching Entries

The PVU does not support multiple entries in a TLB that match the same virtual address. This is an illegal condition for software to program, as there is no mechanism to select one over the other. The PVU will have unpredictable results if this occurs, with the one requirement that the input transactions will never transition from a non-secure attribute to a secure attribute even in this illegal condition. This means that there is no support for background and foreground translating entries in the same TLB (which would be the result if a real MMU were used also). If software user wants to use a foreground and background method (that is not MMU compliant), he can place the foreground entries in earlier TLBs that are chained together, and the background entries are in the later TLBs, so that no single TLB has overlapping entries. The background entry would only be hit when the address falls within the background range but not any of the foreground ranges.

#### 8.3.2.3.11 TLB Disable

A special case is when a TLB should go through a lookup but is disabled through the PVU\_CHAIN\_j[31] EN bit, or the entire PVU is disabled through the PVU\_ENABLE[0] EN bit. If a lookup is requested when the TLB or PVU is disabled, then the transaction will pass and no translation will occur (output address is the same as the input address). Also, if the atype=1 then it will change to atype=0 on the output so that the resulting bypass transaction does not get routed right back to the PVU. Any other atype value will be maintained on the output. Any entries set in the disabled TLB are ignored as well as the chaining setting (described in [Section 8.3.2.3.12](#)) is also ignored, so once a disabled TLB is used then the chain is ended. Software must take care to not include any disabled TLBs in chains that must be fully searched – if a TLB must be disabled then the chain must be updated first to skip over that TLB and any virtIDs that directly map to that TLB initially must not be used.

#### 8.3.2.3.12 TLB Chaining

If the TLB has a chain enabled by writing a non-zero chain value, then the TLB lookup will continue for the transaction using the chained TLB number. There is a latency penalty to resume the lookup at the chained TLB,

but it allows more TLB resources to be used for virtIDs that might need more translations. If a TLB misses and has a chain value of zero then the overall result is a miss. Any TLB logs are always applied to the original TLB of the chain, and not the chained TLB.

TLB chaining can be defined in the PVU\_CHAIN\_j[11:0] CHAIN register bitfield.

### 8.3.2.3.13 TLB Permissions

The TLB entry contains permissions for the page that must be checked before the translation can be used. These include if the page has *supervisor* access for read, write, or execute, and has *user* access for read, write, or execute. The supervisor or user is based on the transaction priv[0] signal (0 = user, 1 = supervisor). The access type is based on the transaction dtype[0] and dir signals (dtype[0] = 0 and dir = 1 is a read, dtype[0] = 0 and dir = 0 is a write, dtype[0] = 1 and dir = 1 is an execute). The transaction signals are checked against these permissions. If they are allowed then the translation continues. If the page does not allow this transaction then the check fails and flush signal set is on the bus, like for a firewall, indicating that the transaction is no longer valid and must return an error. PVU does not perform the error signaling, so a later module must support it. A fault interrupt is produced.

The pperm entry bits define four additional checks. If pperm[0] is clear then user privilege access is not allowed, and a matching transaction with the CBA bus priv signal set to 0 (indicating user access) will result in an error. If pperm[1] is set then write access is not allowed, and a matching transaction that is a write will result in an error. If pperm[2] is set then instruction execute is not allowed, and a matching transaction that has the CBA bus dtype signal set to 1 (instruction fetch) will result in an error. If pperm[3] is set then supervisor instruction execute is not allowed, and a matching transaction that has the CBA bus priv signal non-zero (above user) and the CBA bus dtype signal set 1 (instruction fetch) will result in an error. If all these checks pass then the translation is allowed.

The pprefetch entry bit determines whether the translation allows prefetch transactions. If set, then any transaction with the pfable signal set are allowed. But if pprefetch=0 then those transactions with pfable set will fault as prefetch is defined as not supported.

### 8.3.2.3.14 Translation

If all the checks pass, then the translated address is replaced and the transaction can proceed to the output. Only the address bits above the page size are replaced, and all the lower bits are copied from the transaction. The memory attributes are also replaced on the transaction. These attributes include the memory type, shareability, and allocation policies. In addition, the prefetchable request is only copied if the page allows prefetch, and otherwise it is forced clear, but the transaction is still allowed to continue (just the prefetch will not occur).

### 8.3.2.3.15 Memory Attributes

Table 8-38 shows the mapping of page attributes to the bus memory attributes.

**Table 8-38. Page Attributes to Bus Memory Attributes Mapping**

Bus Signal	Page Attribute	Translated Bus Signal
cinner	piallocpol	cinner = piallocpol
couter	poallocpol	couter = poallocpol
memtype	pmemtype	memtype = pmemtype
sdomain	posable, pisable	sdomain[1] = posable, sdomain[0] = pisable
pable	pprefetch	pable = pable AND pprefetch

### 8.3.2.3.16 Faulted Transactions

A faulted transaction will have the flush bit set on the outgoing transaction so that it will return errors by the interconnect. The atype will be set to 0 since the memory attributes have not been replaced. And the outgoing transaction address will have the upper bits set to the fault\_addr parameter value (see Section 8.3.2.1.2), which defines an interconnect address that will always return errors and not interfere with any valid slave traffic (such as to memory).

#### **8.3.2.3.17 Non-Virtual Transactions**

The module must not receive a transaction that does not have the atype set to an intermediate address, but if it does, there is no translation performed for that transaction. The transaction is simply copied to the output without modification.

#### **8.3.2.3.18 Allowed virtIDs**

Software can control the limit of usable virtIDs in the PVU by PVU\_VIRTID\_MAP2[11:0] MAX\_CNT register. This sets the maximum virtID allowed value. If the PVU receives a transaction with a higher virtID signal, then the transaction will be treated as a fault and error, and no TLB lookup will be performed.

#### **8.3.2.3.19 Software Control**

The software can control all the TLB entries directly. The TLB can be programmed at any time by software, and writes to the TLB take precedence over lookups. This allows the update to be reflected as soon as possible in the translations. But there is no guarantee about the timing of the update against any bus transactions, so the only way to guarantee an endpoint uses the updated table is to stall the endpoint until the update has been completed (using the write status). This allows software to directly populate TLB entries for known translation as well as locking entries down so that they cannot be replaced.

#### **8.3.2.3.20 Fault Logging**

The module will log faults into exception logging registers. It will capture that type of fault, the address that faulted, and the attributes of the transaction that faulted. It will also set an interrupt. The module can only store one fault. When the software clears the interrupt status then the fault is also cleared and another can be captured. If another fault occurs while one is pending then that fault log is lost.

There is a logging Enable/Disable bit per TLB and if disabled then faults from that TLB will not be logged. PVU\_EXCEPTION\_LOGGING\_DISABLE register allows to disable types of faults, and when disabled that type of fault will not be logged. There are additional fault status fields per TLB that capture fault status that would have been logged but could not because another fault was already logged. These status bits stay set until cleared by software.

A TLB fault that occurs after chaining will set the fault status of the original TLB, and not of the last TLB in the chain.

#### **8.3.2.3.21 Alignment Restrictions**

All accesses through the module must not cross 4-kB page boundaries. The module does not perform any checks for this requirement.

### 8.3.3 Page Based Address Translation Unit (PAT)

This chapter describes the PAT module, part of the main NAVSS.

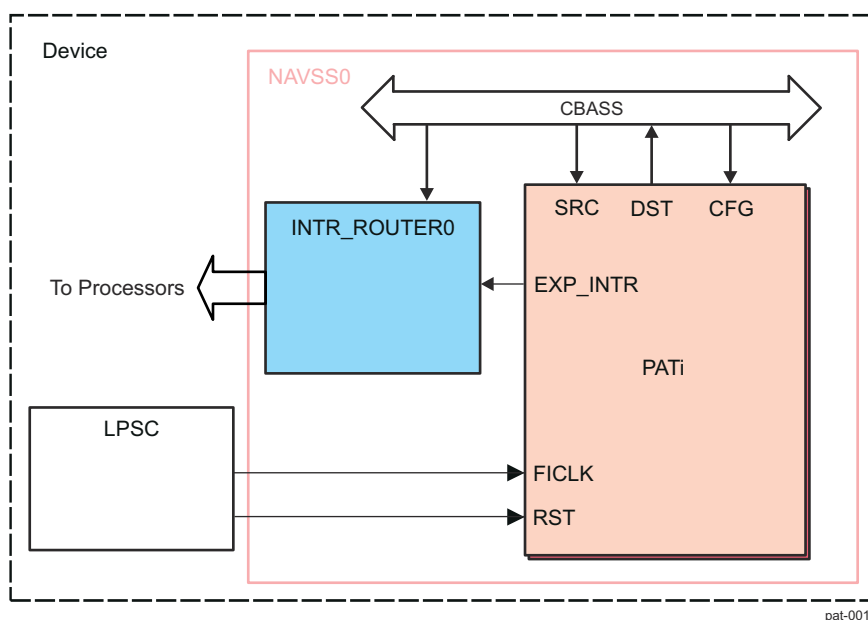
#### 8.3.3.1 PAT Overview

PAT performs a page-based address translation on a CBA VBUSM bus. [Table 8-39](#) lists PAT instances in the device.

**Table 8-39. PAT Allocation Within Device Domains**

Module Instance	Domain		
	WKUP	MCU	MAIN
NAVSS0_PAT0	-	-	✓ (NAVSS)
NAVSS0_PAT1	-	-	✓ (NAVSS)
NAVSS0_PAT2	-	-	✓ (NAVSS)
NAVSS0_PAT3	-	-	✓ (NAVSS)
NAVSS0_PAT4	-	-	✓ (NAVSS)

[Figure 8-21](#) shows a top-level overview of the PAT module.



**Figure 8-21. PAT Overview**

#### 8.3.3.1.1 PAT Features

Each PAT supports the following features:

- Implements a VBUSM retiming bridge with page based address translation
- Supports a number of pages (see [Table 8-40](#)), each occupying a programmable 4-KB, 16-KB, 64-KB, or 1-MB of address space
- Supports a 48-bit output address size
- Allows fast programming of the table, or fast generic SRAM access when the table is disabled
- Supports *transaction id* reassignment
- Supports isolation of registers by placing every 256 pages worth of control registers in their own 4-KB MMU page.

### 8.3.3.1.2 PAT Parameters

Table 8-40 shows the PAT configuration parameters set during SoC design.

**Table 8-40. PAT Configuration Parameters**

Module Instance	Parameters				
	Number of Pages <sup>(1)</sup>	Large	Source ID	Config FW	RouteID
NAVSS0_PAT0	16384	0	0	0x0000:6:3:4736	224
NAVSS0_PAT1	16384	0	0	0x0400:6:3:4737	225
NAVSS0_PAT2	16384	0	0	0x0800:6:3:4738	226
NAVSS0_PAT3	2048	1	1	0x0c00:6:3:4739	227
NAVSS0_PAT4	2048	1	1	0x1000:6:3:4740	228

(1) Can be read off from the PAT\_CONFIG read-only register.

### 8.3.3.1.3 PAT Not Supported Features

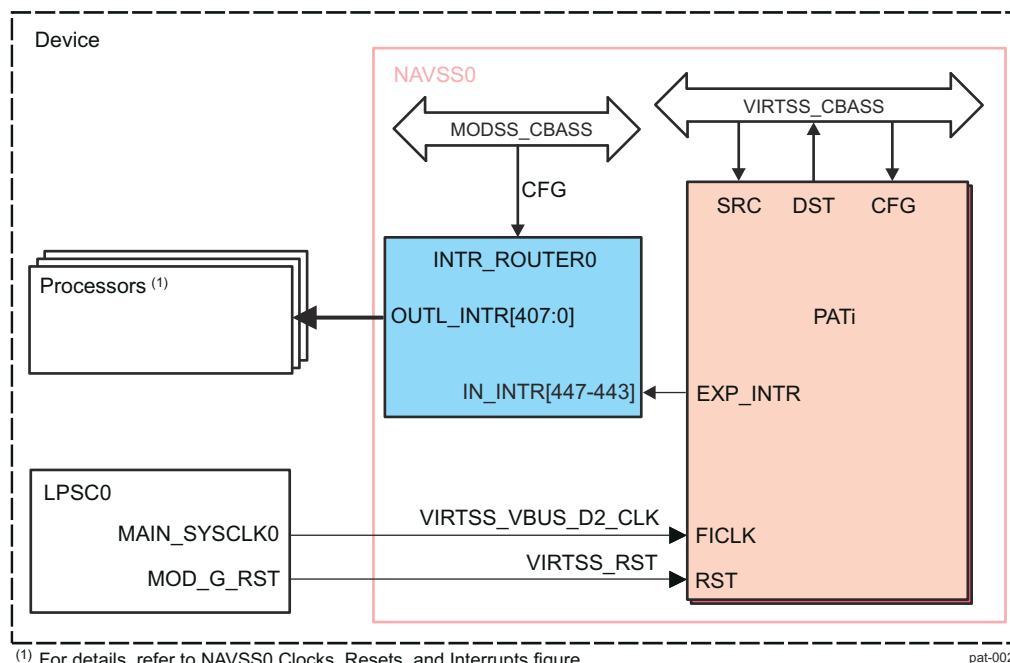
The following features are not supported by the module:

- Does not support transactions that cross a 4-KB page boundary

### 8.3.3.2 PAT Integration

This section describes module integration in the device, including information about clocks, resets, and hardware requests.

Figure 8-22 shows the PAT integration in the device.



**Figure 8-22. PAT Integration**

i = 0 to 4

Table 8-41 and Table 8-42 summarize the integration of the module in the device.

**Table 8-41. PAT Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
NAVSS0_PAT0	PSC0	GP	LPSC0	VIRTSS_CBASS
NAVSS0_PAT1	PSC0	GP	LPSC0	VIRTSS_CBASS
NAVSS0_PAT2	PSC0	GP	LPSC0	VIRTSS_CBASS
NAVSS0_PAT3	PSC0	GP	LPSC0	VIRTSS_CBASS
NAVSS0_PAT4	PSC0	GP	LPSC0	VIRTSS_CBASS

**Table 8-42. PAT Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
NAVSS0_PAT0	PAT0_FICLK	VIRTSS_VBUS_D2_CLK	MAIN_SYSCLK0	PAT clock. This clock is used for all interface and functional operations.
NAVSS0_PAT1	PAT1_FICLK	VIRTSS_VBUS_D2_CLK	MAIN_SYSCLK0	PAT clock. This clock is used for all interface and functional operations.
NAVSS0_PAT2	PAT2_FICLK	VIRTSS_VBUS_D2_CLK	MAIN_SYSCLK0	PAT clock. This clock is used for all interface and functional operations.
NAVSS0_PAT3	PAT3_FICLK	VIRTSS_VBUS_D2_CLK	MAIN_SYSCLK0	PAT clock. This clock is used for all interface and functional operations.
NAVSS0_PAT4	PAT4_FICLK	VIRTSS_VBUS_D2_CLK	MAIN_SYSCLK0	PAT clock. This clock is used for all interface and functional operations.



**Table 8-42. PAT Clocks and Resets (continued)**

Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
NAVSS0_PAT0	PAT0_RST	VIRTSS_RST	LPSC0	PAT hardware reset
NAVSS0_PAT1	PAT1_RST	VIRTSS_RST	LPSC0	PAT hardware reset
NAVSS0_PAT2	PAT2_RST	VIRTSS_RST	LPSC0	PAT hardware reset
NAVSS0_PAT3	PAT3_RST	VIRTSS_RST	LPSC0	PAT hardware reset
NAVSS0_PAT4	PAT4_RST	VIRTSS_RST	LPSC0	PAT hardware reset

**Table 8-43. PAT Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
NAVSS0_PAT0	PAT0_EXP_INTR	IN_INTR[447]	INTR_ROUTER0	Exception interrupt	Level
NAVSS0_PAT1	PAT1_EXP_INTR	IN_INTR[446]	INTR_ROUTER0	Exception interrupt	Level
NAVSS0_PAT2	PAT2_EXP_INTR	IN_INTR[445]	INTR_ROUTER0	Exception interrupt	Level
NAVSS0_PAT3	PAT3_EXP_INTR	IN_INTR[444]	INTR_ROUTER0	Exception interrupt	Level
NAVSS0_PAT4	PAT4_EXP_INTR	IN_INTR[443]	INTR_ROUTER0	Exception interrupt	Level
DMA Events					
Module Instance	Module DMA Event	Destination DMA Event Input	Destination	Description	Type
NAVSS0_PAT0	-	-	-	No PDMA channels to external DMA engines	-
NAVSS0_PAT1	-	-	-	No PDMA channels to external DMA engines	-
NAVSS0_PAT2	-	-	-	No PDMA channels to external DMA engines	-
NAVSS0_PAT3	-	-	-	No PDMA channels to external DMA engines	-
NAVSS0_PAT4	-	-	-	No PDMA channels to external DMA engines	-

### Note

For more information on the interconnects, see [Chapter 3, System Interconnect](#).

For more information on the power, reset, and clock management, see the corresponding sections within [Chapter 5, Device Configuration](#).

### 8.3.3.3 PAT Functional Description

#### 8.3.3.3.1 Functional Operation Overview

PAT performs a page-based address translation on a CBA VBUSM bus. The module acts as a VBUSM retiming bridge that also performs the address translation.

The entire PAT can be enabled or disabled via the PAT\_CONTROL register.

#### 8.3.3.3.2 Page Table

The table consists of a programmed translated base address per page. Each page is a programmable 4 KB, 16 KB, 64 KB, or 1 MB in size and each translated base is 36 bits. The input address uses the bits just above the page size, shown in [Table 8-44](#), to define the index within the table. A lookup is performed using that index to find the associated translated base. Then, that translated base replaces the upper bits of the final output address. The lower bits of the output address are copied from the input address, all shown in [Table 8-44](#). If the table is disabled or the index is beyond the non-power-of-2 size of the table, then the address is not translated, and the output address is copied from the input address and zero extended. Table sizes that are an exact power-of-2 cannot detect a beyond index and the index will just wrap and alias.

The table itself is built using an SRAM.

**Table 8-44. Index Bits and Final Address vs Page Size**

Page Size	Index Bits (N depends on table size)	Final Address
4 KB	address[N:12]	trans_address[47:12], address[11:0]
16 KB	address[N:14]	trans_address[47:14], address[13:0]
64 KB	address[N:16]	trans_address[47:16], address[15:0]
1 MB	address[N:20]	trans_address[47:20], address[19:0]

#### 8.3.3.3.3 Alignment

All accesses through the PAT must not cross 4-KB page boundaries. The module does check for this condition. If a transaction crosses a 4-KB boundary, the entire transaction will set the bus *cflush* signal to return an error. The error is also logged in the PAT\_EXCEPTION\_LOGGING registers and an interrupt can be generated.

#### 8.3.3.3.4 Page Enables

There is an overall PAT enable bit PAT\_CONTROL register, and each page entry has an enable bit - PAT\_BASE\_REG\_H\_j\_k (where j = block, k = page). If the transaction is to the PAT and the PAT\_CONTROL[0] ENABLE bit is clear, then no pages are enabled and all lookups to the table will result in an error. Similarly, if the page entry bit is clear the page is not enabled and any lookups to that page will result in an error. The transaction will be flushed and the error logged in the PAT\_EXCEPTION\_LOGGING registers. If this bit is set, then the page is enabled and the translation will be performed. When changing a page entry, the safest approach is to clear the ENABLE bit first, then write the lower address value, and finally the upper address value along with setting the ENABLE bit. This will guarantee that a lookup cannot use the entry while it is in the middle of the update.

#### 8.3.3.3.5 Table Arbitration

Access to the table can follow two arbitration methods.

- The first is to always prioritize table updates before lookups. In this method, the updates will always be allowed and the lookups will be stalled until there is no update.
- If the table is being updated frequently, there is a second method that performs round robin between the accesses. This will alternate between the two access types if both are pending, and give a balanced bandwidth to each. If only one is pending then it is given access regardless of the round robin state.

#### 8.3.3.3.6 Programming

The table can be programmed at any time. There is no guarantee about the timing of the update against any bus transactions, so the only way to guarantee an endpoint uses the updated table is to stall the endpoint until the table update has been completed. Or the page update can first disable that page, and then write the updated

page entry. If a transaction arrives while the page is disabled it will result in an error. All table accesses must be for full 32-bit words, as byte access is not supported for the table registers, that is, PAT\_BASE\_REG\_L\_j\_k and PAT\_BASE\_REG\_H\_j\_k.

#### **8.3.3.3.7 Scratch RAM**

When the PAT table is disabled, the table RAM can be used as a scratch RAM. Software simply needs to access the PAT scratch (PAT\_MEM\_y) locations. The module supports bursts on the config VBUSP port to keep bandwidth as high as the burst can support. A DMA, for instance, could probably achieve full bandwidth on the 32-bit bus. Since the PAT table is disabled, the SRAM accesses will not stall any transactions through the VBUSM bridge datapath. It is illegal to access the scratch RAM when the PAT table is enabled, and unexpected behavior for translations may occur.

#### **8.3.3.3.8 Error Reporting**

The PAT will capture details of any transaction that causes an error. There are two types of errors:

- The first type is when an incoming transaction crosses a 4-KB boundary, which is illegal on the CBA bus anyway
- The second error type is when the entire PAT or requested page entry is disabled, or the translation is not valid.

If an error occurs, the transaction will be flushed and the error details logged into the PAT\_EXCEPTION\_LOGGING registers. The error logging can generate an interrupt, and can be automatically cleared by reading the bottom error logging register - PAT\_EXCEPTION\_LOGGING\_DATA3.

Only one error can be captured until the interrupt is cleared, and any further errors are lost.

## 8.4 Region-based Address Translation (RAT) Module

This section describes the RAT functionality.

### 8.4.1 RAT Functional Description

#### 8.4.1.1 RAT Availability

Table 8-45 lists the device modules and subsystems which have RAT module.

**Table 8-45. Device Modules and Subsystems with RAT Module**

Module Instance	Domain		
	WKUP	MCU	MAIN
WKUP_DMSC0	✓	-	-
MCU_R5FSS0_CORE0	-	✓	-
MCU_R5FSS0_CORE1	-	✓	-
GPU0	-	-	✓
PRU_ICSSG0_CPU0	-	-	✓
PRU_ICSSG0_CPU1	-	-	✓
PRU_ICSSG1_CPU0	-	-	✓
PRU_ICSSG1_CPU1	-	-	✓
R5FSS0_CORE0	-	-	✓
R5FSS0_CORE1	-	-	✓
R5FSS1_CORE0	-	-	✓
R5FSS1_CORE1	-	-	✓
MLBSS0	-	-	✓
VPFE0	-	-	✓
C66X_CORE0	-	-	✓
C66X_CORE1	-	-	✓

**Table 8-46. RAT Clocks and Resets**

<b>Clocks</b>	
<b>Module Clock Input</b>	<b>Description</b>
RAT_ICLK	RAT Interface Clock
<b>Resets</b>	
<b>Module Reset Input</b>	<b>Description</b>
RAT_RST	RAT reset

**Table 8-47. RAT Hardware Requests**

<b>Interrupt Requests</b>		
<b>Module Interrupt Signal</b>	<b>Description</b>	<b>Type</b>
RAT_PLS_INTR	RAT Pulse Interrupt Output	Pulse
RAT_INTR	RAT Interrupt Output	Level

#### 8.4.1.2 RAT Operation

The RAT module performs a region based address translation. It translates a 32-bit input address into a 48-bit output address. Any input transaction that starts inside of a programmed region will have its address translated, if the region is enabled. Any disabled region is ignored from the translation lookup.

RAT has 16 regions, with each region having dedicated registers that define its attributes:

- Base address of the region, via the RAT\_BASE\_k register.
- Size of the region, via the RAT\_CTRL\_k register.

- Translated base address (48-bit). It is defined by 32-bit lower address via the RAT\_TRANS\_L\_k register, and 16-bit upper address via the RAT\_TRANS\_U\_k register.

The input addresses are compared against all the enabled regions in parallel. The region size defines how many of the lowest address bits are included in the region space, starting from a 1B space to a 4GB space. Then the upper bits not part of the region are used for the lookup and comparison, matching those bits of the input addresses against the region base address. The region matches when it is enabled and the compare matches. The first region that matches has the output address set with the region translated base address upper bits. The lower bits that are part of the region size are copied from the input address to the output address. If there are no region matches then the input address is copied to the output address entirely.

RAT does not automatically correct transactions that cross region boundaries, that is, start inside of a region and end outside of it, or start outside of a region and end inside of it. If a boundary crossing transaction is detected, then RAT annuls the transaction as illegal, generates an interrupt and logs an error, as explained in *RAT Error Logging*.

#### Note

The region base address and translated base address must be aligned to the defined region size. For example, if the defined region size is 64KB, then the two addresses must be 64KB aligned. This is software responsibility as RAT does not perform such alignment check. Regions that are not aligned have unpredictable results.

Moreover, multiple region definitions must not overlap in their covered address space. RAT does not check for this, so it is software responsibility to take care of it. Overlapping regions may lead to unpredictable results.

#### 8.4.1.3 RAT Error Logging

The error log consists of a series of registers that capture the details of the transaction, including the input address that caused the error. These registers are the following:

- RAT\_EXCEPTION\_LOGGING\_HEADER0 and RAT\_EXCEPTION\_LOGGING\_HEADER1
- RAT\_EXCEPTION\_LOGGING\_DATA0 through RAT\_EXCEPTION\_LOGGING\_DATA3

The RAT module can capture one error before it is cleared by software. The error logging is enabled by default, but can be disabled via the RAT\_EXCEPTION\_LOGGING\_CONTROL[0] DISABLE\_F bit. Upon error logging an interrupt is also generated. To clear the log, software must either read the final error logging register, or manually clear the RAT\_EXCEPTION\_PEND\_CLEAR[0] PEND\_CLR bit by setting it to 0x1. This will clear the error status, and not the actual log registers, but it does allow the next error to be captured into the log registers. If the status is not cleared and additional errors are detected, they are not logged.

After an error occurs and is cleared (whether by reading the final error logging register or by clearing the status bit), the RAT\_EOI\_REG register must be written to guarantee the next interrupt pulse will be produced.

Table 8-48 shows the RAT source ID mapping which is associated with the RAT\_EXCEPTION\_LOGGING\_HEADER0[23-8] SRC\_ID field.

**Table 8-48. RAT Source ID Mapping**

Module Instance	Source ID Value (Decimal)
WKUP_DMSC0	0
MCU_R5FSS0_CORE0	2
MCU_R5FSS0_CORE1	3
GPU0	32
PRU_ICSSG0_CPU0	16
PRU_ICSSG0_CPU1	17
PRU_ICSSG1_CPU0	18
PRU_ICSSG1_CPU1	19

**Table 8-48. RAT Source ID Mapping (continued)**

Module Instance	Source ID Value (Decimal)
R5FSS0_CORE0	4
R5FSS0_CORE1	5
R5FSS1_CORE0	6
R5FSS1_CORE1	7
MLBSS0	56
VPFE0	52
C66X_CORE0	64
C66X_CORE1	65



9.1 Interrupt Architecture.....	934
9.2 Interrupt Controllers.....	936
9.3 Interrupt Routers.....	954
9.4 Interrupt Sources.....	966

## 9.1 Interrupt Architecture

The SoC has many peripherals and a large number of event sources (interrupt, time sync or DMA). The use of events is completely dependent on a user's specific application, which drives a need for maximum flexibility in which event sources are used in the system. It is also completely up to software control as to how the events are serviced.

The SoC includes the following interrupt servicing modules (hosts):

- Device management and security controller (WKUP\_DMSC0):
  - Single Arm Cortex-M3 core
  - Nested vectored interrupt controller (NVIC)
- Compute cluster:
  - Two processor subsystems:
    - Dual-core Arm Cortex-A72 microprocessor unit (A72SS0)
    - Single-core C71x DSP subsystem (C71SS0)
  - Two interrupt controllers, both integrated inside the MSMC wrapper:
    - Arm generic interrupt controller (GIC-500) supports both cores in the dual-A72 cluster
    - Cluster level event controller (CLEC) supports the C71x DSP
- C66x DSP subsystems (C66SS0, C66SS1), each implementing:
  - Single C66x CPU core
  - Local interrupt controller
- Microcontroller units (R5FSS0, R5FSS1, MCU\_R5FSS0), each implementing:
  - Two Arm Cortex-R5F cores
  - Vectored interrupt manager (VIM)
- Industrial communications subsystems (PRU\_ICSSG0, PRU\_ICSSG1), each implementing:
  - Two programmable real-time units (PRUs)
  - Local interrupt controller (PRU\_ICSSGn\_INTC0)

Most of the system events are routed directly to the various processing elements but in some cases it is impractical to route all events of a certain group (for example, GPIO events) to each processing element. For this purpose, the SoC integrates several interrupt router (INTRTR) instances. Each interrupt router aggregates a number of system events and can route each event to a given processing element by using simple combinational logic (a set of multiplexors). Event selection is controlled through the associated registers within each interrupt router.

The following interrupt router instances are part of the SoC interrupt architecture:

- WKUP domain GPIO interrupt router (WKUP\_GPIOMUX\_INTRTR0):
  - Provides selection of active module interrupts for the WKUP\_GPIO0 module
- MAIN domain GPIO interrupt router (GPIOMUX\_INTRTR0):
  - Provides selection of active module interrupts for the MAIN domain GPIO0 and GPIO1 modules
- MAIN-to-MCU domain level interrupt router (MAIN2MCU\_LVL\_INTRTR0):
  - Provides selection of active MAIN domain module *level* interrupts for routing to MCU domain processing elements
  - Reduces the number of event voltage domain crossings
- MAIN-to-MCU domain pulse interrupt router (MAIN2MCU\_PLS\_INTRTR0):
  - Provides selection of active MAIN domain module *pulse* interrupts for routing to MCU domain processing elements
  - Reduces the number of event voltage domain crossings
- R5FSS0 interrupt router (R5FSS0\_INTRTR0):
  - Provides selection of active module interrupts for the R5FSS0 module
- R5FSS1 interrupt router (R5FSS1\_INTRTR0):
  - Provides selection of active module interrupts for the R5FSS1 module
- C66SS0 interrupt router (C66SS0\_INTRTR0):



- Provides selection of active module interrupts for the C66SS0 module
- C66SS1 interrupt router (C66SS1\_INTRTR0):
  - Provides selection of active module interrupts for the C66SS1 module

In addition, the following interrupt router instances are part of the SoC time sync architecture:

- Time sync event router (TIMESYNC\_INTRTR0):
  - Provides selection of active common platform time synchronization (CPTS) events for routing to CPTS capable modules
- Compare event router (CMPEVT\_INTRTR0):
  - Provides selection of active CPTS counter compare events for routing as processor interrupts or DMA events

For more details on time sync routers, see [Section 11.3.2, Time Sync Routers](#).

## 9.2 Interrupt Controllers

### 9.2.1 Generic Interrupt Controller (GIC)

This section describes the Generic Interrupt Controller (GIC) module in the device.

#### 9.2.1.1 GIC Overview

The device GIC is a TI module that is based on the Arm GIC-500 interrupt controller. The Arm GIC-500 is a high-performance, area-optimized, built-time configurable interrupt controller which detects, manages, and distributes system interrupts to the Arm® Cortex®-A72 processors in the Compute Cluster.

The Arm GIC-500 is compliant to the Arm GICv3 standard. It only supports cores that implement the Armv8 architecture and the GIC CPU interface with the standard GIC Stream Protocol Interface (such as A72).

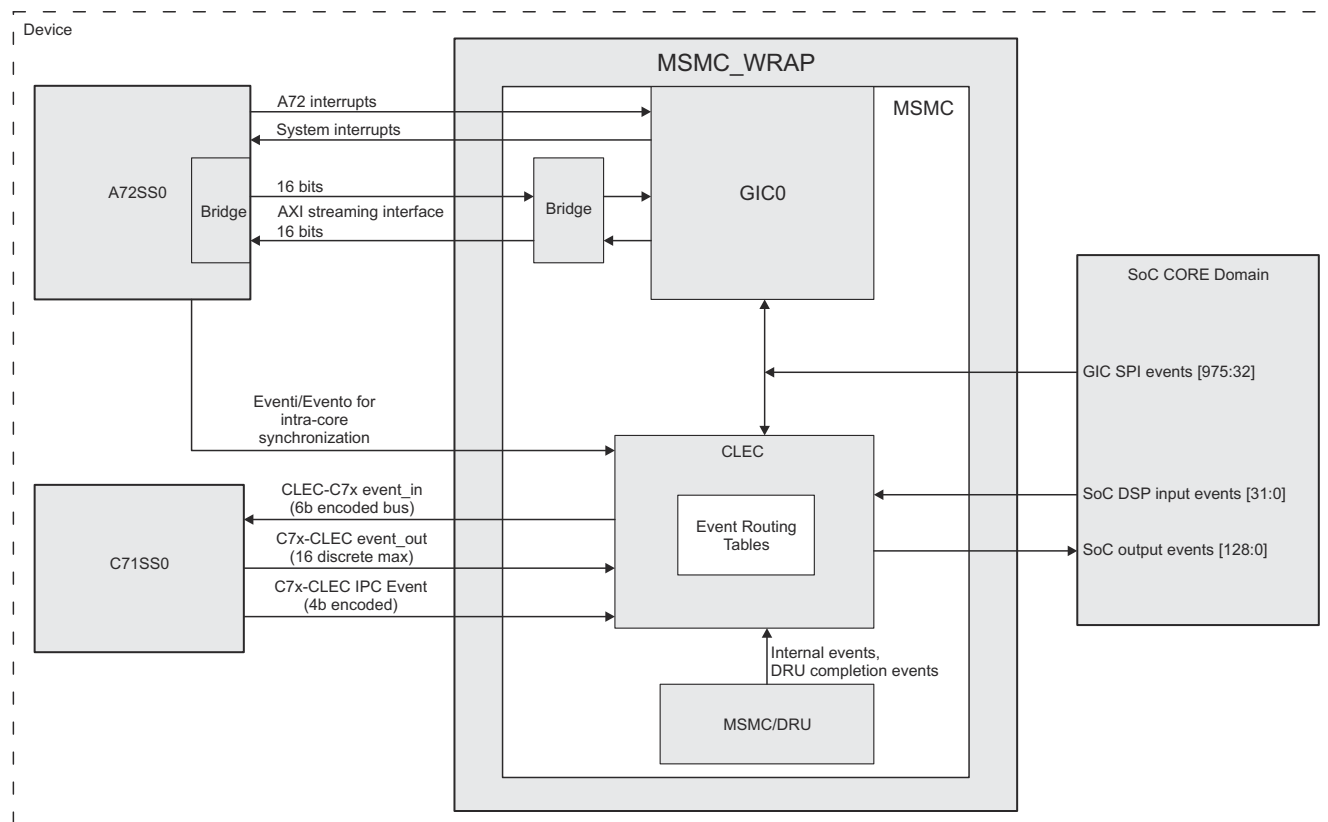
The GIC includes additional logic (TI wrapper) to fully integrate the Arm GIC-500 into the SoC.

The device includes one GIC instance named GIC0. [Table 9-1](#) shows GIC module allocation within device domains.

**Table 9-1. GIC Module Allocation within Device Domains**

Module Instance	Domain		
	WKUP	MCU	MAIN
GIC0	–	–	✓

The GIC module is located inside the MSMC\_WRAP logical block in the Compute Cluster, along with the cluster level interrupt controller (CLEC), which serves the C71x DSP. [Figure 9-1](#) shows an overview of the Compute Cluster interrupt architecture.



**Figure 9-1. Compute Cluster Interrupt Architecture**

### 9.2.1.1.1 GIC Features

#### Note

This chapter provides only a brief description of the GIC functionality and features. For more details on this module, refer to the *Arm®CoreLink™ GIC-500 Generic Interrupt Controller Technical Reference Manual*.

The Arm GIC-500 supports the following features in the device:

- Module revision: r1p1
- Supports both cores in the dual-A72 MPU cluster
- 16 software generated interrupts (SGIs) per core
- 16 private peripheral interrupts (PPIs) per core
- 960 shared peripheral interrupts (SPIs)
- Locality-specific peripheral interrupts (LPIs), used for message-based interrupts
- Interrupt translation service (ITS)
  - Device isolation
  - ID translation for message-based interrupts
  - This allows virtual machines to program devices directly
- Interrupt masking and prioritization
- Programmable, affinity-based interrupt routing
- 32 priorities for each interrupt
- Three different interrupt groups
  - Group 0
  - Non-secure group 1
  - Secure group 1

Additionally, the GIC wrapper logic provides the following features:

- Synchronizes interrupt inputs to GIC clock
- Implements AXI2VBUSM and VBUSM2AXI bridges
  - Used for bus protocol conversion (AXI-to-VBUSM and VBUSM-to-AXI, respectively)
- Implements an integrated ECC aggregator
  - Only used to inject errors (for testing purposes)

### 9.2.1.1.2 GIC Not Supported Features

The GIC does not support the following features:

- GICv2 backwards compatibility
- AWID-based LPI mapping

### 9.2.1.1.3 GIC Configuration Summary

[Table 9-2](#) summarizes the GIC configuration for this device.

**Table 9-2. GIC Configuration Summary**

Parameter	SoC Value
Number of affinity-level 1 CPU Clusters (Corepacs)	1
Number of CPUs in cluster 0	2
Number of SPIs	960
ITS/LPI support <ul style="list-style-type: none"> <li>• TRUE: ITS is present. LPIs are supported</li> <li>• FALSE: ITS is not present. LPIs are not supported</li> </ul>	TRUE
Device ID width	20
Number of LPI cache entries	64

**Table 9-2. GIC Configuration Summary (continued)**

Parameter	SoC Value
Width of AXI4 <b>awid_s</b> and <b>bid_s</b> signals	4
Width of AXI4 <b>arid_s</b> and <b>rid_s</b> signals	4
GICv2 backwards compatibility support <ul style="list-style-type: none"> <li>TRUE: No support for GICv2 backward compatibility</li> <li>FALSE: Software-programmable GICv2 backward compatibility</li> </ul>	TRUE
Security support <ul style="list-style-type: none"> <li>TRUE: No concept of security</li> <li>FALSE: Security enabled. At least one processor must be able to make secure accesses. (Secure access requirement can be disabled by secure software at boot)</li> </ul>	FALSE

### 9.2.1.2 GIC Integration

This section describes the GIC module integration in the device, including information about clocks, resets, and hardware requests.

Figure 9-2 shows the GIC module integration.

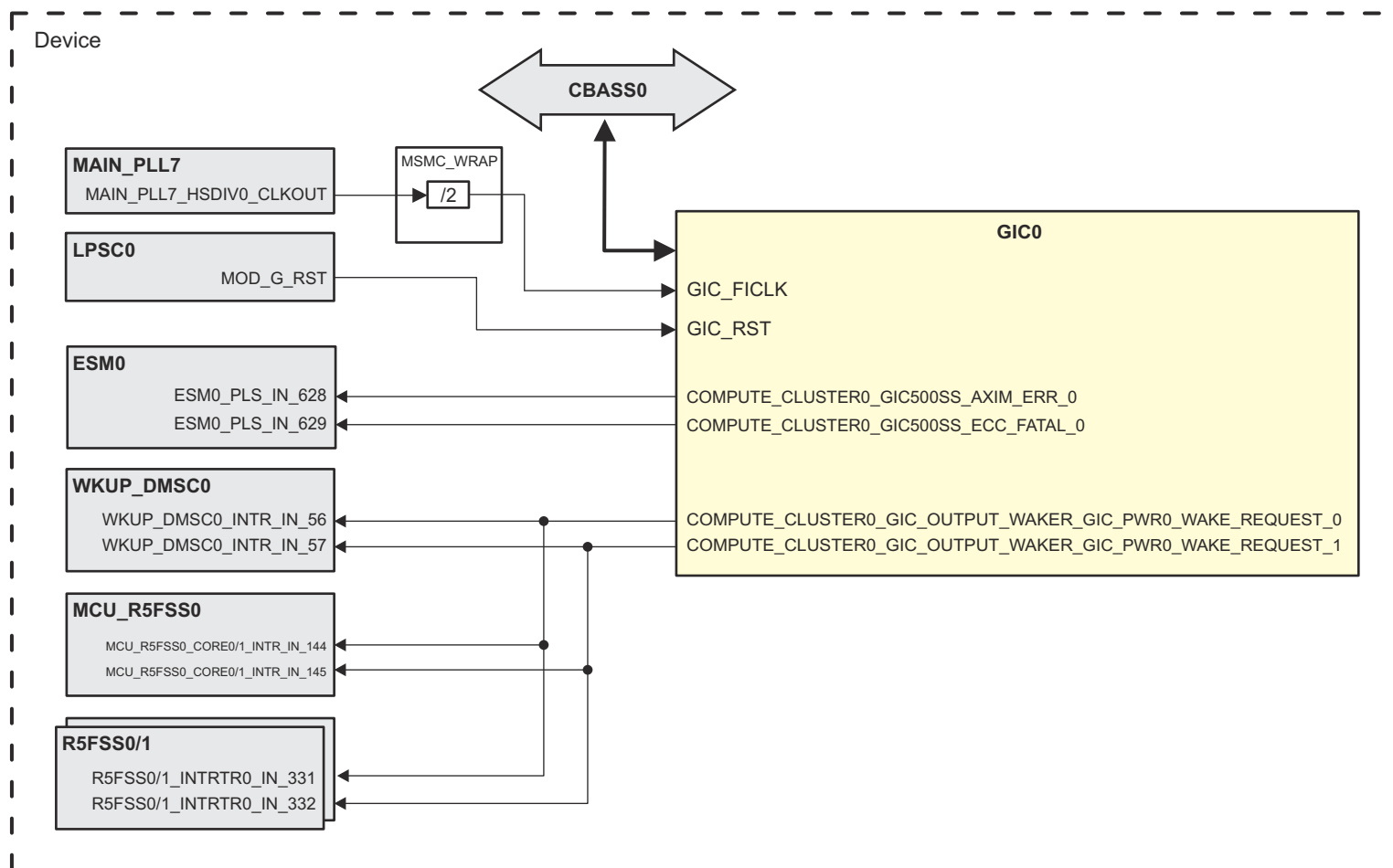


Figure 9-2. GIC Integration

Table 9-3 through Table 9-5 summarize the GIC integration.

**Table 9-3. GIC Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
GIC0	PSC0	PD0	LPSC0	CBASS0

**Table 9-4. GIC Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
GIC0	GIC_FICLK	MAIN_PLL7_HSDIV0_CLKOUT/2 <sup>(1)</sup>	MAIN_PLL7	Module functional and interface clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
GIC0	GIC_RST	MOD_G_RST	LPSC0	Module hardware reset

(1) Division by 2 is done internally in the Compute Cluster.

**Table 9-5. GIC Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
GIC0	COMPUTE_CLUSTER0_GIC500SS_AXIM_ERR_0	ESM0_PLS_IN_628	ESM0	GIC0 bus error interrupt	Pulse
	COMPUTE_CLUSTER0_GIC500SS_ECC_FATAL_0	ESM0_PLS_IN_629	ESM0	GIC0 uncorrectable ECC error interrupt	Pulse
	COMPUTE_CLUSTER0_GIC_OUTPUT_WAKER_GIC_PWR0_WAKE_REQUEST_0	WKUP_DMSC0_INTR_IN_56 MCU_R5FSS0_CORE0_INTR_IN_144 MCU_R5FSS0_CORE1_INTR_IN_144 R5FSS0_INTRTR0_IN_331 R5FSS1_INTRTR0_IN_331	WKUP_DMSC0 MCU_R5FSS0_CORE0 MCU_R5FSS0_CORE1 R5FSS0_INTRTR0 R5FSS1_INTRTR0	GIC0 wake request for A72 core 0	Level
	COMPUTE_CLUSTER0_GIC_OUTPUT_WAKER_GIC_PWR0_WAKE_REQUEST_1	WKUP_DMSC0_INTR_IN_57 MCU_R5FSS0_CORE0_INTR_IN_145 MCU_R5FSS0_CORE1_INTR_IN_145 R5FSS0_INTRTR0_IN_332 R5FSS1_INTRTR0_IN_332	WKUP_DMSC0 MCU_R5FSS0_CORE0 MCU_R5FSS0_CORE1 R5FSS0_INTRTR0 R5FSS1_INTRTR0	GIC0 wake request for A72 core 1	Level

### Note

Table 9-5 lists only the GIC interrupt outputs. These GIC interrupts are further described in [Section 9.2.1.3.5, GIC Interrupt Outputs](#).

The mapping of system interrupts to GIC interrupt inputs is presented in [Section 9.4.3.1](#).

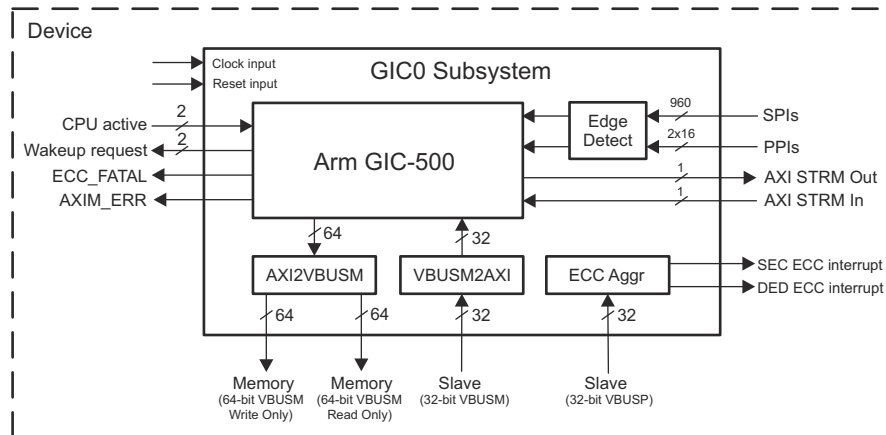
### 9.2.1.3 GIC Functional Description

#### Note

This chapter provides only a brief description of the GIC functionality and features. For more details on this module, refer to the *Arm® CoreLink™ GIC-500 Generic Interrupt Controller Technical Reference Manual*.

#### 9.2.1.3.1 GIC Block Diagram

Figure 9-3 shows the GIC block diagram.



**Figure 9-3. GIC Block Diagram**

#### 9.2.1.3.2 Arm GIC-500

The Arm GIC-500 is responsible for detecting, managing and communicating system interrupts to the two dual-core A72 clusters implemented on the SoC. The main difference between the Arm GIC-500 and previous Arm GIC versions is the direct communication with the CPU(s). Previously, an interrupt controller would set a pending bit to a CPU and that CPU would then have to query an interrupt controller MMR to find out what to do. Now, the Arm GIC-500 sends interrupts to a CPU via a dedicated message interface and the CPU communicates back about interrupts through the same interface. The CPU now uses writes and reads to system registers instead of over the memory interface. This leads to reduced latency and the ability to route interrupts to different CPUs based on a set of rules.

The Arm GIC-500 is divided into three main sections:

- **Distributor:** The Distributor receives interrupts from the wire interrupts and the programming interface. It is responsible for prioritizing these interrupts and sending them to the CPU interface via the GIC Stream Protocol Interface.
- **Redistributor:** There is one Redistributor per core. Each Redistributor holds the state that is individual to a particular core (such as the settings for PPIs and SGIs). It also stores the LPIs for that core after they have been generated using the ITS.
- **ITS:** The ITS is responsible for translating message-based interrupts from device peripherals into LPIs. The user can also use the ITS to manage existing LPIs. Note that the ITS is not used for other types of interrupt

#### 9.2.1.3.3 GIC Interrupt Types

The GIC supports four types of interrupts:

- **Software Generated Interrupts (SGIs):** There are 16 SGIs (ID0-ID15). These are inter-processor interrupts. SGIs can be sent using either Arm system registers or by writing to the Software Generated Interrupt Register (GICD\_SGIR).

- **Private Peripheral Interrupts (PPIs):** There are 16 PPIs (ID16-ID31). These are wired interrupts dedicated to a specific CPU. Many are reserved to specific functions via convention. These can be either active-low levels (by default), or rising edge triggered (software programmable).
- **Shared Peripheral Interrupts (SPIs):** There are 960 SPIs (ID32-ID991). These are wired interrupts that can be routed to any core or cluster, based on the programming of that interrupt in the GIC. These are active-high levels (by default), or rising edge triggered (software programmable).
- **Locality-Specific Peripheral Interrupts (LPIs):** There are 57,344 of these. These interrupts are used for message-based interrupts from a peripheral. They are generated through writes to the GIC slave interface. The information associated with these is located in memory. The GIC keeps a small 64-entries cache on-board in order to reduce latency either for specific interrupts or for most-recently-used. LPIs can be routed to different CPUs based on programmed rules.

The mapping of PPIs and SPIs can be found in section .

#### 9.2.1.3.4 GIC Interfaces

The GIC has the following main interfaces:

- A pair of 64-bit VBUSM write-only and read-only master interfaces
  - AXI2VBUSM bridge converts the standard 64-bit AXI4 master interface into this pair of interfaces
  - The AXI4 master interface allows the GIC ITS and redistributors to access main memory
- 32-bit VBUSM slave interface
  - Provides access to Arm GIC-500 internal resources
  - Handles all message-based interrupts. Message-based interrupts can generate SPIs or LPIs, depending on the register that is written
  - VBUSM2AXI bridge converts this interface into an 32-bit AXI4 slave interface
- 32-bit VBUSP slave interface
  - Provides access to internal ECC aggregator
- Physical interrupt and power signals
  - Inputs: CPU active signals, SPIs and PPIs (can be programmed as either level-sensitive, or edge-triggered)
  - Outputs: Error signals and wake-up requests
- GIC Stream Protocol Interface
  - Consists of a pair of AXI4-Stream interfaces (one upstream and one downstream 16-bit AXI4-Stream interface) that the GIC uses to send interrupts to the core and receive notifications when the core activates interrupts
  - There is a pair of physical interfaces, one in each direction, for each cluster

#### 9.2.1.3.5 GIC Interrupt Outputs

Table 9-6 lists the interrupts that are generated by the GIC.

**Table 9-6. GIC Interrupt Outputs**

Interrupt Name	Description
COMPUTE_CLUSTER0_GIC500SS_AXIM_ERR_0	GIC bus error interrupt. This interrupt is triggered if the GIC receives an error on a bus transaction, such as a decode or protection error. This is a pulse-high interrupt. If this interrupt occurs, the GIC might lose interrupts and must be reset. If it is not reset, behavior becomes unpredictable.
COMPUTE_CLUSTER0_GIC500SS_ECC_FATAL_0	GIC uncorrectable ECC error interrupt. Indicates an uncorrectable 2-bit error detected in one of the ITS cache memories. This is a pulse-high interrupt. If this interrupt occurs, the GIC might lose interrupts and must be reset. If it is not reset, behavior becomes unpredictable.
COMPUTE_CLUSTER0_GIC_OUTPUT_WAKER_GIC_PWR0_WAKE_REQUEST_0 <sup>(1)</sup>	GIC wake requests for A72 cores. Asserted state indicates that an interrupt is pending for a processor that has set the PROCESSORSLEEP bit in the GICR_WAKER register. This bit indicates that the SoC should wake up the indicated CPU so that it can process the interrupt.
COMPUTE_CLUSTER0_GIC_OUTPUT_WAKER_GIC_PWR0_WAKE_REQUEST_1 <sup>(2)</sup>	

(1) Applies to A72SS0\_CORE0



(2) Applies to A72SS0\_CORE1

#### **9.2.1.3.6 GIC ECC Support**

The Arm GIC-500 has built-in SECDED ECC on its memories to protect against errors. The syndrome generation and checking is done internally.

Additionally, the TI GIC wrapper integrates an ECC aggregator (GIC\_ECC\_AGGR) in order to allow errors to be injected for testing purposes. The generic ECC aggregator functionality is described in [Section 12.11.4, ECC Aggregator](#). Note that the GIC\_ECC\_AGGR supports only a subset of this functionality.

#### **9.2.1.3.7 GIC Interrupt Edge Detection**

All interrupt inputs are put through a standard edge-detection logic. This logic can either synchronize levels, or capture the rising edge of pulses. In this way, the GIC is agnostic to the clock frequency of the source, or pulse widths.

#### **9.2.1.3.8 GIC AXI2VBUSM and VBUSM2AXI Bridges**

The GIC uses the AXI4 master interface to allow the ITS and redistributors to access main memory. It is expected that the main memory (attached to a CBASS VBUSM port) holds the following:

- ITS translation tables
- ITS command queue, which is written by software in order to program the ITS
- LPI configuration table, which holds the priorities and enable bits for LPIs
- LPI pending tables, which can hold information on whether each LPI is pending on each core

The AXI2VBUSM bridge converts the standard 64-bit AXI4 master interface into a pair of 64-bit VBUSM write-only and read-only master interfaces. AXI4 has a separate command channel for read and write, and the AXI2VBUSM bridge keeps them separate, so that the system may prioritize or arbitrate the traffic between them as appropriate.

The GIC uses the AXI4 slave interface to provide access to the programming interfaces of all its parts (distributor, redistributors, ITS). The VBUSM2AXI bridge converts the standard 32-bit VBUSM master interface into an 32-bit AXI4 slave interface.

---

#### **Note**

The AXI2VBUSM and VBUSM2AXI bridges do not implement any MMRs.

---

## 9.2.2 Cluster Level Event Controller (CLEC)

### 9.2.2.1 CLEC Overview

The CLEC supports the following features:

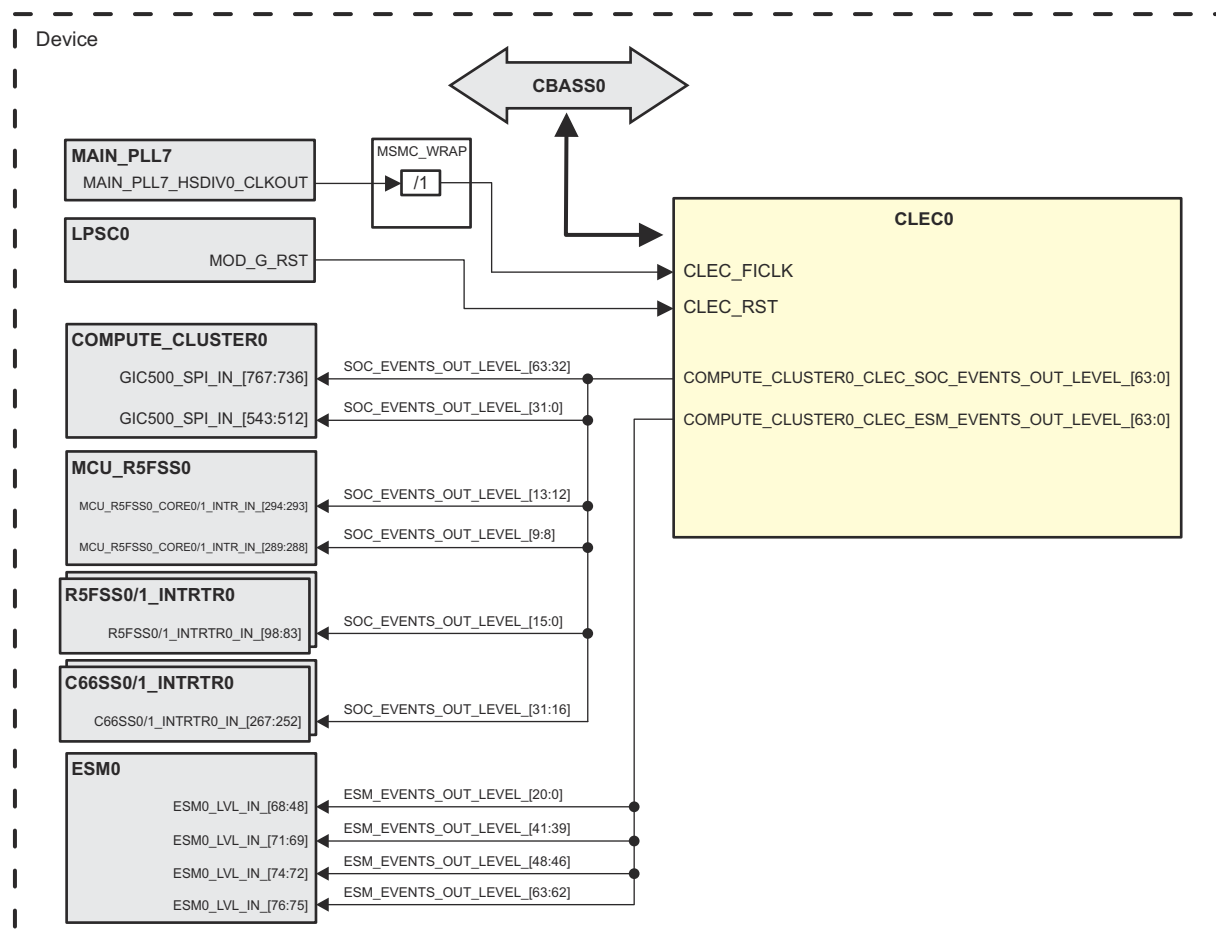
- Supports software scalable event routing mechanism
  - RAM-based interrupt event routing table
- Supports aggregation of incoming interrupt events from:
  - SoC peripherals
  - MSMC
  - DRU
  - C71SS
- Supports distribution of each aggregated interrupt event to one or more of the following:
  - SoC (could be routed to the GIC externally)
  - DRU
  - A72SS
  - C71SS
- Supports originating interrupt events through the following mechanisms:
  - MMR writes to trigger new events
  - Reporting errors detected by the CLEC
- 2048 input events
- 128 output events
- Can handle both level and pulse input interrupts
- Can be used for inter-processor communication (IPC) between A72 and C71x
- Shares SPI (shared peripheral interrupt) input events with GIC

See [Figure 9-1](#) for an overview of CLEC interrupt architecture.

### 9.2.2.2 CLEC Integration

This section describes the CLEC integration in the device, including information about clocks, resets, and hardware requests.

Figure 9-4 shows the CLEC integration.



**Figure 9-4. CLEC Integration**

Table 9-7 through Table 9-9 summarize the CLEC integration.

**Table 9-7. CLEC Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
CLEC	PSC0	PD0	LPSC0	CBASS0

**Table 9-8. CLEC Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
CLEC	CLEC_FICLK	MAIN_PLL7_HSDIV0_CLKOUT	MAIN_PLL7	Module functional and interface clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
CLEC	CLEC_RST	MOD_G_RST	LPSC0	Module hardware reset

**Table 9-9. CLEC Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
CLEC	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_[31:0]	GIC500_SPI_IN_[543:512]	Compute Cluster	CLEC SoC event outputs	Level
	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_[63:32]	GIC500_SPI_IN_[767:736]	Compute Cluster		
	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_[9:8]	MCU_R5FSS0_CORE0_INTR_IN_[289:288] MCU_R5FSS0_CORE1_INTR_IN_[289:288]	MCU_R5FSS0_CORE0 MCU_R5FSS0_CORE1		
	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_[13:12]	MCU_R5FSS0_CORE0_INTR_IN_[294:293] MCU_R5FSS0_CORE1_INTR_IN_[294:293]	MCU_R5FSS0_CORE0 MCU_R5FSS0_CORE1		
	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_[15:0]	R5FSS0_INTRTR0_IN_[98:83] R5FSS1_INTRTR0_IN_[98:83]	R5FSS0_INTRTR0 R5FSS1_INTRTR0		
	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_[31:16]	C66SS0_INTRTR0_IN_[267:252] C66SS1_INTRTR0_IN_[267:252]	C66SS0_INTRTR0 C66SS1_INTRTR0		

**Table 9-9. CLEC Hardware Requests (continued)**

		ESM0	CLEC ESM event outputs	Level
COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_0	ESM0_LVL_IN_48			
COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_1	ESM0_LVL_IN_49			
COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_2	ESM0_LVL_IN_50			
COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_3	ESM0_LVL_IN_51			
COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_4	ESM0_LVL_IN_52			
COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_5	ESM0_LVL_IN_53			
COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_6	ESM0_LVL_IN_54			
COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_7	ESM0_LVL_IN_55			
COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_8	ESM0_LVL_IN_56			
COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_9	ESM0_LVL_IN_57			
COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_10	ESM0_LVL_IN_58			
COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_11	ESM0_LVL_IN_59			
COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_12	ESM0_LVL_IN_60			
COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_13	ESM0_LVL_IN_61			
COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_14	ESM0_LVL_IN_62			
COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_15	ESM0_LVL_IN_63			
COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_16	ESM0_LVL_IN_64			
COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_17	ESM0_LVL_IN_65			
COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_18	ESM0_LVL_IN_66			
COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_19	ESM0_LVL_IN_67			
COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_20	ESM0_LVL_IN_68			
COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_39	ESM0_LVL_IN_69			
COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_40	ESM0_LVL_IN_70			
COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_41	ESM0_LVL_IN_71			
COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_46	ESM0_LVL_IN_72			
COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_47	ESM0_LVL_IN_73			
COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_48	ESM0_LVL_IN_74			
COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_62	ESM0_LVL_IN_75			
COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_63	ESM0_LVL_IN_76			

**Note**

The COMPUTE\_CLUSTER0\_CLEC\_SOC\_EVENTS\_OUT\_LEVEL\_[128:64] outputs are not used in this device.

---

**Note**

[Table 9-9](#) lists only the CLEC interrupt outputs. The mapping of interrupt sources to CLEC interrupt inputs is presented in [Section 9.2.2.3.6.2](#).

---

### 9.2.2.3 CLEC Functional Description

#### 9.2.2.3.1 CLEC Interrupt Event Routing

The CLEC routes events according to a RAM-based interrupt event routing table. This table holds one entry for each supported external interrupt/event number, plus some additional entries for CLEC internal events. External in this context means "external to the CLEC", and corresponds to all interrupt events arriving at the CLEC from other sources. Internal refers to events that the CLEC generates internally, or in response to an MMR write that triggers the event.

The routing table memory has 2048 entries in size. Each routing table entry contains the following details:

- Outgoing event number: Each incoming event can be mapped to any outgoing event number
- Secure / Non-secure: Each incoming event can be claimed as a secure event
- Destination port configuration: This configuration allows for both single-cast and broadcast events

For each incoming interrupt event, the CLEC does the following processing:

- Validates the interrupt event number. For out-of-range event numbers, the CLEC triggers an "invalid event number" event
- Non-secure read/write to registers associated with a secure event will generate a privilege error
- Looks up the event in the routing table
- For each port in the destination bitmap that has a bit set, it queues the event to that destination
  - Queuing to multiple destinations happens in parallel, not sequentially, to minimize latency
  - If no bits are set, the event is dropped silently, but an interrupt dropped event is asserted. This event gets routed as determined by the routing table

#### Note

For events mapped to DRU and ARM, only event mapping number is looked at and they are not qualified by CLEC\_MRR\_j[21-16] RTMAP. Hence, this bit field is "don't care" for events that have CLEC\_MRR\_j[15-8] EXT\_EVNUM set to 128 or higher.

Since the routing table is a shared resource, the CLEC only routes one new event per cycle. The routing state machine arbitrates among all incoming event sources round-robin, with the exception that internally generated error events have priority over all other event sources.

The CLEC routes events to their destinations and also maintains pending interrupt information in an CLEC\_EFR\_k register. Firmware needs to clear this pending interrupt bit in an interrupt service routine (ISR).

#### 9.2.2.3.2 CLEC Virtualization, Isolation and Access Control

The CLEC provides direct support for virtual-machine isolation via a hypervisor. Each event's register window is deliberately spaced 64KB apart. This enables allocating individual event numbers to virtual machines via stage-2 translation. The 64KB spacing allows this to work even with the largest translation granule supported by ARM (64KB).

Individual functions associated with an event within each event's window are spaced 4KB apart. This allows a standard operating system to open up, for instance, write access to an CLEC\_ESR\_j register for a single task, while blocking access to the event routing details in CLEC\_MRR\_j register.

The CLEC implements event lock registers (CLEC\_GELRS / CLEC\_GELRNS) that allow hypervisor to lock down the configuration of events, while still permitting write access to the event send register (CLEC\_ESR\_j) for that event.

In current implementation, the CLEC provides basic support for isolating virtual machines by partitioning incoming events in the address map. However, it does not provide any support for limiting event numbers assigned for outgoing events in CLEC\_MRR\_j[15-8] EXT\_EVNUM. If such isolation is necessary, software can implement it in the hypervisor via para-virtualization calls, via trap-and-emulate for full virtualization.

### 9.2.2.3.3 CLEC Memory Protection

The CLEC implements minimal memory protection checks. It relies on protection mechanisms outside to control accesses from various masters. Chip-level firewalls control which masters may access the CLEC. MMUs provide finer-grain control for programmable processors that access the CLEC.

Specifically, the CLEC returns the following errors:

- Address error on a read or write to unimplemented space
- Address error on a read to a write-only register
- Address error on a write to a read-only register
- Privilege error on a non-secure read or write to registers associated with a secure event (CLEC\_MRR\_j[31] S = 1 for that event)
- Privilege error on a write to CLEC\_ESR\_j on an event whose CLEC\_MRR\_j[30] ESE = 0
- Privilege error on a write to CLEC\_MRR\_j on an event when CLEC\_GELRS[0] LOCK = 1

Any further access control must be implemented with firewalls and/or MMU configuration outside the CLEC.

### 9.2.2.3.4 CLEC ECC Support

Standard ECC aggregator strategy is used to implement ECC protection on the SRAM used to implement the CLEC routing table. The CLEC has two ports connected to the ECC\_AGGR at the MSMC\_WRAP level. For RAM\_ID values of CLEC ECC-protected memories, see [Section 6.1.2.3, Compute Cluster ECC Aggregators](#). For details on the generic ECC\_AGGR functionality, see [Section 12.11.4, ECC Aggregator](#).

### 9.2.2.3.5 CLEC Intra-Core Communication

In order to support intra-core synchronization inside Compute Cluster, associated C71/A72 events are routed through the CLEC.

### 9.2.2.3.6 CLEC Event Maps

#### 9.2.2.3.6.1 CLEC Output Event Routing

CLEC output events can be routed to SoC, DRU or C71/A72 corepacs. For Soc, DRU, and ARM events, the exact event number mapping is controlled by the CLEC\_MRR\_j[15-8] EXT\_EVNUM bit field. For C71 events, the exact event number mapping is controlled by the CLEC\_MRR\_j[5-0] C7X\_EVNUM bit field.

[Table 9-10](#) shows the CLEC output event map.

**Table 9-10. CLEC Output Event Map**

Destination	Event Number	Register Setting	Triggered Event
SoC	127-0	CLEC_MRR_j[15-8] EXT_EVNUM	SoC output events
DRU	159-128	CLEC_MRR_j[15-8] EXT_EVNUM	DRU local input trigger events
Reserved	191-160	Reserved	Reserved
A72 corepac	192	CLEC_MRR_j[15-8] EXT_EVNUM	A72 event for intra-core synchronization with C71
Reserved	255-193	Reserved	Reserved
C71 corepac	63-0	CLEC_MRR_j[5-0] C7x_EVNUM	C71SS events

#### 9.2.2.3.6.2 CLEC Input Event Map

[Table 9-11](#) shows the events that are mapped to CLEC event inputs. Any of these events can be routed to any of the CLEC event outputs.

**Table 9-11. CLEC Input Event Map**

Interrupt Input Line	Interrupt Name	Description
CLEC_INTR_IN_0	Reserved	Reserved
CLEC_INTR_IN_1	CLEC_ADDR_ERR	CLEC address error
CLEC_INTR_IN_2	CLEC_PRIV_ERR	CLEC privilege error
CLEC_INTR_IN_3	CLEC_INVALID_EVT	CLEC invalid event



**Table 9-11. CLEC Input Event Map (continued)**

Interrupt Input Line	Interrupt Name	Description
CLEC_INTR_IN_4	CLEC_DROPPED_EVT	CLEC dropped event
CLEC_INTR_IN_5 - CLEC_INTR_IN_7	Reserved	Reserved
CLEC_INTR_IN_8	ARM0_DFT_PBIST_CPU_ERROR_INTR	A72 DFT PBIST CPU error interrupt
CLEC_INTR_IN_9 - CLEC_INTR_IN_11	Reserved	Reserved
CLEC_INTR_IN_12	C7X_DFT_PBIST_CPU_ERROR_INTR	C71x DFT PBIST CPU error interrupt
CLEC_INTR_IN_13 - CLEC_INTR_IN_15	Reserved	Reserved
CLEC_INTR_IN_16	MSMC_CTSET_INTR	MSMC CT-SET (debug) interrupt
CLEC_INTR_IN_17	DRU_PROTOCOL_ERROR	DRU protocol error
CLEC_INTR_IN_18 - CLEC_INTR_IN_23	Reserved	Reserved
CLEC_INTR_IN_24	MSMC_NULL_SLAVE_ERROR	MSMC null slave error
CLEC_INTR_IN_25 - CLEC_INTR_IN_63	Reserved	Reserved
CLEC_INTR_IN_64 - CLEC_INTR_IN_127	DRU_COMPLETE_EVENT_IN_0 - DRU_COMPLETE_EVENT_IN_63	DRU complete events 0-63
CLEC_INTR_IN_128 - CLEC_INTR_IN_191	DRU_ERROR_EVENT_IN_0 - DRU_ERROR_EVENT_IN_63	DRU error events 0-63
CLEC_INTR_IN_192 - CLEC_INTR_IN_255	DRU_LOCALOUT_EVENT_IN_0 - DRU_LOCALOUT_EVENT_IN_63	DRU local events 0-63
CLEC_INTR_IN_256 - CLEC_INTR_IN_383	Reserved	Reserved
CLEC_INTR_IN_384 - CLEC_INTR_IN_415	C7X_DISCRETE_EVENT_0 - C7X_DISCRETE_EVENT_31	C71x discrete events 0-31
CLEC_INTR_IN_416 - CLEC_INTR_IN_511	Reserved	Reserved
CLEC_INTR_IN_512	ARM0_EVENTO	A72 intra-core (IPC) event
CLEC_INTR_IN_513 - CLEC_INTR_IN_567	Reserved	Reserved
CLEC_INTR_IN_568 - CLEC_INTR_IN_599	SOC_EVENT_IN_0 - SOC_EVENT_IN_31	SoC DSP input events 0-31
CLEC_INTR_IN_600 - CLEC_INTR_IN_1023	Reserved	Reserved
CLEC_INTR_IN_1024 - CLEC_INTR_IN_1983	GIC_SPI_IN_32 - GIC_SPI_IN_975	SoC GIC SPI events 32-975
CLEC_INTR_IN_1984 - CLEC_INTR_IN_2047	Reserved	Reserved

### 9.2.2.3.6.3 CLEC ESM Event Routing

Table 9-12 shows the CLEC ESM event output map.

**Table 9-12. CLEC ESM Event Output Map**

Destination	Source Interrupt	Description
ESM_EVENTS_OUT_LEVEL_0	MSMC_ECCAGGR0_UNCORR_LEVEL_INTR	MSMC ECC_AGGR0 uncorrectable error
ESM_EVENTS_OUT_LEVEL_1	MSMC_ECCAGGR0_CORR_LEVEL_INTR	MSMC ECC_AGGR0 correctable error
ESM_EVENTS_OUT_LEVEL_2	MSMC_ECCAGGR1_UNCORR_LEVEL_INTR	MSMC ECC_AGGR1 uncorrectable error
ESM_EVENTS_OUT_LEVEL_3	MSMC_ECCAGGR1_CORR_LEVEL_INTR	MSMC ECC_AGGR1 correctable error
ESM_EVENTS_OUT_LEVEL_4	MSMC_ECCAGGR2_UNCORR_LEVEL_INTR	MSMC ECC_AGGR2 uncorrectable error
ESM_EVENTS_OUT_LEVEL_5	MSMC_ECCAGGR2_CORR_LEVEL_INTR	MSMC ECC_AGGR2 correctable error

**Table 9-12. CLEC ESM Event Output Map (continued)**

Destination	Source Interrupt	Description
ESM_EVENTS_OUT_LEVEL_6	MSMC_DFT_PBIST_SAFETY_ERROR_INTR	MSMC DFT PBIST internal diagnostic error
ESM_EVENTS_OUT_LEVEL_7	ARM0_ECCAGGR0_UNCORR_LEVEL_INTR	A72 ECC_AGGR0 uncorrectable error
ESM_EVENTS_OUT_LEVEL_8	ARM0_ECCAGGR0_CORR_LEVEL_INTR	A72 ECC_AGGR0 correctable error
ESM_EVENTS_OUT_LEVEL_9	ARM0_ECCAGGR1_UNCORR_LEVEL_INTR	A72 ECC_AGGR1 uncorrectable error
ESM_EVENTS_OUT_LEVEL_10	ARM0_ECCAGGR1_CORR_LEVEL_INTR	A72 ECC_AGGR1 correctable error
ESM_EVENTS_OUT_LEVEL_11	ARM0_ECCAGGR2_UNCORR_LEVEL_INTR	A72 ECC_AGGR2 uncorrectable error
ESM_EVENTS_OUT_LEVEL_12	ARM0_ECCAGGR2_CORR_LEVEL_INTR	A72 ECC_AGGR2 correctable error
ESM_EVENTS_OUT_LEVEL_13	ARM0_DFT_PBIST_SAFETY_ERROR_INTR	A72 DFT PBIST internal diagnostic error
ESM_EVENTS_OUT_LEVEL_14 - ESM_EVENTS_OUT_LEVEL_38	Reserved	Reserved
ESM_EVENTS_OUT_LEVEL_39	C7X_ECCAGGR_UNCORR_LEVEL_INTR	C7x ECC_AGGR uncorrectable error
ESM_EVENTS_OUT_LEVEL_40	C7X_ECCAGGR_CORR_LEVEL_INTR	C7x ECC_AGGR correctable error
ESM_EVENTS_OUT_LEVEL_41	C7X_DFT_PBIST_SAFETY_ERROR_INTR	C7x DFT PBIST internal diagnostic error
ESM_EVENTS_OUT_LEVEL_42 - ESM_EVENTS_OUT_LEVEL_48	Reserved	Reserved
ESM_EVENTS_OUT_LEVEL_49	DDRSS0_CFG_ECCAGGR_UNCORR_LVL_INTR	DDRSS0 cfg ECC_AGGR uncorrectable error
ESM_EVENTS_OUT_LEVEL_50	DDRSS0_CFG_ECCAGGR_CORR_LVL_INTR	DDRSS0 cfg ECC_AGGR correctable error
ESM_EVENTS_OUT_LEVEL_51	DDRSS0_CTL_ECCAGGR_UNCORR_LVL_INTR	DDRSS0 ctl ECC_AGGR uncorrectable error
ESM_EVENTS_OUT_LEVEL_52	DDRSS0_CTL_ECCAGGR_CORR_LVL_INTR	DDRSS0 ctl ECC_AGGR correctable error
ESM_EVENTS_OUT_LEVEL_53	DDRSS0_VBUS_ECC_GGR_UNCORR_LVL_INTR	DDRSS0 vbus ECC_AGGR uncorrectable error
ESM_EVENTS_OUT_LEVEL_54	DDRSS0_VBUS_ECCAGGR_CORR_LVL_INTR	DDRSS0 vbus ECC_AGGR correctable error
ESM_EVENTS_OUT_LEVEL_55 - ESM_EVENTS_OUT_LEVEL_61	Reserved	Reserved
ESM_EVENTS_OUT_LEVEL_62	GIC0_ECCAGGR_UNCORR_LEVEL_INTR	GIC0 ECC_AGGR uncorrectable error
ESM_EVENTS_OUT_LEVEL_63	GIC0_ECCAGGR_CORR_LEVEL_INTR	GIC0 ECC_AGGR correctable error

#### 9.2.2.3.6.4 CLEC C7x DSP Input Event Map

Table 9-13 shows the mapping of C7x discrete events (including IPC) to associated CLEC inputs.

**Table 9-13. CLEC C7x DSP Input Event Map**

Interrupt Line	Interrupt Name	Description
C7X_DISCRETE_EVENT_0	C7X_IPC_DROP_EVT	C7x IPC dropped event
C7X_DISCRETE_EVENT_1	C7X_MEMSYS_INTR	C7x memsys interrupt
C7X_DISCRETE_EVENT_2	C7X_DBG_CTM_INTR	C7x CTM (debug) interrupt
C7X_DISCRETE_EVENT_3	C7X_DBG_AET_INTR	C7x AET (debug) interrupt
C7X_DISCRETE_EVENT_4 - C7X_DISCRETE_EVENT_15	Reserved	Reserved
C7X_DISCRETE_EVENT_16 - C7X_DISCRETE_EVENT_31	C7X_IPC_EVENT_0 - C7X_IPC_EVENT_15	C7x IPC events 0-15

### 9.2.3 Other Interrupt Controllers

For details on R5FSS vectored interrupt manager (VIM), see [Section 6.3.3.6](#), *R5FSS Vectored Interrupt Manager (VIM)*.

For details on DMSC nested vectored interrupt controller (NVIC), see *TI Device Management Security Controller and Power Management Addendum*, and *Arm Cortex-M3 Technical Reference Manual*.

For details on PRU-ICSSG local interrupt controller (INTC), see *PRU\_ICSSG Local INTC*.

## 9.3 Interrupt Routers

### 9.3.1 INTRTR Overview

The interrupt router (INTRTR) module provides a mechanism to mux  $M$  interrupt inputs to  $N$  interrupt outputs, where all  $M$  inputs are selectable to be driven per  $N$  output. There is one register per output (MUXCNTL\_N) that controls the selection.

There are several INTRTR modules in the device. Their purpose is described in [Section 9.1, Interrupt Architecture](#). [Table 9-14](#) summarizes the configuration details for the various interrupt routers.

**Table 9-14. INTRTR Modules Configuration**

Module	Number of Inputs	Number of Outputs	Input Interrupt Type
WKUP_GPIOMUX_INTRTR0	120	32	Pulse
GPIOMUX_INTRTR0	304	64	Pulse
MAIN2MCU_LVL_INTRTR0	320	64	Level
MAIN2MCU_PLS_INTRTR0	104	48	Pulse
R5FSS0_INTRTR0	432	256	Level
R5FSS1_INTRTR0	432	256	Level
C66SS0_INTRTR0	400	97	Level
C66SS1_INTRTR0	400	97	Level

The user should take the following into account when programming the MUXCNTL\_N register:

- Avoid programming this register when input interrupts are active. This could lead to spurious asynchronous output toggles which may lead to unpredictable behavior.

The recommended general programming sequence is as follows:

1. Disable interrupt by writing '0' to the INT\_ENABLE bit field. Do not change mux control configuration settings (ENABLE bit field) at this time.
2. Change the mux control configuration settings. INT\_ENABLE needs to remain '0' at this time.
3. Enable interrupt by writing '1' to INT\_ENABLE. Do not change mux control configuration settings at this time.

### **9.3.2 INTRTR Integration**

This section describes the INTRTR integration in the device, including information about clocks, resets, and hardware requests.

### 9.3.2.1 WKUP\_GPIOMUX\_INTRTR0 Integration

Figure 9-5 shows the WKUP\_GPIOMUX\_INTRTR0 integration.

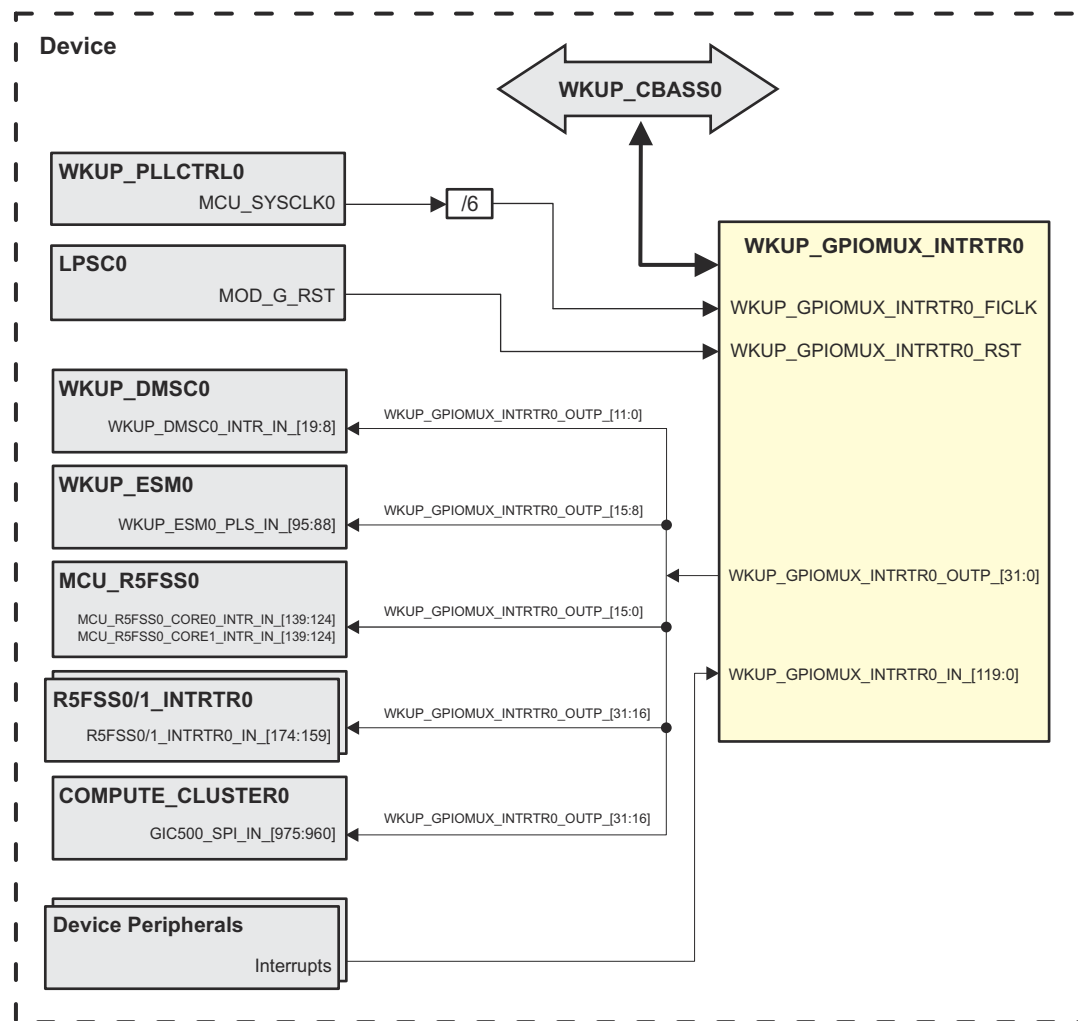


Figure 9-5. WKUP\_GPIOMUX\_INTRTR0 Integration

Table 9-15 through Table 9-17 summarize the WKUP\_GPIOMUX\_INTRTR0 integration.

**Table 9-15. WKUP\_GPIOMUX\_INTRTR0 Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
WKUP_GPIOMUX_INTRTR0	WKUP_PSC0	PD0	LPSC0	WKUP_CBASS0

**Table 9-16. WKUP\_GPIOMUX\_INTRTR0 Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
WKUP_GPIOMUX_INTRTR0	WKUP_GPIOMUX_INTRTR0_FICLK	MCU_SYSCCLK0/6	WKUP_PLLCTRL0	Module functional and interface clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
WKUP_GPIOMUX_INTRTR0	WKUP_GPIOMUX_INTRTR0_RST	MOD_G_RST	LPSC0	Module hardware reset

**Table 9-17. WKUP\_GPIOMUX\_INTRTR0 Hardware Requests**

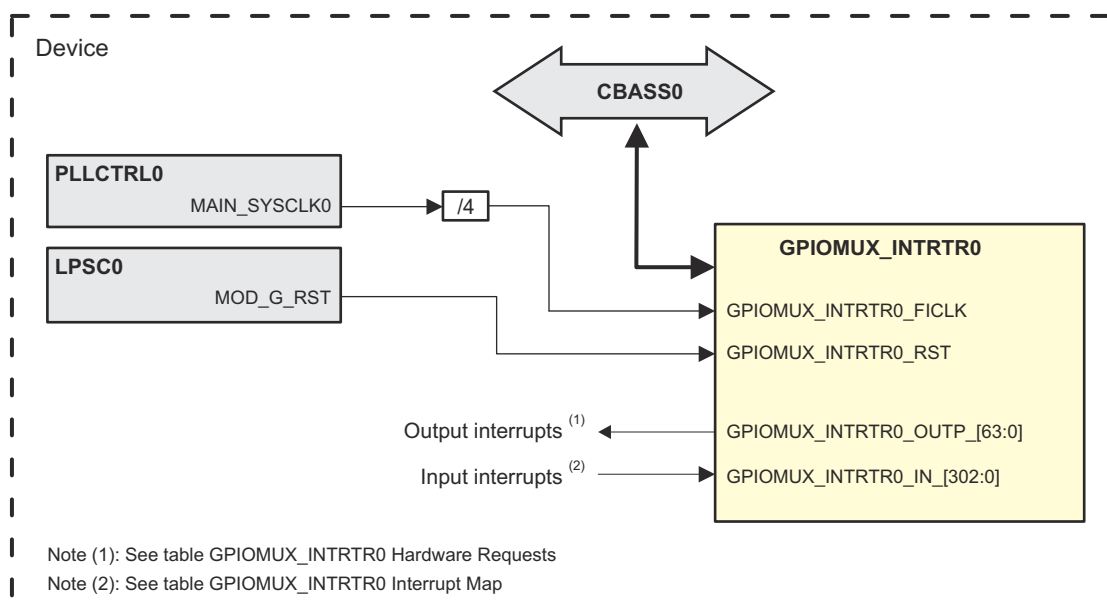
Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
WKUP_GPIOMUX_INTRTR0	WKUP_GPIOMUX_INTRTR0_OUTP_[31:16]	GIC500_SPI_IN_[975:960] R5FSS0_INTRTR0_IN_[174:159] R5FSS1_INTRTR0_IN_[174:159]	COMPUTE_CLUSTER0 R5FSS0_INTRTR0 R5FSS1_INTRTR0	Module interrupt outputs [31:0]	Pulse
	WKUP_GPIOMUX_INTRTR0_OUTP_[15:8]	WKUP_ESM0_PLS_IN_[95:88]	WKUP_ESM0		
	WKUP_GPIOMUX_INTRTR0_OUTP_[15:0]	MCU_R5FSS0_CORE0_INTR_IN_[139:124] MCU_R5FSS0_CORE1_INTR_IN_[139:124]	MCU_R5FSS0_CORE0 MCU_R5FSS0_CORE1		
	WKUP_GPIOMUX_INTRTR0_OUTP_[11:0]	WKUP_DMSC0_INTR_IN_[19:8]	WKUP_DMSC0		

### Note

Table 9-17 lists only the WKUP\_GPIOMUX\_INTRTR0 interrupt outputs. The mapping of interrupt sources to WKUP\_GPIOMUX\_INTRTR0 interrupt inputs is presented in [Section 9.4.1.2](#).

### 9.3.2.2 GPIOMUX\_INTRTR0 Integration

Figure 9-6 shows the GPIOMUX\_INTRTR0 integration.



**Figure 9-6. GPIOMUX\_INTRTR0 Integration**

Table 9-18 through Table 9-20 summarize the GPIOMUX\_INTRTR0 integration.

**Table 9-18. GPIOMUX\_INTRTR0 Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
GPIOMUX_INTRTR0	PSC0	PD0	LPSC0	CBASS0

**Table 9-19. GPIOMUX\_INTRTR0 Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
GPIOMUX_INTRTR0	GPIOMUX_INTRTR0_FICLK	MAIN_SYCLK0/4	PLLCTRL0	Module functional and interface clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
GPIOMUX_INTRTR0	GPIOMUX_INTRTR0_RST	MOD_G_RST	LPSC0	Module hardware reset

**Table 9-20. GPIOMUX\_INTRTR0 Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
GPIOMUX_INTRTR0	GPIOMUX_INTRTR0_OUTP_[63:8]	GIC500_SPI_IN_[447:392]	COMPUTE_CLUSTER0	Module interrupt outputs	Pulse
	GPIOMUX_INTRTR0_OUTP_[31:16]	R5FSS0_CORE0_INTR_IN_[191:176]	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_[191:176]	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_[191:176]	R5FSS1_CORE0		



**Table 9-20. GPIOMUX\_INTRTR0 Hardware Requests (continued)**

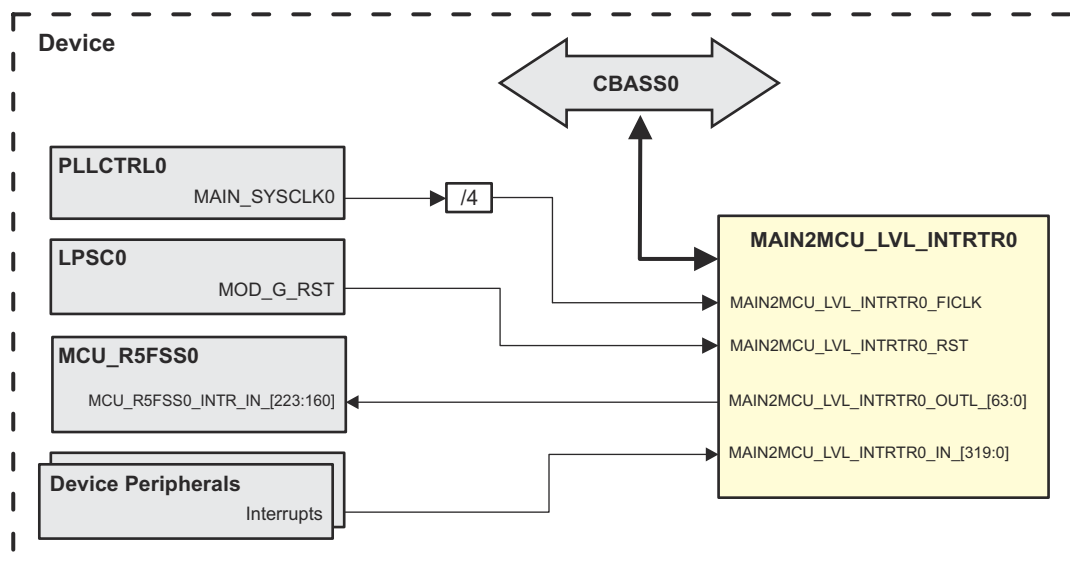
	R5FSS1_CORE1_INTR_IN_[191:176]	R5FSS1_CORE1
GPIOMUX_INTRTR0_OUTP_[15:0]	R5FSS0_INTRTR0_IN_[82:67]	R5FSS0_INTRTR0
	R5FSS1_INTRTR0_IN_[82:67]	R5FSS1_INTRTR0
GPIOMUX_INTRTR0_OUTP_[63:40]	C66SS0_INTRTR0_IN_[150:127]	C66SS0_INTRTR0
	C66SS1_INTRTR0_IN_[150:127]	C66SS1_INTRTR0
GPIOMUX_INTRTR0_OUTP_[39:32]	C66SS0_INTRTR0_IN_[397:390]	C66SS0_INTRTR0
	C66SS1_INTRTR0_IN_[397:390]	C66SS1_INTRTR0
GPIOMUX_INTRTR0_OUTP_[57:52]	PRU_ICSSG0_PR1_IEP0_CAP_IN_TR_REQ[5:0]	PRU_ICSSG0_PR1_IEP0
	PRU_ICSSG1_PR1_IEP0_CAP_IN_TR_REQ[5:0]	PRU_ICSSG1_PR1_IEP0
GPIOMUX_INTRTR0_OUTP_[63:58]	PRU_ICSSG0_PR1_IEP1_CAP_IN_TR_REQ[5:0]	PRU_ICSSG0_PR1_IEP1
	PRU_ICSSG1_PR1_IEP1_CAP_IN_TR_REQ[5:0]	PRU_ICSSG1_PR1_IEP1
GPIOMUX_INTRTR0_OUTP_[31:0]	MAIN2MCU_PLS_INTRTR0_IN_[94:63]	MAIN2MCU_PLS_INT_RTR0
GPIOMUX_INTRTR0_OUTP_[7:0]	ESM0_PLS_IN_[639:632]	ESM0

#### Note

Table 9-20 lists only the GPIOMUX\_INTRTR0 interrupt outputs. The mapping of interrupt sources to GPIOMUX\_INTRTR0 interrupt inputs is presented in [Section 9.4.3.16](#).

### 9.3.2.3 MAIN2MCU\_LVL\_INTRTR0 Integration

Figure 9-7 shows the MAIN2MCU\_LVL\_INTRTR0 integration.



**Figure 9-7. MAIN2MCU\_LVL\_INTRTR0 Integration (TBD)**

Table 9-21 through Table 9-23 summarize the MAIN2MCU\_LVL\_INTRTR0 integration.

**Table 9-21. MAIN2MCU\_LVL\_INTRTR0 Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
MAIN2MCU_LVL_INTRTR0	PSC0	PD0	LPSC0	CBASS0

**Table 9-22. MAIN2MCU\_LVL\_INTRTR0 Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
MAIN2MCU_LVL_INTRTR0	MAIN2MCU_LVL_INTRTR0_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	Module functional and interface clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
MAIN2MCU_LVL_INTRTR0	MAIN2MCU_LVL_INTRTR0_RST	MOD_G_RST	LPSC0	Module hardware reset

**Table 9-23. MAIN2MCU\_LVL\_INTRTR0 Hardware Requests**

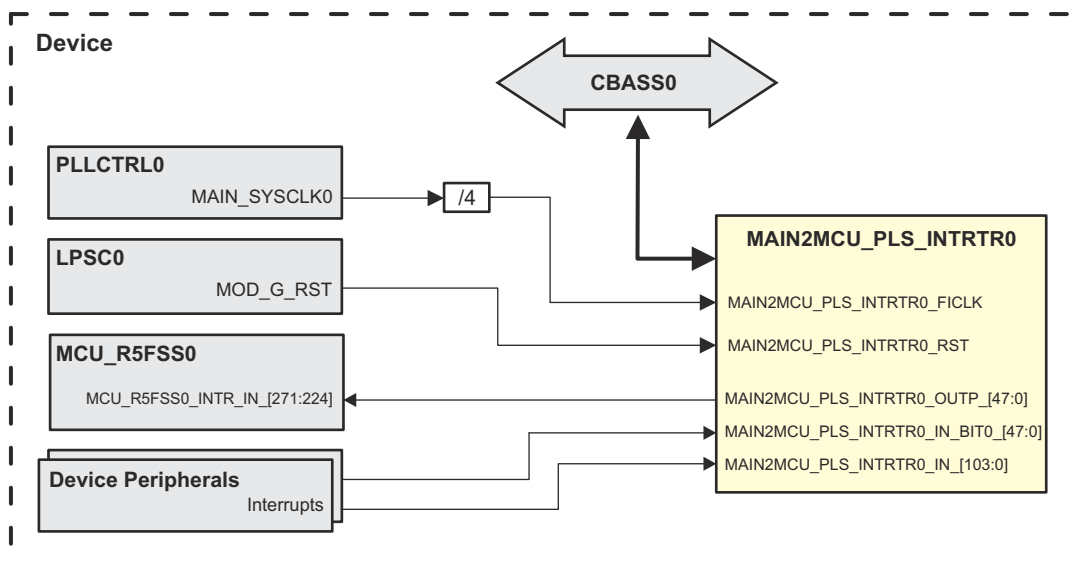
Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MAIN2MCU_LVL_INTRTR0	MAIN2MCU_LVL_INTRTR0_OUTL_[63:0]	MCU_R5FSS0_INTR_IN_[23:160]	MCU_R5FSS0	Module interrupt outputs [63:0]	Level

#### Note

Table 9-23 lists only the MAIN2MCU\_LVL\_INTRTR0 interrupt outputs. The mapping of interrupt sources to MAIN2MCU\_LVL\_INTRTR0 interrupt inputs is presented in Section 9.4.3.14.

### 9.3.2.4 MAIN2MCU\_PLS\_INTRTR0 Integration

Figure 9-8 shows the MAIN2MCU\_PLS\_INTRTR0 integration.



**Figure 9-8. MAIN2MCU\_PLS\_INTRTR0 Integration**

Table 9-24 through Table 9-26 summarize the MAIN2MCU\_PLS\_INTRTR0 integration.

**Table 9-24. MAIN2MCU\_PLS\_INTRTR0 Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
MAIN2MCU_PLS_INTRTR0	PSC0	PD0	LPSC0	CBASS0

**Table 9-25. MAIN2MCU\_PLS\_INTRTR0 Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
MAIN2MCU_PLS_INTRTR0	MAIN2MCU_PLS_INTRTR0_FICLK	MAIN_SYSCCLK0/4	PLLCTRL0	Module functional and interface clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
MAIN2MCU_PLS_INTRTR0	MAIN2MCU_PLS_INTRTR0_RST	MOD_G_RST	LPSC0	Module hardware reset

**Table 9-26. MAIN2MCU\_PLS\_INTRTR0 Hardware Requests**

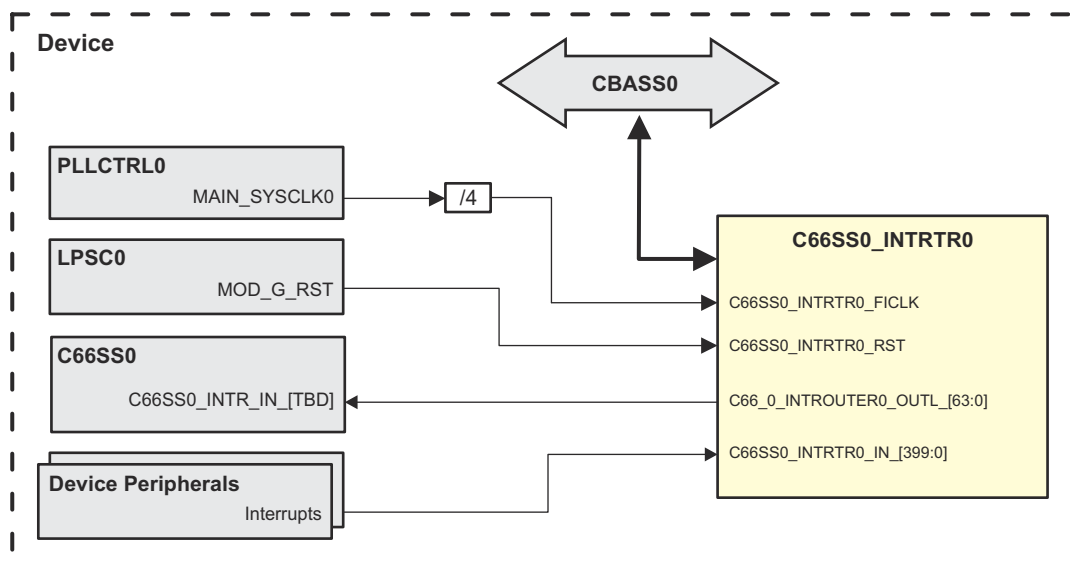
Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MAIN2MCU_PLS_INTRTR0	MAIN2MCU_PLS_INTRTR0_OUTP_[47:0]	MCU_R5FSS0_INTR_IN_[71:224]	MCU_R5FSS0	Module interrupt outputs [47:0]	Pulse

#### Note

Table 9-26 lists only the MAIN2MCU\_PLS\_INTRTR0 interrupt outputs. The mapping of interrupt sources to MAIN2MCU\_PLS\_INTRTR0 interrupt inputs is presented in Section 9.4.3.15.

### 9.3.2.5 C66SS0\_INTRTR0 Integration

Figure 9-9 shows the C66SS0\_INTRTR0 integration.



**Figure 9-9. C66SS0\_INTRTR0 Integration**

Table 9-27 through Table 9-29 summarize the C66SS0\_INTRTR0 integration.

**Table 9-27. C66SS0\_INTRTR0 Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
C66SS0_INTRTR0	PSC0	PD0	LPSC0	CBASS0

**Table 9-28. C66SS0\_INTRTR0 Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
C66SS0_INTRTR0	C66SS0_INTRTR0_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	Module functional and interface clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
C66SS0_INTRTR0	C66SS0_INTRTR0_RST	MOD_G_RST	LPSC0	Module hardware reset

**Table 9-29. C66SS0\_INTRTR0 Hardware Requests**

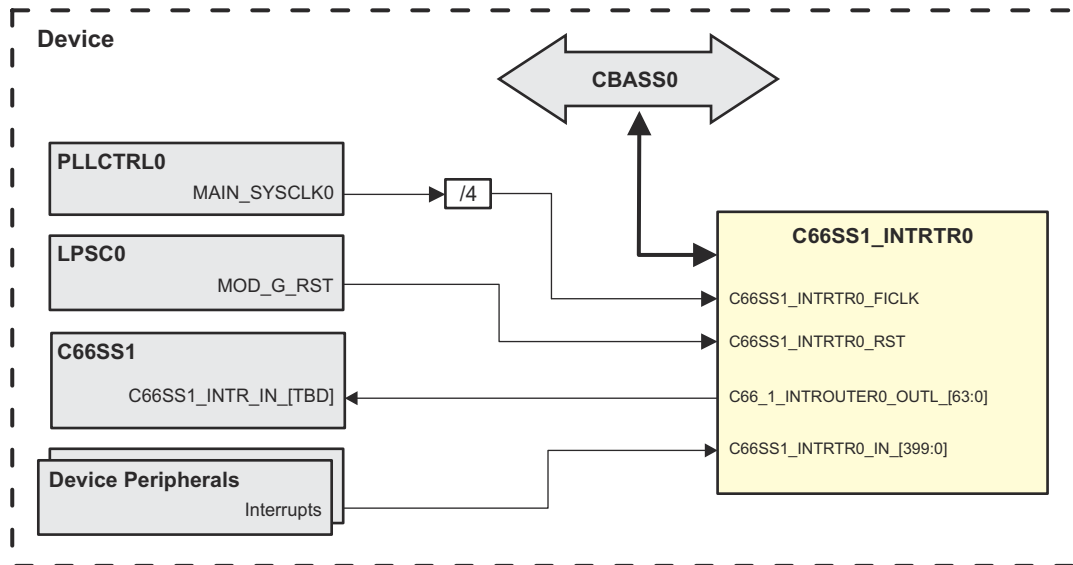
Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
C66SS0_INTRTR0	C66_0_INTROUTER0_OUTL_[96:0]	See <a href="#">Section 9.4</a>	C66SS0	Module interrupt outputs [47:0]	Pulse

#### Note

Table 9-26 lists only the C66SS0\_INTRTR0 interrupt outputs. The mapping of interrupt sources to C66SS0\_INTRTR0 interrupt inputs is presented in [Section 9.4.3.8](#).

### 9.3.2.6 C66SS1\_INTRTR0 Integration

Figure 9-10 shows the C66SS1\_INTRTR0 integration.



**Figure 9-10. C66SS1\_INTRTR0 Integration**

Table 9-30 through Table 9-32 summarize the C66SS1\_INTRTR0 integration.

**Table 9-30. C66SS1\_INTRTR0 Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
C66SS1_INTRTR0	PSC0	PD0	LPSC0	CBASS0

**Table 9-31. C66SS1\_INTRTR0 Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
C66SS1_INTRTR0	C66SS1_INTRTR0_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	Module functional and interface clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
C66SS1_INTRTR0	C66SS1_INTRTR0_RST	MOD_G_RST	LPSC0	Module hardware reset

**Table 9-32. C66SS1\_INTRTR0 Hardware Requests**

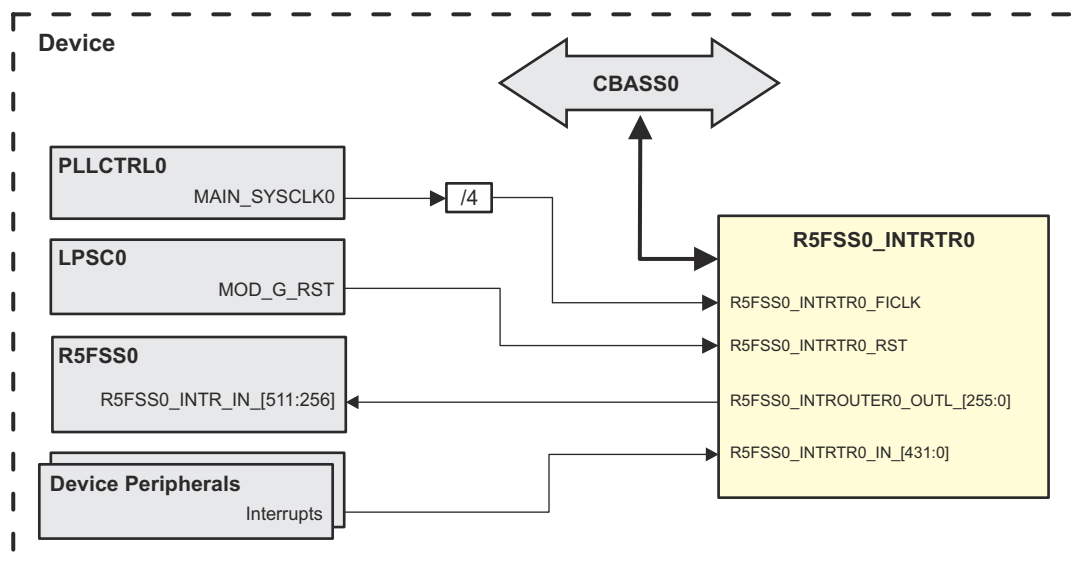
Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
C66SS1_INTRTR0	C66_1_INTROUTER0_OUTL_[96:0]	See <a href="#">Section 9.4</a>	C66SS1	Module interrupt outputs [47:0]	Pulse

#### Note

Table 9-26 lists only the C66SS1\_INTRTR0 interrupt outputs. The mapping of interrupt sources to C66SS1\_INTRTR0 interrupt inputs is presented in [Section 9.4.3.9](#).

### 9.3.2.7 R5FSS0\_INTRTR0 Integration

Figure 9-9 shows the R5FSS0\_INTRTR0 integration.



**Figure 9-11. R5FSS0\_INTRTR0 Integration**

Table 9-33 through Table 9-35 summarize the R5FSS0\_INTRTR0 integration.

**Table 9-33. R5FSS0\_INTRTR0 Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
R5FSS0_INTRTR0	PSC0	PD0	LPSC0	CBASS0

**Table 9-34. R5FSS0\_INTRTR0 Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
R5FSS0_INTRTR0	R5FSS0_INTRTR0_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	Module functional and interface clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
R5FSS0_INTRTR0	R5FSS0_INTRTR0_RST	MOD_G_RST	LPSC0	Module hardware reset

**Table 9-35. R5FSS0\_INTRTR0 Hardware Requests**

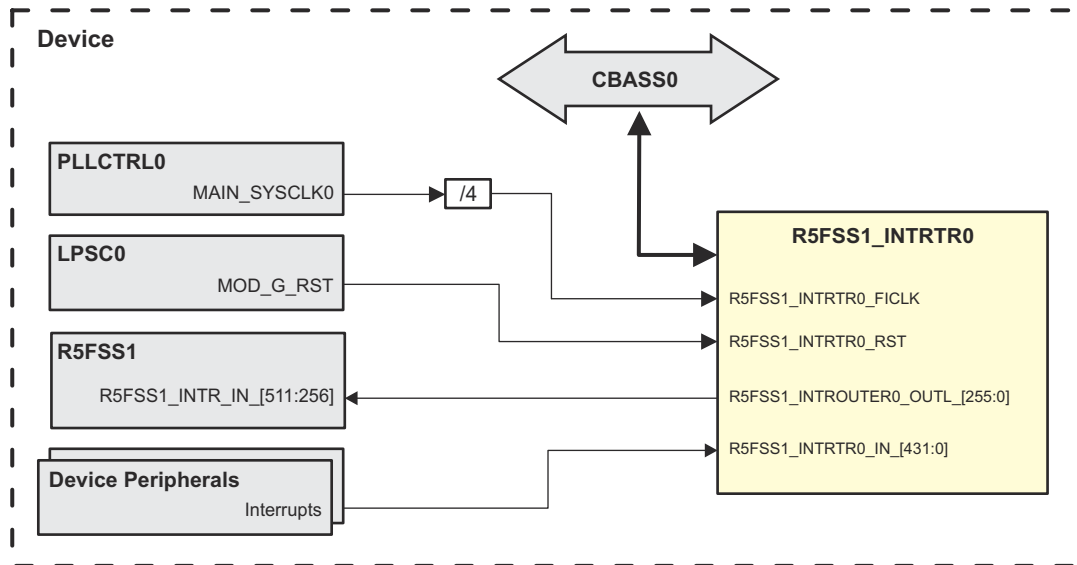
Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
R5FSS0_INTRTR0	R5FSS0_INTROUTER0_OUTL_[255:0]	R5FSS0_INTR_IN_[511:256]	R5FSS0	Module interrupt outputs [255:0]	Pulse

#### Note

Table 9-26 lists only the R5FSS0\_INTRTR0 interrupt outputs. The mapping of interrupt sources to R5FSS0\_INTRTR0 interrupt inputs is presented in .

### 9.3.2.8 R5FSS1\_INTRTR0 Integration

Figure 9-12 shows the R5FSS1\_INTRTR0 integration.



**Figure 9-12. R5FSS1\_INTRTR0 Integration**

Table 9-36 through Table 9-38 summarize the R5FSS1\_INTRTR0 integration.

**Table 9-36. R5FSS1\_INTRTR0 Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
R5FSS1_INTRTR0	PSC0	PD0	LPSC0	CBASS0

**Table 9-37. R5FSS1\_INTRTR0 Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
R5FSS1_INTRTR0	R5FSS1_INTRTR0_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	Module functional and interface clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
R5FSS1_INTRTR0	R5FSS1_INTRTR0_RST	MOD_G_RST	LPSC0	Module hardware reset

**Table 9-38. R5FSS1\_INTRTR0 Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
R5FSS1_INTRTR0	R5FSS1_INTROUTER0_OUTL_[255:0]	R5FSS1_INTR_IN_[511:256]	R5FSS1	Module interrupt outputs [255:0]	Pulse

#### Note

Table 9-26 lists only the R5FSS1\_INTRTR0 interrupt outputs. The mapping of interrupt sources to R5FSS1\_INTRTR0 interrupt inputs is presented in .

## 9.4 Interrupt Sources

### 9.4.1 WKUP Domain Interrupt Maps

#### 9.4.1.1 WKUP\_DMSC0 Interrupt Map

Table 9-39 shows the mapping of events to the WKUP\_DMSC0 external interrupt inputs. Note that the WKUP\_DMSC0 interrupt controller (NVIC) includes WKUP\_DMSC0 internal inputs as well, so external interrupt input 0 actually represents NVIC event 176.

**Table 9-39. WKUP\_DMSC0 Interrupt Map**

Interrupt Input Line	Interrupt ID	Interrupt Name
WKUP_DMSC0_INTR_IN_0	176	WKUP_PORZ_SYNC0_MAIN_PORZ_LATCHED_0
WKUP_DMSC0_INTR_IN_2	178	WKUP_RESETZ_SYNC0_MAIN_RESETZ_LATCHED_0
WKUP_DMSC0_INTR_IN_3	179	GLUELOGIC_FW_CBASS_INTR_OR_GLUE_FW_CBASS_AGG_ERR_INTR_0
WKUP_DMSC0_INTR_IN_4	180	WKUP_I2C0_POINTRPEND_0
WKUP_DMSC0_INTR_IN_5	181	WKUP_I2C0_CLKSTOP_WAKEUP_0
WKUP_DMSC0_INTR_IN_6	182	WKUP_UART0_USART_IRQ_0
WKUP_DMSC0_INTR_IN_7	183	WKUP_UART0_CLKSTOP_WAKEUP_0
WKUP_DMSC0_INTR_IN_8	184	WKUP_GPIOMUX_INTRTR0_OUTP_0
WKUP_DMSC0_INTR_IN_9	185	WKUP_GPIOMUX_INTRTR0_OUTP_1
WKUP_DMSC0_INTR_IN_10	186	WKUP_GPIOMUX_INTRTR0_OUTP_2
WKUP_DMSC0_INTR_IN_11	187	WKUP_GPIOMUX_INTRTR0_OUTP_3
WKUP_DMSC0_INTR_IN_12	188	WKUP_GPIOMUX_INTRTR0_OUTP_4
WKUP_DMSC0_INTR_IN_13	189	WKUP_GPIOMUX_INTRTR0_OUTP_5
WKUP_DMSC0_INTR_IN_14	190	WKUP_GPIOMUX_INTRTR0_OUTP_6
WKUP_DMSC0_INTR_IN_15	191	WKUP_GPIOMUX_INTRTR0_OUTP_7
WKUP_DMSC0_INTR_IN_16	192	WKUP_GPIOMUX_INTRTR0_OUTP_8
WKUP_DMSC0_INTR_IN_17	193	WKUP_GPIOMUX_INTRTR0_OUTP_9
WKUP_DMSC0_INTR_IN_18	194	WKUP_GPIOMUX_INTRTR0_OUTP_10
WKUP_DMSC0_INTR_IN_19	195	WKUP_GPIOMUX_INTRTR0_OUTP_11
WKUP_DMSC0_INTR_IN_20	196	WKUP_VTM0_THERM_LVL_GT_TH1_INTR_0
WKUP_DMSC0_INTR_IN_21	197	WKUP_VTM0_THERM_LVL_LT_TH0_INTR_0
WKUP_DMSC0_INTR_IN_22	198	WKUP_VTM0_THERM_LVL_GT_TH2_INTR_0
WKUP_DMSC0_INTR_IN_23	199	WKUP_ESM0_ESM_INT_CFG_LVL_0
WKUP_DMSC0_INTR_IN_24	200	WKUP_ESM0_ESM_INT_LOW_LVL_0
WKUP_DMSC0_INTR_IN_25	201	WKUP_ESM0_ESM_INT_HI_LVL_0
WKUP_DMSC0_INTR_IN_26	202	WKUP_CTRL_MMR0_ACCESS_ERR_0
WKUP_DMSC0_INTR_IN_27	203	WKUP_PSC0_PSC_ALLINT_0
WKUP_DMSC0_INTR_IN_28	204	WKUP_GPIO0_GPIO_LVL_0
WKUP_DMSC0_INTR_IN_29	205	WKUP_GPIO1_GPIO_LVL_0
WKUP_DMSC0_INTR_IN_31	207	WKUP_DMSC0_RAT_0_EXP_INTR_0
WKUP_DMSC0_INTR_IN_36	212	MCU_PBI01_DFT_PBI01_CPU_0
WKUP_DMSC0_INTR_IN_37	213	GLUELOGIC_NONFW_CBASS_INTR_OR_GLUE_NONFW_CBASS_AGG_ERR_INTR_0
WKUP_DMSC0_INTR_IN_38	214	MCU_I2C1_POINTRPEND_0
WKUP_DMSC0_INTR_IN_39	215	MCU_I2C1_CLKSTOP_WAKEUP_0
WKUP_DMSC0_INTR_IN_40	216	GLUELOGIC_MCU_R5FSS_LBIST_GLUE_DFT_LBIST_BIST_DONE_0
WKUP_DMSC0_INTR_IN_41	217	MCU_PBI00_DFT_PBI00_CPU_0
WKUP_DMSC0_INTR_IN_42	218	MCU_CTRL_MMR0_IPC_SET8_IPC_SET_IPCFG_0
WKUP_DMSC0_INTR_IN_43	219	MCU_CTRL_MMR0_ACCESS_ERR_0



**Table 9-39. WKUP\_DMSC0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
WKUP_DMSC0_INTR_IN_44	220	MCU_DCC0_INTR_DONE_LEVEL_0
WKUP_DMSC0_INTR_IN_45	221	MCU_DCC1_INTR_DONE_LEVEL_0
WKUP_DMSC0_INTR_IN_46	222	MCU_DCC2_INTR_DONE_LEVEL_0
WKUP_DMSC0_INTR_IN_47	223	MCU_UART0_USART_IRQ_0
WKUP_DMSC0_INTR_IN_48	224	MCU_FSS0_OSPI0_LVL_INTR_0
WKUP_DMSC0_INTR_IN_49	225	MCU_FSS0_OSPI1_LVL_INTR_0
WKUP_DMSC0_INTR_IN_50	226	MCU_FSS0_HPB_INTR_0
WKUP_DMSC0_INTR_IN_51	227	MCU_FSS0_OTFE_INTR_ERR_PEND_0
WKUP_DMSC0_INTR_IN_52	228	MCU_FSS0_FSAS_ECC_INTR_ERR_PEND_0
WKUP_DMSC0_INTR_IN_53	229	PSC0_PSC_ALLINT_0
WKUP_DMSC0_INTR_IN_54	230	MCU_I2C0_POINTRPEND_0
WKUP_DMSC0_INTR_IN_55	231	MCU_I2C0_CLKSTOP_WAKEUP_0
WKUP_DMSC0_INTR_IN_56	232	COMPUTE_CLUSTER0_GIC_OUTPUT_WAKER_GIC_PWR0_WAKE_REQUEST_0
WKUP_DMSC0_INTR_IN_57	233	COMPUTE_CLUSTER0_GIC_OUTPUT_WAKER_GIC_PWR0_WAKE_REQUEST_1

### 9.4.1.2 WKUP\_GPIOMUX\_INTRTR0 Interrupt Map

Table 9-40 lists all the interrupts that are mapped to the WKUP\_GPIOMUX\_INTRTR0 inputs, along with the corresponding register programming values used for mux control. Any WKUP\_GPIOMUX\_INTRTR0 interrupt input that is not listed in this table should be considered as unused.

**Table 9-40. WKUP\_GPIOMUX\_INTRTR0 Interrupt Map**

Interrupt Input Line	Peripheral Interrupt	WKUP_GPIOMUX_INTRTR0_MUXCNTL_n [6-0] ENABLE Field Value (DEC)
WKUP_GPIOMUX_INTRTR0_IN_0	WKUP_GPIO_VIRT_OUT0_0	0
WKUP_GPIOMUX_INTRTR0_IN_1	WKUP_GPIO_VIRT_OUT0_1_0	1
WKUP_GPIOMUX_INTRTR0_IN_2	WKUP_GPIO_VIRT_OUT0_2_0	2
WKUP_GPIOMUX_INTRTR0_IN_3	WKUP_GPIO_VIRT_OUT0_3_0	3
WKUP_GPIOMUX_INTRTR0_IN_4	WKUP_GPIO_VIRT_OUT0_4_0	4
WKUP_GPIOMUX_INTRTR0_IN_5	WKUP_GPIO_VIRT_OUT0_5_0	5
WKUP_GPIOMUX_INTRTR0_IN_6	WKUP_GPIO_VIRT_OUT0_6_0	6
WKUP_GPIOMUX_INTRTR0_IN_7	WKUP_GPIO_VIRT_OUT0_7_0	7
WKUP_GPIOMUX_INTRTR0_IN_8	WKUP_GPIO_VIRT_OUT0_8_0	8
WKUP_GPIOMUX_INTRTR0_IN_9	WKUP_GPIO_VIRT_OUT0_9_0	9
WKUP_GPIOMUX_INTRTR0_IN_10	WKUP_GPIO_VIRT_OUT0_10_0	10
WKUP_GPIOMUX_INTRTR0_IN_11	WKUP_GPIO_VIRT_OUT0_11_0	11
WKUP_GPIOMUX_INTRTR0_IN_12	WKUP_GPIO_VIRT_OUT0_12_0	12
WKUP_GPIOMUX_INTRTR0_IN_13	WKUP_GPIO_VIRT_OUT0_13_0	13
WKUP_GPIOMUX_INTRTR0_IN_14	WKUP_GPIO_VIRT_OUT0_14_0	14
WKUP_GPIOMUX_INTRTR0_IN_15	WKUP_GPIO_VIRT_OUT0_15_0	15
WKUP_GPIOMUX_INTRTR0_IN_16	WKUP_GPIO_VIRT_OUT0_16_0	16
WKUP_GPIOMUX_INTRTR0_IN_17	WKUP_GPIO_VIRT_OUT0_17_0	17
WKUP_GPIOMUX_INTRTR0_IN_18	WKUP_GPIO_VIRT_OUT0_18_0	18
WKUP_GPIOMUX_INTRTR0_IN_19	WKUP_GPIO_VIRT_OUT0_19_0	19
WKUP_GPIOMUX_INTRTR0_IN_20	WKUP_GPIO_VIRT_OUT0_20_0	20
WKUP_GPIOMUX_INTRTR0_IN_21	WKUP_GPIO_VIRT_OUT0_21_0	21
WKUP_GPIOMUX_INTRTR0_IN_22	WKUP_GPIO_VIRT_OUT0_22_0	22
WKUP_GPIOMUX_INTRTR0_IN_23	WKUP_GPIO_VIRT_OUT0_23_0	23
WKUP_GPIOMUX_INTRTR0_IN_24	WKUP_GPIO_VIRT_OUT0_24_0	24
WKUP_GPIOMUX_INTRTR0_IN_25	WKUP_GPIO_VIRT_OUT0_25_0	25
WKUP_GPIOMUX_INTRTR0_IN_26	WKUP_GPIO_VIRT_OUT0_26_0	26
WKUP_GPIOMUX_INTRTR0_IN_27	WKUP_GPIO_VIRT_OUT0_27_0	27
WKUP_GPIOMUX_INTRTR0_IN_28	WKUP_GPIO_VIRT_OUT0_28_0	28
WKUP_GPIOMUX_INTRTR0_IN_29	WKUP_GPIO_VIRT_OUT0_29_0	29
WKUP_GPIOMUX_INTRTR0_IN_30	WKUP_GPIO_VIRT_OUT0_30_0	30
WKUP_GPIOMUX_INTRTR0_IN_31	WKUP_GPIO_VIRT_OUT0_31_0	31
WKUP_GPIOMUX_INTRTR0_IN_32	WKUP_GPIO_VIRT_OUT0_32_0	32
WKUP_GPIOMUX_INTRTR0_IN_33	WKUP_GPIO_VIRT_OUT0_33_0	33
WKUP_GPIOMUX_INTRTR0_IN_34	WKUP_GPIO_VIRT_OUT0_34_0	34
WKUP_GPIOMUX_INTRTR0_IN_35	WKUP_GPIO_VIRT_OUT0_35_0	35
WKUP_GPIOMUX_INTRTR0_IN_36	WKUP_GPIO_VIRT_OUT0_36_0	36
WKUP_GPIOMUX_INTRTR0_IN_37	WKUP_GPIO_VIRT_OUT0_37_0	37
WKUP_GPIOMUX_INTRTR0_IN_38	WKUP_GPIO_VIRT_OUT0_38_0	38
WKUP_GPIOMUX_INTRTR0_IN_39	WKUP_GPIO_VIRT_OUT0_39_0	39

**Table 9-40. WKUP\_GPIOMUX\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	WKUP_GPIOMUX_INTRTR0_MUXCNTL_n [6-0] ENABLE Field Value (DEC)
WKUP_GPIOMUX_INTRTR0_IN_40	WKUP_GPIO_VIRT_OUT0_40_0	40
WKUP_GPIOMUX_INTRTR0_IN_41	WKUP_GPIO_VIRT_OUT0_41_0	41
WKUP_GPIOMUX_INTRTR0_IN_42	WKUP_GPIO_VIRT_OUT0_42_0	42
WKUP_GPIOMUX_INTRTR0_IN_43	WKUP_GPIO_VIRT_OUT0_43_0	43
WKUP_GPIOMUX_INTRTR0_IN_44	WKUP_GPIO_VIRT_OUT0_44_0	44
WKUP_GPIOMUX_INTRTR0_IN_45	WKUP_GPIO_VIRT_OUT0_45_0	45
WKUP_GPIOMUX_INTRTR0_IN_46	WKUP_GPIO_VIRT_OUT0_46_0	46
WKUP_GPIOMUX_INTRTR0_IN_47	WKUP_GPIO_VIRT_OUT0_47_0	47
WKUP_GPIOMUX_INTRTR0_IN_48	WKUP_GPIO_VIRT_OUT0_48_0	48
WKUP_GPIOMUX_INTRTR0_IN_49	WKUP_GPIO_VIRT_OUT0_49_0	49
WKUP_GPIOMUX_INTRTR0_IN_50	WKUP_GPIO_VIRT_OUT0_50_0	50
WKUP_GPIOMUX_INTRTR0_IN_51	WKUP_GPIO_VIRT_OUT0_51_0	51
WKUP_GPIOMUX_INTRTR0_IN_52	WKUP_GPIO_VIRT_OUT0_52_0	52
WKUP_GPIOMUX_INTRTR0_IN_53	WKUP_GPIO_VIRT_OUT0_53_0	53
WKUP_GPIOMUX_INTRTR0_IN_54	WKUP_GPIO_VIRT_OUT0_54_0	54
WKUP_GPIOMUX_INTRTR0_IN_55	WKUP_GPIO_VIRT_OUT0_55_0	55
WKUP_GPIOMUX_INTRTR0_IN_56	WKUP_GPIO_VIRT_OUT0_56_0	56
WKUP_GPIOMUX_INTRTR0_IN_57	WKUP_GPIO_VIRT_OUT0_57_0	57
WKUP_GPIOMUX_INTRTR0_IN_58	WKUP_GPIO_VIRT_OUT0_58_0	58
WKUP_GPIOMUX_INTRTR0_IN_59	WKUP_GPIO_VIRT_OUT0_59_0	59
WKUP_GPIOMUX_INTRTR0_IN_60	WKUP_GPIO_VIRT_OUT0_60_0	60
WKUP_GPIOMUX_INTRTR0_IN_61	WKUP_GPIO_VIRT_OUT0_61_0	61
WKUP_GPIOMUX_INTRTR0_IN_62	WKUP_GPIO_VIRT_OUT0_62_0	62
WKUP_GPIOMUX_INTRTR0_IN_63	WKUP_GPIO_VIRT_OUT0_63_0	63
WKUP_GPIOMUX_INTRTR0_IN_64	WKUP_GPIO_VIRT_OUT0_64_0	64
WKUP_GPIOMUX_INTRTR0_IN_65	WKUP_GPIO_VIRT_OUT0_65_0	65
WKUP_GPIOMUX_INTRTR0_IN_66	WKUP_GPIO_VIRT_OUT0_66_0	66
WKUP_GPIOMUX_INTRTR0_IN_67	WKUP_GPIO_VIRT_OUT0_67_0	67
WKUP_GPIOMUX_INTRTR0_IN_68	WKUP_GPIO_VIRT_OUT0_68_0	68
WKUP_GPIOMUX_INTRTR0_IN_69	WKUP_GPIO_VIRT_OUT0_69_0	69
WKUP_GPIOMUX_INTRTR0_IN_70	WKUP_GPIO_VIRT_OUT0_70_0	70
WKUP_GPIOMUX_INTRTR0_IN_71	WKUP_GPIO_VIRT_OUT0_71_0	71
WKUP_GPIOMUX_INTRTR0_IN_72	WKUP_GPIO_VIRT_OUT0_72_0	72
WKUP_GPIOMUX_INTRTR0_IN_73	WKUP_GPIO_VIRT_OUT0_73_0	73
WKUP_GPIOMUX_INTRTR0_IN_74	WKUP_GPIO_VIRT_OUT0_74_0	74
WKUP_GPIOMUX_INTRTR0_IN_75	WKUP_GPIO_VIRT_OUT0_75_0	75
WKUP_GPIOMUX_INTRTR0_IN_76	WKUP_GPIO_VIRT_OUT0_76_0	76
WKUP_GPIOMUX_INTRTR0_IN_77	WKUP_GPIO_VIRT_OUT0_77_0	77
WKUP_GPIOMUX_INTRTR0_IN_78	WKUP_GPIO_VIRT_OUT0_78_0	78
WKUP_GPIOMUX_INTRTR0_IN_79	WKUP_GPIO_VIRT_OUT0_79_0	79
WKUP_GPIOMUX_INTRTR0_IN_80	WKUP_GPIO_VIRT_OUT0_80_0	80
WKUP_GPIOMUX_INTRTR0_IN_81	WKUP_GPIO_VIRT_OUT0_81_0	81
WKUP_GPIOMUX_INTRTR0_IN_82	WKUP_GPIO_VIRT_OUT0_82_0	82
WKUP_GPIOMUX_INTRTR0_IN_83	WKUP_GPIO_VIRT_OUT0_83_0	83

**Table 9-40. WKUP\_GPIOMUX\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	WKUP_GPIOMUX_INTRTR0_MUXCNTL_n [6-0] ENABLE Field Value (DEC)
WKUP_GPIOMUX_INTRTR0_IN_103	WKUP_GPIO0_GPIO_BANK_0	103
WKUP_GPIOMUX_INTRTR0_IN_104	WKUP_GPIO0_GPIO_BANK_1	104
WKUP_GPIOMUX_INTRTR0_IN_105	WKUP_GPIO0_GPIO_BANK_2	105
WKUP_GPIOMUX_INTRTR0_IN_106	WKUP_GPIO0_GPIO_BANK_3	106
WKUP_GPIOMUX_INTRTR0_IN_107	WKUP_GPIO0_GPIO_BANK_4	107
WKUP_GPIOMUX_INTRTR0_IN_108	WKUP_GPIO0_GPIO_BANK_5	108
WKUP_GPIOMUX_INTRTR0_IN_112	WKUP_GPIO1_GPIO_BANK_0	112
WKUP_GPIOMUX_INTRTR0_IN_113	WKUP_GPIO1_GPIO_BANK_1	113
WKUP_GPIOMUX_INTRTR0_IN_114	WKUP_GPIO1_GPIO_BANK_2	114
WKUP_GPIOMUX_INTRTR0_IN_115	WKUP_GPIO1_GPIO_BANK_3	115
WKUP_GPIOMUX_INTRTR0_IN_116	WKUP_GPIO1_GPIO_BANK_4	116
WKUP_GPIOMUX_INTRTR0_IN_117	WKUP_GPIO1_GPIO_BANK_5	117

#### 9.4.1.3 WKUP\_ESM0 Interrupt Map

Table 9-41 shows the mapping of events to the WKUP\_ESM0. Note that pulse event inputs are triple redundant so that there are three separate inputs for each pulse event (all to the same source signal) but only one connection is shown here.

**Table 9-41. WKUP\_ESM0 Interrupt Map**

Interrupt Input Line	Interrupt ID	Interrupt Name
WKUP_ESM0_LVL_IN_0	0	WKUP_DMSC0_DMTIMER_0_INTR_PEND_0
WKUP_ESM0_LVL_IN_1	1	WKUP_DMSC0_DMTIMER_1_INTR_PEND_1
WKUP_ESM0_LVL_IN_2	2	WKUP_DMSC0_DMTIMER_2_INTR_PEND_2
WKUP_ESM0_LVL_IN_3	3	WKUP_DMSC0_DMTIMER_3_INTR_PEND_3
WKUP_ESM0_LVL_IN_4	4	WKUP_DMSC0_RTI_0_INTR_WWD_4
WKUP_ESM0_LVL_IN_5	5	WKUP_DMSC0_RAT_0_EXP_INTR_0
WKUP_ESM0_LVL_IN_6	6	WKUP_DMSC0_ECC_AGGR_0_ECC_CORRECTED_ERR_LEVEL_0
WKUP_ESM0_LVL_IN_7	7	WKUP_DMSC0_ECC_AGGR_0_ECC_UNCORRECTED_ERR_LEVEL_0
WKUP_ESM0_LVL_IN_8	8	WKUP_VTM0_THERM_LVL_GT_TH1_INTR_0
WKUP_ESM0_LVL_IN_9	9	WKUP_VTM0_THERM_LVL_LT_TH0_INTR_0
WKUP_ESM0_LVL_IN_10	10	WKUP_VTM0_THERM_LVL_GT_TH2_INTR_0
WKUP_ESM0_LVL_IN_11	11	WKUP_VTM0_CORR_LEVEL_0
WKUP_ESM0_LVL_IN_12	12	WKUP_VTM0_UNCORR_LEVEL_0
WKUP_ESM0_LVL_IN_13	13	GLUELOGIC_HFOSC0_CLKLOSS_GLUE_REF_CLK_LOSS_DETECT_OUT_0
WKUP_ESM0_LVL_IN_14	14	WKUP_ECC_AGGR0_CORR_LEVEL_0
WKUP_ESM0_LVL_IN_15	15	WKUP_ECC_AGGR0_UNCORR_LEVEL_0
WKUP_ESM0_LVL_IN_16	16	WKUP_TIMEOUT_INFRA0_SAFEG_TRANS_ERR_LVL_0
WKUP_ESM0_LVL_IN_17	17	WKUP_PORZ_SYNC0_PORZ_TIMEOUT_0
WKUP_ESM0_LVL_IN_20	20	WKUP_WAKEIP0_VDD_MCU_GLDTC_STAT_THRESH_LOW_FLAG_IPCFG_0
WKUP_ESM0_LVL_IN_21	21	WKUP_WAKEIP0_VDD_MCU_GLDTC_STAT_THRESH_HI_FLAG_IPCFG_0
WKUP_ESM0_LVL_IN_22	22	WKUP_WAKEIP0_VDDR_MCU_GLDTC_STAT_THRESH_LOW_FLAG_IPCFG_0
WKUP_ESM0_LVL_IN_23	23	WKUP_WAKEIP0_VDDR_MCU_GLDTC_STAT_THRESH_HI_FLAG_IPCFG_0
WKUP_ESM0_LVL_IN_24	24	MAIN0_VDD_CORE_GLDTC_STAT_THRESH_LOW_FLAG_IPCFG_0
WKUP_ESM0_LVL_IN_25	25	MAIN0_VDD_CORE_GLDTC_STAT_THRESH_HI_FLAG_IPCFG_0
WKUP_ESM0_LVL_IN_26	26	MAIN0_VDDR_CORE_GLDTC_STAT_THRESH_LOW_FLAG_IPCFG_0
WKUP_ESM0_LVL_IN_27	27	MAIN0_VDDR_CORE_GLDTC_STAT_THRESH_HI_FLAG_IPCFG_0
WKUP_ESM0_LVL_IN_28	28	MAIN0_VDD_CPU_GLDTC_STAT_THRESH_LOW_FLAG_IPCFG_0
WKUP_ESM0_LVL_IN_29	29	MAIN0_VDD_CPU_GLDTC_STAT_THRESH_HI_FLAG_IPCFG_0
WKUP_ESM0_LVL_IN_30	30	MAIN0_VDDR_CPU_GLDTC_STAT_THRESH_LOW_FLAG_IPCFG_0
WKUP_ESM0_LVL_IN_31	31	MAIN0_VDDR_CPU_GLDTC_STAT_THRESH_HI_FLAG_IPCFG_0
WKUP_ESM0_LVL_IN_32	32	WKUP_PSC0_PSC_MOD_WKLP_WKUP_ALWAYS_ON_CS1_CLKSTOP_REQ_0
WKUP_ESM0_LVL_IN_33	33	WKUP_PSC0_PSC_MOD_WKLP_DMSC_CS1_CLKSTOP_REQ_0
WKUP_ESM0_LVL_IN_34	34	WKUP_PSC0_PSC_MOD_WKLP_DEBUG2DMSC_CS1_CLKSTOP_REQ_0
WKUP_ESM0_LVL_IN_35	35	WKUP_PSC0_PSC_MOD_WKLP_WKUP_GPIO_CS1_CLKSTOP_REQ_0
WKUP_ESM0_LVL_IN_36	36	WKUP_PSC0_PSC_MOD_WKLP_WKUPMCU2MAIN_CS1_CLKSTOP_REQ_0
WKUP_ESM0_LVL_IN_37	37	WKUP_PSC0_PSC_MOD_WKLP_MAIN2WKUPMCU_CS1_CLKSTOP_REQ_0
WKUP_ESM0_LVL_IN_38	38	WKUP_PSC0_PSC_MOD_WKLP_MCU_TEST_CS1_CLKSTOP_REQ_0
WKUP_ESM0_LVL_IN_39	39	WKUP_PSC0_PSC_MOD_WKLP_MCU_DEBUG_CS1_CLKSTOP_REQ_0
WKUP_ESM0_LVL_IN_40	40	WKUP_PSC0_PSC_MOD_WKLP_MCU_MCAN_0_CS1_CLKSTOP_REQ_0
WKUP_ESM0_LVL_IN_41	41	WKUP_PSC0_PSC_MOD_WKLP_MCU_MCAN_1_CS1_CLKSTOP_REQ_0

**Table 9-41. WKUP\_ESM0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
WKUP_ESM0_LVL_IN_42	42	WKUP_PSC0_PSC_MOD_WKLP_MCU_OSPI_0_CS1_CLKSTOP_REQ_0
WKUP_ESM0_LVL_IN_43	43	WKUP_PSC0_PSC_MOD_WKLP_MCU_OSPI_1_CS1_CLKSTOP_REQ_0
WKUP_ESM0_LVL_IN_44	44	WKUP_PSC0_PSC_MOD_WKLP_MCU_HYPERBUS_CS1_CLKSTOP_REQ_0
WKUP_ESM0_LVL_IN_45	45	WKUP_PSC0_PSC_MOD_WKLP_MCU_I3C_0_CS1_CLKSTOP_REQ_0
WKUP_ESM0_LVL_IN_46	46	WKUP_PSC0_PSC_MOD_WKLP_MCU_I3C_1_CS1_CLKSTOP_REQ_0
WKUP_ESM0_LVL_IN_47	47	WKUP_PSC0_PSC_MOD_WKLP_MCU_ADC_0_CS1_CLKSTOP_REQ_0
WKUP_ESM0_LVL_IN_48	48	WKUP_PSC0_PSC_MOD_WKLP_MCU_ADC_1_CS1_CLKSTOP_REQ_0
WKUP_ESM0_LVL_IN_49	49	WKUP_PSC0_PSC_MOD_WKLP_WKUP_SPARE0_CS1_CLKSTOP_REQ_0
WKUP_ESM0_LVL_IN_50	50	WKUP_PSC0_PSC_MOD_WKLP_WKUP_SPARE1_CS1_CLKSTOP_REQ_0
WKUP_ESM0_LVL_IN_51	51	WKUP_PSC0_PSC_MOD_WKLP_MCU_R5_0_CS1_CLKSTOP_REQ_0
WKUP_ESM0_LVL_IN_52	52	WKUP_PSC0_PSC_MOD_WKLP_MCU_R5_1_CS1_CLKSTOP_REQ_0
WKUP_ESM0_LVL_IN_53	53	WKUP_PSC0_PSC_MOD_WKLP_MCU_R5FSS0_PBI0_0_CS1_CLKSTOP_REQ_0
WKUP_ESM0_PLS_IN_64	64	WKUP_PRG_MCU0_POK_PGOOD_OUT_N_TO_ESM_0
WKUP_ESM0_PLS_IN_65	65	WKUP_PRG_MCU0_POK_PGOOD_OUT_N_TO_ESM_1
WKUP_ESM0_PLS_IN_66	66	WKUP_PRG_MCU0_POK_PGOOD_OUT_N_TO_ESM_2
WKUP_ESM0_PLS_IN_67	67	WKUP_PRG_MCU0_POK_PGOOD_OUT_N_TO_ESM_3
WKUP_ESM0_PLS_IN_68	68	WKUP_PRG_MCU0_POK_PGOOD_OUT_N_TO_ESM_4
WKUP_ESM0_PLS_IN_69	69	WKUP_PRG_MCU0_POK_PGOOD_OUT_N_TO_ESM_5
WKUP_ESM0_PLS_IN_70	70	WKUP_PRG_MCU_3POKS0_POK_PGOOD_OUT_N_TO_ESM_0
WKUP_ESM0_PLS_IN_71	71	WKUP_PRG_MCU_3POKS0_POK_PGOOD_OUT_N_TO_ESM_1
WKUP_ESM0_PLS_IN_72	72	WKUP_PRG_MCU_3POKS0_POK_PGOOD_OUT_N_TO_ESM_2
WKUP_ESM0_PLS_IN_76	76	WKUP_PRG0_POK_PGOOD_OUT_N_TO_ESM_0
WKUP_ESM0_PLS_IN_77	77	WKUP_PRG0_POK_PGOOD_OUT_N_TO_ESM_1
WKUP_ESM0_PLS_IN_78	78	WKUP_PRG0_POK_PGOOD_OUT_N_TO_ESM_2
WKUP_ESM0_PLS_IN_79	79	WKUP_PRG0_POK_PGOOD_OUT_N_TO_ESM_3
WKUP_ESM0_PLS_IN_80	80	WKUP_PRG0_POK_PGOOD_OUT_N_TO_ESM_4
WKUP_ESM0_PLS_IN_81	81	WKUP_PRG0_POK_PGOOD_OUT_N_TO_ESM_5
WKUP_ESM0_PLS_IN_82	82	WKUP_PRG0_POK_PGOOD_OUT_N_TO_ESM_6
WKUP_ESM0_PLS_IN_83	83	WKUP_PRG0_POK_PGOOD_OUT_N_TO_ESM_7
WKUP_ESM0_PLS_IN_88	88	WKUP_GPIOMUX_INTRTR0_OUTP_8
WKUP_ESM0_PLS_IN_89	89	WKUP_GPIOMUX_INTRTR0_OUTP_9
WKUP_ESM0_PLS_IN_90	90	WKUP_GPIOMUX_INTRTR0_OUTP_10
WKUP_ESM0_PLS_IN_91	91	WKUP_GPIOMUX_INTRTR0_OUTP_11
WKUP_ESM0_PLS_IN_92	92	WKUP_GPIOMUX_INTRTR0_OUTP_12
WKUP_ESM0_PLS_IN_93	93	WKUP_GPIOMUX_INTRTR0_OUTP_13
WKUP_ESM0_PLS_IN_94	94	WKUP_GPIOMUX_INTRTR0_OUTP_14
WKUP_ESM0_PLS_IN_95	95	WKUP_GPIOMUX_INTRTR0_OUTP_15

## 9.4.2 MCU Domain Interrupt Maps

### 9.4.2.1 MCU\_R5FSS0\_CORE0 Interrupt Map

[Table 9-42](#) shows the mapping of events to the MCU\_R5FSS0\_CORE0. The MCU\_R5FSS0 VIM supports both R5 cores.

The MCU\_R5FSS0\_CORE0 events are the events used by both processors when operating in lockstep mode.

**Table 9-42. MCU\_R5FSS0\_CORE0 Interrupt Map**

Interrupt Input Line	Interrupt ID	Interrupt Name
MCU_R5FSS0_CORE0_INTR_IN_0	0	MCU_MCAN0_MCANSS_MCAN_LVL_INT_0
MCU_R5FSS0_CORE0_INTR_IN_1	1	MCU_MCAN0_MCANSS_MCAN_LVL_INT_1
MCU_R5FSS0_CORE0_INTR_IN_2	2	MCU_MCAN1_MCANSS_MCAN_LVL_INT_0
MCU_R5FSS0_CORE0_INTR_IN_3	3	MCU_MCAN1_MCANSS_MCAN_LVL_INT_1
MCU_R5FSS0_CORE0_INTR_IN_4	4	MCU_MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
MCU_R5FSS0_CORE0_INTR_IN_5	5	MCU_MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
MCU_R5FSS0_CORE0_INTR_IN_6	6	MCU_ADC0_GEN_LEVEL_0
MCU_R5FSS0_CORE0_INTR_IN_7	7	MCU_ADC1_GEN_LEVEL_0
MCU_R5FSS0_CORE0_INTR_IN_14	14	MCU_SA2_UL0_SA_UL_PKA_0
MCU_R5FSS0_CORE0_INTR_IN_15	15	MCU_SA2_UL0_SA_UL_TRNG_0
MCU_R5FSS0_CORE0_INTR_IN_16	16	MCU_I3C0_I3C_INT_0
MCU_R5FSS0_CORE0_INTR_IN_17	17	MCU_I3C1_I3C_INT_0
MCU_R5FSS0_CORE0_INTR_IN_18	18	MCU_I2C1_POINTRPEND_0
MCU_R5FSS0_CORE0_INTR_IN_20	20	MCU_MCSPI0_INTR_SPI_0
MCU_R5FSS0_CORE0_INTR_IN_21	21	MCU_MCSPI1_INTR_SPI_0
MCU_R5FSS0_CORE0_INTR_IN_22	22	MCU_MCSPI2_INTR_SPI_0
MCU_R5FSS0_CORE0_INTR_IN_23	23	MCU_I2C0_POINTRPEND_0
MCU_R5FSS0_CORE0_INTR_IN_24	24	MCU_FSS0_OSPI_0_OSPI_LVL_INTR_0
MCU_R5FSS0_CORE0_INTR_IN_25	25	MCU_FSS0_OSPI_1_OSPI_LVL_INTR_0
MCU_R5FSS0_CORE0_INTR_IN_26	26	MCU_FSS0_HYPERBUS1P0_0_HPB_INTR_0
MCU_R5FSS0_CORE0_INTR_IN_27	27	MCU_FSS0_FSAS_0_OTFE_INTR_ERR_PEND_0
MCU_R5FSS0_CORE0_INTR_IN_28	28	MCU_FSS0_FSAS_0_ECC_INTR_ERR_PEND_0
MCU_R5FSS0_CORE0_INTR_IN_30	30	MCU_UART0_USART_IRQ_0
MCU_R5FSS0_CORE0_INTR_IN_32	32	MCU_CPSW0_STAT_PEND_0
MCU_R5FSS0_CORE0_INTR_IN_34	34	MCU_CPSW0_EVNT_PEND_0
MCU_R5FSS0_CORE0_INTR_IN_35	35	MCU_CPSW0_MDIO_PEND_0
MCU_R5FSS0_CORE0_INTR_IN_37	37	MCU_PBIST1_DFT_PBIST_CPU_0
MCU_R5FSS0_CORE0_INTR_IN_38	38	MCU_TIMER0_INTR_PEND_0
MCU_R5FSS0_CORE0_INTR_IN_39	39	MCU_TIMER1_INTR_PEND_0
MCU_R5FSS0_CORE0_INTR_IN_40	40	MCU_TIMER2_INTR_PEND_0
MCU_R5FSS0_CORE0_INTR_IN_41	41	MCU_TIMER3_INTR_PEND_0
MCU_R5FSS0_CORE0_INTR_IN_42	42	MCU_RTIO_INTR_WWD_0
MCU_R5FSS0_CORE0_INTR_IN_43	43	MCU_RTII_INTR_WWD_0
MCU_R5FSS0_CORE0_INTR_IN_44	44	MCU_CTRL_MMR0_IPC_SET0_IPC_SET_IPCFG_0
MCU_R5FSS0_CORE0_INTR_IN_45	45	MCU_CTRL_MMR0_IPC_SET1_IPC_SET_IPCFG_0
MCU_R5FSS0_CORE0_INTR_IN_46	46	MCU_CTRL_MMR0_ACCESS_ERR_0
MCU_R5FSS0_CORE0_INTR_IN_47	47	MCU_PBIST0_DFT_PBIST_CPU_0
MCU_R5FSS0_CORE0_INTR_IN_48	48	MCU_ESM0_ESM_INT_LOW_LVL_0



**Table 9-42. MCU\_R5FSS0\_CORE0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
MCU_R5FSS0_CORE0_INTR_IN_49	49	MCU_ESM0_ESM_INT_HI_LVL_0
MCU_R5FSS0_CORE0_INTR_IN_50	50	MCU_DCC0_INTR_DONE_LEVEL_0
MCU_R5FSS0_CORE0_INTR_IN_51	51	MCU_DCC1_INTR_DONE_LEVEL_0
MCU_R5FSS0_CORE0_INTR_IN_52	52	MCU_DCC2_INTR_DONE_LEVEL_0
MCU_R5FSS0_CORE0_INTR_IN_53	53	MCU_ESM0_ESM_INT_CFG_LVL_0
MCU_R5FSS0_CORE0_INTR_IN_54	54	DEBUGSS1_AQCMPINTR_LEVEL_0
MCU_R5FSS0_CORE0_INTR_IN_55	55	DEBUGSS1_CTM_LEVEL_0
MCU_R5FSS0_CORE0_INTR_IN_56	56	MCU_R5FSS0_CORE0_VALIRQ_0
MCU_R5FSS0_CORE0_INTR_IN_57	57	MCU_R5FSS0_CORE0_VALFIQ_0
MCU_R5FSS0_CORE0_INTR_IN_58	58	MCU_R5FSS0_CORE0_CTI_0
MCU_R5FSS0_CORE0_INTR_IN_59	59	MCU_R5FSS0_CORE1_CTI_0
MCU_R5FSS0_CORE0_INTR_IN_60	60	MCU_R5FSS0_COMMRX_LEVEL_0_0
MCU_R5FSS0_CORE0_INTR_IN_61	61	MCU_R5FSS0_COMMTX_LEVEL_0_0
MCU_R5FSS0_CORE0_INTR_IN_64	64	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_0
MCU_R5FSS0_CORE0_INTR_IN_65	65	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_1
MCU_R5FSS0_CORE0_INTR_IN_66	66	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_2
MCU_R5FSS0_CORE0_INTR_IN_67	67	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_3
MCU_R5FSS0_CORE0_INTR_IN_68	68	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_4
MCU_R5FSS0_CORE0_INTR_IN_69	69	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_5
MCU_R5FSS0_CORE0_INTR_IN_70	70	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_6
MCU_R5FSS0_CORE0_INTR_IN_71	71	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_7
MCU_R5FSS0_CORE0_INTR_IN_72	72	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_8
MCU_R5FSS0_CORE0_INTR_IN_73	73	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_9
MCU_R5FSS0_CORE0_INTR_IN_74	74	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_10
MCU_R5FSS0_CORE0_INTR_IN_75	75	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_11
MCU_R5FSS0_CORE0_INTR_IN_76	76	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_12
MCU_R5FSS0_CORE0_INTR_IN_77	77	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_13
MCU_R5FSS0_CORE0_INTR_IN_78	78	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_14
MCU_R5FSS0_CORE0_INTR_IN_79	79	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_15
MCU_R5FSS0_CORE0_INTR_IN_80	80	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_16
MCU_R5FSS0_CORE0_INTR_IN_81	81	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_17
MCU_R5FSS0_CORE0_INTR_IN_82	82	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_18
MCU_R5FSS0_CORE0_INTR_IN_83	83	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_19
MCU_R5FSS0_CORE0_INTR_IN_84	84	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_20
MCU_R5FSS0_CORE0_INTR_IN_85	85	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_21
MCU_R5FSS0_CORE0_INTR_IN_86	86	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_22
MCU_R5FSS0_CORE0_INTR_IN_87	87	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_23
MCU_R5FSS0_CORE0_INTR_IN_88	88	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_24
MCU_R5FSS0_CORE0_INTR_IN_89	89	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_25
MCU_R5FSS0_CORE0_INTR_IN_90	90	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_26
MCU_R5FSS0_CORE0_INTR_IN_91	91	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_27
MCU_R5FSS0_CORE0_INTR_IN_92	92	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_28
MCU_R5FSS0_CORE0_INTR_IN_93	93	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_29
MCU_R5FSS0_CORE0_INTR_IN_94	94	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_30



**Table 9-42. MCU\_R5FSS0\_CORE0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
MCU_R5FSS0_CORE0_INTR_IN_95	95	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_31
MCU_R5FSS0_CORE0_INTR_IN_96	96	WKUP_I2C0_POINTRPEND_0
MCU_R5FSS0_CORE0_INTR_IN_97	97	WKUP_UART0_USART_IRQ_0
MCU_R5FSS0_CORE0_INTR_IN_98	98	WKUP_ESM0_ESM_INT_LOW_LVL_0
MCU_R5FSS0_CORE0_INTR_IN_99	99	WKUP_ESM0_ESM_INT_HI_LVL_0
MCU_R5FSS0_CORE0_INTR_IN_100	100	WKUP_ESM0_ESM_INT_CFG_LVL_0
MCU_R5FSS0_CORE0_INTR_IN_102	102	WKUP_DMSC0_RAT_0_EXP_INTR_0
MCU_R5FSS0_CORE0_INTR_IN_103	103	WKUP_DMSC0_DBG_AUTH_0_DEBUG_AUTH_INTR_0
MCU_R5FSS0_CORE0_INTR_IN_104	104	WKUP_DMSC0_AES_0_HIB_PUBLIC_0
MCU_R5FSS0_CORE0_INTR_IN_105	105	WKUP_DMSC0_AES_0_HIB_SECURE_0
MCU_R5FSS0_CORE0_INTR_IN_106	106	WKUP_DMSC0_CORTEX_M3_0_SEC_OUT_0
MCU_R5FSS0_CORE0_INTR_IN_107	107	WKUP_DMSC0_CORTEX_M3_0_SEC_OUT_1
MCU_R5FSS0_CORE0_INTR_IN_108	108	MCU_TIMER4_INTR_PEND_0
MCU_R5FSS0_CORE0_INTR_IN_109	109	MCU_TIMER5_INTR_PEND_0
MCU_R5FSS0_CORE0_INTR_IN_110	110	MCU_TIMER6_INTR_PEND_0
MCU_R5FSS0_CORE0_INTR_IN_111	111	MCU_TIMER7_INTR_PEND_0
MCU_R5FSS0_CORE0_INTR_IN_112	112	MCU_TIMER8_INTR_PEND_0
MCU_R5FSS0_CORE0_INTR_IN_113	113	MCU_TIMER9_INTR_PEND_0
MCU_R5FSS0_CORE0_INTR_IN_119	119	WKUP_CTRL_MMR0_ACCESS_ERR_0
MCU_R5FSS0_CORE0_INTR_IN_120	120	WKUP_PORZ_SYNC0_MAIN_PORZ_LATCHED_0
MCU_R5FSS0_CORE0_INTR_IN_122	122	WKUP_RESETZ_SYNC0_MAIN_RESETZ_LATCHED_0
MCU_R5FSS0_CORE0_INTR_IN_124	124	WKUP_GPIOMUX_INTRTR0_OUTP_0
MCU_R5FSS0_CORE0_INTR_IN_125	125	WKUP_GPIOMUX_INTRTR0_OUTP_1
MCU_R5FSS0_CORE0_INTR_IN_126	126	WKUP_GPIOMUX_INTRTR0_OUTP_2
MCU_R5FSS0_CORE0_INTR_IN_127	127	WKUP_GPIOMUX_INTRTR0_OUTP_3
MCU_R5FSS0_CORE0_INTR_IN_128	128	WKUP_GPIOMUX_INTRTR0_OUTP_4
MCU_R5FSS0_CORE0_INTR_IN_129	129	WKUP_GPIOMUX_INTRTR0_OUTP_5
MCU_R5FSS0_CORE0_INTR_IN_130	130	WKUP_GPIOMUX_INTRTR0_OUTP_6
MCU_R5FSS0_CORE0_INTR_IN_131	131	WKUP_GPIOMUX_INTRTR0_OUTP_7
MCU_R5FSS0_CORE0_INTR_IN_132	132	WKUP_GPIOMUX_INTRTR0_OUTP_8
MCU_R5FSS0_CORE0_INTR_IN_133	133	WKUP_GPIOMUX_INTRTR0_OUTP_9
MCU_R5FSS0_CORE0_INTR_IN_134	134	WKUP_GPIOMUX_INTRTR0_OUTP_10
MCU_R5FSS0_CORE0_INTR_IN_135	135	WKUP_GPIOMUX_INTRTR0_OUTP_11
MCU_R5FSS0_CORE0_INTR_IN_136	136	WKUP_GPIOMUX_INTRTR0_OUTP_12
MCU_R5FSS0_CORE0_INTR_IN_137	137	WKUP_GPIOMUX_INTRTR0_OUTP_13
MCU_R5FSS0_CORE0_INTR_IN_138	138	WKUP_GPIOMUX_INTRTR0_OUTP_14
MCU_R5FSS0_CORE0_INTR_IN_139	139	WKUP_GPIOMUX_INTRTR0_OUTP_15
MCU_R5FSS0_CORE0_INTR_IN_140	140	ESM0_ESM_INT_LOW_LVL_0
MCU_R5FSS0_CORE0_INTR_IN_141	141	ESM0_ESM_INT_HI_LVL_0
MCU_R5FSS0_CORE0_INTR_IN_142	142	ESM0_ESM_INT_CFG_LVL_0
MCU_R5FSS0_CORE0_INTR_IN_144	144	COMPUTE_CLUSTER0_GIC_OUTPUT_WAKER_GIC_PWR0_WAKE_REQUEST_0
MCU_R5FSS0_CORE0_INTR_IN_145	145	COMPUTE_CLUSTER0_GIC_OUTPUT_WAKER_GIC_PWR0_WAKE_REQUEST_1
MCU_R5FSS0_CORE0_INTR_IN_146	146	MCU_R5FSS0_CORE0_EXP_INTR_0
MCU_R5FSS0_CORE0_INTR_IN_147	147	MCU_R5FSS0_CORE1_EXP_INTR_0

**Table 9-42. MCU\_R5FSS0\_CORE0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
MCU_R5FSS0_CORE0_INTR_IN_148	148	MCU_CBASS0_LPSC_MCU_COMMON_ERR_INTR_0
MCU_R5FSS0_CORE0_INTR_IN_149	149	GLUELOGIC_DBG_CBASS_INTR_OR_GLUE_DBG_CBASS_AGG_ERR_INTR_0
MCU_R5FSS0_CORE0_INTR_IN_150	150	GLUELOGIC_FW_CBASS_INTR_OR_GLUE_FW_CBASS_AGG_ERR_INTR_0
MCU_R5FSS0_CORE0_INTR_IN_151	151	WKUP_CBASS0_LPSC_WKUP_COMMON_ERR_INTR_0
MCU_R5FSS0_CORE0_INTR_IN_156	156	WKUP_VTM0_THERM_LVL_GT_TH1_INTR_0
MCU_R5FSS0_CORE0_INTR_IN_157	157	WKUP_VTM0_THERM_LVL_LT_TH0_INTR_0
MCU_R5FSS0_CORE0_INTR_IN_158	158	WKUP_VTM0_THERM_LVL_GT_TH2_INTR_0
MCU_R5FSS0_CORE0_INTR_IN_160	160	MAIN2MCU_LVL_INTRTR0_OUTL_0
MCU_R5FSS0_CORE0_INTR_IN_161	161	MAIN2MCU_LVL_INTRTR0_OUTL_1
MCU_R5FSS0_CORE0_INTR_IN_162	162	MAIN2MCU_LVL_INTRTR0_OUTL_2
MCU_R5FSS0_CORE0_INTR_IN_163	163	MAIN2MCU_LVL_INTRTR0_OUTL_3
MCU_R5FSS0_CORE0_INTR_IN_164	164	MAIN2MCU_LVL_INTRTR0_OUTL_4
MCU_R5FSS0_CORE0_INTR_IN_165	165	MAIN2MCU_LVL_INTRTR0_OUTL_5
MCU_R5FSS0_CORE0_INTR_IN_166	166	MAIN2MCU_LVL_INTRTR0_OUTL_6
MCU_R5FSS0_CORE0_INTR_IN_167	167	MAIN2MCU_LVL_INTRTR0_OUTL_7
MCU_R5FSS0_CORE0_INTR_IN_168	168	MAIN2MCU_LVL_INTRTR0_OUTL_8
MCU_R5FSS0_CORE0_INTR_IN_169	169	MAIN2MCU_LVL_INTRTR0_OUTL_9
MCU_R5FSS0_CORE0_INTR_IN_170	170	MAIN2MCU_LVL_INTRTR0_OUTL_10
MCU_R5FSS0_CORE0_INTR_IN_171	171	MAIN2MCU_LVL_INTRTR0_OUTL_11
MCU_R5FSS0_CORE0_INTR_IN_172	172	MAIN2MCU_LVL_INTRTR0_OUTL_12
MCU_R5FSS0_CORE0_INTR_IN_173	173	MAIN2MCU_LVL_INTRTR0_OUTL_13
MCU_R5FSS0_CORE0_INTR_IN_174	174	MAIN2MCU_LVL_INTRTR0_OUTL_14
MCU_R5FSS0_CORE0_INTR_IN_175	175	MAIN2MCU_LVL_INTRTR0_OUTL_15
MCU_R5FSS0_CORE0_INTR_IN_176	176	MAIN2MCU_LVL_INTRTR0_OUTL_16
MCU_R5FSS0_CORE0_INTR_IN_177	177	MAIN2MCU_LVL_INTRTR0_OUTL_17
MCU_R5FSS0_CORE0_INTR_IN_178	178	MAIN2MCU_LVL_INTRTR0_OUTL_18
MCU_R5FSS0_CORE0_INTR_IN_179	179	MAIN2MCU_LVL_INTRTR0_OUTL_19
MCU_R5FSS0_CORE0_INTR_IN_180	180	MAIN2MCU_LVL_INTRTR0_OUTL_20
MCU_R5FSS0_CORE0_INTR_IN_181	181	MAIN2MCU_LVL_INTRTR0_OUTL_21
MCU_R5FSS0_CORE0_INTR_IN_182	182	MAIN2MCU_LVL_INTRTR0_OUTL_22
MCU_R5FSS0_CORE0_INTR_IN_183	183	MAIN2MCU_LVL_INTRTR0_OUTL_23
MCU_R5FSS0_CORE0_INTR_IN_184	184	MAIN2MCU_LVL_INTRTR0_OUTL_24
MCU_R5FSS0_CORE0_INTR_IN_185	185	MAIN2MCU_LVL_INTRTR0_OUTL_25
MCU_R5FSS0_CORE0_INTR_IN_186	186	MAIN2MCU_LVL_INTRTR0_OUTL_26
MCU_R5FSS0_CORE0_INTR_IN_187	187	MAIN2MCU_LVL_INTRTR0_OUTL_27
MCU_R5FSS0_CORE0_INTR_IN_188	188	MAIN2MCU_LVL_INTRTR0_OUTL_28
MCU_R5FSS0_CORE0_INTR_IN_189	189	MAIN2MCU_LVL_INTRTR0_OUTL_29
MCU_R5FSS0_CORE0_INTR_IN_190	190	MAIN2MCU_LVL_INTRTR0_OUTL_30
MCU_R5FSS0_CORE0_INTR_IN_191	191	MAIN2MCU_LVL_INTRTR0_OUTL_31
MCU_R5FSS0_CORE0_INTR_IN_192	192	MAIN2MCU_LVL_INTRTR0_OUTL_32
MCU_R5FSS0_CORE0_INTR_IN_193	193	MAIN2MCU_LVL_INTRTR0_OUTL_33
MCU_R5FSS0_CORE0_INTR_IN_194	194	MAIN2MCU_LVL_INTRTR0_OUTL_34
MCU_R5FSS0_CORE0_INTR_IN_195	195	MAIN2MCU_LVL_INTRTR0_OUTL_35
MCU_R5FSS0_CORE0_INTR_IN_196	196	MAIN2MCU_LVL_INTRTR0_OUTL_36

**Table 9-42. MCU\_R5FSS0\_CORE0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
MCU_R5FSS0_CORE0_INTR_IN_197	197	MAIN2MCU_LVL_INTRTR0_OUTL_37
MCU_R5FSS0_CORE0_INTR_IN_198	198	MAIN2MCU_LVL_INTRTR0_OUTL_38
MCU_R5FSS0_CORE0_INTR_IN_199	199	MAIN2MCU_LVL_INTRTR0_OUTL_39
MCU_R5FSS0_CORE0_INTR_IN_200	200	MAIN2MCU_LVL_INTRTR0_OUTL_40
MCU_R5FSS0_CORE0_INTR_IN_201	201	MAIN2MCU_LVL_INTRTR0_OUTL_41
MCU_R5FSS0_CORE0_INTR_IN_202	202	MAIN2MCU_LVL_INTRTR0_OUTL_42
MCU_R5FSS0_CORE0_INTR_IN_203	203	MAIN2MCU_LVL_INTRTR0_OUTL_43
MCU_R5FSS0_CORE0_INTR_IN_204	204	MAIN2MCU_LVL_INTRTR0_OUTL_44
MCU_R5FSS0_CORE0_INTR_IN_205	205	MAIN2MCU_LVL_INTRTR0_OUTL_45
MCU_R5FSS0_CORE0_INTR_IN_206	206	MAIN2MCU_LVL_INTRTR0_OUTL_46
MCU_R5FSS0_CORE0_INTR_IN_207	207	MAIN2MCU_LVL_INTRTR0_OUTL_47
MCU_R5FSS0_CORE0_INTR_IN_208	208	MAIN2MCU_LVL_INTRTR0_OUTL_48
MCU_R5FSS0_CORE0_INTR_IN_209	209	MAIN2MCU_LVL_INTRTR0_OUTL_49
MCU_R5FSS0_CORE0_INTR_IN_210	210	MAIN2MCU_LVL_INTRTR0_OUTL_50
MCU_R5FSS0_CORE0_INTR_IN_211	211	MAIN2MCU_LVL_INTRTR0_OUTL_51
MCU_R5FSS0_CORE0_INTR_IN_212	212	MAIN2MCU_LVL_INTRTR0_OUTL_52
MCU_R5FSS0_CORE0_INTR_IN_213	213	MAIN2MCU_LVL_INTRTR0_OUTL_53
MCU_R5FSS0_CORE0_INTR_IN_214	214	MAIN2MCU_LVL_INTRTR0_OUTL_54
MCU_R5FSS0_CORE0_INTR_IN_215	215	MAIN2MCU_LVL_INTRTR0_OUTL_55
MCU_R5FSS0_CORE0_INTR_IN_216	216	MAIN2MCU_LVL_INTRTR0_OUTL_56
MCU_R5FSS0_CORE0_INTR_IN_217	217	MAIN2MCU_LVL_INTRTR0_OUTL_57
MCU_R5FSS0_CORE0_INTR_IN_218	218	MAIN2MCU_LVL_INTRTR0_OUTL_58
MCU_R5FSS0_CORE0_INTR_IN_219	219	MAIN2MCU_LVL_INTRTR0_OUTL_59
MCU_R5FSS0_CORE0_INTR_IN_220	220	MAIN2MCU_LVL_INTRTR0_OUTL_60
MCU_R5FSS0_CORE0_INTR_IN_221	221	MAIN2MCU_LVL_INTRTR0_OUTL_61
MCU_R5FSS0_CORE0_INTR_IN_222	222	MAIN2MCU_LVL_INTRTR0_OUTL_62
MCU_R5FSS0_CORE0_INTR_IN_223	223	MAIN2MCU_LVL_INTRTR0_OUTL_63
MCU_R5FSS0_CORE0_INTR_IN_224	224	MAIN2MCU_PLS_INTRTR0_OUTP_0
MCU_R5FSS0_CORE0_INTR_IN_225	225	MAIN2MCU_PLS_INTRTR0_OUTP_1
MCU_R5FSS0_CORE0_INTR_IN_226	226	MAIN2MCU_PLS_INTRTR0_OUTP_2
MCU_R5FSS0_CORE0_INTR_IN_227	227	MAIN2MCU_PLS_INTRTR0_OUTP_3
MCU_R5FSS0_CORE0_INTR_IN_228	228	MAIN2MCU_PLS_INTRTR0_OUTP_4
MCU_R5FSS0_CORE0_INTR_IN_229	229	MAIN2MCU_PLS_INTRTR0_OUTP_5
MCU_R5FSS0_CORE0_INTR_IN_230	230	MAIN2MCU_PLS_INTRTR0_OUTP_6
MCU_R5FSS0_CORE0_INTR_IN_231	231	MAIN2MCU_PLS_INTRTR0_OUTP_7
MCU_R5FSS0_CORE0_INTR_IN_232	232	MAIN2MCU_PLS_INTRTR0_OUTP_8
MCU_R5FSS0_CORE0_INTR_IN_233	233	MAIN2MCU_PLS_INTRTR0_OUTP_9
MCU_R5FSS0_CORE0_INTR_IN_234	234	MAIN2MCU_PLS_INTRTR0_OUTP_10
MCU_R5FSS0_CORE0_INTR_IN_235	235	MAIN2MCU_PLS_INTRTR0_OUTP_11
MCU_R5FSS0_CORE0_INTR_IN_236	236	MAIN2MCU_PLS_INTRTR0_OUTP_12
MCU_R5FSS0_CORE0_INTR_IN_237	237	MAIN2MCU_PLS_INTRTR0_OUTP_13
MCU_R5FSS0_CORE0_INTR_IN_238	238	MAIN2MCU_PLS_INTRTR0_OUTP_14
MCU_R5FSS0_CORE0_INTR_IN_239	239	MAIN2MCU_PLS_INTRTR0_OUTP_15
MCU_R5FSS0_CORE0_INTR_IN_240	240	MAIN2MCU_PLS_INTRTR0_OUTP_16

**Table 9-42. MCU\_R5FSS0\_CORE0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
MCU_R5FSS0_CORE0_INTR_IN_241	241	MAIN2MCU_PLS_INTRTR0_OUTP_17
MCU_R5FSS0_CORE0_INTR_IN_242	242	MAIN2MCU_PLS_INTRTR0_OUTP_18
MCU_R5FSS0_CORE0_INTR_IN_243	243	MAIN2MCU_PLS_INTRTR0_OUTP_19
MCU_R5FSS0_CORE0_INTR_IN_244	244	MAIN2MCU_PLS_INTRTR0_OUTP_20
MCU_R5FSS0_CORE0_INTR_IN_245	245	MAIN2MCU_PLS_INTRTR0_OUTP_21
MCU_R5FSS0_CORE0_INTR_IN_246	246	MAIN2MCU_PLS_INTRTR0_OUTP_22
MCU_R5FSS0_CORE0_INTR_IN_247	247	MAIN2MCU_PLS_INTRTR0_OUTP_23
MCU_R5FSS0_CORE0_INTR_IN_248	248	MAIN2MCU_PLS_INTRTR0_OUTP_24
MCU_R5FSS0_CORE0_INTR_IN_249	249	MAIN2MCU_PLS_INTRTR0_OUTP_25
MCU_R5FSS0_CORE0_INTR_IN_250	250	MAIN2MCU_PLS_INTRTR0_OUTP_26
MCU_R5FSS0_CORE0_INTR_IN_251	251	MAIN2MCU_PLS_INTRTR0_OUTP_27
MCU_R5FSS0_CORE0_INTR_IN_252	252	MAIN2MCU_PLS_INTRTR0_OUTP_28
MCU_R5FSS0_CORE0_INTR_IN_253	253	MAIN2MCU_PLS_INTRTR0_OUTP_29
MCU_R5FSS0_CORE0_INTR_IN_254	254	MAIN2MCU_PLS_INTRTR0_OUTP_30
MCU_R5FSS0_CORE0_INTR_IN_255	255	MAIN2MCU_PLS_INTRTR0_OUTP_31
MCU_R5FSS0_CORE0_INTR_IN_256	256	MAIN2MCU_PLS_INTRTR0_OUTP_32
MCU_R5FSS0_CORE0_INTR_IN_257	257	MAIN2MCU_PLS_INTRTR0_OUTP_33
MCU_R5FSS0_CORE0_INTR_IN_258	258	MAIN2MCU_PLS_INTRTR0_OUTP_34
MCU_R5FSS0_CORE0_INTR_IN_259	259	MAIN2MCU_PLS_INTRTR0_OUTP_35
MCU_R5FSS0_CORE0_INTR_IN_260	260	MAIN2MCU_PLS_INTRTR0_OUTP_36
MCU_R5FSS0_CORE0_INTR_IN_261	261	MAIN2MCU_PLS_INTRTR0_OUTP_37
MCU_R5FSS0_CORE0_INTR_IN_262	262	MAIN2MCU_PLS_INTRTR0_OUTP_38
MCU_R5FSS0_CORE0_INTR_IN_263	263	MAIN2MCU_PLS_INTRTR0_OUTP_39
MCU_R5FSS0_CORE0_INTR_IN_264	264	MAIN2MCU_PLS_INTRTR0_OUTP_40
MCU_R5FSS0_CORE0_INTR_IN_265	265	MAIN2MCU_PLS_INTRTR0_OUTP_41
MCU_R5FSS0_CORE0_INTR_IN_266	266	MAIN2MCU_PLS_INTRTR0_OUTP_42
MCU_R5FSS0_CORE0_INTR_IN_267	267	MAIN2MCU_PLS_INTRTR0_OUTP_43
MCU_R5FSS0_CORE0_INTR_IN_268	268	MAIN2MCU_PLS_INTRTR0_OUTP_44
MCU_R5FSS0_CORE0_INTR_IN_269	269	MAIN2MCU_PLS_INTRTR0_OUTP_45
MCU_R5FSS0_CORE0_INTR_IN_270	270	MAIN2MCU_PLS_INTRTR0_OUTP_46
MCU_R5FSS0_CORE0_INTR_IN_271	271	MAIN2MCU_PLS_INTRTR0_OUTP_47
MCU_R5FSS0_CORE0_INTR_IN_288	288	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_8
MCU_R5FSS0_CORE0_INTR_IN_289	289	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_9
MCU_R5FSS0_CORE0_INTR_IN_290	290	COMPUTE_CLUSTER0_PBIIST_WRAP_DFT_PBIIST_CPU_0
MCU_R5FSS0_CORE0_INTR_IN_291	291	COMPUTE_CLUSTER0_ARM0_DFT_LBIST_DFT_LBIST_BIST_DONE_0
MCU_R5FSS0_CORE0_INTR_IN_293	293	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_12
MCU_R5FSS0_CORE0_INTR_IN_294	294	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_13
MCU_R5FSS0_CORE0_INTR_IN_295	295	COMPUTE_CLUSTER0_C7X_4_DFT_LBIST_DFT_LBIST_BIST_DONE_0
MCU_R5FSS0_CORE0_INTR_IN_297	297	GLUELOGIC_MAIN_PULSAR0_LBIST_GLUE_DFT_LBIST_BIST_DONE_0
MCU_R5FSS0_CORE0_INTR_IN_298	298	GLUELOGIC_MAIN_PULSAR1_LBIST_GLUE_DFT_LBIST_BIST_DONE_0
MCU_R5FSS0_CORE0_INTR_IN_299	299	GLUELOGIC_DMPAC_LBIST_GLUE_DFT_LBIST_BIST_DONE_0
MCU_R5FSS0_CORE0_INTR_IN_300	300	GLUELOGIC_VPAC_LBIST_GLUE_DFT_LBIST_BIST_DONE_0
MCU_R5FSS0_CORE0_INTR_IN_302	302	PBIIST2_DFT_PBIIST_CPU_0
MCU_R5FSS0_CORE0_INTR_IN_303	303	PBIIST0_DFT_PBIIST_CPU_0

**Table 9-42. MCU\_R5FSS0\_CORE0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
MCU_R5FSS0_CORE0_INTR_IN_304	304	PBIST1_DFT_PBIST_CPU_0
MCU_R5FSS0_CORE0_INTR_IN_305	305	PBIST3_DFT_PBIST_CPU_0
MCU_R5FSS0_CORE0_INTR_IN_306	306	PBIST4_DFT_PBIST_CPU_0
MCU_R5FSS0_CORE0_INTR_IN_307	307	PBIST5_DFT_PBIST_CPU_0
MCU_R5FSS0_CORE0_INTR_IN_308	308	PBIST7_DFT_PBIST_CPU_0
MCU_R5FSS0_CORE0_INTR_IN_309	309	PBIST9_DFT_PBIST_CPU_0
MCU_R5FSS0_CORE0_INTR_IN_310	310	PBIST10_DFT_PBIST_CPU_0
MCU_R5FSS0_CORE0_INTR_IN_312	312	PBIST6_DFT_PBIST_CPU_0
MCU_R5FSS0_CORE0_INTR_IN_315	315	C66SS0_PBIST0_DFT_PBIST_CPU_0
MCU_R5FSS0_CORE0_INTR_IN_316	316	C66SS1_PBIST0_DFT_PBIST_CPU_0
MCU_R5FSS0_CORE0_INTR_IN_317	317	GPU0_DFT_PBIST_0_DFT_PBIST_CPU_0
MCU_R5FSS0_CORE0_INTR_IN_320	320	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_0
MCU_R5FSS0_CORE0_INTR_IN_321	321	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_1
MCU_R5FSS0_CORE0_INTR_IN_322	322	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_2
MCU_R5FSS0_CORE0_INTR_IN_323	323	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_3
MCU_R5FSS0_CORE0_INTR_IN_324	324	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_4
MCU_R5FSS0_CORE0_INTR_IN_325	325	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_5
MCU_R5FSS0_CORE0_INTR_IN_326	326	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_6
MCU_R5FSS0_CORE0_INTR_IN_327	327	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_7
MCU_R5FSS0_CORE0_INTR_IN_328	328	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_8
MCU_R5FSS0_CORE0_INTR_IN_329	329	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_9
MCU_R5FSS0_CORE0_INTR_IN_330	330	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_10
MCU_R5FSS0_CORE0_INTR_IN_331	331	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_11
MCU_R5FSS0_CORE0_INTR_IN_332	332	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_12
MCU_R5FSS0_CORE0_INTR_IN_333	333	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_13
MCU_R5FSS0_CORE0_INTR_IN_334	334	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_14
MCU_R5FSS0_CORE0_INTR_IN_335	335	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_15
MCU_R5FSS0_CORE0_INTR_IN_336	336	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_16
MCU_R5FSS0_CORE0_INTR_IN_337	337	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_17
MCU_R5FSS0_CORE0_INTR_IN_338	338	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_18
MCU_R5FSS0_CORE0_INTR_IN_339	339	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_19
MCU_R5FSS0_CORE0_INTR_IN_340	340	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_20
MCU_R5FSS0_CORE0_INTR_IN_341	341	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_21
MCU_R5FSS0_CORE0_INTR_IN_342	342	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_22
MCU_R5FSS0_CORE0_INTR_IN_343	343	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_23
MCU_R5FSS0_CORE0_INTR_IN_344	344	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_24
MCU_R5FSS0_CORE0_INTR_IN_345	345	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_25
MCU_R5FSS0_CORE0_INTR_IN_346	346	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_26
MCU_R5FSS0_CORE0_INTR_IN_347	347	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_27
MCU_R5FSS0_CORE0_INTR_IN_348	348	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_28
MCU_R5FSS0_CORE0_INTR_IN_349	349	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_29
MCU_R5FSS0_CORE0_INTR_IN_350	350	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_30
MCU_R5FSS0_CORE0_INTR_IN_351	351	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_31
MCU_R5FSS0_CORE0_INTR_IN_376	376	NAVSS0_INTR_ROUTER_0_OUTL_INTR_400

**Table 9-42. MCU\_R5FSS0\_CORE0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
MCU_R5FSS0_CORE0_INTR_IN_377	377	NAVSS0_INTR_ROUTER_0_OUTL_INTR_401
MCU_R5FSS0_CORE0_INTR_IN_378	378	NAVSS0_INTR_ROUTER_0_OUTL_INTR_402
MCU_R5FSS0_CORE0_INTR_IN_379	379	NAVSS0_INTR_ROUTER_0_OUTL_INTR_403
MCU_R5FSS0_CORE0_INTR_IN_380	380	NAVSS0_INTR_ROUTER_0_OUTL_INTR_404
MCU_R5FSS0_CORE0_INTR_IN_381	381	NAVSS0_INTR_ROUTER_0_OUTL_INTR_405
MCU_R5FSS0_CORE0_INTR_IN_382	382	NAVSS0_INTR_ROUTER_0_OUTL_INTR_406
MCU_R5FSS0_CORE0_INTR_IN_383	383	NAVSS0_INTR_ROUTER_0_OUTL_INTR_407

#### 9.4.2.2 MCU\_R5FSS0\_CORE1 Interrupt Map

Table 9-43 shows the mapping of events to the MCU\_R5FSS0\_CORE1. The MCU\_R5FSS0 VIM supports both R5 cores.

The MCU\_R5FSS0\_CORE1 events are not used when operating in lockstep mode.

**Table 9-43. MCU\_R5FSS0\_CORE1 Interrupt Map**

Interrupt Input Line	Interrupt ID	Interrupt Name
MCU_R5FSS0_CORE1_INTR_IN_0	0	MCU_MCAN0_MCANSS_MCAN_LVL_INT_0
MCU_R5FSS0_CORE1_INTR_IN_1	1	MCU_MCAN0_MCANSS_MCAN_LVL_INT_1
MCU_R5FSS0_CORE1_INTR_IN_2	2	MCU_MCAN1_MCANSS_MCAN_LVL_INT_0
MCU_R5FSS0_CORE1_INTR_IN_3	3	MCU_MCAN1_MCANSS_MCAN_LVL_INT_1
MCU_R5FSS0_CORE1_INTR_IN_4	4	MCU_MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
MCU_R5FSS0_CORE1_INTR_IN_5	5	MCU_MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
MCU_R5FSS0_CORE1_INTR_IN_6	6	MCU_ADC0_GEN_LEVEL_0
MCU_R5FSS0_CORE1_INTR_IN_7	7	MCU_ADC1_GEN_LEVEL_0
MCU_R5FSS0_CORE1_INTR_IN_14	14	MCU_SA2_UL0_SA_UL_PKA_0
MCU_R5FSS0_CORE1_INTR_IN_15	15	MCU_SA2_UL0_SA_UL_TRNG_0
MCU_R5FSS0_CORE1_INTR_IN_16	16	MCU_I3C0_I3C_INT_0
MCU_R5FSS0_CORE1_INTR_IN_17	17	MCU_I3C1_I3C_INT_0
MCU_R5FSS0_CORE1_INTR_IN_18	18	MCU_I2C1_POINTRPEND_0
MCU_R5FSS0_CORE1_INTR_IN_20	20	MCU_MCSPI0_INTR_SPI_0
MCU_R5FSS0_CORE1_INTR_IN_21	21	MCU_MCSPI1_INTR_SPI_0
MCU_R5FSS0_CORE1_INTR_IN_22	22	MCU_MCSPI2_INTR_SPI_0
MCU_R5FSS0_CORE1_INTR_IN_23	23	MCU_I2C0_POINTRPEND_0
MCU_R5FSS0_CORE1_INTR_IN_24	24	MCU_FSS0_OSPI_0_OSPI_LVL_INTR_0
MCU_R5FSS0_CORE1_INTR_IN_25	25	MCU_FSS0_OSPI_1_OSPI_LVL_INTR_0
MCU_R5FSS0_CORE1_INTR_IN_26	26	MCU_FSS0_HYPERBUS1P0_0_HPB_INTR_0
MCU_R5FSS0_CORE1_INTR_IN_27	27	MCU_FSS0_FSAS_0_OTFE_INTR_ERR_PEND_0
MCU_R5FSS0_CORE1_INTR_IN_28	28	MCU_FSS0_FSAS_0_ECC_INTR_ERR_PEND_0
MCU_R5FSS0_CORE1_INTR_IN_30	30	MCU_UART0_USART_IRQ_0
MCU_R5FSS0_CORE1_INTR_IN_32	32	MCU_CPSW0_STAT_PEND_0
MCU_R5FSS0_CORE1_INTR_IN_34	34	MCU_CPSW0_EVTN_PEND_0
MCU_R5FSS0_CORE1_INTR_IN_35	35	MCU_CPSW0_MDIO_PEND_0
MCU_R5FSS0_CORE1_INTR_IN_37	37	MCU_PBIST1_DFT_PBIST_CPU_0
MCU_R5FSS0_CORE1_INTR_IN_38	38	MCU_TIMER0_INTR_PEND_0
MCU_R5FSS0_CORE1_INTR_IN_39	39	MCU_TIMER1_INTR_PEND_0
MCU_R5FSS0_CORE1_INTR_IN_40	40	MCU_TIMER2_INTR_PEND_0
MCU_R5FSS0_CORE1_INTR_IN_41	41	MCU_TIMER3_INTR_PEND_0
MCU_R5FSS0_CORE1_INTR_IN_42	42	MCU_RTIO_INTR_WWD_0
MCU_R5FSS0_CORE1_INTR_IN_43	43	MCU_RT11_INTR_WWD_0
MCU_R5FSS0_CORE1_INTR_IN_44	44	MCU_CTRL_MMR0_IPC_SET0_IPC_SET_IPCFG_0
MCU_R5FSS0_CORE1_INTR_IN_45	45	MCU_CTRL_MMR0_IPC_SET1_IPC_SET_IPCFG_0
MCU_R5FSS0_CORE1_INTR_IN_46	46	MCU_CTRL_MMR0_ACCESS_ERR_0
MCU_R5FSS0_CORE1_INTR_IN_47	47	MCU_PBIST0_DFT_PBIST_CPU_0
MCU_R5FSS0_CORE1_INTR_IN_48	48	MCU_ESM0_ESM_INT_LOW_LVL_0
MCU_R5FSS0_CORE1_INTR_IN_49	49	MCU_ESM0_ESM_INT_HI_LVL_0



**Table 9-43. MCU\_R5FSS0\_CORE1 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
MCU_R5FSS0_CORE1_INTR_IN_50	50	MCU_DCC0_INTR_DONE_LEVEL_0
MCU_R5FSS0_CORE1_INTR_IN_51	51	MCU_DCC1_INTR_DONE_LEVEL_0
MCU_R5FSS0_CORE1_INTR_IN_52	52	MCU_DCC2_INTR_DONE_LEVEL_0
MCU_R5FSS0_CORE1_INTR_IN_53	53	MCU_ESM0_ESM_INT_CFG_LVL_0
MCU_R5FSS0_CORE1_INTR_IN_54	54	DEBUGSS1_AQCMPINTR_LEVEL_0
MCU_R5FSS0_CORE1_INTR_IN_55	55	DEBUGSS1_CTM_LEVEL_0
MCU_R5FSS0_CORE1_INTR_IN_56	56	MCU_R5FSS0_CORE1_VALIRQ_0
MCU_R5FSS0_CORE1_INTR_IN_57	57	MCU_R5FSS0_CORE1_VALFIQ_0
MCU_R5FSS0_CORE1_INTR_IN_58	58	MCU_R5FSS0_CORE0_CTI_0
MCU_R5FSS0_CORE1_INTR_IN_59	59	MCU_R5FSS0_CORE1_CTI_0
MCU_R5FSS0_CORE1_INTR_IN_60	60	MCU_R5FSS0_COMMRX_LEVEL_1_0
MCU_R5FSS0_CORE1_INTR_IN_61	61	MCU_R5FSS0_COMMTX_LEVEL_1_0
MCU_R5FSS0_CORE1_INTR_IN_64	64	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_0
MCU_R5FSS0_CORE1_INTR_IN_65	65	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_1
MCU_R5FSS0_CORE1_INTR_IN_66	66	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_2
MCU_R5FSS0_CORE1_INTR_IN_67	67	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_3
MCU_R5FSS0_CORE1_INTR_IN_68	68	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_4
MCU_R5FSS0_CORE1_INTR_IN_69	69	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_5
MCU_R5FSS0_CORE1_INTR_IN_70	70	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_6
MCU_R5FSS0_CORE1_INTR_IN_71	71	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_7
MCU_R5FSS0_CORE1_INTR_IN_72	72	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_8
MCU_R5FSS0_CORE1_INTR_IN_73	73	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_9
MCU_R5FSS0_CORE1_INTR_IN_74	74	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_10
MCU_R5FSS0_CORE1_INTR_IN_75	75	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_11
MCU_R5FSS0_CORE1_INTR_IN_76	76	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_12
MCU_R5FSS0_CORE1_INTR_IN_77	77	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_13
MCU_R5FSS0_CORE1_INTR_IN_78	78	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_14
MCU_R5FSS0_CORE1_INTR_IN_79	79	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_15
MCU_R5FSS0_CORE1_INTR_IN_80	80	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_16
MCU_R5FSS0_CORE1_INTR_IN_81	81	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_17
MCU_R5FSS0_CORE1_INTR_IN_82	82	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_18
MCU_R5FSS0_CORE1_INTR_IN_83	83	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_19
MCU_R5FSS0_CORE1_INTR_IN_84	84	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_20
MCU_R5FSS0_CORE1_INTR_IN_85	85	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_21
MCU_R5FSS0_CORE1_INTR_IN_86	86	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_22
MCU_R5FSS0_CORE1_INTR_IN_87	87	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_23
MCU_R5FSS0_CORE1_INTR_IN_88	88	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_24
MCU_R5FSS0_CORE1_INTR_IN_89	89	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_25
MCU_R5FSS0_CORE1_INTR_IN_90	90	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_26
MCU_R5FSS0_CORE1_INTR_IN_91	91	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_27
MCU_R5FSS0_CORE1_INTR_IN_92	92	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_28
MCU_R5FSS0_CORE1_INTR_IN_93	93	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_29
MCU_R5FSS0_CORE1_INTR_IN_94	94	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_30
MCU_R5FSS0_CORE1_INTR_IN_95	95	MCU_NAVSS0_INTR_ROUTER_0_OUTL_INTR_31



**Table 9-43. MCU\_R5FSS0\_CORE1 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
MCU_R5FSS0_CORE1_INTR_IN_96	96	WKUP_I2C0_POINTRPEND_0
MCU_R5FSS0_CORE1_INTR_IN_97	97	WKUP_UART0_USART_IRQ_0
MCU_R5FSS0_CORE1_INTR_IN_98	98	WKUP_ESM0_ESM_INT_LOW_LVL_0
MCU_R5FSS0_CORE1_INTR_IN_99	99	WKUP_ESM0_ESM_INT_HI_LVL_0
MCU_R5FSS0_CORE1_INTR_IN_100	100	WKUP_ESM0_ESM_INT_CFG_LVL_0
MCU_R5FSS0_CORE1_INTR_IN_102	102	WKUP_DMSC0_RAT_0_EXP_INTR_0
MCU_R5FSS0_CORE1_INTR_IN_103	103	WKUP_DMSC0_DBG_AUTH_0_DEBUG_AUTH_INTR_0
MCU_R5FSS0_CORE1_INTR_IN_104	104	WKUP_DMSC0_AES_0_HIB_PUBLIC_0
MCU_R5FSS0_CORE1_INTR_IN_105	105	WKUP_DMSC0_AES_0_HIB_SECURE_0
MCU_R5FSS0_CORE1_INTR_IN_106	106	WKUP_DMSC0_CORTEX_M3_0_SEC_OUT_0
MCU_R5FSS0_CORE1_INTR_IN_107	107	WKUP_DMSC0_CORTEX_M3_0_SEC_OUT_1
MCU_R5FSS0_CORE1_INTR_IN_108	108	MCU_TIMER4_INTR_PEND_0
MCU_R5FSS0_CORE1_INTR_IN_109	109	MCU_TIMER5_INTR_PEND_0
MCU_R5FSS0_CORE1_INTR_IN_110	110	MCU_TIMER6_INTR_PEND_0
MCU_R5FSS0_CORE1_INTR_IN_111	111	MCU_TIMER7_INTR_PEND_0
MCU_R5FSS0_CORE1_INTR_IN_112	112	MCU_TIMER8_INTR_PEND_0
MCU_R5FSS0_CORE1_INTR_IN_113	113	MCU_TIMER9_INTR_PEND_0
MCU_R5FSS0_CORE1_INTR_IN_119	119	WKUP_CTRL_MMR0_ACCESS_ERR_0
MCU_R5FSS0_CORE1_INTR_IN_120	120	WKUP_PORZ_SYNC0_MAIN_PORZ_LATCHED_0
MCU_R5FSS0_CORE1_INTR_IN_122	122	WKUP_RESETZ_SYNC0_MAIN_RESETZ_LATCHED_0
MCU_R5FSS0_CORE1_INTR_IN_124	124	WKUP_GPIOMUX_INTRTR0_OUTP_0
MCU_R5FSS0_CORE1_INTR_IN_125	125	WKUP_GPIOMUX_INTRTR0_OUTP_1
MCU_R5FSS0_CORE1_INTR_IN_126	126	WKUP_GPIOMUX_INTRTR0_OUTP_2
MCU_R5FSS0_CORE1_INTR_IN_127	127	WKUP_GPIOMUX_INTRTR0_OUTP_3
MCU_R5FSS0_CORE1_INTR_IN_128	128	WKUP_GPIOMUX_INTRTR0_OUTP_4
MCU_R5FSS0_CORE1_INTR_IN_129	129	WKUP_GPIOMUX_INTRTR0_OUTP_5
MCU_R5FSS0_CORE1_INTR_IN_130	130	WKUP_GPIOMUX_INTRTR0_OUTP_6
MCU_R5FSS0_CORE1_INTR_IN_131	131	WKUP_GPIOMUX_INTRTR0_OUTP_7
MCU_R5FSS0_CORE1_INTR_IN_132	132	WKUP_GPIOMUX_INTRTR0_OUTP_8
MCU_R5FSS0_CORE1_INTR_IN_133	133	WKUP_GPIOMUX_INTRTR0_OUTP_9
MCU_R5FSS0_CORE1_INTR_IN_134	134	WKUP_GPIOMUX_INTRTR0_OUTP_10
MCU_R5FSS0_CORE1_INTR_IN_135	135	WKUP_GPIOMUX_INTRTR0_OUTP_11
MCU_R5FSS0_CORE1_INTR_IN_136	136	WKUP_GPIOMUX_INTRTR0_OUTP_12
MCU_R5FSS0_CORE1_INTR_IN_137	137	WKUP_GPIOMUX_INTRTR0_OUTP_13
MCU_R5FSS0_CORE1_INTR_IN_138	138	WKUP_GPIOMUX_INTRTR0_OUTP_14
MCU_R5FSS0_CORE1_INTR_IN_139	139	WKUP_GPIOMUX_INTRTR0_OUTP_15
MCU_R5FSS0_CORE1_INTR_IN_140	140	ESM0_ESM_INT_LOW_LVL_0
MCU_R5FSS0_CORE1_INTR_IN_141	141	ESM0_ESM_INT_HI_LVL_0
MCU_R5FSS0_CORE1_INTR_IN_142	142	ESM0_ESM_INT_CFG_LVL_0
MCU_R5FSS0_CORE1_INTR_IN_144	144	COMPUTE_CLUSTER0_GIC_OUTPUT_WAKER_GIC_PWR0_WAKE_REQUEST_0
MCU_R5FSS0_CORE1_INTR_IN_145	145	COMPUTE_CLUSTER0_GIC_OUTPUT_WAKER_GIC_PWR0_WAKE_REQUEST_1
MCU_R5FSS0_CORE1_INTR_IN_146	146	MCU_R5FSS0_CORE0_EXP_INTR_0
MCU_R5FSS0_CORE1_INTR_IN_147	147	MCU_R5FSS0_CORE1_EXP_INTR_0
MCU_R5FSS0_CORE1_INTR_IN_148	148	MCU_CBASS0_LPSC_MCU_COMMON_ERR_INTR_0

**Table 9-43. MCU\_R5FSS0\_CORE1 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
MCU_R5FSS0_CORE1_INTR_IN_149	149	GLUELOGIC_DBG_CBASS_INTR_OR_GLUE_DBG_CBASS_AGG_ERR_INTR_0
MCU_R5FSS0_CORE1_INTR_IN_150	150	GLUELOGIC_FW_CBASS_INTR_OR_GLUE_FW_CBASS_AGG_ERR_INTR_0
MCU_R5FSS0_CORE1_INTR_IN_151	151	WKUP_CBASS0_LPSC_WKUP_COMMON_ERR_INTR_0
MCU_R5FSS0_CORE1_INTR_IN_156	156	WKUP_VTM0_THERM_LVL_GT_TH1_INTR_0
MCU_R5FSS0_CORE1_INTR_IN_157	157	WKUP_VTM0_THERM_LVL_LT_TH0_INTR_0
MCU_R5FSS0_CORE1_INTR_IN_158	158	WKUP_VTM0_THERM_LVL_GT_TH2_INTR_0
MCU_R5FSS0_CORE1_INTR_IN_160	160	MAIN2MCU_LVL_INTRTR0_OUTL_0
MCU_R5FSS0_CORE1_INTR_IN_161	161	MAIN2MCU_LVL_INTRTR0_OUTL_1
MCU_R5FSS0_CORE1_INTR_IN_162	162	MAIN2MCU_LVL_INTRTR0_OUTL_2
MCU_R5FSS0_CORE1_INTR_IN_163	163	MAIN2MCU_LVL_INTRTR0_OUTL_3
MCU_R5FSS0_CORE1_INTR_IN_164	164	MAIN2MCU_LVL_INTRTR0_OUTL_4
MCU_R5FSS0_CORE1_INTR_IN_165	165	MAIN2MCU_LVL_INTRTR0_OUTL_5
MCU_R5FSS0_CORE1_INTR_IN_166	166	MAIN2MCU_LVL_INTRTR0_OUTL_6
MCU_R5FSS0_CORE1_INTR_IN_167	167	MAIN2MCU_LVL_INTRTR0_OUTL_7
MCU_R5FSS0_CORE1_INTR_IN_168	168	MAIN2MCU_LVL_INTRTR0_OUTL_8
MCU_R5FSS0_CORE1_INTR_IN_169	169	MAIN2MCU_LVL_INTRTR0_OUTL_9
MCU_R5FSS0_CORE1_INTR_IN_170	170	MAIN2MCU_LVL_INTRTR0_OUTL_10
MCU_R5FSS0_CORE1_INTR_IN_171	171	MAIN2MCU_LVL_INTRTR0_OUTL_11
MCU_R5FSS0_CORE1_INTR_IN_172	172	MAIN2MCU_LVL_INTRTR0_OUTL_12
MCU_R5FSS0_CORE1_INTR_IN_173	173	MAIN2MCU_LVL_INTRTR0_OUTL_13
MCU_R5FSS0_CORE1_INTR_IN_174	174	MAIN2MCU_LVL_INTRTR0_OUTL_14
MCU_R5FSS0_CORE1_INTR_IN_175	175	MAIN2MCU_LVL_INTRTR0_OUTL_15
MCU_R5FSS0_CORE1_INTR_IN_176	176	MAIN2MCU_LVL_INTRTR0_OUTL_16
MCU_R5FSS0_CORE1_INTR_IN_177	177	MAIN2MCU_LVL_INTRTR0_OUTL_17
MCU_R5FSS0_CORE1_INTR_IN_178	178	MAIN2MCU_LVL_INTRTR0_OUTL_18
MCU_R5FSS0_CORE1_INTR_IN_179	179	MAIN2MCU_LVL_INTRTR0_OUTL_19
MCU_R5FSS0_CORE1_INTR_IN_180	180	MAIN2MCU_LVL_INTRTR0_OUTL_20
MCU_R5FSS0_CORE1_INTR_IN_181	181	MAIN2MCU_LVL_INTRTR0_OUTL_21
MCU_R5FSS0_CORE1_INTR_IN_182	182	MAIN2MCU_LVL_INTRTR0_OUTL_22
MCU_R5FSS0_CORE1_INTR_IN_183	183	MAIN2MCU_LVL_INTRTR0_OUTL_23
MCU_R5FSS0_CORE1_INTR_IN_184	184	MAIN2MCU_LVL_INTRTR0_OUTL_24
MCU_R5FSS0_CORE1_INTR_IN_185	185	MAIN2MCU_LVL_INTRTR0_OUTL_25
MCU_R5FSS0_CORE1_INTR_IN_186	186	MAIN2MCU_LVL_INTRTR0_OUTL_26
MCU_R5FSS0_CORE1_INTR_IN_187	187	MAIN2MCU_LVL_INTRTR0_OUTL_27
MCU_R5FSS0_CORE1_INTR_IN_188	188	MAIN2MCU_LVL_INTRTR0_OUTL_28
MCU_R5FSS0_CORE1_INTR_IN_189	189	MAIN2MCU_LVL_INTRTR0_OUTL_29
MCU_R5FSS0_CORE1_INTR_IN_190	190	MAIN2MCU_LVL_INTRTR0_OUTL_30
MCU_R5FSS0_CORE1_INTR_IN_191	191	MAIN2MCU_LVL_INTRTR0_OUTL_31
MCU_R5FSS0_CORE1_INTR_IN_192	192	MAIN2MCU_LVL_INTRTR0_OUTL_32
MCU_R5FSS0_CORE1_INTR_IN_193	193	MAIN2MCU_LVL_INTRTR0_OUTL_33
MCU_R5FSS0_CORE1_INTR_IN_194	194	MAIN2MCU_LVL_INTRTR0_OUTL_34
MCU_R5FSS0_CORE1_INTR_IN_195	195	MAIN2MCU_LVL_INTRTR0_OUTL_35
MCU_R5FSS0_CORE1_INTR_IN_196	196	MAIN2MCU_LVL_INTRTR0_OUTL_36
MCU_R5FSS0_CORE1_INTR_IN_197	197	MAIN2MCU_LVL_INTRTR0_OUTL_37

**Table 9-43. MCU\_R5FSS0\_CORE1 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
MCU_R5FSS0_CORE1_INTR_IN_198	198	MAIN2MCU_LVL_INTRTR0_OUTL_38
MCU_R5FSS0_CORE1_INTR_IN_199	199	MAIN2MCU_LVL_INTRTR0_OUTL_39
MCU_R5FSS0_CORE1_INTR_IN_200	200	MAIN2MCU_LVL_INTRTR0_OUTL_40
MCU_R5FSS0_CORE1_INTR_IN_201	201	MAIN2MCU_LVL_INTRTR0_OUTL_41
MCU_R5FSS0_CORE1_INTR_IN_202	202	MAIN2MCU_LVL_INTRTR0_OUTL_42
MCU_R5FSS0_CORE1_INTR_IN_203	203	MAIN2MCU_LVL_INTRTR0_OUTL_43
MCU_R5FSS0_CORE1_INTR_IN_204	204	MAIN2MCU_LVL_INTRTR0_OUTL_44
MCU_R5FSS0_CORE1_INTR_IN_205	205	MAIN2MCU_LVL_INTRTR0_OUTL_45
MCU_R5FSS0_CORE1_INTR_IN_206	206	MAIN2MCU_LVL_INTRTR0_OUTL_46
MCU_R5FSS0_CORE1_INTR_IN_207	207	MAIN2MCU_LVL_INTRTR0_OUTL_47
MCU_R5FSS0_CORE1_INTR_IN_208	208	MAIN2MCU_LVL_INTRTR0_OUTL_48
MCU_R5FSS0_CORE1_INTR_IN_209	209	MAIN2MCU_LVL_INTRTR0_OUTL_49
MCU_R5FSS0_CORE1_INTR_IN_210	210	MAIN2MCU_LVL_INTRTR0_OUTL_50
MCU_R5FSS0_CORE1_INTR_IN_211	211	MAIN2MCU_LVL_INTRTR0_OUTL_51
MCU_R5FSS0_CORE1_INTR_IN_212	212	MAIN2MCU_LVL_INTRTR0_OUTL_52
MCU_R5FSS0_CORE1_INTR_IN_213	213	MAIN2MCU_LVL_INTRTR0_OUTL_53
MCU_R5FSS0_CORE1_INTR_IN_214	214	MAIN2MCU_LVL_INTRTR0_OUTL_54
MCU_R5FSS0_CORE1_INTR_IN_215	215	MAIN2MCU_LVL_INTRTR0_OUTL_55
MCU_R5FSS0_CORE1_INTR_IN_216	216	MAIN2MCU_LVL_INTRTR0_OUTL_56
MCU_R5FSS0_CORE1_INTR_IN_217	217	MAIN2MCU_LVL_INTRTR0_OUTL_57
MCU_R5FSS0_CORE1_INTR_IN_218	218	MAIN2MCU_LVL_INTRTR0_OUTL_58
MCU_R5FSS0_CORE1_INTR_IN_219	219	MAIN2MCU_LVL_INTRTR0_OUTL_59
MCU_R5FSS0_CORE1_INTR_IN_220	220	MAIN2MCU_LVL_INTRTR0_OUTL_60
MCU_R5FSS0_CORE1_INTR_IN_221	221	MAIN2MCU_LVL_INTRTR0_OUTL_61
MCU_R5FSS0_CORE1_INTR_IN_222	222	MAIN2MCU_LVL_INTRTR0_OUTL_62
MCU_R5FSS0_CORE1_INTR_IN_223	223	MAIN2MCU_LVL_INTRTR0_OUTL_63
MCU_R5FSS0_CORE1_INTR_IN_224	224	MAIN2MCU_PLS_INTRTR0_OUTP_0
MCU_R5FSS0_CORE1_INTR_IN_225	225	MAIN2MCU_PLS_INTRTR0_OUTP_1
MCU_R5FSS0_CORE1_INTR_IN_226	226	MAIN2MCU_PLS_INTRTR0_OUTP_2
MCU_R5FSS0_CORE1_INTR_IN_227	227	MAIN2MCU_PLS_INTRTR0_OUTP_3
MCU_R5FSS0_CORE1_INTR_IN_228	228	MAIN2MCU_PLS_INTRTR0_OUTP_4
MCU_R5FSS0_CORE1_INTR_IN_229	229	MAIN2MCU_PLS_INTRTR0_OUTP_5
MCU_R5FSS0_CORE1_INTR_IN_230	230	MAIN2MCU_PLS_INTRTR0_OUTP_6
MCU_R5FSS0_CORE1_INTR_IN_231	231	MAIN2MCU_PLS_INTRTR0_OUTP_7
MCU_R5FSS0_CORE1_INTR_IN_232	232	MAIN2MCU_PLS_INTRTR0_OUTP_8
MCU_R5FSS0_CORE1_INTR_IN_233	233	MAIN2MCU_PLS_INTRTR0_OUTP_9
MCU_R5FSS0_CORE1_INTR_IN_234	234	MAIN2MCU_PLS_INTRTR0_OUTP_10
MCU_R5FSS0_CORE1_INTR_IN_235	235	MAIN2MCU_PLS_INTRTR0_OUTP_11
MCU_R5FSS0_CORE1_INTR_IN_236	236	MAIN2MCU_PLS_INTRTR0_OUTP_12
MCU_R5FSS0_CORE1_INTR_IN_237	237	MAIN2MCU_PLS_INTRTR0_OUTP_13
MCU_R5FSS0_CORE1_INTR_IN_238	238	MAIN2MCU_PLS_INTRTR0_OUTP_14
MCU_R5FSS0_CORE1_INTR_IN_239	239	MAIN2MCU_PLS_INTRTR0_OUTP_15
MCU_R5FSS0_CORE1_INTR_IN_240	240	MAIN2MCU_PLS_INTRTR0_OUTP_16
MCU_R5FSS0_CORE1_INTR_IN_241	241	MAIN2MCU_PLS_INTRTR0_OUTP_17

**Table 9-43. MCU\_R5FSS0\_CORE1 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
MCU_R5FSS0_CORE1_INTR_IN_242	242	MAIN2MCU_PLS_INTRTR0_OUTP_18
MCU_R5FSS0_CORE1_INTR_IN_243	243	MAIN2MCU_PLS_INTRTR0_OUTP_19
MCU_R5FSS0_CORE1_INTR_IN_244	244	MAIN2MCU_PLS_INTRTR0_OUTP_20
MCU_R5FSS0_CORE1_INTR_IN_245	245	MAIN2MCU_PLS_INTRTR0_OUTP_21
MCU_R5FSS0_CORE1_INTR_IN_246	246	MAIN2MCU_PLS_INTRTR0_OUTP_22
MCU_R5FSS0_CORE1_INTR_IN_247	247	MAIN2MCU_PLS_INTRTR0_OUTP_23
MCU_R5FSS0_CORE1_INTR_IN_248	248	MAIN2MCU_PLS_INTRTR0_OUTP_24
MCU_R5FSS0_CORE1_INTR_IN_249	249	MAIN2MCU_PLS_INTRTR0_OUTP_25
MCU_R5FSS0_CORE1_INTR_IN_250	250	MAIN2MCU_PLS_INTRTR0_OUTP_26
MCU_R5FSS0_CORE1_INTR_IN_251	251	MAIN2MCU_PLS_INTRTR0_OUTP_27
MCU_R5FSS0_CORE1_INTR_IN_252	252	MAIN2MCU_PLS_INTRTR0_OUTP_28
MCU_R5FSS0_CORE1_INTR_IN_253	253	MAIN2MCU_PLS_INTRTR0_OUTP_29
MCU_R5FSS0_CORE1_INTR_IN_254	254	MAIN2MCU_PLS_INTRTR0_OUTP_30
MCU_R5FSS0_CORE1_INTR_IN_255	255	MAIN2MCU_PLS_INTRTR0_OUTP_31
MCU_R5FSS0_CORE1_INTR_IN_256	256	MAIN2MCU_PLS_INTRTR0_OUTP_32
MCU_R5FSS0_CORE1_INTR_IN_257	257	MAIN2MCU_PLS_INTRTR0_OUTP_33
MCU_R5FSS0_CORE1_INTR_IN_258	258	MAIN2MCU_PLS_INTRTR0_OUTP_34
MCU_R5FSS0_CORE1_INTR_IN_259	259	MAIN2MCU_PLS_INTRTR0_OUTP_35
MCU_R5FSS0_CORE1_INTR_IN_260	260	MAIN2MCU_PLS_INTRTR0_OUTP_36
MCU_R5FSS0_CORE1_INTR_IN_261	261	MAIN2MCU_PLS_INTRTR0_OUTP_37
MCU_R5FSS0_CORE1_INTR_IN_262	262	MAIN2MCU_PLS_INTRTR0_OUTP_38
MCU_R5FSS0_CORE1_INTR_IN_263	263	MAIN2MCU_PLS_INTRTR0_OUTP_39
MCU_R5FSS0_CORE1_INTR_IN_264	264	MAIN2MCU_PLS_INTRTR0_OUTP_40
MCU_R5FSS0_CORE1_INTR_IN_265	265	MAIN2MCU_PLS_INTRTR0_OUTP_41
MCU_R5FSS0_CORE1_INTR_IN_266	266	MAIN2MCU_PLS_INTRTR0_OUTP_42
MCU_R5FSS0_CORE1_INTR_IN_267	267	MAIN2MCU_PLS_INTRTR0_OUTP_43
MCU_R5FSS0_CORE1_INTR_IN_268	268	MAIN2MCU_PLS_INTRTR0_OUTP_44
MCU_R5FSS0_CORE1_INTR_IN_269	269	MAIN2MCU_PLS_INTRTR0_OUTP_45
MCU_R5FSS0_CORE1_INTR_IN_270	270	MAIN2MCU_PLS_INTRTR0_OUTP_46
MCU_R5FSS0_CORE1_INTR_IN_271	271	MAIN2MCU_PLS_INTRTR0_OUTP_47
MCU_R5FSS0_CORE1_INTR_IN_288	288	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_8
MCU_R5FSS0_CORE1_INTR_IN_289	289	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_9
MCU_R5FSS0_CORE1_INTR_IN_290	290	COMPUTE_CLUSTER0_PBI2_WRAP_DFT_PBI2_CPU_0
MCU_R5FSS0_CORE1_INTR_IN_291	291	COMPUTE_CLUSTER0_ARM0_DFT_LBIST_DFT_LBIST_BIST_DONE_0
MCU_R5FSS0_CORE1_INTR_IN_293	293	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_12
MCU_R5FSS0_CORE1_INTR_IN_294	294	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_13
MCU_R5FSS0_CORE1_INTR_IN_295	295	COMPUTE_CLUSTER0_C7X_4_DFT_LBIST_DFT_LBIST_BIST_DONE_0
MCU_R5FSS0_CORE1_INTR_IN_297	297	GLUELOGIC_MAIN_PULSAR0_LBIST_GLUE_DFT_LBIST_BIST_DONE_0
MCU_R5FSS0_CORE1_INTR_IN_298	298	GLUELOGIC_MAIN_PULSAR1_LBIST_GLUE_DFT_LBIST_BIST_DONE_0
MCU_R5FSS0_CORE1_INTR_IN_299	299	GLUELOGIC_DMPAC_LBIST_GLUE_DFT_LBIST_BIST_DONE_0
MCU_R5FSS0_CORE1_INTR_IN_300	300	GLUELOGIC_VPAC_LBIST_GLUE_DFT_LBIST_BIST_DONE_0
MCU_R5FSS0_CORE1_INTR_IN_302	302	PBI2_DFT_PBI2_CPU_0
MCU_R5FSS0_CORE1_INTR_IN_303	303	PBI0_DFT_PBI2_CPU_0
MCU_R5FSS0_CORE1_INTR_IN_304	304	PBI1_DFT_PBI2_CPU_0

**Table 9-43. MCU\_R5FSS0\_CORE1 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
MCU_R5FSS0_CORE1_INTR_IN_305	305	PBIST3_DFT_PBIST_CPU_0
MCU_R5FSS0_CORE1_INTR_IN_306	306	PBIST4_DFT_PBIST_CPU_0
MCU_R5FSS0_CORE1_INTR_IN_307	307	PBIST5_DFT_PBIST_CPU_0
MCU_R5FSS0_CORE1_INTR_IN_308	308	PBIST7_DFT_PBIST_CPU_0
MCU_R5FSS0_CORE1_INTR_IN_309	309	PBIST9_DFT_PBIST_CPU_0
MCU_R5FSS0_CORE1_INTR_IN_310	310	PBIST10_DFT_PBIST_CPU_0
MCU_R5FSS0_CORE1_INTR_IN_312	312	PBIST6_DFT_PBIST_CPU_0
MCU_R5FSS0_CORE1_INTR_IN_315	315	C66SS0_PBIST0_DFT_PBIST_CPU_0
MCU_R5FSS0_CORE1_INTR_IN_316	316	C66SS1_PBIST0_DFT_PBIST_CPU_0
MCU_R5FSS0_CORE1_INTR_IN_317	317	GPU0_DFT_PBIST_0_DFT_PBIST_CPU_0
MCU_R5FSS0_CORE1_INTR_IN_320	320	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_0
MCU_R5FSS0_CORE1_INTR_IN_321	321	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_1
MCU_R5FSS0_CORE1_INTR_IN_322	322	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_2
MCU_R5FSS0_CORE1_INTR_IN_323	323	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_3
MCU_R5FSS0_CORE1_INTR_IN_324	324	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_4
MCU_R5FSS0_CORE1_INTR_IN_325	325	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_5
MCU_R5FSS0_CORE1_INTR_IN_326	326	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_6
MCU_R5FSS0_CORE1_INTR_IN_327	327	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_7
MCU_R5FSS0_CORE1_INTR_IN_328	328	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_8
MCU_R5FSS0_CORE1_INTR_IN_329	329	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_9
MCU_R5FSS0_CORE1_INTR_IN_330	330	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_10
MCU_R5FSS0_CORE1_INTR_IN_331	331	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_11
MCU_R5FSS0_CORE1_INTR_IN_332	332	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_12
MCU_R5FSS0_CORE1_INTR_IN_333	333	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_13
MCU_R5FSS0_CORE1_INTR_IN_334	334	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_14
MCU_R5FSS0_CORE1_INTR_IN_335	335	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_15
MCU_R5FSS0_CORE1_INTR_IN_336	336	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_16
MCU_R5FSS0_CORE1_INTR_IN_337	337	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_17
MCU_R5FSS0_CORE1_INTR_IN_338	338	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_18
MCU_R5FSS0_CORE1_INTR_IN_339	339	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_19
MCU_R5FSS0_CORE1_INTR_IN_340	340	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_20
MCU_R5FSS0_CORE1_INTR_IN_341	341	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_21
MCU_R5FSS0_CORE1_INTR_IN_342	342	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_22
MCU_R5FSS0_CORE1_INTR_IN_343	343	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_23
MCU_R5FSS0_CORE1_INTR_IN_344	344	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_24
MCU_R5FSS0_CORE1_INTR_IN_345	345	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_25
MCU_R5FSS0_CORE1_INTR_IN_346	346	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_26
MCU_R5FSS0_CORE1_INTR_IN_347	347	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_27
MCU_R5FSS0_CORE1_INTR_IN_348	348	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_28
MCU_R5FSS0_CORE1_INTR_IN_349	349	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_29
MCU_R5FSS0_CORE1_INTR_IN_350	350	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_30
MCU_R5FSS0_CORE1_INTR_IN_351	351	WKUP_DMSC0_INTAGGR_0_INTAGGR_VINTR_PEND_31
MCU_R5FSS0_CORE1_INTR_IN_376	376	NAVSS0_INTR_ROUTER_0_OUTL_INTR_400
MCU_R5FSS0_CORE1_INTR_IN_377	377	NAVSS0_INTR_ROUTER_0_OUTL_INTR_401

**Table 9-43. MCU\_R5FSS0\_CORE1 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
MCU_R5FSS0_CORE1_INTR_IN_378	378	NAVSS0_INTR_ROUTER_0_OUTL_INTR_402
MCU_R5FSS0_CORE1_INTR_IN_379	379	NAVSS0_INTR_ROUTER_0_OUTL_INTR_403
MCU_R5FSS0_CORE1_INTR_IN_380	380	NAVSS0_INTR_ROUTER_0_OUTL_INTR_404
MCU_R5FSS0_CORE1_INTR_IN_381	381	NAVSS0_INTR_ROUTER_0_OUTL_INTR_405
MCU_R5FSS0_CORE1_INTR_IN_382	382	NAVSS0_INTR_ROUTER_0_OUTL_INTR_406
MCU_R5FSS0_CORE1_INTR_IN_383	383	NAVSS0_INTR_ROUTER_0_OUTL_INTR_407

### 9.4.2.3 MCU\_ESM0 Interrupt Map

Table 9-44 shows the mapping of events to the MCU\_ESM0. Note that pulse event inputs are triple redundant so that there are three separate inputs for each pulse event (all to the same source signal) but only one connection is shown here.

**Table 9-44. MCU\_ESM0 Interrupt Map**

Interrupt Input Line	Interrupt ID	Interrupt Name
MCU_ESM0_LVL_IN_0	0	MCU_PLLFRAC2_SSMOD0_LOCKLOSS_IPCFG_0
MCU_ESM0_LVL_IN_1	1	MCU_PLLFRAC2_SSMOD1_LOCKLOSS_IPCFG_0
MCU_ESM0_LVL_IN_2	2	MCU_PLLFRAC2_SSMOD2_LOCKLOSS_IPCFG_0
MCU_ESM0_LVL_IN_10	10	MCU_ADC0_ECC_CORRECTED_ERR_LEVEL_0
MCU_ESM0_LVL_IN_11	11	MCU_ADC0_ECC_UNCORRECTED_ERR_LEVEL_0
MCU_ESM0_LVL_IN_12	12	MCU_ADC1_ECC_CORRECTED_ERR_LEVEL_0
MCU_ESM0_LVL_IN_13	13	MCU_ADC1_ECC_UNCORRECTED_ERR_LEVEL_0
MCU_ESM0_LVL_IN_14	14	MCU_CPSW0_ECC_SEC_PEND_0
MCU_ESM0_LVL_IN_15	15	MCU_CPSW0_ECC_DED_PEND_0
MCU_ESM0_LVL_IN_16	16	MCU_MCAN0_MCANSS_ECC_CORR_LVL_INT_0
MCU_ESM0_LVL_IN_17	17	MCU_MCAN0_MCANSS_ECC_UNCORR_LVL_INT_0
MCU_ESM0_LVL_IN_18	18	MCU_MCAN1_MCANSS_ECC_CORR_LVL_INT_0
MCU_ESM0_LVL_IN_19	19	MCU_MCAN1_MCANSS_ECC_UNCORR_LVL_INT_0
MCU_ESM0_LVL_IN_20	20	MCU_I3C0_PCLK_ECC_UNCORR_LVL_0
MCU_ESM0_LVL_IN_21	21	MCU_I3C0_SCLK_ECC_UNCORR_LVL_0
MCU_ESM0_LVL_IN_22	22	MCU_I3C1_PCLK_ECC_UNCORR_LVL_0
MCU_ESM0_LVL_IN_23	23	MCU_I3C1_SCLK_ECC_UNCORR_LVL_0
MCU_ESM0_LVL_IN_24	24	MCU_FSS0_OSPI_0_OSPI_ECC_CORR_LVL_INTR_0
MCU_ESM0_LVL_IN_25	25	MCU_FSS0_OSPI_0_OSPI_ECC_UNCORR_LVL_INTR_0
MCU_ESM0_LVL_IN_26	26	MCU_FSS0_OSPI_1_OSPI_ECC_CORR_LVL_INTR_0
MCU_ESM0_LVL_IN_27	27	MCU_FSS0_OSPI_1_OSPI_ECC_UNCORR_LVL_INTR_0
MCU_ESM0_LVL_IN_28	28	MCU_FSS0_HYPERBUS1P0_0_HPB_ECC_CORR_LEVEL_0
MCU_ESM0_LVL_IN_29	29	MCU_FSS0_HYPERBUS1P0_0_HPB_ECC_UNCORR_LEVEL_0
MCU_ESM0_LVL_IN_30	30	MCU_FSS0_MISC_0_ECC_INTR_ERR_PEND_0
MCU_ESM0_LVL_IN_31	31	MCU_FSS0_FSAS_0_ECC_INTR_ERR_PEND_0
MCU_ESM0_LVL_IN_32	32	MCU_R5FSS0_CORE0_ECC_CORRECTED_LEVEL_0
MCU_ESM0_LVL_IN_33	33	MCU_R5FSS0_CORE0_ECC_UNCORRECTED_LEVEL_0
MCU_ESM0_LVL_IN_34	34	MCU_R5FSS0_CORE1_ECC_CORRECTED_LEVEL_0
MCU_ESM0_LVL_IN_35	35	MCU_R5FSS0_CORE1_ECC_UNCORRECTED_LEVEL_0
MCU_ESM0_LVL_IN_36	36	MCU_R5FSS0_CORE0_EXP_INTR_0
MCU_ESM0_LVL_IN_37	37	MCU_R5FSS0_CORE1_EXP_INTR_0
MCU_ESM0_LVL_IN_40	40	MCU_NAVSS0_MODSS_ECC_AGGR_0_ECC_CORRECTED_ERR_LEVEL_0
MCU_ESM0_LVL_IN_41	41	MCU_NAVSS0_MODSS_ECC_AGGR_0_ECC_UNCORRECTED_ERR_LEVEL_0
MCU_ESM0_LVL_IN_42	42	MCU_NAVSS0_UDMASS_ECC_AGGR_0_ECC_CORRECTED_ERR_LEVEL_0
MCU_ESM0_LVL_IN_43	43	MCU_NAVSS0_UDMASS_ECC_AGGR_0_ECC_UNCORRECTED_ERR_LEVEL_0
MCU_ESM0_LVL_IN_54	54	MCU_TIMER0_INTR_PEND_0
MCU_ESM0_LVL_IN_55	55	MCU_TIMER1_INTR_PEND_0
MCU_ESM0_LVL_IN_56	56	MCU_TIMER2_INTR_PEND_0
MCU_ESM0_LVL_IN_57	57	MCU_TIMER3_INTR_PEND_0
MCU_ESM0_LVL_IN_58	58	MCU_TIMER4_INTR_PEND_0
MCU_ESM0_LVL_IN_59	59	MCU_TIMER5_INTR_PEND_0



**Table 9-44. MCU\_ESM0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
MCU_ESM0_LVL_IN_60	60	MCU_TIMER6_INTR_PEND_0
MCU_ESM0_LVL_IN_61	61	MCU_TIMER7_INTR_PEND_0
MCU_ESM0_LVL_IN_62	62	MCU_TIMER8_INTR_PEND_0
MCU_ESM0_LVL_IN_63	63	MCU_TIMER9_INTR_PEND_0
MCU_ESM0_LVL_IN_64	64	MCU_MSRAM_1MB0_ECC_CORR_LEVEL_0
MCU_ESM0_LVL_IN_65	65	MCU_MSRAM_1MB0_ECC_UNCORR_LEVEL_0
MCU_ESM0_LVL_IN_66	66	MCU_SA2_UL0_SA_UL_ECC_CORR_LEVEL_0
MCU_ESM0_LVL_IN_67	67	MCU_SA2_UL0_SA_UL_ECC_UNCORR_LEVEL_0
MCU_ESM0_LVL_IN_70	70	MCU_ECC_AGGR0_CORR_LEVEL_0
MCU_ESM0_LVL_IN_71	71	MCU_ECC_AGGR0_UNCORR_LEVEL_0
MCU_ESM0_LVL_IN_74	74	MCU_PBIST0_DFT_PBIST_SAFETY_ERROR_0
MCU_ESM0_LVL_IN_75	75	MCU_PBIST1_DFT_PBIST_SAFETY_ERROR_0
MCU_ESM0_LVL_IN_76	76	MCU_TIMEOUT_64B2_TRANS_ERR_LVL_0
MCU_ESM0_LVL_IN_77	77	MCU_TIMEOUT_INFRA0_SAFEG_TRANS_ERR_LVL_0
MCU_ESM0_LVL_IN_78	78	MCU_TIMEOUT_FW1_SAFEG_TRANS_ERR_LVL_0
MCU_ESM0_LVL_IN_80	80	MCU_I3C0_I3C_NONFATAL_INT_0
MCU_ESM0_LVL_IN_81	81	MCU_I3C0_I3C_FATAL_INT_0
MCU_ESM0_LVL_IN_82	82	MCU_I3C1_I3C_NONFATAL_INT_0
MCU_ESM0_LVL_IN_83	83	MCU_I3C1_I3C_FATAL_INT_0
MCU_ESM0_LVL_IN_84	84	MCU_RTIO_INTR_WWD_0
MCU_ESM0_LVL_IN_85	85	MCU_RT11_INTR_WWD_0
MCU_ESM0_LVL_IN_86	86	MCU_DCC0_INTR_ERR_LEVEL_0
MCU_ESM0_LVL_IN_87	87	MCU_DCC1_INTR_ERR_LEVEL_0
MCU_ESM0_LVL_IN_88	88	MCU_DCC2_INTR_ERR_LEVEL_0
MCU_ESM0_LVL_IN_95	95	GLUELOGIC_ESM_MAIN_ERR_GLUE_ERR_I_N_0
MCU_ESM0_PLS_IN_96	96	MCU_R5FSS0_SELFTEST_ERR_PULSE_0
MCU_ESM0_PLS_IN_97	97	MCU_R5FSS0_COMPARE_ERR_PULSE_0
MCU_ESM0_PLS_IN_98	98	MCU_R5FSS0_BUS_MONITOR_ERR_PULSE_0
MCU_ESM0_PLS_IN_99	99	MCU_R5FSS0_VIM_COMPARE_ERR_PULSE_0
MCU_ESM0_PLS_IN_100	100	MCU_R5FSS0_CCM_COMPARE_STAT_PULSE_INTR_0



### 9.4.3 MAIN Domain Interrupt Maps

#### 9.4.3.1 COMPUTE\_CLUSTER0 Interrupt Map

##### 9.4.3.1.1 GIC500 PPI Interrupt Map

[Table 9-45](#) shows the mapping of events to the GIC500 PPI inputs. There are 16 PPI events per A72 core.

PPI events can be configured for either low-level (default) or high-pulse operation.

PPI events represent events 16-31 of each A72 core.

**Table 9-45. GIC500 PPI Interrupt Map**

Interrupt Input Line	Interrupt ID	Interrupt Name
<b>PPI events for A72SS0_CORE0</b>		
GIC500_PPI0_0_IN_16	16	GLUELOGIC_A72_IPC_INTR_GLUE_IIPC_INTR_Z_8_0
GIC500_PPI0_0_IN_22	22	A72SS0_COMMIRQ0_0
GIC500_PPI0_0_IN_23	23	A72SS0_PMUIRQ0_0
GIC500_PPI0_0_IN_24	24	A72SS0_CTIIRQ0_0
GIC500_PPI0_0_IN_25	25	A72SS0_VCPUMNTIRQ0_0
GIC500_PPI0_0_IN_26	26	A72SS0_CNTHPIRQ0_0
GIC500_PPI0_0_IN_27	27	A72SS0_CNTVIRQ0_0
GIC500_PPI0_0_IN_29	29	A72SS0_CNTPSIRQ0_0
GIC500_PPI0_0_IN_30	30	A72SS0_CNTPNSIRQ0_0
<b>PPI events for A72SS0_CORE1</b>		
GIC500_PPI0_1_IN_16	16	GLUELOGIC_A72_IPC_INTR_GLUE_IIPC_INTR_Z_9_0
GIC500_PPI0_1_IN_22	22	A72SS0_COMMIRQ1_0
GIC500_PPI0_1_IN_23	23	A72SS0_PMUIRQ1_0
GIC500_PPI0_1_IN_24	24	A72SS0_CTIIRQ1_0
GIC500_PPI0_1_IN_25	25	A72SS0_VCPUMNTIRQ1_0
GIC500_PPI0_1_IN_26	26	A72SS0_CNTHPIRQ1_0
GIC500_PPI0_1_IN_27	27	A72SS0_CNTVIRQ1_0
GIC500_PPI0_1_IN_29	29	A72SS0_CNTPSIRQ1_0
GIC500_PPI0_1_IN_30	30	A72SS0_CNTPNSIRQ1_0

### 9.4.3.1.2 GIC500 SPI Interrupt Map

Table 9-46 shows the mapping of events to the GIC500 SPI inputs. SPI events may be mapped by the GIC500 to signal any (or both) of the A72 cores integrated in the Compute Cluster.

SPI events may be configured by software for either level (default) or pulse operation.

SPI events represent events 32-991 of each A72 core.

**Table 9-46. GIC500 SPI Interrupt Map**

Interrupt Input Line	Interrupt ID	Interrupt Name
GIC500_SPI_IN_32	32	ESM0_ESM_INT_CFG_LVL_0
GIC500_SPI_IN_33	33	ESM0_ESM_INT_HI_LVL_0
GIC500_SPI_IN_34	34	ESM0_ESM_INT_LOW_LVL_0
GIC500_SPI_IN_35	35	MMCSD0_EMMCSS_INTR_0
GIC500_SPI_IN_36	36	MMCSD1_EMMCSDSS_INTR_0
GIC500_SPI_IN_37	37	MMCSD2_EMMCSDSS_INTR_0
GIC500_SPI_IN_39	39	GLUELOGIC_EXT_INTN_GLUE_EXT_INT_LVL_0
GIC500_SPI_IN_40	40	GPMC0_GPMC_SINTERRUPT_0
GIC500_SPI_IN_41	41	ELM0_ELM_POROCPSINTERRUPT_LVL_0
GIC500_SPI_IN_42	42	SA2_UL0_SA_UL_PKA_0
GIC500_SPI_IN_43	43	SA2_UL0_SA_UL_TRNG_0
GIC500_SPI_IN_46	46	CPSW0_STAT_PEND_0
GIC500_SPI_IN_47	47	CPSW0_MDIO_PEND_0
GIC500_SPI_IN_48	48	CPSW0_EVTN_PEND_0
GIC500_SPI_IN_49	49	UFS0_UFS_INTR_0
GIC500_SPI_IN_54	54	GLUELOGIC_GPU_GPIO_REQACK_GLUE_GPU_GPIO_REQINT_LVL_0
GIC500_SPI_IN_55	55	GLUELOGIC_GPU_GPIO_REQACK_GLUE_GPU_GPIO_ACKINT_LVL_0
GIC500_SPI_IN_56	56	GPU0_MISC_0_IRQ_0
GIC500_SPI_IN_64	64	NAVSS0_INTR_ROUTER_0_OUTL_INTR_0
GIC500_SPI_IN_65	65	NAVSS0_INTR_ROUTER_0_OUTL_INTR_1
GIC500_SPI_IN_66	66	NAVSS0_INTR_ROUTER_0_OUTL_INTR_2
GIC500_SPI_IN_67	67	NAVSS0_INTR_ROUTER_0_OUTL_INTR_3
GIC500_SPI_IN_68	68	NAVSS0_INTR_ROUTER_0_OUTL_INTR_4
GIC500_SPI_IN_69	69	NAVSS0_INTR_ROUTER_0_OUTL_INTR_5
GIC500_SPI_IN_70	70	NAVSS0_INTR_ROUTER_0_OUTL_INTR_6
GIC500_SPI_IN_71	71	NAVSS0_INTR_ROUTER_0_OUTL_INTR_7
GIC500_SPI_IN_72	72	NAVSS0_INTR_ROUTER_0_OUTL_INTR_8
GIC500_SPI_IN_73	73	NAVSS0_INTR_ROUTER_0_OUTL_INTR_9
GIC500_SPI_IN_74	74	NAVSS0_INTR_ROUTER_0_OUTL_INTR_10
GIC500_SPI_IN_75	75	NAVSS0_INTR_ROUTER_0_OUTL_INTR_11
GIC500_SPI_IN_76	76	NAVSS0_INTR_ROUTER_0_OUTL_INTR_12
GIC500_SPI_IN_77	77	NAVSS0_INTR_ROUTER_0_OUTL_INTR_13
GIC500_SPI_IN_78	78	NAVSS0_INTR_ROUTER_0_OUTL_INTR_14
GIC500_SPI_IN_79	79	NAVSS0_INTR_ROUTER_0_OUTL_INTR_15
GIC500_SPI_IN_80	80	NAVSS0_INTR_ROUTER_0_OUTL_INTR_16
GIC500_SPI_IN_81	81	NAVSS0_INTR_ROUTER_0_OUTL_INTR_17
GIC500_SPI_IN_82	82	NAVSS0_INTR_ROUTER_0_OUTL_INTR_18
GIC500_SPI_IN_83	83	NAVSS0_INTR_ROUTER_0_OUTL_INTR_19
GIC500_SPI_IN_84	84	NAVSS0_INTR_ROUTER_0_OUTL_INTR_20

**Table 9-46. GIC500 SPI Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
GIC500_SPI_IN_85	85	NAVSS0_INTR_ROUTER_0_OUTL_INTR_21
GIC500_SPI_IN_86	86	NAVSS0_INTR_ROUTER_0_OUTL_INTR_22
GIC500_SPI_IN_87	87	NAVSS0_INTR_ROUTER_0_OUTL_INTR_23
GIC500_SPI_IN_88	88	NAVSS0_INTR_ROUTER_0_OUTL_INTR_24
GIC500_SPI_IN_89	89	NAVSS0_INTR_ROUTER_0_OUTL_INTR_25
GIC500_SPI_IN_90	90	NAVSS0_INTR_ROUTER_0_OUTL_INTR_26
GIC500_SPI_IN_91	91	NAVSS0_INTR_ROUTER_0_OUTL_INTR_27
GIC500_SPI_IN_92	92	NAVSS0_INTR_ROUTER_0_OUTL_INTR_28
GIC500_SPI_IN_93	93	NAVSS0_INTR_ROUTER_0_OUTL_INTR_29
GIC500_SPI_IN_94	94	NAVSS0_INTR_ROUTER_0_OUTL_INTR_30
GIC500_SPI_IN_95	95	NAVSS0_INTR_ROUTER_0_OUTL_INTR_31
GIC500_SPI_IN_96	96	NAVSS0_INTR_ROUTER_0_OUTL_INTR_32
GIC500_SPI_IN_97	97	NAVSS0_INTR_ROUTER_0_OUTL_INTR_33
GIC500_SPI_IN_98	98	NAVSS0_INTR_ROUTER_0_OUTL_INTR_34
GIC500_SPI_IN_99	99	NAVSS0_INTR_ROUTER_0_OUTL_INTR_35
GIC500_SPI_IN_100	100	NAVSS0_INTR_ROUTER_0_OUTL_INTR_36
GIC500_SPI_IN_101	101	NAVSS0_INTR_ROUTER_0_OUTL_INTR_37
GIC500_SPI_IN_102	102	NAVSS0_INTR_ROUTER_0_OUTL_INTR_38
GIC500_SPI_IN_103	103	NAVSS0_INTR_ROUTER_0_OUTL_INTR_39
GIC500_SPI_IN_104	104	NAVSS0_INTR_ROUTER_0_OUTL_INTR_40
GIC500_SPI_IN_105	105	NAVSS0_INTR_ROUTER_0_OUTL_INTR_41
GIC500_SPI_IN_106	106	NAVSS0_INTR_ROUTER_0_OUTL_INTR_42
GIC500_SPI_IN_107	107	NAVSS0_INTR_ROUTER_0_OUTL_INTR_43
GIC500_SPI_IN_108	108	NAVSS0_INTR_ROUTER_0_OUTL_INTR_44
GIC500_SPI_IN_109	109	NAVSS0_INTR_ROUTER_0_OUTL_INTR_45
GIC500_SPI_IN_110	110	NAVSS0_INTR_ROUTER_0_OUTL_INTR_46
GIC500_SPI_IN_111	111	NAVSS0_INTR_ROUTER_0_OUTL_INTR_47
GIC500_SPI_IN_112	112	NAVSS0_INTR_ROUTER_0_OUTL_INTR_48
GIC500_SPI_IN_113	113	NAVSS0_INTR_ROUTER_0_OUTL_INTR_49
GIC500_SPI_IN_114	114	NAVSS0_INTR_ROUTER_0_OUTL_INTR_50
GIC500_SPI_IN_115	115	NAVSS0_INTR_ROUTER_0_OUTL_INTR_51
GIC500_SPI_IN_116	116	NAVSS0_INTR_ROUTER_0_OUTL_INTR_52
GIC500_SPI_IN_117	117	NAVSS0_INTR_ROUTER_0_OUTL_INTR_53
GIC500_SPI_IN_118	118	NAVSS0_INTR_ROUTER_0_OUTL_INTR_54
GIC500_SPI_IN_119	119	NAVSS0_INTR_ROUTER_0_OUTL_INTR_55
GIC500_SPI_IN_120	120	NAVSS0_INTR_ROUTER_0_OUTL_INTR_56
GIC500_SPI_IN_121	121	NAVSS0_INTR_ROUTER_0_OUTL_INTR_57
GIC500_SPI_IN_122	122	NAVSS0_INTR_ROUTER_0_OUTL_INTR_58
GIC500_SPI_IN_123	123	NAVSS0_INTR_ROUTER_0_OUTL_INTR_59
GIC500_SPI_IN_124	124	NAVSS0_INTR_ROUTER_0_OUTL_INTR_60
GIC500_SPI_IN_125	125	NAVSS0_INTR_ROUTER_0_OUTL_INTR_61
GIC500_SPI_IN_126	126	NAVSS0_INTR_ROUTER_0_OUTL_INTR_62
GIC500_SPI_IN_127	127	NAVSS0_INTR_ROUTER_0_OUTL_INTR_63
GIC500_SPI_IN_128	128	USB0_IRQ_0
GIC500_SPI_IN_129	129	USB0_IRQ_1

**Table 9-46. GIC500 SPI Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
GIC500_SPI_IN_130	130	USB0_IRQ_2
GIC500_SPI_IN_131	131	USB0_IRQ_3
GIC500_SPI_IN_132	132	USB0_IRQ_4
GIC500_SPI_IN_133	133	USB0_IRQ_5
GIC500_SPI_IN_134	134	USB0_IRQ_6
GIC500_SPI_IN_135	135	USB0_IRQ_7
GIC500_SPI_IN_136	136	USB1_IRQ_0
GIC500_SPI_IN_137	137	USB1_IRQ_1
GIC500_SPI_IN_138	138	USB1_IRQ_2
GIC500_SPI_IN_139	139	USB1_IRQ_3
GIC500_SPI_IN_140	140	USB1_IRQ_4
GIC500_SPI_IN_141	141	USB1_IRQ_5
GIC500_SPI_IN_142	142	USB1_IRQ_6
GIC500_SPI_IN_143	143	USB1_IRQ_7
GIC500_SPI_IN_152	152	USB0_OTGIRQ_0
GIC500_SPI_IN_153	153	USB1_OTGIRQ_0
GIC500_SPI_IN_156	156	MCAN0_MCANSS_MCAN_LVL_INT_0
GIC500_SPI_IN_157	157	MCAN0_MCANSS_MCAN_LVL_INT_1
GIC500_SPI_IN_158	158	MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
GIC500_SPI_IN_159	159	MCAN1_MCANSS_MCAN_LVL_INT_0
GIC500_SPI_IN_160	160	MCAN1_MCANSS_MCAN_LVL_INT_1
GIC500_SPI_IN_161	161	MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
GIC500_SPI_IN_162	162	MCAN2_MCANSS_MCAN_LVL_INT_0
GIC500_SPI_IN_163	163	MCAN2_MCANSS_MCAN_LVL_INT_1
GIC500_SPI_IN_164	164	MCAN2_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
GIC500_SPI_IN_165	165	MCAN3_MCANSS_MCAN_LVL_INT_0
GIC500_SPI_IN_166	166	MCAN3_MCANSS_MCAN_LVL_INT_1
GIC500_SPI_IN_167	167	MCAN3_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
GIC500_SPI_IN_168	168	MCAN4_MCANSS_MCAN_LVL_INT_0
GIC500_SPI_IN_169	169	MCAN4_MCANSS_MCAN_LVL_INT_1
GIC500_SPI_IN_170	170	MCAN4_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
GIC500_SPI_IN_171	171	MCAN5_MCANSS_MCAN_LVL_INT_0
GIC500_SPI_IN_172	172	MCAN5_MCANSS_MCAN_LVL_INT_1
GIC500_SPI_IN_173	173	MCAN5_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
GIC500_SPI_IN_174	174	MCAN6_MCANSS_MCAN_LVL_INT_0
GIC500_SPI_IN_175	175	MCAN6_MCANSS_MCAN_LVL_INT_1
GIC500_SPI_IN_176	176	MCAN6_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
GIC500_SPI_IN_177	177	MCAN7_MCANSS_MCAN_LVL_INT_0
GIC500_SPI_IN_178	178	MCAN7_MCANSS_MCAN_LVL_INT_1
GIC500_SPI_IN_179	179	MCAN7_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
GIC500_SPI_IN_180	180	CSI_TX_IF0_CSI_INTERRUPT_0
GIC500_SPI_IN_181	181	CSI_TX_IF0_CSI_LEVEL_0
GIC500_SPI_IN_184	184	CSI_RX_IF0_CSI_IRQ_0
GIC500_SPI_IN_185	185	CSI_RX_IF0_CSI_ERR_IRQ_0
GIC500_SPI_IN_186	186	CSI_RX_IF0_CSI_LEVEL_0

**Table 9-46. GIC500 SPI Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
GIC500_SPI_IN_188	188	CSI_RX_IF1_CSI_IRQ_0
GIC500_SPI_IN_189	189	CSI_RX_IF1_CSI_ERR_IRQ_0
GIC500_SPI_IN_190	190	CSI_RX_IF1_CSI_LEVEL_0
GIC500_SPI_IN_200	200	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_0
GIC500_SPI_IN_201	201	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_1
GIC500_SPI_IN_202	202	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_2
GIC500_SPI_IN_203	203	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_3
GIC500_SPI_IN_204	204	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_4
GIC500_SPI_IN_205	205	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_5
GIC500_SPI_IN_206	206	DMPAC0_INTD_0_SYSTEM_INTR_LEVEL_0
GIC500_SPI_IN_207	207	DMPAC0_INTD_0_SYSTEM_INTR_LEVEL_1
GIC500_SPI_IN_210	210	VPFE0_CCDC_INTR_PEND_0
GIC500_SPI_IN_211	211	VPFE0_RAT_EXP_INTR_0
GIC500_SPI_IN_212	212	DECODER0_IRQ_0
GIC500_SPI_IN_213	213	ENCODER0_IRQ_0
GIC500_SPI_IN_216	216	MCSPi0_INTR_SPI_0
GIC500_SPI_IN_217	217	MCSPi1_INTR_SPI_0
GIC500_SPI_IN_218	218	MCSPi2_INTR_SPI_0
GIC500_SPI_IN_219	219	MCSPi3_INTR_SPI_0
GIC500_SPI_IN_220	220	MCSPi4_INTR_SPI_0
GIC500_SPI_IN_221	221	MCSPi5_INTR_SPI_0
GIC500_SPI_IN_222	222	MCSPi6_INTR_SPI_0
GIC500_SPI_IN_223	223	MCSPi7_INTR_SPI_0
GIC500_SPI_IN_224	224	UART0_USART_IRQ_0
GIC500_SPI_IN_225	225	UART1_USART_IRQ_0
GIC500_SPI_IN_226	226	UART2_USART_IRQ_0
GIC500_SPI_IN_227	227	UART3_USART_IRQ_0
GIC500_SPI_IN_228	228	UART4_USART_IRQ_0
GIC500_SPI_IN_229	229	UART5_USART_IRQ_0
GIC500_SPI_IN_230	230	UART6_USART_IRQ_0
GIC500_SPI_IN_231	231	UART7_USART_IRQ_0
GIC500_SPI_IN_232	232	I2C0_POINTRPEND_0
GIC500_SPI_IN_233	233	I2C1_POINTRPEND_0
GIC500_SPI_IN_234	234	I2C2_POINTRPEND_0
GIC500_SPI_IN_235	235	I2C3_POINTRPEND_0
GIC500_SPI_IN_236	236	I2C4_POINTRPEND_0
GIC500_SPI_IN_237	237	I2C5_POINTRPEND_0
GIC500_SPI_IN_238	238	I2C6_POINTRPEND_0
GIC500_SPI_IN_240	240	RTi0_INTR_WWD_0
GIC500_SPI_IN_241	241	RTi1_INTR_WWD_0
GIC500_SPI_IN_248	248	DDR0_DDRSS_CONTROLLER_0
GIC500_SPI_IN_249	249	DDR0_DDRSS_V2A_OTHER_ERR_LVL_0
GIC500_SPI_IN_250	250	DDR0_DDRSS_HS_PHY_GLOBAL_ERROR_0
GIC500_SPI_IN_251	251	DDR0_DDRSS_PLL_FREQ_CHANGE_REQ_0
GIC500_SPI_IN_256	256	TIMER0_INTR_PEND_0

**Table 9-46. GIC500 SPI Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
GIC500_SPI_IN_257	257	TIMER1_INTR_PEND_0
GIC500_SPI_IN_258	258	TIMER2_INTR_PEND_0
GIC500_SPI_IN_259	259	TIMER3_INTR_PEND_0
GIC500_SPI_IN_260	260	TIMER4_INTR_PEND_0
GIC500_SPI_IN_261	261	TIMER5_INTR_PEND_0
GIC500_SPI_IN_262	262	TIMER6_INTR_PEND_0
GIC500_SPI_IN_263	263	TIMER7_INTR_PEND_0
GIC500_SPI_IN_264	264	TIMER8_INTR_PEND_0
GIC500_SPI_IN_265	265	TIMER9_INTR_PEND_0
GIC500_SPI_IN_266	266	TIMER10_INTR_PEND_0
GIC500_SPI_IN_267	267	TIMER11_INTR_PEND_0
GIC500_SPI_IN_268	268	TIMER12_INTR_PEND_0
GIC500_SPI_IN_269	269	TIMER13_INTR_PEND_0
GIC500_SPI_IN_270	270	TIMER14_INTR_PEND_0
GIC500_SPI_IN_271	271	TIMER15_INTR_PEND_0
GIC500_SPI_IN_272	272	TIMER16_INTR_PEND_0
GIC500_SPI_IN_273	273	TIMER17_INTR_PEND_0
GIC500_SPI_IN_274	274	TIMER18_INTR_PEND_0
GIC500_SPI_IN_275	275	TIMER19_INTR_PEND_0
GIC500_SPI_IN_280	280	UART8_USART_IRQ_0
GIC500_SPI_IN_281	281	UART9_USART_IRQ_0
GIC500_SPI_IN_282	282	GLUELOGIC_SOCA_INT_GLUE_SOCA_INT_0
GIC500_SPI_IN_283	283	GLUELOGIC_SOCB_INT_GLUE_SOCB_INT_0
GIC500_SPI_IN_284	284	I3C0_I3C_INT_0
GIC500_SPI_IN_286	286	PRU_ICSSG0_PR1_HOST_INTR_PEND_0
GIC500_SPI_IN_287	287	PRU_ICSSG0_PR1_HOST_INTR_PEND_1
GIC500_SPI_IN_288	288	PRU_ICSSG0_PR1_HOST_INTR_PEND_2
GIC500_SPI_IN_289	289	PRU_ICSSG0_PR1_HOST_INTR_PEND_3
GIC500_SPI_IN_290	290	PRU_ICSSG0_PR1_HOST_INTR_PEND_4
GIC500_SPI_IN_291	291	PRU_ICSSG0_PR1_HOST_INTR_PEND_5
GIC500_SPI_IN_292	292	PRU_ICSSG0_PR1_HOST_INTR_PEND_6
GIC500_SPI_IN_293	293	PRU_ICSSG0_PR1_HOST_INTR_PEND_7
GIC500_SPI_IN_294	294	PRU_ICSSG1_PR1_HOST_INTR_PEND_0
GIC500_SPI_IN_295	295	PRU_ICSSG1_PR1_HOST_INTR_PEND_1
GIC500_SPI_IN_296	296	PRU_ICSSG1_PR1_HOST_INTR_PEND_2
GIC500_SPI_IN_297	297	PRU_ICSSG1_PR1_HOST_INTR_PEND_3
GIC500_SPI_IN_298	298	PRU_ICSSG1_PR1_HOST_INTR_PEND_4
GIC500_SPI_IN_299	299	PRU_ICSSG1_PR1_HOST_INTR_PEND_5
GIC500_SPI_IN_300	300	PRU_ICSSG1_PR1_HOST_INTR_PEND_6
GIC500_SPI_IN_301	301	PRU_ICSSG1_PR1_HOST_INTR_PEND_7
GIC500_SPI_IN_302	302	PRU_ICSSG0_PR1_TX_SOF_INTR_REQ_0
GIC500_SPI_IN_303	303	PRU_ICSSG0_PR1_TX_SOF_INTR_REQ_1
GIC500_SPI_IN_304	304	PRU_ICSSG0_PR1_RX_SOF_INTR_REQ_0
GIC500_SPI_IN_305	305	PRU_ICSSG0_PR1_RX_SOF_INTR_REQ_1
GIC500_SPI_IN_306	306	PRU_ICSSG1_PR1_TX_SOF_INTR_REQ_0

**Table 9-46. GIC500 SPI Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
GIC500_SPI_IN_307	307	PRU_ICSSG1_PR1_TX_SOF_INTR_REQ_1
GIC500_SPI_IN_308	308	PRU_ICSSG1_PR1_RX_SOF_INTR_REQ_0
GIC500_SPI_IN_309	309	PRU_ICSSG1_PR1_RX_SOF_INTR_REQ_1
GIC500_SPI_IN_310	310	EHRPWM0_EPWM_ETINT_0
GIC500_SPI_IN_311	311	EHRPWM1_EPWM_ETINT_0
GIC500_SPI_IN_312	312	EHRPWM2_EPWM_ETINT_0
GIC500_SPI_IN_313	313	EHRPWM3_EPWM_ETINT_0
GIC500_SPI_IN_314	314	EHRPWM4_EPWM_ETINT_0
GIC500_SPI_IN_315	315	EHRPWM5_EPWM_ETINT_0
GIC500_SPI_IN_316	316	EHRPWM0_EPWM_TRIPZINT_0
GIC500_SPI_IN_317	317	EHRPWM1_EPWM_TRIPZINT_0
GIC500_SPI_IN_318	318	EHRPWM2_EPWM_TRIPZINT_0
GIC500_SPI_IN_319	319	EHRPWM3_EPWM_TRIPZINT_0
GIC500_SPI_IN_320	320	EHRPWM4_EPWM_TRIPZINT_0
GIC500_SPI_IN_321	321	EHRPWM5_EPWM_TRIPZINT_0
GIC500_SPI_IN_322	322	EQEP0_EQEP_INT_0
GIC500_SPI_IN_323	323	EQEP1_EQEP_INT_0
GIC500_SPI_IN_324	324	EQEP2_EQEP_INT_0
GIC500_SPI_IN_325	325	ECAP0_ECAP_INT_0
GIC500_SPI_IN_326	326	ECAP1_ECAP_INT_0
GIC500_SPI_IN_327	327	ECAP2_ECAP_INT_0
GIC500_SPI_IN_328	328	DCC0_INTR_DONE_LEVEL_0
GIC500_SPI_IN_329	329	DCC1_INTR_DONE_LEVEL_0
GIC500_SPI_IN_330	330	DCC2_INTR_DONE_LEVEL_0
GIC500_SPI_IN_331	331	DCC3_INTR_DONE_LEVEL_0
GIC500_SPI_IN_332	332	DCC4_INTR_DONE_LEVEL_0
GIC500_SPI_IN_333	333	DCC5_INTR_DONE_LEVEL_0
GIC500_SPI_IN_334	334	DCC6_INTR_DONE_LEVEL_0
GIC500_SPI_IN_335	335	DCC7_INTR_DONE_LEVEL_0
GIC500_SPI_IN_336	336	DCC8_INTR_DONE_LEVEL_0
GIC500_SPI_IN_337	337	DCC9_INTR_DONE_LEVEL_0
GIC500_SPI_IN_338	338	DCC10_INTR_DONE_LEVEL_0
GIC500_SPI_IN_339	339	DCC11_INTR_DONE_LEVEL_0
GIC500_SPI_IN_340	340	DCC12_INTR_DONE_LEVEL_0
GIC500_SPI_IN_344	344	PCIE0_PCIE_LEGACY_PULSE_0
GIC500_SPI_IN_345	345	PCIE0_PCIE_DOWNSTREAM_PULSE_0
GIC500_SPI_IN_346	346	PCIE0_PCIE_FLR_PULSE_0
GIC500_SPI_IN_347	347	PCIE0_PCIE_PHY_LEVEL_0
GIC500_SPI_IN_348	348	PCIE0_PCIE_LOCAL_LEVEL_0
GIC500_SPI_IN_349	349	PCIE0_PCIE_ERROR_PULSE_0
GIC500_SPI_IN_350	350	PCIE0_PCIE_LINK_STATE_PULSE_0
GIC500_SPI_IN_351	351	PCIE0_PCIE_PWR_STATE_PULSE_0
GIC500_SPI_IN_352	352	PCIE0_PCIE_PTM_VALID_PULSE_0
GIC500_SPI_IN_353	353	PCIE0_PCIE_HOT_RESET_PULSE_0
GIC500_SPI_IN_354	354	PCIE0_PCIE_CPTS_PEND_0

**Table 9-46. GIC500 SPI Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
GIC500_SPI_IN_356	356	PCIE1_PCIE_LEGACY_PULSE_0
GIC500_SPI_IN_357	357	PCIE1_PCIE_DOWNSTREAM_PULSE_0
GIC500_SPI_IN_358	358	PCIE1_PCIE_FLR_PULSE_0
GIC500_SPI_IN_359	359	PCIE1_PCIE_PHY_LEVEL_0
GIC500_SPI_IN_360	360	PCIE1_PCIE_LOCAL_LEVEL_0
GIC500_SPI_IN_361	361	PCIE1_PCIE_ERROR_PULSE_0
GIC500_SPI_IN_362	362	PCIE1_PCIE_LINK_STATE_PULSE_0
GIC500_SPI_IN_363	363	PCIE1_PCIE_PWR_STATE_PULSE_0
GIC500_SPI_IN_364	364	PCIE1_PCIE_PTM_VALID_PULSE_0
GIC500_SPI_IN_365	365	PCIE1_PCIE_HOT_RESET_PULSE_0
GIC500_SPI_IN_366	366	PCIE1_PCIE_CPTS_PEND_0
GIC500_SPI_IN_368	368	PCIE2_PCIE_LEGACY_PULSE_0
GIC500_SPI_IN_369	369	PCIE2_PCIE_DOWNSTREAM_PULSE_0
GIC500_SPI_IN_370	370	PCIE2_PCIE_FLR_PULSE_0
GIC500_SPI_IN_371	371	PCIE2_PCIE_PHY_LEVEL_0
GIC500_SPI_IN_372	372	PCIE2_PCIE_LOCAL_LEVEL_0
GIC500_SPI_IN_373	373	PCIE2_PCIE_ERROR_PULSE_0
GIC500_SPI_IN_374	374	PCIE2_PCIE_LINK_STATE_PULSE_0
GIC500_SPI_IN_375	375	PCIE2_PCIE_PWR_STATE_PULSE_0
GIC500_SPI_IN_376	376	PCIE2_PCIE_PTM_VALID_PULSE_0
GIC500_SPI_IN_377	377	PCIE2_PCIE_HOT_RESET_PULSE_0
GIC500_SPI_IN_378	378	PCIE2_PCIE_CPTS_PEND_0
GIC500_SPI_IN_380	380	PCIE3_PCIE_LEGACY_PULSE_0
GIC500_SPI_IN_381	381	PCIE3_PCIE_DOWNSTREAM_PULSE_0
GIC500_SPI_IN_382	382	PCIE3_PCIE_FLR_PULSE_0
GIC500_SPI_IN_383	383	PCIE3_PCIE_PHY_LEVEL_0
GIC500_SPI_IN_384	384	PCIE3_PCIE_LOCAL_LEVEL_0
GIC500_SPI_IN_385	385	PCIE3_PCIE_ERROR_PULSE_0
GIC500_SPI_IN_386	386	PCIE3_PCIE_LINK_STATE_PULSE_0
GIC500_SPI_IN_387	387	PCIE3_PCIE_PWR_STATE_PULSE_0
GIC500_SPI_IN_388	388	PCIE3_PCIE_PTM_VALID_PULSE_0
GIC500_SPI_IN_389	389	PCIE3_PCIE_HOT_RESET_PULSE_0
GIC500_SPI_IN_390	390	PCIE3_PCIE_CPTS_PEND_0
GIC500_SPI_IN_392	392	GPIOMUX_INTRTR0_OUTP_8
GIC500_SPI_IN_393	393	GPIOMUX_INTRTR0_OUTP_9
GIC500_SPI_IN_394	394	GPIOMUX_INTRTR0_OUTP_10
GIC500_SPI_IN_395	395	GPIOMUX_INTRTR0_OUTP_11
GIC500_SPI_IN_396	396	GPIOMUX_INTRTR0_OUTP_12
GIC500_SPI_IN_397	397	GPIOMUX_INTRTR0_OUTP_13
GIC500_SPI_IN_398	398	GPIOMUX_INTRTR0_OUTP_14
GIC500_SPI_IN_399	399	GPIOMUX_INTRTR0_OUTP_15
GIC500_SPI_IN_400	400	GPIOMUX_INTRTR0_OUTP_16
GIC500_SPI_IN_401	401	GPIOMUX_INTRTR0_OUTP_17
GIC500_SPI_IN_402	402	GPIOMUX_INTRTR0_OUTP_18
GIC500_SPI_IN_403	403	GPIOMUX_INTRTR0_OUTP_19



**Table 9-46. GIC500 SPI Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
GIC500_SPI_IN_404	404	GPIOMUX_INTRTR0_OUTP_20
GIC500_SPI_IN_405	405	GPIOMUX_INTRTR0_OUTP_21
GIC500_SPI_IN_406	406	GPIOMUX_INTRTR0_OUTP_22
GIC500_SPI_IN_407	407	GPIOMUX_INTRTR0_OUTP_23
GIC500_SPI_IN_408	408	GPIOMUX_INTRTR0_OUTP_24
GIC500_SPI_IN_409	409	GPIOMUX_INTRTR0_OUTP_25
GIC500_SPI_IN_410	410	GPIOMUX_INTRTR0_OUTP_26
GIC500_SPI_IN_411	411	GPIOMUX_INTRTR0_OUTP_27
GIC500_SPI_IN_412	412	GPIOMUX_INTRTR0_OUTP_28
GIC500_SPI_IN_413	413	GPIOMUX_INTRTR0_OUTP_29
GIC500_SPI_IN_414	414	GPIOMUX_INTRTR0_OUTP_30
GIC500_SPI_IN_415	415	GPIOMUX_INTRTR0_OUTP_31
GIC500_SPI_IN_416	416	GPIOMUX_INTRTR0_OUTP_32
GIC500_SPI_IN_417	417	GPIOMUX_INTRTR0_OUTP_33
GIC500_SPI_IN_418	418	GPIOMUX_INTRTR0_OUTP_34
GIC500_SPI_IN_419	419	GPIOMUX_INTRTR0_OUTP_35
GIC500_SPI_IN_420	420	GPIOMUX_INTRTR0_OUTP_36
GIC500_SPI_IN_421	421	GPIOMUX_INTRTR0_OUTP_37
GIC500_SPI_IN_422	422	GPIOMUX_INTRTR0_OUTP_38
GIC500_SPI_IN_423	423	GPIOMUX_INTRTR0_OUTP_39
GIC500_SPI_IN_424	424	GPIOMUX_INTRTR0_OUTP_40
GIC500_SPI_IN_425	425	GPIOMUX_INTRTR0_OUTP_41
GIC500_SPI_IN_426	426	GPIOMUX_INTRTR0_OUTP_42
GIC500_SPI_IN_427	427	GPIOMUX_INTRTR0_OUTP_43
GIC500_SPI_IN_428	428	GPIOMUX_INTRTR0_OUTP_44
GIC500_SPI_IN_429	429	GPIOMUX_INTRTR0_OUTP_45
GIC500_SPI_IN_430	430	GPIOMUX_INTRTR0_OUTP_46
GIC500_SPI_IN_431	431	GPIOMUX_INTRTR0_OUTP_47
GIC500_SPI_IN_432	432	GPIOMUX_INTRTR0_OUTP_48
GIC500_SPI_IN_433	433	GPIOMUX_INTRTR0_OUTP_49
GIC500_SPI_IN_434	434	GPIOMUX_INTRTR0_OUTP_50
GIC500_SPI_IN_435	435	GPIOMUX_INTRTR0_OUTP_51
GIC500_SPI_IN_436	436	GPIOMUX_INTRTR0_OUTP_52
GIC500_SPI_IN_437	437	GPIOMUX_INTRTR0_OUTP_53
GIC500_SPI_IN_438	438	GPIOMUX_INTRTR0_OUTP_54
GIC500_SPI_IN_439	439	GPIOMUX_INTRTR0_OUTP_55
GIC500_SPI_IN_440	440	GPIOMUX_INTRTR0_OUTP_56
GIC500_SPI_IN_441	441	GPIOMUX_INTRTR0_OUTP_57
GIC500_SPI_IN_442	442	GPIOMUX_INTRTR0_OUTP_58
GIC500_SPI_IN_443	443	GPIOMUX_INTRTR0_OUTP_59
GIC500_SPI_IN_444	444	GPIOMUX_INTRTR0_OUTP_60
GIC500_SPI_IN_445	445	GPIOMUX_INTRTR0_OUTP_61
GIC500_SPI_IN_446	446	GPIOMUX_INTRTR0_OUTP_62
GIC500_SPI_IN_447	447	GPIOMUX_INTRTR0_OUTP_63
GIC500_SPI_IN_448	448	NAVSS0_INTR_ROUTER_0_OUTL_INTR_64

**Table 9-46. GIC500 SPI Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
GIC500_SPI_IN_449	449	NAVSS0_INTR_ROUTER_0_OUTL_INTR_65
GIC500_SPI_IN_450	450	NAVSS0_INTR_ROUTER_0_OUTL_INTR_66
GIC500_SPI_IN_451	451	NAVSS0_INTR_ROUTER_0_OUTL_INTR_67
GIC500_SPI_IN_452	452	NAVSS0_INTR_ROUTER_0_OUTL_INTR_68
GIC500_SPI_IN_453	453	NAVSS0_INTR_ROUTER_0_OUTL_INTR_69
GIC500_SPI_IN_454	454	NAVSS0_INTR_ROUTER_0_OUTL_INTR_70
GIC500_SPI_IN_455	455	NAVSS0_INTR_ROUTER_0_OUTL_INTR_71
GIC500_SPI_IN_456	456	NAVSS0_INTR_ROUTER_0_OUTL_INTR_72
GIC500_SPI_IN_457	457	NAVSS0_INTR_ROUTER_0_OUTL_INTR_73
GIC500_SPI_IN_458	458	NAVSS0_INTR_ROUTER_0_OUTL_INTR_74
GIC500_SPI_IN_459	459	NAVSS0_INTR_ROUTER_0_OUTL_INTR_75
GIC500_SPI_IN_460	460	NAVSS0_INTR_ROUTER_0_OUTL_INTR_76
GIC500_SPI_IN_461	461	NAVSS0_INTR_ROUTER_0_OUTL_INTR_77
GIC500_SPI_IN_462	462	NAVSS0_INTR_ROUTER_0_OUTL_INTR_78
GIC500_SPI_IN_463	463	NAVSS0_INTR_ROUTER_0_OUTL_INTR_79
GIC500_SPI_IN_464	464	NAVSS0_INTR_ROUTER_0_OUTL_INTR_80
GIC500_SPI_IN_465	465	NAVSS0_INTR_ROUTER_0_OUTL_INTR_81
GIC500_SPI_IN_466	466	NAVSS0_INTR_ROUTER_0_OUTL_INTR_82
GIC500_SPI_IN_467	467	NAVSS0_INTR_ROUTER_0_OUTL_INTR_83
GIC500_SPI_IN_468	468	NAVSS0_INTR_ROUTER_0_OUTL_INTR_84
GIC500_SPI_IN_469	469	NAVSS0_INTR_ROUTER_0_OUTL_INTR_85
GIC500_SPI_IN_470	470	NAVSS0_INTR_ROUTER_0_OUTL_INTR_86
GIC500_SPI_IN_471	471	NAVSS0_INTR_ROUTER_0_OUTL_INTR_87
GIC500_SPI_IN_472	472	NAVSS0_INTR_ROUTER_0_OUTL_INTR_88
GIC500_SPI_IN_473	473	NAVSS0_INTR_ROUTER_0_OUTL_INTR_89
GIC500_SPI_IN_474	474	NAVSS0_INTR_ROUTER_0_OUTL_INTR_90
GIC500_SPI_IN_475	475	NAVSS0_INTR_ROUTER_0_OUTL_INTR_91
GIC500_SPI_IN_476	476	NAVSS0_INTR_ROUTER_0_OUTL_INTR_92
GIC500_SPI_IN_477	477	NAVSS0_INTR_ROUTER_0_OUTL_INTR_93
GIC500_SPI_IN_478	478	NAVSS0_INTR_ROUTER_0_OUTL_INTR_94
GIC500_SPI_IN_479	479	NAVSS0_INTR_ROUTER_0_OUTL_INTR_95
GIC500_SPI_IN_480	480	NAVSS0_INTR_ROUTER_0_OUTL_INTR_96
GIC500_SPI_IN_481	481	NAVSS0_INTR_ROUTER_0_OUTL_INTR_97
GIC500_SPI_IN_482	482	NAVSS0_INTR_ROUTER_0_OUTL_INTR_98
GIC500_SPI_IN_483	483	NAVSS0_INTR_ROUTER_0_OUTL_INTR_99
GIC500_SPI_IN_484	484	NAVSS0_INTR_ROUTER_0_OUTL_INTR_100
GIC500_SPI_IN_485	485	NAVSS0_INTR_ROUTER_0_OUTL_INTR_101
GIC500_SPI_IN_486	486	NAVSS0_INTR_ROUTER_0_OUTL_INTR_102
GIC500_SPI_IN_487	487	NAVSS0_INTR_ROUTER_0_OUTL_INTR_103
GIC500_SPI_IN_488	488	NAVSS0_INTR_ROUTER_0_OUTL_INTR_104
GIC500_SPI_IN_489	489	NAVSS0_INTR_ROUTER_0_OUTL_INTR_105
GIC500_SPI_IN_490	490	NAVSS0_INTR_ROUTER_0_OUTL_INTR_106
GIC500_SPI_IN_491	491	NAVSS0_INTR_ROUTER_0_OUTL_INTR_107
GIC500_SPI_IN_492	492	NAVSS0_INTR_ROUTER_0_OUTL_INTR_108
GIC500_SPI_IN_493	493	NAVSS0_INTR_ROUTER_0_OUTL_INTR_109

**Table 9-46. GIC500 SPI Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
GIC500_SPI_IN_494	494	NAVSS0_INTR_ROUTER_0_OUTL_INTR_110
GIC500_SPI_IN_495	495	NAVSS0_INTR_ROUTER_0_OUTL_INTR_111
GIC500_SPI_IN_496	496	NAVSS0_INTR_ROUTER_0_OUTL_INTR_112
GIC500_SPI_IN_497	497	NAVSS0_INTR_ROUTER_0_OUTL_INTR_113
GIC500_SPI_IN_498	498	NAVSS0_INTR_ROUTER_0_OUTL_INTR_114
GIC500_SPI_IN_499	499	NAVSS0_INTR_ROUTER_0_OUTL_INTR_115
GIC500_SPI_IN_500	500	NAVSS0_INTR_ROUTER_0_OUTL_INTR_116
GIC500_SPI_IN_501	501	NAVSS0_INTR_ROUTER_0_OUTL_INTR_117
GIC500_SPI_IN_502	502	NAVSS0_INTR_ROUTER_0_OUTL_INTR_118
GIC500_SPI_IN_503	503	NAVSS0_INTR_ROUTER_0_OUTL_INTR_119
GIC500_SPI_IN_504	504	NAVSS0_INTR_ROUTER_0_OUTL_INTR_120
GIC500_SPI_IN_505	505	NAVSS0_INTR_ROUTER_0_OUTL_INTR_121
GIC500_SPI_IN_506	506	NAVSS0_INTR_ROUTER_0_OUTL_INTR_122
GIC500_SPI_IN_507	507	NAVSS0_INTR_ROUTER_0_OUTL_INTR_123
GIC500_SPI_IN_508	508	NAVSS0_INTR_ROUTER_0_OUTL_INTR_124
GIC500_SPI_IN_509	509	NAVSS0_INTR_ROUTER_0_OUTL_INTR_125
GIC500_SPI_IN_510	510	NAVSS0_INTR_ROUTER_0_OUTL_INTR_126
GIC500_SPI_IN_511	511	NAVSS0_INTR_ROUTER_0_OUTL_INTR_127
GIC500_SPI_IN_512	512	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_0
GIC500_SPI_IN_513	513	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_1
GIC500_SPI_IN_514	514	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_2
GIC500_SPI_IN_515	515	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_3
GIC500_SPI_IN_516	516	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_4
GIC500_SPI_IN_517	517	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_5
GIC500_SPI_IN_518	518	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_6
GIC500_SPI_IN_519	519	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_7
GIC500_SPI_IN_520	520	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_8
GIC500_SPI_IN_521	521	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_9
GIC500_SPI_IN_522	522	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_10
GIC500_SPI_IN_523	523	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_11
GIC500_SPI_IN_524	524	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_12
GIC500_SPI_IN_525	525	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_13
GIC500_SPI_IN_526	526	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_14
GIC500_SPI_IN_527	527	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_15
GIC500_SPI_IN_528	528	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_16
GIC500_SPI_IN_529	529	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_17
GIC500_SPI_IN_530	530	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_18
GIC500_SPI_IN_531	531	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_19
GIC500_SPI_IN_532	532	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_20
GIC500_SPI_IN_533	533	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_21
GIC500_SPI_IN_534	534	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_22
GIC500_SPI_IN_535	535	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_23
GIC500_SPI_IN_536	536	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_24
GIC500_SPI_IN_537	537	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_25
GIC500_SPI_IN_538	538	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_26

**Table 9-46. GIC500 SPI Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
GIC500_SPI_IN_539	539	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_27
GIC500_SPI_IN_540	540	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_28
GIC500_SPI_IN_541	541	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_29
GIC500_SPI_IN_542	542	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_30
GIC500_SPI_IN_543	543	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_31
GIC500_SPI_IN_544	544	CMPEVENT_INTRTR0_OUTP_0
GIC500_SPI_IN_545	545	CMPEVENT_INTRTR0_OUTP_1
GIC500_SPI_IN_546	546	CMPEVENT_INTRTR0_OUTP_2
GIC500_SPI_IN_547	547	CMPEVENT_INTRTR0_OUTP_3
GIC500_SPI_IN_548	548	CMPEVENT_INTRTR0_OUTP_4
GIC500_SPI_IN_549	549	CMPEVENT_INTRTR0_OUTP_5
GIC500_SPI_IN_550	550	CMPEVENT_INTRTR0_OUTP_6
GIC500_SPI_IN_551	551	CMPEVENT_INTRTR0_OUTP_7
GIC500_SPI_IN_552	552	CMPEVENT_INTRTR0_OUTP_8
GIC500_SPI_IN_553	553	CMPEVENT_INTRTR0_OUTP_9
GIC500_SPI_IN_554	554	CMPEVENT_INTRTR0_OUTP_10
GIC500_SPI_IN_555	555	CMPEVENT_INTRTR0_OUTP_11
GIC500_SPI_IN_556	556	CMPEVENT_INTRTR0_OUTP_12
GIC500_SPI_IN_557	557	CMPEVENT_INTRTR0_OUTP_13
GIC500_SPI_IN_558	558	CMPEVENT_INTRTR0_OUTP_14
GIC500_SPI_IN_559	559	CMPEVENT_INTRTR0_OUTP_15
GIC500_SPI_IN_576	576	MCASP0_XMIT_INTR_PEND_0
GIC500_SPI_IN_577	577	MCASP0_REC_INTR_PEND_0
GIC500_SPI_IN_578	578	MCASP1_XMIT_INTR_PEND_0
GIC500_SPI_IN_579	579	MCASP1_REC_INTR_PEND_0
GIC500_SPI_IN_580	580	MCASP2_XMIT_INTR_PEND_0
GIC500_SPI_IN_581	581	MCASP2_REC_INTR_PEND_0
GIC500_SPI_IN_582	582	MCASP3_XMIT_INTR_PEND_0
GIC500_SPI_IN_583	583	MCASP3_REC_INTR_PEND_0
GIC500_SPI_IN_584	584	MCASP4_XMIT_INTR_PEND_0
GIC500_SPI_IN_585	585	MCASP4_REC_INTR_PEND_0
GIC500_SPI_IN_586	586	MCASP5_XMIT_INTR_PEND_0
GIC500_SPI_IN_587	587	MCASP5_REC_INTR_PEND_0
GIC500_SPI_IN_588	588	MCASP6_XMIT_INTR_PEND_0
GIC500_SPI_IN_589	589	MCASP6_REC_INTR_PEND_0
GIC500_SPI_IN_590	590	MCASP7_XMIT_INTR_PEND_0
GIC500_SPI_IN_591	591	MCASP7_REC_INTR_PEND_0
GIC500_SPI_IN_592	592	MCASP8_XMIT_INTR_PEND_0
GIC500_SPI_IN_593	593	MCASP8_REC_INTR_PEND_0
GIC500_SPI_IN_594	594	MCASP9_XMIT_INTR_PEND_0
GIC500_SPI_IN_595	595	MCASP9_REC_INTR_PEND_0
GIC500_SPI_IN_596	596	MCASP10_XMIT_INTR_PEND_0
GIC500_SPI_IN_597	597	MCASP10_REC_INTR_PEND_0
GIC500_SPI_IN_598	598	MCASP11_XMIT_INTR_PEND_0
GIC500_SPI_IN_599	599	MCASP11_REC_INTR_PEND_0

**Table 9-46. GIC500 SPI Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
GIC500_SPI_IN_600	600	AASRC0_INFIFO_LEVEL_0
GIC500_SPI_IN_601	601	AASRC0_INGROUP_LEVEL_0
GIC500_SPI_IN_602	602	AASRC0_OUTFIFO_LEVEL_0
GIC500_SPI_IN_603	603	AASRC0_OUTGROUP_LEVEL_0
GIC500_SPI_IN_604	604	AASRC0_ERR_LEVEL_0
GIC500_SPI_IN_605	605	MLB0_MLBSS_MLB_INT_0
GIC500_SPI_IN_606	606	MLB0_MLBSS_MLB_AHB_INT_0
GIC500_SPI_IN_607	607	MLB0_MLBSS_MLB_AHB_INT_1
GIC500_SPI_IN_608	608	MCAN8_MCANSS_MCAN_LVL_INT_0
GIC500_SPI_IN_609	609	MCAN8_MCANSS_MCAN_LVL_INT_1
GIC500_SPI_IN_610	610	MCAN8_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
GIC500_SPI_IN_611	611	MCAN9_MCANSS_MCAN_LVL_INT_0
GIC500_SPI_IN_612	612	MCAN9_MCANSS_MCAN_LVL_INT_1
GIC500_SPI_IN_613	613	MCAN9_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
GIC500_SPI_IN_614	614	MCAN10_MCANSS_MCAN_LVL_INT_0
GIC500_SPI_IN_615	615	MCAN10_MCANSS_MCAN_LVL_INT_1
GIC500_SPI_IN_616	616	MCAN10_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
GIC500_SPI_IN_617	617	MCAN11_MCANSS_MCAN_LVL_INT_0
GIC500_SPI_IN_618	618	MCAN11_MCANSS_MCAN_LVL_INT_1
GIC500_SPI_IN_619	619	MCAN11_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
GIC500_SPI_IN_620	620	MCAN12_MCANSS_MCAN_LVL_INT_0
GIC500_SPI_IN_621	621	MCAN12_MCANSS_MCAN_LVL_INT_1
GIC500_SPI_IN_622	622	MCAN12_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
GIC500_SPI_IN_623	623	MCAN13_MCANSS_MCAN_LVL_INT_0
GIC500_SPI_IN_624	624	MCAN13_MCANSS_MCAN_LVL_INT_1
GIC500_SPI_IN_625	625	MCAN13_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
GIC500_SPI_IN_632	632	DSS_DSI0_DSI_0_FUNC_INTR_0
GIC500_SPI_IN_634	634	DSS0_DSS_INST0_DISPC_FUNC_IRQ_PROC0_0
GIC500_SPI_IN_635	635	DSS0_DSS_INST0_DISPC_FUNC_IRQ_PROC1_0
GIC500_SPI_IN_636	636	DSS0_DSS_INST0_DISPC_FUNC_IRQ_PROC2_0
GIC500_SPI_IN_637	637	DSS0_DSS_INST0_DISPC_FUNC_IRQ_PROC3_0
GIC500_SPI_IN_638	638	DSS0_DSS_INST0_DISPC_SECURE_IRQ_PROC0_0
GIC500_SPI_IN_639	639	DSS0_DSS_INST0_DISPC_SECURE_IRQ_PROC1_0
GIC500_SPI_IN_640	640	DSS0_DSS_INST0_DISPC_SECURE_IRQ_PROC2_0
GIC500_SPI_IN_641	641	DSS0_DSS_INST0_DISPC_SECURE_IRQ_PROC3_0
GIC500_SPI_IN_642	642	DSS0_DSS_INST0_DISPC_SAFETY_ERROR_IRQ_PROC0_0
GIC500_SPI_IN_643	643	DSS0_DSS_INST0_DISPC_SAFETY_ERROR_IRQ_PROC1_0
GIC500_SPI_IN_644	644	DSS0_DSS_INST0_DISPC_SAFETY_ERROR_IRQ_PROC2_0
GIC500_SPI_IN_645	645	DSS0_DSS_INST0_DISPC_SAFETY_ERROR_IRQ_PROC3_0
GIC500_SPI_IN_646	646	DSS_EDP0_INTR_0
GIC500_SPI_IN_647	647	DSS_EDP0_INTR_1
GIC500_SPI_IN_648	648	DSS_EDP0_INTR_2
GIC500_SPI_IN_649	649	DSS_EDP0_INTR_3
GIC500_SPI_IN_670	670	C66SS0_KSBUS_RAT_0_C66_RAT_INTR_0
GIC500_SPI_IN_671	671	C66SS1_KSBUS_RAT_0_C66_RAT_INTR_0

**Table 9-46. GIC500 SPI Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
GIC500_SPI_IN_672	672	NAVSS0_INTR_ROUTER_0_OUTL_INTR_128
GIC500_SPI_IN_673	673	NAVSS0_INTR_ROUTER_0_OUTL_INTR_129
GIC500_SPI_IN_674	674	NAVSS0_INTR_ROUTER_0_OUTL_INTR_130
GIC500_SPI_IN_675	675	NAVSS0_INTR_ROUTER_0_OUTL_INTR_131
GIC500_SPI_IN_676	676	NAVSS0_INTR_ROUTER_0_OUTL_INTR_132
GIC500_SPI_IN_677	677	NAVSS0_INTR_ROUTER_0_OUTL_INTR_133
GIC500_SPI_IN_678	678	NAVSS0_INTR_ROUTER_0_OUTL_INTR_134
GIC500_SPI_IN_679	679	NAVSS0_INTR_ROUTER_0_OUTL_INTR_135
GIC500_SPI_IN_680	680	NAVSS0_INTR_ROUTER_0_OUTL_INTR_136
GIC500_SPI_IN_681	681	NAVSS0_INTR_ROUTER_0_OUTL_INTR_137
GIC500_SPI_IN_682	682	NAVSS0_INTR_ROUTER_0_OUTL_INTR_138
GIC500_SPI_IN_683	683	NAVSS0_INTR_ROUTER_0_OUTL_INTR_139
GIC500_SPI_IN_684	684	NAVSS0_INTR_ROUTER_0_OUTL_INTR_140
GIC500_SPI_IN_685	685	NAVSS0_INTR_ROUTER_0_OUTL_INTR_141
GIC500_SPI_IN_686	686	NAVSS0_INTR_ROUTER_0_OUTL_INTR_142
GIC500_SPI_IN_687	687	NAVSS0_INTR_ROUTER_0_OUTL_INTR_143
GIC500_SPI_IN_688	688	NAVSS0_INTR_ROUTER_0_OUTL_INTR_144
GIC500_SPI_IN_689	689	NAVSS0_INTR_ROUTER_0_OUTL_INTR_145
GIC500_SPI_IN_690	690	NAVSS0_INTR_ROUTER_0_OUTL_INTR_146
GIC500_SPI_IN_691	691	NAVSS0_INTR_ROUTER_0_OUTL_INTR_147
GIC500_SPI_IN_692	692	NAVSS0_INTR_ROUTER_0_OUTL_INTR_148
GIC500_SPI_IN_693	693	NAVSS0_INTR_ROUTER_0_OUTL_INTR_149
GIC500_SPI_IN_694	694	NAVSS0_INTR_ROUTER_0_OUTL_INTR_150
GIC500_SPI_IN_695	695	NAVSS0_INTR_ROUTER_0_OUTL_INTR_151
GIC500_SPI_IN_696	696	NAVSS0_INTR_ROUTER_0_OUTL_INTR_152
GIC500_SPI_IN_697	697	NAVSS0_INTR_ROUTER_0_OUTL_INTR_153
GIC500_SPI_IN_698	698	NAVSS0_INTR_ROUTER_0_OUTL_INTR_154
GIC500_SPI_IN_699	699	NAVSS0_INTR_ROUTER_0_OUTL_INTR_155
GIC500_SPI_IN_700	700	NAVSS0_INTR_ROUTER_0_OUTL_INTR_156
GIC500_SPI_IN_701	701	NAVSS0_INTR_ROUTER_0_OUTL_INTR_157
GIC500_SPI_IN_702	702	NAVSS0_INTR_ROUTER_0_OUTL_INTR_158
GIC500_SPI_IN_703	703	NAVSS0_INTR_ROUTER_0_OUTL_INTR_159
GIC500_SPI_IN_704	704	NAVSS0_INTR_ROUTER_0_OUTL_INTR_160
GIC500_SPI_IN_705	705	NAVSS0_INTR_ROUTER_0_OUTL_INTR_161
GIC500_SPI_IN_706	706	NAVSS0_INTR_ROUTER_0_OUTL_INTR_162
GIC500_SPI_IN_707	707	NAVSS0_INTR_ROUTER_0_OUTL_INTR_163
GIC500_SPI_IN_708	708	NAVSS0_INTR_ROUTER_0_OUTL_INTR_164
GIC500_SPI_IN_709	709	NAVSS0_INTR_ROUTER_0_OUTL_INTR_165
GIC500_SPI_IN_710	710	NAVSS0_INTR_ROUTER_0_OUTL_INTR_166
GIC500_SPI_IN_711	711	NAVSS0_INTR_ROUTER_0_OUTL_INTR_167
GIC500_SPI_IN_712	712	NAVSS0_INTR_ROUTER_0_OUTL_INTR_168
GIC500_SPI_IN_713	713	NAVSS0_INTR_ROUTER_0_OUTL_INTR_169
GIC500_SPI_IN_714	714	NAVSS0_INTR_ROUTER_0_OUTL_INTR_170
GIC500_SPI_IN_715	715	NAVSS0_INTR_ROUTER_0_OUTL_INTR_171
GIC500_SPI_IN_716	716	NAVSS0_INTR_ROUTER_0_OUTL_INTR_172

**Table 9-46. GIC500 SPI Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
GIC500_SPI_IN_717	717	NAVSS0_INTR_ROUTER_0_OUTL_INTR_173
GIC500_SPI_IN_718	718	NAVSS0_INTR_ROUTER_0_OUTL_INTR_174
GIC500_SPI_IN_719	719	NAVSS0_INTR_ROUTER_0_OUTL_INTR_175
GIC500_SPI_IN_720	720	NAVSS0_INTR_ROUTER_0_OUTL_INTR_176
GIC500_SPI_IN_721	721	NAVSS0_INTR_ROUTER_0_OUTL_INTR_177
GIC500_SPI_IN_722	722	NAVSS0_INTR_ROUTER_0_OUTL_INTR_178
GIC500_SPI_IN_723	723	NAVSS0_INTR_ROUTER_0_OUTL_INTR_179
GIC500_SPI_IN_724	724	NAVSS0_INTR_ROUTER_0_OUTL_INTR_180
GIC500_SPI_IN_725	725	NAVSS0_INTR_ROUTER_0_OUTL_INTR_181
GIC500_SPI_IN_726	726	NAVSS0_INTR_ROUTER_0_OUTL_INTR_182
GIC500_SPI_IN_727	727	NAVSS0_INTR_ROUTER_0_OUTL_INTR_183
GIC500_SPI_IN_728	728	NAVSS0_INTR_ROUTER_0_OUTL_INTR_184
GIC500_SPI_IN_729	729	NAVSS0_INTR_ROUTER_0_OUTL_INTR_185
GIC500_SPI_IN_730	730	NAVSS0_INTR_ROUTER_0_OUTL_INTR_186
GIC500_SPI_IN_731	731	NAVSS0_INTR_ROUTER_0_OUTL_INTR_187
GIC500_SPI_IN_732	732	NAVSS0_INTR_ROUTER_0_OUTL_INTR_188
GIC500_SPI_IN_733	733	NAVSS0_INTR_ROUTER_0_OUTL_INTR_189
GIC500_SPI_IN_734	734	NAVSS0_INTR_ROUTER_0_OUTL_INTR_190
GIC500_SPI_IN_735	735	NAVSS0_INTR_ROUTER_0_OUTL_INTR_191
GIC500_SPI_IN_736	736	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_32
GIC500_SPI_IN_737	737	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_33
GIC500_SPI_IN_738	738	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_34
GIC500_SPI_IN_739	739	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_35
GIC500_SPI_IN_740	740	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_36
GIC500_SPI_IN_741	741	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_37
GIC500_SPI_IN_742	742	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_38
GIC500_SPI_IN_743	743	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_39
GIC500_SPI_IN_744	744	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_40
GIC500_SPI_IN_745	745	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_41
GIC500_SPI_IN_746	746	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_42
GIC500_SPI_IN_747	747	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_43
GIC500_SPI_IN_748	748	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_44
GIC500_SPI_IN_749	749	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_45
GIC500_SPI_IN_750	750	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_46
GIC500_SPI_IN_751	751	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_47
GIC500_SPI_IN_752	752	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_48
GIC500_SPI_IN_753	753	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_49
GIC500_SPI_IN_754	754	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_50
GIC500_SPI_IN_755	755	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_51
GIC500_SPI_IN_756	756	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_52
GIC500_SPI_IN_757	757	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_53
GIC500_SPI_IN_758	758	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_54
GIC500_SPI_IN_759	759	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_55
GIC500_SPI_IN_760	760	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_56
GIC500_SPI_IN_761	761	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_57



**Table 9-46. GIC500 SPI Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
GIC500_SPI_IN_762	762	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_58
GIC500_SPI_IN_763	763	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_59
GIC500_SPI_IN_764	764	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_60
GIC500_SPI_IN_765	765	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_61
GIC500_SPI_IN_766	766	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_62
GIC500_SPI_IN_767	767	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_63
GIC500_SPI_IN_768	768	CCDEBUGSS0_AQCMPIINTR_LEVEL_0
GIC500_SPI_IN_769	769	DEBUGSS0_AQCMPIINTR_LEVEL_0
GIC500_SPI_IN_770	770	C66DEBUGSS0_AQCMPIINTR_LEVEL_0
GIC500_SPI_IN_771	771	DEBUGSS1_AQCMPIINTR_LEVEL_0
GIC500_SPI_IN_772	772	C66DEBUGSS1_AQCMPIINTR_LEVEL_0
GIC500_SPI_IN_773	773	DEBUGSS0_CTM_LEVEL_0
GIC500_SPI_IN_775	775	DEBUGSS1_CTM_LEVEL_0
GIC500_SPI_IN_776	776	R5FSS0_CORE0_PMU_0
GIC500_SPI_IN_777	777	R5FSS0_CORE1_PMU_0
GIC500_SPI_IN_778	778	R5FSS1_CORE0_PMU_0
GIC500_SPI_IN_779	779	R5FSS1_CORE1_PMU_0
GIC500_SPI_IN_780	780	CTRL_MMR0_ACCESS_ERR_0
GIC500_SPI_IN_781	781	MCU_CTRL_MMR0_ACCESS_ERR_0
GIC500_SPI_IN_782	782	USB0_HOST_SYSTEM_ERROR_0
GIC500_SPI_IN_783	783	USB1_HOST_SYSTEM_ERROR_0
GIC500_SPI_IN_787	787	PSC0_PSC_ALLINT_0
GIC500_SPI_IN_791	791	CBASS_INFRA0_DEFAULT_ERR_INTR_0
GIC500_SPI_IN_793	793	GLUELOGIC_MAIN_CBASS_INTR_OR_GLUE_MAIN_CBASS_AGG_ERR_INTR_0
GIC500_SPI_IN_848	848	MCU_TIMER0_INTR_PEND_0
GIC500_SPI_IN_849	849	MCU_TIMER1_INTR_PEND_0
GIC500_SPI_IN_850	850	MCU_TIMER2_INTR_PEND_0
GIC500_SPI_IN_851	851	MCU_TIMER3_INTR_PEND_0
GIC500_SPI_IN_852	852	MCU_TIMER4_INTR_PEND_0
GIC500_SPI_IN_853	853	MCU_TIMER5_INTR_PEND_0
GIC500_SPI_IN_854	854	MCU_TIMER6_INTR_PEND_0
GIC500_SPI_IN_855	855	MCU_TIMER7_INTR_PEND_0
GIC500_SPI_IN_856	856	MCU_TIMER8_INTR_PEND_0
GIC500_SPI_IN_857	857	MCU_TIMER9_INTR_PEND_0
GIC500_SPI_IN_864	864	MCU_MCAN0_MCANSS_MCAN_LVL_INT_0
GIC500_SPI_IN_865	865	MCU_MCAN0_MCANSS_MCAN_LVL_INT_1
GIC500_SPI_IN_866	866	MCU_MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
GIC500_SPI_IN_867	867	MCU_MCAN1_MCANSS_MCAN_LVL_INT_0
GIC500_SPI_IN_868	868	MCU_MCAN1_MCANSS_MCAN_LVL_INT_1
GIC500_SPI_IN_869	869	MCU_MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
GIC500_SPI_IN_872	872	MCU_FSS0_OSPI_0_OSPI_LVL_INTR_0
GIC500_SPI_IN_873	873	MCU_FSS0_OSPI_1_OSPI_LVL_INTR_0
GIC500_SPI_IN_874	874	MCU_FSS0_HYPERBUS1P0_0_HPB_INTR_0
GIC500_SPI_IN_875	875	MCU_FSS0_FSAS_0_OTFE_INTR_ERR_PEND_0
GIC500_SPI_IN_876	876	MCU_FSS0_FSAS_0_ECC_INTR_ERR_PEND_0



**Table 9-46. GIC500 SPI Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
GIC500_SPI_IN_878	878	MCU_UART0_USART_IRQ_0
GIC500_SPI_IN_880	880	MCU_MCSPI0_INTR_SPI_0
GIC500_SPI_IN_881	881	MCU_MCSPI1_INTR_SPI_0
GIC500_SPI_IN_882	882	MCU_MCSPI2_INTR_SPI_0
GIC500_SPI_IN_884	884	MCU_I2C0_POINTRPEND_0
GIC500_SPI_IN_885	885	MCU_I2C1_POINTRPEND_0
GIC500_SPI_IN_886	886	MCU_I3C0_I3C__INT_0
GIC500_SPI_IN_887	887	MCU_I3C1_I3C__INT_0
GIC500_SPI_IN_888	888	MCU_CPSW0_STAT_PEND_0
GIC500_SPI_IN_889	889	MCU_CPSW0_MDIO_PEND_0
GIC500_SPI_IN_890	890	MCU_CPSW0_EVTN_PEND_0
GIC500_SPI_IN_892	892	MCU_ADC0_GEN_LEVEL_0
GIC500_SPI_IN_893	893	MCU_ADC1_GEN_LEVEL_0
GIC500_SPI_IN_912	912	MCU_SA2_UL0_SA_UL_PKA_0
GIC500_SPI_IN_913	913	MCU_SA2_UL0_SA_UL_TRNG_0
GIC500_SPI_IN_918	918	MCU_R5FSS0_CORE0_PMU_0
GIC500_SPI_IN_919	919	MCU_R5FSS0_CORE1_PMU_0
GIC500_SPI_IN_920	920	MCU_CBASS0_LPSC_MCU_COMMON_ERR_INTR_0
GIC500_SPI_IN_921	921	GLUELOGIC_DBG_CBASS_INTR_OR_GLUE_DBG_CBASS_AGG_ERR_INTR_0
GIC500_SPI_IN_928	928	WKUP_I2C0_POINTRPEND_0
GIC500_SPI_IN_929	929	WKUP_UART0_USART_IRQ_0
GIC500_SPI_IN_932	932	WKUP_DMSC0_CORTEX_M3_0_SEC_OUT_0
GIC500_SPI_IN_933	933	WKUP_DMSC0_CORTEX_M3_0_SEC_OUT_1
GIC500_SPI_IN_936	936	WKUP_VTM0_THERM_LVL_GT_TH1_INTR_0
GIC500_SPI_IN_937	937	WKUP_VTM0_THERM_LVL_LT_TH0_INTR_0
GIC500_SPI_IN_938	938	WKUP_VTM0_THERM_LVL_GT_TH2_INTR_0
GIC500_SPI_IN_952	952	WKUP_CBASS0_LPSC_WKUP_COMMON_ERR_INTR_0
GIC500_SPI_IN_953	953	GLUELOGIC_FW_CBASS_INTR_OR_GLUE_FW_CBASS_AGG_ERR_INTR_0
GIC500_SPI_IN_960	960	WKUP_GPIOMUX_INTRTR0_OUTP_16
GIC500_SPI_IN_961	961	WKUP_GPIOMUX_INTRTR0_OUTP_17
GIC500_SPI_IN_962	962	WKUP_GPIOMUX_INTRTR0_OUTP_18
GIC500_SPI_IN_963	963	WKUP_GPIOMUX_INTRTR0_OUTP_19
GIC500_SPI_IN_964	964	WKUP_GPIOMUX_INTRTR0_OUTP_20
GIC500_SPI_IN_965	965	WKUP_GPIOMUX_INTRTR0_OUTP_21
GIC500_SPI_IN_966	966	WKUP_GPIOMUX_INTRTR0_OUTP_22
GIC500_SPI_IN_967	967	WKUP_GPIOMUX_INTRTR0_OUTP_23
GIC500_SPI_IN_968	968	WKUP_GPIOMUX_INTRTR0_OUTP_24
GIC500_SPI_IN_969	969	WKUP_GPIOMUX_INTRTR0_OUTP_25
GIC500_SPI_IN_970	970	WKUP_GPIOMUX_INTRTR0_OUTP_26
GIC500_SPI_IN_971	971	WKUP_GPIOMUX_INTRTR0_OUTP_27
GIC500_SPI_IN_972	972	WKUP_GPIOMUX_INTRTR0_OUTP_28
GIC500_SPI_IN_973	973	WKUP_GPIOMUX_INTRTR0_OUTP_29
GIC500_SPI_IN_974	974	WKUP_GPIOMUX_INTRTR0_OUTP_30
GIC500_SPI_IN_975	975	WKUP_GPIOMUX_INTRTR0_OUTP_31
GIC500_SPI_IN_968	968	WKUP_GPIOMUX_INTRTR0_OUTP_24

**Table 9-46. GIC500 SPI Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
GIC500_SPI_IN_969	969	WKUP_GPIOMUX_INTRTR0_OUTP_25
GIC500_SPI_IN_970	970	WKUP_GPIOMUX_INTRTR0_OUTP_26
GIC500_SPI_IN_971	971	WKUP_GPIOMUX_INTRTR0_OUTP_27
GIC500_SPI_IN_972	972	WKUP_GPIOMUX_INTRTR0_OUTP_28
GIC500_SPI_IN_973	973	WKUP_GPIOMUX_INTRTR0_OUTP_29
GIC500_SPI_IN_974	974	WKUP_GPIOMUX_INTRTR0_OUTP_30
GIC500_SPI_IN_975	975	WKUP_GPIOMUX_INTRTR0_OUTP_31

### 9.4.3.2 R5FSS0\_CORE0 Interrupt Map

Table 9-47 shows the mapping of events to the R5FSS0\_CORE0. The R5FSS0 VIM supports both R5FSS0 cores.

The R5FSS0\_CORE0 events are the events used by both processors when operating in lockstep mode.

**Table 9-47. R5FSS0\_CORE0 Interrupt Map**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS0_CORE0_INTR_IN_0	0	CTRL_MMR0_IPC_SET16_IPC_SET_IPCFG_0
R5FSS0_CORE0_INTR_IN_1	1	CTRL_MMR0_IPC_SET17_IPC_SET_IPCFG_0
R5FSS0_CORE0_INTR_IN_2	2	RTI28_INTR_WWD_0
R5FSS0_CORE0_INTR_IN_3	3	RTI29_INTR_WWD_0
R5FSS0_CORE0_INTR_IN_4	4	R5FSS0_COMMRX_LEVEL_0_0
R5FSS0_CORE0_INTR_IN_5	5	R5FSS0_COMMTX_LEVEL_0_0
R5FSS0_CORE0_INTR_IN_6	6	R5FSS0_CORE0_VALFIQ_0
R5FSS0_CORE0_INTR_IN_7	7	R5FSS0_CORE0_VALIRQ_0
R5FSS0_CORE0_INTR_IN_8	8	R5FSS0_CORE0_CTI_0
R5FSS0_CORE0_INTR_IN_9	9	R5FSS0_CORE1_CTI_0
R5FSS0_CORE0_INTR_IN_10	10	ESM0_ESM_INT_LOW_LVL_0
R5FSS0_CORE0_INTR_IN_11	11	ESM0_ESM_INT_HI_LVL_0
R5FSS0_CORE0_INTR_IN_12	12	ESM0_ESM_INT_CFG_LVL_0
R5FSS0_CORE0_INTR_IN_13	13	GLUELOGIC_EXT_INTN_GLUE_EXT_INT_LVL_0
R5FSS0_CORE0_INTR_IN_16	16	R5FSS0_CORE0_EXP_INTR_0
R5FSS0_CORE0_INTR_IN_17	17	R5FSS0_CORE1_EXP_INTR_0
R5FSS0_CORE0_INTR_IN_32	32	DMPAC0_INTD_0_SYSTEM_INTR_LEVEL_0
R5FSS0_CORE0_INTR_IN_33	33	DMPAC0_INTD_0_SYSTEM_INTR_LEVEL_1
R5FSS0_CORE0_INTR_IN_34	34	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_0
R5FSS0_CORE0_INTR_IN_35	35	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_1
R5FSS0_CORE0_INTR_IN_36	36	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_2
R5FSS0_CORE0_INTR_IN_37	37	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_3
R5FSS0_CORE0_INTR_IN_38	38	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_4
R5FSS0_CORE0_INTR_IN_39	39	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_5
R5FSS0_CORE0_INTR_IN_40	40	GPU0_MISC_0_IRQ_0
R5FSS0_CORE0_INTR_IN_48	48	GLUELOGIC_GPU_GPIO_REQACK_GLUE_GPU_GPIO_REQINT_LVL_0
R5FSS0_CORE0_INTR_IN_49	49	GLUELOGIC_GPU_GPIO_REQACK_GLUE_GPU_GPIO_ACKINT_LVL_0
R5FSS0_CORE0_INTR_IN_52	52	DSS0_DSS_INST0_DISPC_FUNC_IRQ_PROC0_0
R5FSS0_CORE0_INTR_IN_53	53	DSS0_DSS_INST0_DISPC_FUNC_IRQ_PROC1_0
R5FSS0_CORE0_INTR_IN_54	54	DSS0_DSS_INST0_DISPC_SAFETY_ERROR_IRQ_PROC0_0
R5FSS0_CORE0_INTR_IN_55	55	DSS0_DSS_INST0_DISPC_SAFETY_ERROR_IRQ_PROC1_0
R5FSS0_CORE0_INTR_IN_56	56	DSS0_DSS_INST0_DISPC_SECURE_IRQ_PROC0_0
R5FSS0_CORE0_INTR_IN_57	57	DSS0_DSS_INST0_DISPC_SECURE_IRQ_PROC1_0
R5FSS0_CORE0_INTR_IN_64	64	DSS_EDP0_INTR_0
R5FSS0_CORE0_INTR_IN_65	65	DSS_EDP0_INTR_1
R5FSS0_CORE0_INTR_IN_66	66	DSS_EDP0_INTR_2
R5FSS0_CORE0_INTR_IN_67	67	DSS_EDP0_INTR_3
R5FSS0_CORE0_INTR_IN_76	76	DSS_DSI0_DSI_0_FUNC_INTR_0
R5FSS0_CORE0_INTR_IN_78	78	CSI_RX_IF0_CSI_ERR_IRQ_0
R5FSS0_CORE0_INTR_IN_79	79	CSI_RX_IF0_CSI_IRQ_0

**Table 9-47. R5FSS0\_CORE0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS0_CORE0_INTR_IN_80	80	CSI_RX_IF0_CSI_LEVEL_0
R5FSS0_CORE0_INTR_IN_81	81	CSI_RX_IF1_CSI_ERR_IRQ_0
R5FSS0_CORE0_INTR_IN_82	82	CSI_RX_IF1_CSI_IRQ_0
R5FSS0_CORE0_INTR_IN_83	83	CSI_RX_IF1_CSI_LEVEL_0
R5FSS0_CORE0_INTR_IN_87	87	DECODER0_IRQ_0
R5FSS0_CORE0_INTR_IN_88	88	ENCODER0_IRQ_0
R5FSS0_CORE0_INTR_IN_96	96	CPSW0_STAT_PEND_0
R5FSS0_CORE0_INTR_IN_97	97	CPSW0_MDIO_PEND_0
R5FSS0_CORE0_INTR_IN_98	98	CPSW0_EVTN_PEND_0
R5FSS0_CORE0_INTR_IN_99	99	MMCS00_EMMCSS_INTR_0
R5FSS0_CORE0_INTR_IN_104	104	EHRPWM0_EPWM_ETINT_0
R5FSS0_CORE0_INTR_IN_105	105	EHRPWM1_EPWM_ETINT_0
R5FSS0_CORE0_INTR_IN_106	106	EHRPWM2_EPWM_ETINT_0
R5FSS0_CORE0_INTR_IN_107	107	EHRPWM3_EPWM_ETINT_0
R5FSS0_CORE0_INTR_IN_108	108	EHRPWM4_EPWM_ETINT_0
R5FSS0_CORE0_INTR_IN_109	109	EHRPWM5_EPWM_ETINT_0
R5FSS0_CORE0_INTR_IN_110	110	EHRPWM0_EPWM_TRIPZINT_0
R5FSS0_CORE0_INTR_IN_111	111	EHRPWM1_EPWM_TRIPZINT_0
R5FSS0_CORE0_INTR_IN_112	112	EHRPWM2_EPWM_TRIPZINT_0
R5FSS0_CORE0_INTR_IN_113	113	EHRPWM3_EPWM_TRIPZINT_0
R5FSS0_CORE0_INTR_IN_114	114	EHRPWM4_EPWM_TRIPZINT_0
R5FSS0_CORE0_INTR_IN_115	115	EHRPWM5_EPWM_TRIPZINT_0
R5FSS0_CORE0_INTR_IN_116	116	EQEP0_EQEP_INT_0
R5FSS0_CORE0_INTR_IN_117	117	EQEP1_EQEP_INT_0
R5FSS0_CORE0_INTR_IN_118	118	EQEP2_EQEP_INT_0
R5FSS0_CORE0_INTR_IN_120	120	MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS0_CORE0_INTR_IN_121	121	MCAN0_MCANSS_MCAN_LVL_INT_0
R5FSS0_CORE0_INTR_IN_122	122	MCAN0_MCANSS_MCAN_LVL_INT_1
R5FSS0_CORE0_INTR_IN_123	123	MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS0_CORE0_INTR_IN_124	124	MCAN1_MCANSS_MCAN_LVL_INT_0
R5FSS0_CORE0_INTR_IN_125	125	MCAN1_MCANSS_MCAN_LVL_INT_1
R5FSS0_CORE0_INTR_IN_126	126	MCAN2_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS0_CORE0_INTR_IN_127	127	MCAN2_MCANSS_MCAN_LVL_INT_0
R5FSS0_CORE0_INTR_IN_128	128	MCAN2_MCANSS_MCAN_LVL_INT_1
R5FSS0_CORE0_INTR_IN_129	129	MCAN3_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS0_CORE0_INTR_IN_130	130	MCAN3_MCANSS_MCAN_LVL_INT_0
R5FSS0_CORE0_INTR_IN_131	131	MCAN3_MCANSS_MCAN_LVL_INT_1
R5FSS0_CORE0_INTR_IN_132	132	MCASP0_XMIT_INTR_PEND_0
R5FSS0_CORE0_INTR_IN_133	133	MCASP0_REC_INTR_PEND_0
R5FSS0_CORE0_INTR_IN_134	134	MCASP1_XMIT_INTR_PEND_0
R5FSS0_CORE0_INTR_IN_135	135	MCASP1_REC_INTR_PEND_0
R5FSS0_CORE0_INTR_IN_136	136	PCIE0_PCIE_LEGACY_PULSE_0
R5FSS0_CORE0_INTR_IN_137	137	PCIE0_PCIE_DOWNSTREAM_PULSE_0
R5FSS0_CORE0_INTR_IN_138	138	PCIE0_PCIE_FLR_PULSE_0
R5FSS0_CORE0_INTR_IN_139	139	PCIE0_PCIE_PHY_LEVEL_0

**Table 9-47. R5FSS0\_CORE0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS0_CORE0_INTR_IN_140	140	PCIE0_PCIE_LOCAL_LEVEL_0
R5FSS0_CORE0_INTR_IN_141	141	PCIE0_PCIE_ERROR_PULSE_0
R5FSS0_CORE0_INTR_IN_142	142	PCIE0_PCIE_LINK_STATE_PULSE_0
R5FSS0_CORE0_INTR_IN_143	143	PCIE0_PCIE_PWR_STATE_PULSE_0
R5FSS0_CORE0_INTR_IN_144	144	PCIE0_PCIE_PTM_VALID_PULSE_0
R5FSS0_CORE0_INTR_IN_145	145	PCIE0_PCIE_HOT_RESET_PULSE_0
R5FSS0_CORE0_INTR_IN_146	146	PCIE0_PCIE_CPTS_PEND_0
R5FSS0_CORE0_INTR_IN_150	150	I2C0_POINTRPEND_0
R5FSS0_CORE0_INTR_IN_151	151	I2C1_POINTRPEND_0
R5FSS0_CORE0_INTR_IN_152	152	MCSPi0_INTR_SPI_0
R5FSS0_CORE0_INTR_IN_153	153	MCSPi1_INTR_SPI_0
R5FSS0_CORE0_INTR_IN_154	154	MLB0_MLBSS_MLB_INT_0
R5FSS0_CORE0_INTR_IN_155	155	MLB0_MLBSS_MLB_AHB_INT_0
R5FSS0_CORE0_INTR_IN_156	156	MLB0_MLBSS_MLB_AHB_INT_1
R5FSS0_CORE0_INTR_IN_158	158	UART0_USART_IRQ_0
R5FSS0_CORE0_INTR_IN_159	159	UART1_USART_IRQ_0
R5FSS0_CORE0_INTR_IN_160	160	UART2_USART_IRQ_0
R5FSS0_CORE0_INTR_IN_168	168	TIMER12_INTR_PEND_0
R5FSS0_CORE0_INTR_IN_169	169	TIMER13_INTR_PEND_0
R5FSS0_CORE0_INTR_IN_170	170	TIMER14_INTR_PEND_0
R5FSS0_CORE0_INTR_IN_171	171	TIMER15_INTR_PEND_0
R5FSS0_CORE0_INTR_IN_172	172	TIMER16_INTR_PEND_0
R5FSS0_CORE0_INTR_IN_173	173	TIMER17_INTR_PEND_0
R5FSS0_CORE0_INTR_IN_174	174	TIMER18_INTR_PEND_0
R5FSS0_CORE0_INTR_IN_175	175	TIMER19_INTR_PEND_0
R5FSS0_CORE0_INTR_IN_176	176	GPIOMUX_INTRTR0_OUTP_16
R5FSS0_CORE0_INTR_IN_177	177	GPIOMUX_INTRTR0_OUTP_17
R5FSS0_CORE0_INTR_IN_178	178	GPIOMUX_INTRTR0_OUTP_18
R5FSS0_CORE0_INTR_IN_179	179	GPIOMUX_INTRTR0_OUTP_19
R5FSS0_CORE0_INTR_IN_180	180	GPIOMUX_INTRTR0_OUTP_20
R5FSS0_CORE0_INTR_IN_181	181	GPIOMUX_INTRTR0_OUTP_21
R5FSS0_CORE0_INTR_IN_182	182	GPIOMUX_INTRTR0_OUTP_22
R5FSS0_CORE0_INTR_IN_183	183	GPIOMUX_INTRTR0_OUTP_23
R5FSS0_CORE0_INTR_IN_184	184	GPIOMUX_INTRTR0_OUTP_24
R5FSS0_CORE0_INTR_IN_185	185	GPIOMUX_INTRTR0_OUTP_25
R5FSS0_CORE0_INTR_IN_186	186	GPIOMUX_INTRTR0_OUTP_26
R5FSS0_CORE0_INTR_IN_187	187	GPIOMUX_INTRTR0_OUTP_27
R5FSS0_CORE0_INTR_IN_188	188	GPIOMUX_INTRTR0_OUTP_28
R5FSS0_CORE0_INTR_IN_189	189	GPIOMUX_INTRTR0_OUTP_29
R5FSS0_CORE0_INTR_IN_190	190	GPIOMUX_INTRTR0_OUTP_30
R5FSS0_CORE0_INTR_IN_191	191	GPIOMUX_INTRTR0_OUTP_31
R5FSS0_CORE0_INTR_IN_192	192	MCAN4_MCANSS_MCAN_LVL_INT_0
R5FSS0_CORE0_INTR_IN_193	193	MCAN4_MCANSS_MCAN_LVL_INT_1
R5FSS0_CORE0_INTR_IN_194	194	MCAN4_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS0_CORE0_INTR_IN_195	195	MCAN5_MCANSS_MCAN_LVL_INT_0

**Table 9-47. R5FSS0\_CORE0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS0_CORE0_INTR_IN_196	196	MCAN5_MCANSS_MCAN_LVL_INT_1
R5FSS0_CORE0_INTR_IN_197	197	MCAN5_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS0_CORE0_INTR_IN_198	198	MCAN6_MCANSS_MCAN_LVL_INT_0
R5FSS0_CORE0_INTR_IN_199	199	MCAN6_MCANSS_MCAN_LVL_INT_1
R5FSS0_CORE0_INTR_IN_200	200	MCAN6_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS0_CORE0_INTR_IN_201	201	MCAN7_MCANSS_MCAN_LVL_INT_0
R5FSS0_CORE0_INTR_IN_202	202	MCAN7_MCANSS_MCAN_LVL_INT_1
R5FSS0_CORE0_INTR_IN_203	203	MCAN7_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS0_CORE0_INTR_IN_204	204	MCAN8_MCANSS_MCAN_LVL_INT_0
R5FSS0_CORE0_INTR_IN_205	205	MCAN8_MCANSS_MCAN_LVL_INT_1
R5FSS0_CORE0_INTR_IN_206	206	MCAN8_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS0_CORE0_INTR_IN_207	207	MCAN9_MCANSS_MCAN_LVL_INT_0
R5FSS0_CORE0_INTR_IN_208	208	MCAN9_MCANSS_MCAN_LVL_INT_1
R5FSS0_CORE0_INTR_IN_209	209	MCAN9_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS0_CORE0_INTR_IN_210	210	MCAN10_MCANSS_MCAN_LVL_INT_0
R5FSS0_CORE0_INTR_IN_211	211	MCAN10_MCANSS_MCAN_LVL_INT_1
R5FSS0_CORE0_INTR_IN_212	212	MCAN10_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS0_CORE0_INTR_IN_213	213	MCAN11_MCANSS_MCAN_LVL_INT_0
R5FSS0_CORE0_INTR_IN_214	214	MCAN11_MCANSS_MCAN_LVL_INT_1
R5FSS0_CORE0_INTR_IN_215	215	MCAN11_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS0_CORE0_INTR_IN_216	216	MCAN12_MCANSS_MCAN_LVL_INT_0
R5FSS0_CORE0_INTR_IN_217	217	MCAN12_MCANSS_MCAN_LVL_INT_1
R5FSS0_CORE0_INTR_IN_218	218	MCAN12_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS0_CORE0_INTR_IN_219	219	MCAN13_MCANSS_MCAN_LVL_INT_0
R5FSS0_CORE0_INTR_IN_220	220	MCAN13_MCANSS_MCAN_LVL_INT_1
R5FSS0_CORE0_INTR_IN_221	221	MCAN13_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS0_CORE0_INTR_IN_224	224	NAVSS0_INTR_ROUTER_0_OUTL_INTR_192
R5FSS0_CORE0_INTR_IN_225	225	NAVSS0_INTR_ROUTER_0_OUTL_INTR_193
R5FSS0_CORE0_INTR_IN_226	226	NAVSS0_INTR_ROUTER_0_OUTL_INTR_194
R5FSS0_CORE0_INTR_IN_227	227	NAVSS0_INTR_ROUTER_0_OUTL_INTR_195
R5FSS0_CORE0_INTR_IN_228	228	NAVSS0_INTR_ROUTER_0_OUTL_INTR_196
R5FSS0_CORE0_INTR_IN_229	229	NAVSS0_INTR_ROUTER_0_OUTL_INTR_197
R5FSS0_CORE0_INTR_IN_230	230	NAVSS0_INTR_ROUTER_0_OUTL_INTR_198
R5FSS0_CORE0_INTR_IN_231	231	NAVSS0_INTR_ROUTER_0_OUTL_INTR_199
R5FSS0_CORE0_INTR_IN_232	232	NAVSS0_INTR_ROUTER_0_OUTL_INTR_200
R5FSS0_CORE0_INTR_IN_233	233	NAVSS0_INTR_ROUTER_0_OUTL_INTR_201
R5FSS0_CORE0_INTR_IN_234	234	NAVSS0_INTR_ROUTER_0_OUTL_INTR_202
R5FSS0_CORE0_INTR_IN_235	235	NAVSS0_INTR_ROUTER_0_OUTL_INTR_203
R5FSS0_CORE0_INTR_IN_236	236	NAVSS0_INTR_ROUTER_0_OUTL_INTR_204
R5FSS0_CORE0_INTR_IN_237	237	NAVSS0_INTR_ROUTER_0_OUTL_INTR_205
R5FSS0_CORE0_INTR_IN_238	238	NAVSS0_INTR_ROUTER_0_OUTL_INTR_206
R5FSS0_CORE0_INTR_IN_239	239	NAVSS0_INTR_ROUTER_0_OUTL_INTR_207
R5FSS0_CORE0_INTR_IN_240	240	NAVSS0_INTR_ROUTER_0_OUTL_INTR_208
R5FSS0_CORE0_INTR_IN_241	241	NAVSS0_INTR_ROUTER_0_OUTL_INTR_209
R5FSS0_CORE0_INTR_IN_242	242	NAVSS0_INTR_ROUTER_0_OUTL_INTR_210

**Table 9-47. R5FSS0\_CORE0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS0_CORE0_INTR_IN_243	243	NAVSS0_INTR_ROUTER_0_OUTL_INTR_211
R5FSS0_CORE0_INTR_IN_244	244	NAVSS0_INTR_ROUTER_0_OUTL_INTR_212
R5FSS0_CORE0_INTR_IN_245	245	NAVSS0_INTR_ROUTER_0_OUTL_INTR_213
R5FSS0_CORE0_INTR_IN_246	246	NAVSS0_INTR_ROUTER_0_OUTL_INTR_214
R5FSS0_CORE0_INTR_IN_247	247	NAVSS0_INTR_ROUTER_0_OUTL_INTR_215
R5FSS0_CORE0_INTR_IN_248	248	NAVSS0_INTR_ROUTER_0_OUTL_INTR_216
R5FSS0_CORE0_INTR_IN_249	249	NAVSS0_INTR_ROUTER_0_OUTL_INTR_217
R5FSS0_CORE0_INTR_IN_250	250	NAVSS0_INTR_ROUTER_0_OUTL_INTR_218
R5FSS0_CORE0_INTR_IN_251	251	NAVSS0_INTR_ROUTER_0_OUTL_INTR_219
R5FSS0_CORE0_INTR_IN_252	252	NAVSS0_INTR_ROUTER_0_OUTL_INTR_220
R5FSS0_CORE0_INTR_IN_253	253	NAVSS0_INTR_ROUTER_0_OUTL_INTR_221
R5FSS0_CORE0_INTR_IN_254	254	NAVSS0_INTR_ROUTER_0_OUTL_INTR_222
R5FSS0_CORE0_INTR_IN_255	255	NAVSS0_INTR_ROUTER_0_OUTL_INTR_223
R5FSS0_CORE0_INTR_IN_256	256	R5FSS0_INTRROUTER0_OUTL_0
R5FSS0_CORE0_INTR_IN_257	257	R5FSS0_INTRROUTER0_OUTL_1
R5FSS0_CORE0_INTR_IN_258	258	R5FSS0_INTRROUTER0_OUTL_2
R5FSS0_CORE0_INTR_IN_259	259	R5FSS0_INTRROUTER0_OUTL_3
R5FSS0_CORE0_INTR_IN_260	260	R5FSS0_INTRROUTER0_OUTL_4
R5FSS0_CORE0_INTR_IN_261	261	R5FSS0_INTRROUTER0_OUTL_5
R5FSS0_CORE0_INTR_IN_262	262	R5FSS0_INTRROUTER0_OUTL_6
R5FSS0_CORE0_INTR_IN_263	263	R5FSS0_INTRROUTER0_OUTL_7
R5FSS0_CORE0_INTR_IN_264	264	R5FSS0_INTRROUTER0_OUTL_8
R5FSS0_CORE0_INTR_IN_265	265	R5FSS0_INTRROUTER0_OUTL_9
R5FSS0_CORE0_INTR_IN_266	266	R5FSS0_INTRROUTER0_OUTL_10
R5FSS0_CORE0_INTR_IN_267	267	R5FSS0_INTRROUTER0_OUTL_11
R5FSS0_CORE0_INTR_IN_268	268	R5FSS0_INTRROUTER0_OUTL_12
R5FSS0_CORE0_INTR_IN_269	269	R5FSS0_INTRROUTER0_OUTL_13
R5FSS0_CORE0_INTR_IN_270	270	R5FSS0_INTRROUTER0_OUTL_14
R5FSS0_CORE0_INTR_IN_271	271	R5FSS0_INTRROUTER0_OUTL_15
R5FSS0_CORE0_INTR_IN_272	272	R5FSS0_INTRROUTER0_OUTL_16
R5FSS0_CORE0_INTR_IN_273	273	R5FSS0_INTRROUTER0_OUTL_17
R5FSS0_CORE0_INTR_IN_274	274	R5FSS0_INTRROUTER0_OUTL_18
R5FSS0_CORE0_INTR_IN_275	275	R5FSS0_INTRROUTER0_OUTL_19
R5FSS0_CORE0_INTR_IN_276	276	R5FSS0_INTRROUTER0_OUTL_20
R5FSS0_CORE0_INTR_IN_277	277	R5FSS0_INTRROUTER0_OUTL_21
R5FSS0_CORE0_INTR_IN_278	278	R5FSS0_INTRROUTER0_OUTL_22
R5FSS0_CORE0_INTR_IN_279	279	R5FSS0_INTRROUTER0_OUTL_23
R5FSS0_CORE0_INTR_IN_280	280	R5FSS0_INTRROUTER0_OUTL_24
R5FSS0_CORE0_INTR_IN_281	281	R5FSS0_INTRROUTER0_OUTL_25
R5FSS0_CORE0_INTR_IN_282	282	R5FSS0_INTRROUTER0_OUTL_26
R5FSS0_CORE0_INTR_IN_283	283	R5FSS0_INTRROUTER0_OUTL_27
R5FSS0_CORE0_INTR_IN_284	284	R5FSS0_INTRROUTER0_OUTL_28
R5FSS0_CORE0_INTR_IN_285	285	R5FSS0_INTRROUTER0_OUTL_29
R5FSS0_CORE0_INTR_IN_286	286	R5FSS0_INTRROUTER0_OUTL_30
R5FSS0_CORE0_INTR_IN_287	287	R5FSS0_INTRROUTER0_OUTL_31

**Table 9-47. R5FSS0\_CORE0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS0_CORE0_INTR_IN_288	288	R5FSS0_INTROUTER0_OUTL_32
R5FSS0_CORE0_INTR_IN_289	289	R5FSS0_INTROUTER0_OUTL_33
R5FSS0_CORE0_INTR_IN_290	290	R5FSS0_INTROUTER0_OUTL_34
R5FSS0_CORE0_INTR_IN_291	291	R5FSS0_INTROUTER0_OUTL_35
R5FSS0_CORE0_INTR_IN_292	292	R5FSS0_INTROUTER0_OUTL_36
R5FSS0_CORE0_INTR_IN_293	293	R5FSS0_INTROUTER0_OUTL_37
R5FSS0_CORE0_INTR_IN_294	294	R5FSS0_INTROUTER0_OUTL_38
R5FSS0_CORE0_INTR_IN_295	295	R5FSS0_INTROUTER0_OUTL_39
R5FSS0_CORE0_INTR_IN_296	296	R5FSS0_INTROUTER0_OUTL_40
R5FSS0_CORE0_INTR_IN_297	297	R5FSS0_INTROUTER0_OUTL_41
R5FSS0_CORE0_INTR_IN_298	298	R5FSS0_INTROUTER0_OUTL_42
R5FSS0_CORE0_INTR_IN_299	299	R5FSS0_INTROUTER0_OUTL_43
R5FSS0_CORE0_INTR_IN_300	300	R5FSS0_INTROUTER0_OUTL_44
R5FSS0_CORE0_INTR_IN_301	301	R5FSS0_INTROUTER0_OUTL_45
R5FSS0_CORE0_INTR_IN_302	302	R5FSS0_INTROUTER0_OUTL_46
R5FSS0_CORE0_INTR_IN_303	303	R5FSS0_INTROUTER0_OUTL_47
R5FSS0_CORE0_INTR_IN_304	304	R5FSS0_INTROUTER0_OUTL_48
R5FSS0_CORE0_INTR_IN_305	305	R5FSS0_INTROUTER0_OUTL_49
R5FSS0_CORE0_INTR_IN_306	306	R5FSS0_INTROUTER0_OUTL_50
R5FSS0_CORE0_INTR_IN_307	307	R5FSS0_INTROUTER0_OUTL_51
R5FSS0_CORE0_INTR_IN_308	308	R5FSS0_INTROUTER0_OUTL_52
R5FSS0_CORE0_INTR_IN_309	309	R5FSS0_INTROUTER0_OUTL_53
R5FSS0_CORE0_INTR_IN_310	310	R5FSS0_INTROUTER0_OUTL_54
R5FSS0_CORE0_INTR_IN_311	311	R5FSS0_INTROUTER0_OUTL_55
R5FSS0_CORE0_INTR_IN_312	312	R5FSS0_INTROUTER0_OUTL_56
R5FSS0_CORE0_INTR_IN_313	313	R5FSS0_INTROUTER0_OUTL_57
R5FSS0_CORE0_INTR_IN_314	314	R5FSS0_INTROUTER0_OUTL_58
R5FSS0_CORE0_INTR_IN_315	315	R5FSS0_INTROUTER0_OUTL_59
R5FSS0_CORE0_INTR_IN_316	316	R5FSS0_INTROUTER0_OUTL_60
R5FSS0_CORE0_INTR_IN_317	317	R5FSS0_INTROUTER0_OUTL_61
R5FSS0_CORE0_INTR_IN_318	318	R5FSS0_INTROUTER0_OUTL_62
R5FSS0_CORE0_INTR_IN_319	319	R5FSS0_INTROUTER0_OUTL_63
R5FSS0_CORE0_INTR_IN_320	320	R5FSS0_INTROUTER0_OUTL_64
R5FSS0_CORE0_INTR_IN_321	321	R5FSS0_INTROUTER0_OUTL_65
R5FSS0_CORE0_INTR_IN_322	322	R5FSS0_INTROUTER0_OUTL_66
R5FSS0_CORE0_INTR_IN_323	323	R5FSS0_INTROUTER0_OUTL_67
R5FSS0_CORE0_INTR_IN_324	324	R5FSS0_INTROUTER0_OUTL_68
R5FSS0_CORE0_INTR_IN_325	325	R5FSS0_INTROUTER0_OUTL_69
R5FSS0_CORE0_INTR_IN_326	326	R5FSS0_INTROUTER0_OUTL_70
R5FSS0_CORE0_INTR_IN_327	327	R5FSS0_INTROUTER0_OUTL_71
R5FSS0_CORE0_INTR_IN_328	328	R5FSS0_INTROUTER0_OUTL_72
R5FSS0_CORE0_INTR_IN_329	329	R5FSS0_INTROUTER0_OUTL_73
R5FSS0_CORE0_INTR_IN_330	330	R5FSS0_INTROUTER0_OUTL_74
R5FSS0_CORE0_INTR_IN_331	331	R5FSS0_INTROUTER0_OUTL_75
R5FSS0_CORE0_INTR_IN_332	332	R5FSS0_INTROUTER0_OUTL_76



**Table 9-47. R5FSS0\_CORE0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS0_CORE0_INTR_IN_333	333	R5FSS0_INTRROUTER0_OUTL_77
R5FSS0_CORE0_INTR_IN_334	334	R5FSS0_INTRROUTER0_OUTL_78
R5FSS0_CORE0_INTR_IN_335	335	R5FSS0_INTRROUTER0_OUTL_79
R5FSS0_CORE0_INTR_IN_336	336	R5FSS0_INTRROUTER0_OUTL_80
R5FSS0_CORE0_INTR_IN_337	337	R5FSS0_INTRROUTER0_OUTL_81
R5FSS0_CORE0_INTR_IN_338	338	R5FSS0_INTRROUTER0_OUTL_82
R5FSS0_CORE0_INTR_IN_339	339	R5FSS0_INTRROUTER0_OUTL_83
R5FSS0_CORE0_INTR_IN_340	340	R5FSS0_INTRROUTER0_OUTL_84
R5FSS0_CORE0_INTR_IN_341	341	R5FSS0_INTRROUTER0_OUTL_85
R5FSS0_CORE0_INTR_IN_342	342	R5FSS0_INTRROUTER0_OUTL_86
R5FSS0_CORE0_INTR_IN_343	343	R5FSS0_INTRROUTER0_OUTL_87
R5FSS0_CORE0_INTR_IN_344	344	R5FSS0_INTRROUTER0_OUTL_88
R5FSS0_CORE0_INTR_IN_345	345	R5FSS0_INTRROUTER0_OUTL_89
R5FSS0_CORE0_INTR_IN_346	346	R5FSS0_INTRROUTER0_OUTL_90
R5FSS0_CORE0_INTR_IN_347	347	R5FSS0_INTRROUTER0_OUTL_91
R5FSS0_CORE0_INTR_IN_348	348	R5FSS0_INTRROUTER0_OUTL_92
R5FSS0_CORE0_INTR_IN_349	349	R5FSS0_INTRROUTER0_OUTL_93
R5FSS0_CORE0_INTR_IN_350	350	R5FSS0_INTRROUTER0_OUTL_94
R5FSS0_CORE0_INTR_IN_351	351	R5FSS0_INTRROUTER0_OUTL_95
R5FSS0_CORE0_INTR_IN_352	352	R5FSS0_INTRROUTER0_OUTL_96
R5FSS0_CORE0_INTR_IN_353	353	R5FSS0_INTRROUTER0_OUTL_97
R5FSS0_CORE0_INTR_IN_354	354	R5FSS0_INTRROUTER0_OUTL_98
R5FSS0_CORE0_INTR_IN_355	355	R5FSS0_INTRROUTER0_OUTL_99
R5FSS0_CORE0_INTR_IN_356	356	R5FSS0_INTRROUTER0_OUTL_100
R5FSS0_CORE0_INTR_IN_357	357	R5FSS0_INTRROUTER0_OUTL_101
R5FSS0_CORE0_INTR_IN_358	358	R5FSS0_INTRROUTER0_OUTL_102
R5FSS0_CORE0_INTR_IN_359	359	R5FSS0_INTRROUTER0_OUTL_103
R5FSS0_CORE0_INTR_IN_360	360	R5FSS0_INTRROUTER0_OUTL_104
R5FSS0_CORE0_INTR_IN_361	361	R5FSS0_INTRROUTER0_OUTL_105
R5FSS0_CORE0_INTR_IN_362	362	R5FSS0_INTRROUTER0_OUTL_106
R5FSS0_CORE0_INTR_IN_363	363	R5FSS0_INTRROUTER0_OUTL_107
R5FSS0_CORE0_INTR_IN_364	364	R5FSS0_INTRROUTER0_OUTL_108
R5FSS0_CORE0_INTR_IN_365	365	R5FSS0_INTRROUTER0_OUTL_109
R5FSS0_CORE0_INTR_IN_366	366	R5FSS0_INTRROUTER0_OUTL_110
R5FSS0_CORE0_INTR_IN_367	367	R5FSS0_INTRROUTER0_OUTL_111
R5FSS0_CORE0_INTR_IN_368	368	R5FSS0_INTRROUTER0_OUTL_112
R5FSS0_CORE0_INTR_IN_369	369	R5FSS0_INTRROUTER0_OUTL_113
R5FSS0_CORE0_INTR_IN_370	370	R5FSS0_INTRROUTER0_OUTL_114
R5FSS0_CORE0_INTR_IN_371	371	R5FSS0_INTRROUTER0_OUTL_115
R5FSS0_CORE0_INTR_IN_372	372	R5FSS0_INTRROUTER0_OUTL_116
R5FSS0_CORE0_INTR_IN_373	373	R5FSS0_INTRROUTER0_OUTL_117
R5FSS0_CORE0_INTR_IN_374	374	R5FSS0_INTRROUTER0_OUTL_118
R5FSS0_CORE0_INTR_IN_375	375	R5FSS0_INTRROUTER0_OUTL_119
R5FSS0_CORE0_INTR_IN_376	376	R5FSS0_INTRROUTER0_OUTL_120
R5FSS0_CORE0_INTR_IN_377	377	R5FSS0_INTRROUTER0_OUTL_121

**Table 9-47. R5FSS0\_CORE0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS0_CORE0_INTR_IN_378	378	R5FSS0_INTROUTER0_OUTL_122
R5FSS0_CORE0_INTR_IN_379	379	R5FSS0_INTROUTER0_OUTL_123
R5FSS0_CORE0_INTR_IN_380	380	R5FSS0_INTROUTER0_OUTL_124
R5FSS0_CORE0_INTR_IN_381	381	R5FSS0_INTROUTER0_OUTL_125
R5FSS0_CORE0_INTR_IN_382	382	R5FSS0_INTROUTER0_OUTL_126
R5FSS0_CORE0_INTR_IN_383	383	R5FSS0_INTROUTER0_OUTL_127
R5FSS0_CORE0_INTR_IN_384	384	R5FSS0_INTROUTER0_OUTL_128
R5FSS0_CORE0_INTR_IN_385	385	R5FSS0_INTROUTER0_OUTL_129
R5FSS0_CORE0_INTR_IN_386	386	R5FSS0_INTROUTER0_OUTL_130
R5FSS0_CORE0_INTR_IN_387	387	R5FSS0_INTROUTER0_OUTL_131
R5FSS0_CORE0_INTR_IN_388	388	R5FSS0_INTROUTER0_OUTL_132
R5FSS0_CORE0_INTR_IN_389	389	R5FSS0_INTROUTER0_OUTL_133
R5FSS0_CORE0_INTR_IN_390	390	R5FSS0_INTROUTER0_OUTL_134
R5FSS0_CORE0_INTR_IN_391	391	R5FSS0_INTROUTER0_OUTL_135
R5FSS0_CORE0_INTR_IN_392	392	R5FSS0_INTROUTER0_OUTL_136
R5FSS0_CORE0_INTR_IN_393	393	R5FSS0_INTROUTER0_OUTL_137
R5FSS0_CORE0_INTR_IN_394	394	R5FSS0_INTROUTER0_OUTL_138
R5FSS0_CORE0_INTR_IN_395	395	R5FSS0_INTROUTER0_OUTL_139
R5FSS0_CORE0_INTR_IN_396	396	R5FSS0_INTROUTER0_OUTL_140
R5FSS0_CORE0_INTR_IN_397	397	R5FSS0_INTROUTER0_OUTL_141
R5FSS0_CORE0_INTR_IN_398	398	R5FSS0_INTROUTER0_OUTL_142
R5FSS0_CORE0_INTR_IN_399	399	R5FSS0_INTROUTER0_OUTL_143
R5FSS0_CORE0_INTR_IN_400	400	R5FSS0_INTROUTER0_OUTL_144
R5FSS0_CORE0_INTR_IN_401	401	R5FSS0_INTROUTER0_OUTL_145
R5FSS0_CORE0_INTR_IN_402	402	R5FSS0_INTROUTER0_OUTL_146
R5FSS0_CORE0_INTR_IN_403	403	R5FSS0_INTROUTER0_OUTL_147
R5FSS0_CORE0_INTR_IN_404	404	R5FSS0_INTROUTER0_OUTL_148
R5FSS0_CORE0_INTR_IN_405	405	R5FSS0_INTROUTER0_OUTL_149
R5FSS0_CORE0_INTR_IN_406	406	R5FSS0_INTROUTER0_OUTL_150
R5FSS0_CORE0_INTR_IN_407	407	R5FSS0_INTROUTER0_OUTL_151
R5FSS0_CORE0_INTR_IN_408	408	R5FSS0_INTROUTER0_OUTL_152
R5FSS0_CORE0_INTR_IN_409	409	R5FSS0_INTROUTER0_OUTL_153
R5FSS0_CORE0_INTR_IN_410	410	R5FSS0_INTROUTER0_OUTL_154
R5FSS0_CORE0_INTR_IN_411	411	R5FSS0_INTROUTER0_OUTL_155
R5FSS0_CORE0_INTR_IN_412	412	R5FSS0_INTROUTER0_OUTL_156
R5FSS0_CORE0_INTR_IN_413	413	R5FSS0_INTROUTER0_OUTL_157
R5FSS0_CORE0_INTR_IN_414	414	R5FSS0_INTROUTER0_OUTL_158
R5FSS0_CORE0_INTR_IN_415	415	R5FSS0_INTROUTER0_OUTL_159
R5FSS0_CORE0_INTR_IN_416	416	R5FSS0_INTROUTER0_OUTL_160
R5FSS0_CORE0_INTR_IN_417	417	R5FSS0_INTROUTER0_OUTL_161
R5FSS0_CORE0_INTR_IN_418	418	R5FSS0_INTROUTER0_OUTL_162
R5FSS0_CORE0_INTR_IN_419	419	R5FSS0_INTROUTER0_OUTL_163
R5FSS0_CORE0_INTR_IN_420	420	R5FSS0_INTROUTER0_OUTL_164
R5FSS0_CORE0_INTR_IN_421	421	R5FSS0_INTROUTER0_OUTL_165
R5FSS0_CORE0_INTR_IN_422	422	R5FSS0_INTROUTER0_OUTL_166

**Table 9-47. R5FSS0\_CORE0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS0_CORE0_INTR_IN_423	423	R5FSS0_INTROUTER0_OUTL_167
R5FSS0_CORE0_INTR_IN_424	424	R5FSS0_INTROUTER0_OUTL_168
R5FSS0_CORE0_INTR_IN_425	425	R5FSS0_INTROUTER0_OUTL_169
R5FSS0_CORE0_INTR_IN_426	426	R5FSS0_INTROUTER0_OUTL_170
R5FSS0_CORE0_INTR_IN_427	427	R5FSS0_INTROUTER0_OUTL_171
R5FSS0_CORE0_INTR_IN_428	428	R5FSS0_INTROUTER0_OUTL_172
R5FSS0_CORE0_INTR_IN_429	429	R5FSS0_INTROUTER0_OUTL_173
R5FSS0_CORE0_INTR_IN_430	430	R5FSS0_INTROUTER0_OUTL_174
R5FSS0_CORE0_INTR_IN_431	431	R5FSS0_INTROUTER0_OUTL_175
R5FSS0_CORE0_INTR_IN_432	432	R5FSS0_INTROUTER0_OUTL_176
R5FSS0_CORE0_INTR_IN_433	433	R5FSS0_INTROUTER0_OUTL_177
R5FSS0_CORE0_INTR_IN_434	434	R5FSS0_INTROUTER0_OUTL_178
R5FSS0_CORE0_INTR_IN_435	435	R5FSS0_INTROUTER0_OUTL_179
R5FSS0_CORE0_INTR_IN_436	436	R5FSS0_INTROUTER0_OUTL_180
R5FSS0_CORE0_INTR_IN_437	437	R5FSS0_INTROUTER0_OUTL_181
R5FSS0_CORE0_INTR_IN_438	438	R5FSS0_INTROUTER0_OUTL_182
R5FSS0_CORE0_INTR_IN_439	439	R5FSS0_INTROUTER0_OUTL_183
R5FSS0_CORE0_INTR_IN_440	440	R5FSS0_INTROUTER0_OUTL_184
R5FSS0_CORE0_INTR_IN_441	441	R5FSS0_INTROUTER0_OUTL_185
R5FSS0_CORE0_INTR_IN_442	442	R5FSS0_INTROUTER0_OUTL_186
R5FSS0_CORE0_INTR_IN_443	443	R5FSS0_INTROUTER0_OUTL_187
R5FSS0_CORE0_INTR_IN_444	444	R5FSS0_INTROUTER0_OUTL_188
R5FSS0_CORE0_INTR_IN_445	445	R5FSS0_INTROUTER0_OUTL_189
R5FSS0_CORE0_INTR_IN_446	446	R5FSS0_INTROUTER0_OUTL_190
R5FSS0_CORE0_INTR_IN_447	447	R5FSS0_INTROUTER0_OUTL_191
R5FSS0_CORE0_INTR_IN_448	448	R5FSS0_INTROUTER0_OUTL_192
R5FSS0_CORE0_INTR_IN_449	449	R5FSS0_INTROUTER0_OUTL_193
R5FSS0_CORE0_INTR_IN_450	450	R5FSS0_INTROUTER0_OUTL_194
R5FSS0_CORE0_INTR_IN_451	451	R5FSS0_INTROUTER0_OUTL_195
R5FSS0_CORE0_INTR_IN_452	452	R5FSS0_INTROUTER0_OUTL_196
R5FSS0_CORE0_INTR_IN_453	453	R5FSS0_INTROUTER0_OUTL_197
R5FSS0_CORE0_INTR_IN_454	454	R5FSS0_INTROUTER0_OUTL_198
R5FSS0_CORE0_INTR_IN_455	455	R5FSS0_INTROUTER0_OUTL_199
R5FSS0_CORE0_INTR_IN_456	456	R5FSS0_INTROUTER0_OUTL_200
R5FSS0_CORE0_INTR_IN_457	457	R5FSS0_INTROUTER0_OUTL_201
R5FSS0_CORE0_INTR_IN_458	458	R5FSS0_INTROUTER0_OUTL_202
R5FSS0_CORE0_INTR_IN_459	459	R5FSS0_INTROUTER0_OUTL_203
R5FSS0_CORE0_INTR_IN_460	460	R5FSS0_INTROUTER0_OUTL_204
R5FSS0_CORE0_INTR_IN_461	461	R5FSS0_INTROUTER0_OUTL_205
R5FSS0_CORE0_INTR_IN_462	462	R5FSS0_INTROUTER0_OUTL_206
R5FSS0_CORE0_INTR_IN_463	463	R5FSS0_INTROUTER0_OUTL_207
R5FSS0_CORE0_INTR_IN_464	464	R5FSS0_INTROUTER0_OUTL_208
R5FSS0_CORE0_INTR_IN_465	465	R5FSS0_INTROUTER0_OUTL_209
R5FSS0_CORE0_INTR_IN_466	466	R5FSS0_INTROUTER0_OUTL_210
R5FSS0_CORE0_INTR_IN_467	467	R5FSS0_INTROUTER0_OUTL_211

**Table 9-47. R5FSS0\_CORE0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS0_CORE0_INTR_IN_468	468	R5FSS0_INTROUTER0_OUTL_212
R5FSS0_CORE0_INTR_IN_469	469	R5FSS0_INTROUTER0_OUTL_213
R5FSS0_CORE0_INTR_IN_470	470	R5FSS0_INTROUTER0_OUTL_214
R5FSS0_CORE0_INTR_IN_471	471	R5FSS0_INTROUTER0_OUTL_215
R5FSS0_CORE0_INTR_IN_472	472	R5FSS0_INTROUTER0_OUTL_216
R5FSS0_CORE0_INTR_IN_473	473	R5FSS0_INTROUTER0_OUTL_217
R5FSS0_CORE0_INTR_IN_474	474	R5FSS0_INTROUTER0_OUTL_218
R5FSS0_CORE0_INTR_IN_475	475	R5FSS0_INTROUTER0_OUTL_219
R5FSS0_CORE0_INTR_IN_476	476	R5FSS0_INTROUTER0_OUTL_220
R5FSS0_CORE0_INTR_IN_477	477	R5FSS0_INTROUTER0_OUTL_221
R5FSS0_CORE0_INTR_IN_478	478	R5FSS0_INTROUTER0_OUTL_222
R5FSS0_CORE0_INTR_IN_479	479	R5FSS0_INTROUTER0_OUTL_223
R5FSS0_CORE0_INTR_IN_480	480	R5FSS0_INTROUTER0_OUTL_224
R5FSS0_CORE0_INTR_IN_481	481	R5FSS0_INTROUTER0_OUTL_225
R5FSS0_CORE0_INTR_IN_482	482	R5FSS0_INTROUTER0_OUTL_226
R5FSS0_CORE0_INTR_IN_483	483	R5FSS0_INTROUTER0_OUTL_227
R5FSS0_CORE0_INTR_IN_484	484	R5FSS0_INTROUTER0_OUTL_228
R5FSS0_CORE0_INTR_IN_485	485	R5FSS0_INTROUTER0_OUTL_229
R5FSS0_CORE0_INTR_IN_486	486	R5FSS0_INTROUTER0_OUTL_230
R5FSS0_CORE0_INTR_IN_487	487	R5FSS0_INTROUTER0_OUTL_231
R5FSS0_CORE0_INTR_IN_488	488	R5FSS0_INTROUTER0_OUTL_232
R5FSS0_CORE0_INTR_IN_489	489	R5FSS0_INTROUTER0_OUTL_233
R5FSS0_CORE0_INTR_IN_490	490	R5FSS0_INTROUTER0_OUTL_234
R5FSS0_CORE0_INTR_IN_491	491	R5FSS0_INTROUTER0_OUTL_235
R5FSS0_CORE0_INTR_IN_492	492	R5FSS0_INTROUTER0_OUTL_236
R5FSS0_CORE0_INTR_IN_493	493	R5FSS0_INTROUTER0_OUTL_237
R5FSS0_CORE0_INTR_IN_494	494	R5FSS0_INTROUTER0_OUTL_238
R5FSS0_CORE0_INTR_IN_495	495	R5FSS0_INTROUTER0_OUTL_239
R5FSS0_CORE0_INTR_IN_496	496	R5FSS0_INTROUTER0_OUTL_240
R5FSS0_CORE0_INTR_IN_497	497	R5FSS0_INTROUTER0_OUTL_241
R5FSS0_CORE0_INTR_IN_498	498	R5FSS0_INTROUTER0_OUTL_242
R5FSS0_CORE0_INTR_IN_499	499	R5FSS0_INTROUTER0_OUTL_243
R5FSS0_CORE0_INTR_IN_500	500	R5FSS0_INTROUTER0_OUTL_244
R5FSS0_CORE0_INTR_IN_501	501	R5FSS0_INTROUTER0_OUTL_245
R5FSS0_CORE0_INTR_IN_502	502	R5FSS0_INTROUTER0_OUTL_246
R5FSS0_CORE0_INTR_IN_503	503	R5FSS0_INTROUTER0_OUTL_247
R5FSS0_CORE0_INTR_IN_504	504	R5FSS0_INTROUTER0_OUTL_248
R5FSS0_CORE0_INTR_IN_505	505	R5FSS0_INTROUTER0_OUTL_249
R5FSS0_CORE0_INTR_IN_506	506	R5FSS0_INTROUTER0_OUTL_250
R5FSS0_CORE0_INTR_IN_507	507	R5FSS0_INTROUTER0_OUTL_251
R5FSS0_CORE0_INTR_IN_508	508	R5FSS0_INTROUTER0_OUTL_252
R5FSS0_CORE0_INTR_IN_509	509	R5FSS0_INTROUTER0_OUTL_253
R5FSS0_CORE0_INTR_IN_510	510	R5FSS0_INTROUTER0_OUTL_254
R5FSS0_CORE0_INTR_IN_511	511	R5FSS0_INTROUTER0_OUTL_255

#### 9.4.3.3 R5FSS0\_CORE1 Interrupt Map

Table 9-48 shows the mapping of events to the R5FSS0\_CORE1. The R5FSS0 VIM supports both R5FSS0 cores.

The R5FSS0\_CORE1 events are not used when operating in lockstep mode.

**Table 9-48. R5FSS0\_CORE1 Interrupt Map**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS0_CORE1_INTR_IN_0	0	CTRL_MMR0_IPC_SET16_IPC_SET_IPCFG_0
R5FSS0_CORE1_INTR_IN_1	1	CTRL_MMR0_IPC_SET17_IPC_SET_IPCFG_0
R5FSS0_CORE1_INTR_IN_2	2	RTI28_INTR_WWD_0
R5FSS0_CORE1_INTR_IN_3	3	RTI29_INTR_WWD_0
R5FSS0_CORE1_INTR_IN_4	4	R5FSS0_COMMRX_LEVEL_1_0
R5FSS0_CORE1_INTR_IN_5	5	R5FSS0_COMMTX_LEVEL_1_0
R5FSS0_CORE1_INTR_IN_6	6	R5FSS0_CORE1_VALFIQ_0
R5FSS0_CORE1_INTR_IN_7	7	R5FSS0_CORE1_VALIRQ_0
R5FSS0_CORE1_INTR_IN_8	8	R5FSS0_CORE0_CTI_0
R5FSS0_CORE1_INTR_IN_9	9	R5FSS0_CORE1_CTI_0
R5FSS0_CORE1_INTR_IN_10	10	ESM0_ESM_INT_LOW_LVL_0
R5FSS0_CORE1_INTR_IN_11	11	ESM0_ESM_INT_HI_LVL_0
R5FSS0_CORE1_INTR_IN_12	12	ESM0_ESM_INT_CFG_LVL_0
R5FSS0_CORE1_INTR_IN_13	13	GLUELOGIC_EXT_INTN_GLUE_EXT_INT_LVL_0
R5FSS0_CORE1_INTR_IN_16	16	R5FSS0_CORE0_EXP_INTR_0
R5FSS0_CORE1_INTR_IN_17	17	R5FSS0_CORE1_EXP_INTR_0
R5FSS0_CORE1_INTR_IN_32	32	DMPAC0_INTD_0_SYSTEM_INTR_LEVEL_0
R5FSS0_CORE1_INTR_IN_33	33	DMPAC0_INTD_0_SYSTEM_INTR_LEVEL_1
R5FSS0_CORE1_INTR_IN_34	34	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_0
R5FSS0_CORE1_INTR_IN_35	35	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_1
R5FSS0_CORE1_INTR_IN_36	36	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_2
R5FSS0_CORE1_INTR_IN_37	37	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_3
R5FSS0_CORE1_INTR_IN_38	38	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_4
R5FSS0_CORE1_INTR_IN_39	39	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_5
R5FSS0_CORE1_INTR_IN_40	40	GPU0_MISC_0_IRQ_0
R5FSS0_CORE1_INTR_IN_48	48	GLUELOGIC_GPU_GPIO_REQACK_GLUE_GPU_GPIO_REQINT_LVL_0
R5FSS0_CORE1_INTR_IN_49	49	GLUELOGIC_GPU_GPIO_REQACK_GLUE_GPU_GPIO_ACKINT_LVL_0
R5FSS0_CORE1_INTR_IN_52	52	DSS0_DSS_INST0_DISPC_FUNC_IRQ_PROC0_0
R5FSS0_CORE1_INTR_IN_53	53	DSS0_DSS_INST0_DISPC_FUNC_IRQ_PROC1_0
R5FSS0_CORE1_INTR_IN_54	54	DSS0_DSS_INST0_DISPC_SAFETY_ERROR_IRQ_PROC0_0
R5FSS0_CORE1_INTR_IN_55	55	DSS0_DSS_INST0_DISPC_SAFETY_ERROR_IRQ_PROC1_0
R5FSS0_CORE1_INTR_IN_56	56	DSS0_DSS_INST0_DISPC_SECURE_IRQ_PROC0_0
R5FSS0_CORE1_INTR_IN_57	57	DSS0_DSS_INST0_DISPC_SECURE_IRQ_PROC1_0
R5FSS0_CORE1_INTR_IN_64	64	DSS_EDP0_INTR_0
R5FSS0_CORE1_INTR_IN_65	65	DSS_EDP0_INTR_1
R5FSS0_CORE1_INTR_IN_66	66	DSS_EDP0_INTR_2
R5FSS0_CORE1_INTR_IN_67	67	DSS_EDP0_INTR_3
R5FSS0_CORE1_INTR_IN_76	76	DSS_DSI0_DSI_0_FUNC_INTR_0
R5FSS0_CORE1_INTR_IN_78	78	CSI_RX_IF0_CSI_ERR_IRQ_0
R5FSS0_CORE1_INTR_IN_79	79	CSI_RX_IF0_CSI_IRQ_0

**Table 9-48. R5FSS0\_CORE1 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS0_CORE1_INTR_IN_80	80	CSI_RX_IF0_CSI_LEVEL_0
R5FSS0_CORE1_INTR_IN_81	81	CSI_RX_IF1_CSI_ERR_IRQ_0
R5FSS0_CORE1_INTR_IN_82	82	CSI_RX_IF1_CSI_IRQ_0
R5FSS0_CORE1_INTR_IN_83	83	CSI_RX_IF1_CSI_LEVEL_0
R5FSS0_CORE1_INTR_IN_87	87	DECODER0_IRQ_0
R5FSS0_CORE1_INTR_IN_88	88	ENCODER0_IRQ_0
R5FSS0_CORE1_INTR_IN_96	96	CPSW0_STAT_PEND_0
R5FSS0_CORE1_INTR_IN_97	97	CPSW0_MDIO_PEND_0
R5FSS0_CORE1_INTR_IN_98	98	CPSW0_EVTN_PEND_0
R5FSS0_CORE1_INTR_IN_99	99	MMCS00_EMMCSS_INTR_0
R5FSS0_CORE1_INTR_IN_104	104	EHRPWM0_EPWM_ETINT_0
R5FSS0_CORE1_INTR_IN_105	105	EHRPWM1_EPWM_ETINT_0
R5FSS0_CORE1_INTR_IN_106	106	EHRPWM2_EPWM_ETINT_0
R5FSS0_CORE1_INTR_IN_107	107	EHRPWM3_EPWM_ETINT_0
R5FSS0_CORE1_INTR_IN_108	108	EHRPWM4_EPWM_ETINT_0
R5FSS0_CORE1_INTR_IN_109	109	EHRPWM5_EPWM_ETINT_0
R5FSS0_CORE1_INTR_IN_110	110	EHRPWM0_EPWM_TRIPZINT_0
R5FSS0_CORE1_INTR_IN_111	111	EHRPWM1_EPWM_TRIPZINT_0
R5FSS0_CORE1_INTR_IN_112	112	EHRPWM2_EPWM_TRIPZINT_0
R5FSS0_CORE1_INTR_IN_113	113	EHRPWM3_EPWM_TRIPZINT_0
R5FSS0_CORE1_INTR_IN_114	114	EHRPWM4_EPWM_TRIPZINT_0
R5FSS0_CORE1_INTR_IN_115	115	EHRPWM5_EPWM_TRIPZINT_0
R5FSS0_CORE1_INTR_IN_116	116	EQEP0_EQEP_INT_0
R5FSS0_CORE1_INTR_IN_117	117	EQEP1_EQEP_INT_0
R5FSS0_CORE1_INTR_IN_118	118	EQEP2_EQEP_INT_0
R5FSS0_CORE1_INTR_IN_120	120	MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS0_CORE1_INTR_IN_121	121	MCAN0_MCANSS_MCAN_LVL_INT_0
R5FSS0_CORE1_INTR_IN_122	122	MCAN0_MCANSS_MCAN_LVL_INT_1
R5FSS0_CORE1_INTR_IN_123	123	MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS0_CORE1_INTR_IN_124	124	MCAN1_MCANSS_MCAN_LVL_INT_0
R5FSS0_CORE1_INTR_IN_125	125	MCAN1_MCANSS_MCAN_LVL_INT_1
R5FSS0_CORE1_INTR_IN_126	126	MCAN2_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS0_CORE1_INTR_IN_127	127	MCAN2_MCANSS_MCAN_LVL_INT_0
R5FSS0_CORE1_INTR_IN_128	128	MCAN2_MCANSS_MCAN_LVL_INT_1
R5FSS0_CORE1_INTR_IN_129	129	MCAN3_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS0_CORE1_INTR_IN_130	130	MCAN3_MCANSS_MCAN_LVL_INT_0
R5FSS0_CORE1_INTR_IN_131	131	MCAN3_MCANSS_MCAN_LVL_INT_1
R5FSS0_CORE1_INTR_IN_132	132	MCASP0_XMIT_INTR_PEND_0
R5FSS0_CORE1_INTR_IN_133	133	MCASP0_REC_INTR_PEND_0
R5FSS0_CORE1_INTR_IN_134	134	MCASP1_XMIT_INTR_PEND_0
R5FSS0_CORE1_INTR_IN_135	135	MCASP1_REC_INTR_PEND_0
R5FSS0_CORE1_INTR_IN_136	136	PCIE0_PCIE_LEGACY_PULSE_0
R5FSS0_CORE1_INTR_IN_137	137	PCIE0_PCIE_DOWNSTREAM_PULSE_0
R5FSS0_CORE1_INTR_IN_138	138	PCIE0_PCIE_FLR_PULSE_0
R5FSS0_CORE1_INTR_IN_139	139	PCIE0_PCIE_PHY_LEVEL_0

**Table 9-48. R5FSS0\_CORE1 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS0_CORE1_INTR_IN_140	140	PCIE0_PCIE_LOCAL_LEVEL_0
R5FSS0_CORE1_INTR_IN_141	141	PCIE0_PCIE_ERROR_PULSE_0
R5FSS0_CORE1_INTR_IN_142	142	PCIE0_PCIE_LINK_STATE_PULSE_0
R5FSS0_CORE1_INTR_IN_143	143	PCIE0_PCIE_PWR_STATE_PULSE_0
R5FSS0_CORE1_INTR_IN_144	144	PCIE0_PCIE_PTM_VALID_PULSE_0
R5FSS0_CORE1_INTR_IN_145	145	PCIE0_PCIE_HOT_RESET_PULSE_0
R5FSS0_CORE1_INTR_IN_146	146	PCIE0_PCIE_CPTS_PEND_0
R5FSS0_CORE1_INTR_IN_150	150	I2C0_POINTRPEND_0
R5FSS0_CORE1_INTR_IN_151	151	I2C1_POINTRPEND_0
R5FSS0_CORE1_INTR_IN_152	152	MCSP10_INTR_SPI_0
R5FSS0_CORE1_INTR_IN_153	153	MCSP11_INTR_SPI_0
R5FSS0_CORE1_INTR_IN_154	154	MLB0_MLBSS_MLB_INT_0
R5FSS0_CORE1_INTR_IN_155	155	MLB0_MLBSS_MLB_AHB_INT_0
R5FSS0_CORE1_INTR_IN_156	156	MLB0_MLBSS_MLB_AHB_INT_1
R5FSS0_CORE1_INTR_IN_158	158	UART0_USART_IRQ_0
R5FSS0_CORE1_INTR_IN_159	159	UART1_USART_IRQ_0
R5FSS0_CORE1_INTR_IN_160	160	UART2_USART_IRQ_0
R5FSS0_CORE1_INTR_IN_168	168	TIMER12_INTR_PEND_0
R5FSS0_CORE1_INTR_IN_169	169	TIMER13_INTR_PEND_0
R5FSS0_CORE1_INTR_IN_170	170	TIMER14_INTR_PEND_0
R5FSS0_CORE1_INTR_IN_171	171	TIMER15_INTR_PEND_0
R5FSS0_CORE1_INTR_IN_172	172	TIMER16_INTR_PEND_0
R5FSS0_CORE1_INTR_IN_173	173	TIMER17_INTR_PEND_0
R5FSS0_CORE1_INTR_IN_174	174	TIMER18_INTR_PEND_0
R5FSS0_CORE1_INTR_IN_175	175	TIMER19_INTR_PEND_0
R5FSS0_CORE1_INTR_IN_176	176	GPIOMUX_INTRTR0_OUTP_16
R5FSS0_CORE1_INTR_IN_177	177	GPIOMUX_INTRTR0_OUTP_17
R5FSS0_CORE1_INTR_IN_178	178	GPIOMUX_INTRTR0_OUTP_18
R5FSS0_CORE1_INTR_IN_179	179	GPIOMUX_INTRTR0_OUTP_19
R5FSS0_CORE1_INTR_IN_180	180	GPIOMUX_INTRTR0_OUTP_20
R5FSS0_CORE1_INTR_IN_181	181	GPIOMUX_INTRTR0_OUTP_21
R5FSS0_CORE1_INTR_IN_182	182	GPIOMUX_INTRTR0_OUTP_22
R5FSS0_CORE1_INTR_IN_183	183	GPIOMUX_INTRTR0_OUTP_23
R5FSS0_CORE1_INTR_IN_184	184	GPIOMUX_INTRTR0_OUTP_24
R5FSS0_CORE1_INTR_IN_185	185	GPIOMUX_INTRTR0_OUTP_25
R5FSS0_CORE1_INTR_IN_186	186	GPIOMUX_INTRTR0_OUTP_26
R5FSS0_CORE1_INTR_IN_187	187	GPIOMUX_INTRTR0_OUTP_27
R5FSS0_CORE1_INTR_IN_188	188	GPIOMUX_INTRTR0_OUTP_28
R5FSS0_CORE1_INTR_IN_189	189	GPIOMUX_INTRTR0_OUTP_29
R5FSS0_CORE1_INTR_IN_190	190	GPIOMUX_INTRTR0_OUTP_30
R5FSS0_CORE1_INTR_IN_191	191	GPIOMUX_INTRTR0_OUTP_31
R5FSS0_CORE1_INTR_IN_192	192	MCAN4_MCANSS_MCAN_LVL_INT_0
R5FSS0_CORE1_INTR_IN_193	193	MCAN4_MCANSS_MCAN_LVL_INT_1
R5FSS0_CORE1_INTR_IN_194	194	MCAN4_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS0_CORE1_INTR_IN_195	195	MCAN5_MCANSS_MCAN_LVL_INT_0



**Table 9-48. R5FSS0\_CORE1 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS0_CORE1_INTR_IN_196	196	MCAN5_MCANSS_MCAN_LVL_INT_1
R5FSS0_CORE1_INTR_IN_197	197	MCAN5_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS0_CORE1_INTR_IN_198	198	MCAN6_MCANSS_MCAN_LVL_INT_0
R5FSS0_CORE1_INTR_IN_199	199	MCAN6_MCANSS_MCAN_LVL_INT_1
R5FSS0_CORE1_INTR_IN_200	200	MCAN6_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS0_CORE1_INTR_IN_201	201	MCAN7_MCANSS_MCAN_LVL_INT_0
R5FSS0_CORE1_INTR_IN_202	202	MCAN7_MCANSS_MCAN_LVL_INT_1
R5FSS0_CORE1_INTR_IN_203	203	MCAN7_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS0_CORE1_INTR_IN_204	204	MCAN8_MCANSS_MCAN_LVL_INT_0
R5FSS0_CORE1_INTR_IN_205	205	MCAN8_MCANSS_MCAN_LVL_INT_1
R5FSS0_CORE1_INTR_IN_206	206	MCAN8_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS0_CORE1_INTR_IN_207	207	MCAN9_MCANSS_MCAN_LVL_INT_0
R5FSS0_CORE1_INTR_IN_208	208	MCAN9_MCANSS_MCAN_LVL_INT_1
R5FSS0_CORE1_INTR_IN_209	209	MCAN9_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS0_CORE1_INTR_IN_210	210	MCAN10_MCANSS_MCAN_LVL_INT_0
R5FSS0_CORE1_INTR_IN_211	211	MCAN10_MCANSS_MCAN_LVL_INT_1
R5FSS0_CORE1_INTR_IN_212	212	MCAN10_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS0_CORE1_INTR_IN_213	213	MCAN11_MCANSS_MCAN_LVL_INT_0
R5FSS0_CORE1_INTR_IN_214	214	MCAN11_MCANSS_MCAN_LVL_INT_1
R5FSS0_CORE1_INTR_IN_215	215	MCAN11_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS0_CORE1_INTR_IN_216	216	MCAN12_MCANSS_MCAN_LVL_INT_0
R5FSS0_CORE1_INTR_IN_217	217	MCAN12_MCANSS_MCAN_LVL_INT_1
R5FSS0_CORE1_INTR_IN_218	218	MCAN12_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS0_CORE1_INTR_IN_219	219	MCAN13_MCANSS_MCAN_LVL_INT_0
R5FSS0_CORE1_INTR_IN_220	220	MCAN13_MCANSS_MCAN_LVL_INT_1
R5FSS0_CORE1_INTR_IN_221	221	MCAN13_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS0_CORE1_INTR_IN_224	224	NAVSS0_INTR_ROUTER_0_OUTL_INTR_224
R5FSS0_CORE1_INTR_IN_225	225	NAVSS0_INTR_ROUTER_0_OUTL_INTR_225
R5FSS0_CORE1_INTR_IN_226	226	NAVSS0_INTR_ROUTER_0_OUTL_INTR_226
R5FSS0_CORE1_INTR_IN_227	227	NAVSS0_INTR_ROUTER_0_OUTL_INTR_227
R5FSS0_CORE1_INTR_IN_228	228	NAVSS0_INTR_ROUTER_0_OUTL_INTR_228
R5FSS0_CORE1_INTR_IN_229	229	NAVSS0_INTR_ROUTER_0_OUTL_INTR_229
R5FSS0_CORE1_INTR_IN_230	230	NAVSS0_INTR_ROUTER_0_OUTL_INTR_230
R5FSS0_CORE1_INTR_IN_231	231	NAVSS0_INTR_ROUTER_0_OUTL_INTR_231
R5FSS0_CORE1_INTR_IN_232	232	NAVSS0_INTR_ROUTER_0_OUTL_INTR_232
R5FSS0_CORE1_INTR_IN_233	233	NAVSS0_INTR_ROUTER_0_OUTL_INTR_233
R5FSS0_CORE1_INTR_IN_234	234	NAVSS0_INTR_ROUTER_0_OUTL_INTR_234
R5FSS0_CORE1_INTR_IN_235	235	NAVSS0_INTR_ROUTER_0_OUTL_INTR_235
R5FSS0_CORE1_INTR_IN_236	236	NAVSS0_INTR_ROUTER_0_OUTL_INTR_236
R5FSS0_CORE1_INTR_IN_237	237	NAVSS0_INTR_ROUTER_0_OUTL_INTR_237
R5FSS0_CORE1_INTR_IN_238	238	NAVSS0_INTR_ROUTER_0_OUTL_INTR_238
R5FSS0_CORE1_INTR_IN_239	239	NAVSS0_INTR_ROUTER_0_OUTL_INTR_239
R5FSS0_CORE1_INTR_IN_240	240	NAVSS0_INTR_ROUTER_0_OUTL_INTR_240
R5FSS0_CORE1_INTR_IN_241	241	NAVSS0_INTR_ROUTER_0_OUTL_INTR_241
R5FSS0_CORE1_INTR_IN_242	242	NAVSS0_INTR_ROUTER_0_OUTL_INTR_242



**Table 9-48. R5FSS0\_CORE1 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS0_CORE1_INTR_IN_243	243	NAVSS0_INTR_ROUTER_0_OUTL_INTR_243
R5FSS0_CORE1_INTR_IN_244	244	NAVSS0_INTR_ROUTER_0_OUTL_INTR_244
R5FSS0_CORE1_INTR_IN_245	245	NAVSS0_INTR_ROUTER_0_OUTL_INTR_245
R5FSS0_CORE1_INTR_IN_246	246	NAVSS0_INTR_ROUTER_0_OUTL_INTR_246
R5FSS0_CORE1_INTR_IN_247	247	NAVSS0_INTR_ROUTER_0_OUTL_INTR_247
R5FSS0_CORE1_INTR_IN_248	248	NAVSS0_INTR_ROUTER_0_OUTL_INTR_248
R5FSS0_CORE1_INTR_IN_249	249	NAVSS0_INTR_ROUTER_0_OUTL_INTR_249
R5FSS0_CORE1_INTR_IN_250	250	NAVSS0_INTR_ROUTER_0_OUTL_INTR_250
R5FSS0_CORE1_INTR_IN_251	251	NAVSS0_INTR_ROUTER_0_OUTL_INTR_251
R5FSS0_CORE1_INTR_IN_252	252	NAVSS0_INTR_ROUTER_0_OUTL_INTR_252
R5FSS0_CORE1_INTR_IN_253	253	NAVSS0_INTR_ROUTER_0_OUTL_INTR_253
R5FSS0_CORE1_INTR_IN_254	254	NAVSS0_INTR_ROUTER_0_OUTL_INTR_254
R5FSS0_CORE1_INTR_IN_255	255	NAVSS0_INTR_ROUTER_0_OUTL_INTR_255
R5FSS0_CORE1_INTR_IN_256	256	R5FSS0_INTRROUTER0_OUTL_0
R5FSS0_CORE1_INTR_IN_257	257	R5FSS0_INTRROUTER0_OUTL_1
R5FSS0_CORE1_INTR_IN_258	258	R5FSS0_INTRROUTER0_OUTL_2
R5FSS0_CORE1_INTR_IN_259	259	R5FSS0_INTRROUTER0_OUTL_3
R5FSS0_CORE1_INTR_IN_260	260	R5FSS0_INTRROUTER0_OUTL_4
R5FSS0_CORE1_INTR_IN_261	261	R5FSS0_INTRROUTER0_OUTL_5
R5FSS0_CORE1_INTR_IN_262	262	R5FSS0_INTRROUTER0_OUTL_6
R5FSS0_CORE1_INTR_IN_263	263	R5FSS0_INTRROUTER0_OUTL_7
R5FSS0_CORE1_INTR_IN_264	264	R5FSS0_INTRROUTER0_OUTL_8
R5FSS0_CORE1_INTR_IN_265	265	R5FSS0_INTRROUTER0_OUTL_9
R5FSS0_CORE1_INTR_IN_266	266	R5FSS0_INTRROUTER0_OUTL_10
R5FSS0_CORE1_INTR_IN_267	267	R5FSS0_INTRROUTER0_OUTL_11
R5FSS0_CORE1_INTR_IN_268	268	R5FSS0_INTRROUTER0_OUTL_12
R5FSS0_CORE1_INTR_IN_269	269	R5FSS0_INTRROUTER0_OUTL_13
R5FSS0_CORE1_INTR_IN_270	270	R5FSS0_INTRROUTER0_OUTL_14
R5FSS0_CORE1_INTR_IN_271	271	R5FSS0_INTRROUTER0_OUTL_15
R5FSS0_CORE1_INTR_IN_272	272	R5FSS0_INTRROUTER0_OUTL_16
R5FSS0_CORE1_INTR_IN_273	273	R5FSS0_INTRROUTER0_OUTL_17
R5FSS0_CORE1_INTR_IN_274	274	R5FSS0_INTRROUTER0_OUTL_18
R5FSS0_CORE1_INTR_IN_275	275	R5FSS0_INTRROUTER0_OUTL_19
R5FSS0_CORE1_INTR_IN_276	276	R5FSS0_INTRROUTER0_OUTL_20
R5FSS0_CORE1_INTR_IN_277	277	R5FSS0_INTRROUTER0_OUTL_21
R5FSS0_CORE1_INTR_IN_278	278	R5FSS0_INTRROUTER0_OUTL_22
R5FSS0_CORE1_INTR_IN_279	279	R5FSS0_INTRROUTER0_OUTL_23
R5FSS0_CORE1_INTR_IN_280	280	R5FSS0_INTRROUTER0_OUTL_24
R5FSS0_CORE1_INTR_IN_281	281	R5FSS0_INTRROUTER0_OUTL_25
R5FSS0_CORE1_INTR_IN_282	282	R5FSS0_INTRROUTER0_OUTL_26
R5FSS0_CORE1_INTR_IN_283	283	R5FSS0_INTRROUTER0_OUTL_27
R5FSS0_CORE1_INTR_IN_284	284	R5FSS0_INTRROUTER0_OUTL_28
R5FSS0_CORE1_INTR_IN_285	285	R5FSS0_INTRROUTER0_OUTL_29
R5FSS0_CORE1_INTR_IN_286	286	R5FSS0_INTRROUTER0_OUTL_30
R5FSS0_CORE1_INTR_IN_287	287	R5FSS0_INTRROUTER0_OUTL_31

**Table 9-48. R5FSS0\_CORE1 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS0_CORE1_INTR_IN_288	288	R5FSS0_INTRROUTER0_OUTL_32
R5FSS0_CORE1_INTR_IN_289	289	R5FSS0_INTRROUTER0_OUTL_33
R5FSS0_CORE1_INTR_IN_290	290	R5FSS0_INTRROUTER0_OUTL_34
R5FSS0_CORE1_INTR_IN_291	291	R5FSS0_INTRROUTER0_OUTL_35
R5FSS0_CORE1_INTR_IN_292	292	R5FSS0_INTRROUTER0_OUTL_36
R5FSS0_CORE1_INTR_IN_293	293	R5FSS0_INTRROUTER0_OUTL_37
R5FSS0_CORE1_INTR_IN_294	294	R5FSS0_INTRROUTER0_OUTL_38
R5FSS0_CORE1_INTR_IN_295	295	R5FSS0_INTRROUTER0_OUTL_39
R5FSS0_CORE1_INTR_IN_296	296	R5FSS0_INTRROUTER0_OUTL_40
R5FSS0_CORE1_INTR_IN_297	297	R5FSS0_INTRROUTER0_OUTL_41
R5FSS0_CORE1_INTR_IN_298	298	R5FSS0_INTRROUTER0_OUTL_42
R5FSS0_CORE1_INTR_IN_299	299	R5FSS0_INTRROUTER0_OUTL_43
R5FSS0_CORE1_INTR_IN_300	300	R5FSS0_INTRROUTER0_OUTL_44
R5FSS0_CORE1_INTR_IN_301	301	R5FSS0_INTRROUTER0_OUTL_45
R5FSS0_CORE1_INTR_IN_302	302	R5FSS0_INTRROUTER0_OUTL_46
R5FSS0_CORE1_INTR_IN_303	303	R5FSS0_INTRROUTER0_OUTL_47
R5FSS0_CORE1_INTR_IN_304	304	R5FSS0_INTRROUTER0_OUTL_48
R5FSS0_CORE1_INTR_IN_305	305	R5FSS0_INTRROUTER0_OUTL_49
R5FSS0_CORE1_INTR_IN_306	306	R5FSS0_INTRROUTER0_OUTL_50
R5FSS0_CORE1_INTR_IN_307	307	R5FSS0_INTRROUTER0_OUTL_51
R5FSS0_CORE1_INTR_IN_308	308	R5FSS0_INTRROUTER0_OUTL_52
R5FSS0_CORE1_INTR_IN_309	309	R5FSS0_INTRROUTER0_OUTL_53
R5FSS0_CORE1_INTR_IN_310	310	R5FSS0_INTRROUTER0_OUTL_54
R5FSS0_CORE1_INTR_IN_311	311	R5FSS0_INTRROUTER0_OUTL_55
R5FSS0_CORE1_INTR_IN_312	312	R5FSS0_INTRROUTER0_OUTL_56
R5FSS0_CORE1_INTR_IN_313	313	R5FSS0_INTRROUTER0_OUTL_57
R5FSS0_CORE1_INTR_IN_314	314	R5FSS0_INTRROUTER0_OUTL_58
R5FSS0_CORE1_INTR_IN_315	315	R5FSS0_INTRROUTER0_OUTL_59
R5FSS0_CORE1_INTR_IN_316	316	R5FSS0_INTRROUTER0_OUTL_60
R5FSS0_CORE1_INTR_IN_317	317	R5FSS0_INTRROUTER0_OUTL_61
R5FSS0_CORE1_INTR_IN_318	318	R5FSS0_INTRROUTER0_OUTL_62
R5FSS0_CORE1_INTR_IN_319	319	R5FSS0_INTRROUTER0_OUTL_63
R5FSS0_CORE1_INTR_IN_320	320	R5FSS0_INTRROUTER0_OUTL_64
R5FSS0_CORE1_INTR_IN_321	321	R5FSS0_INTRROUTER0_OUTL_65
R5FSS0_CORE1_INTR_IN_322	322	R5FSS0_INTRROUTER0_OUTL_66
R5FSS0_CORE1_INTR_IN_323	323	R5FSS0_INTRROUTER0_OUTL_67
R5FSS0_CORE1_INTR_IN_324	324	R5FSS0_INTRROUTER0_OUTL_68
R5FSS0_CORE1_INTR_IN_325	325	R5FSS0_INTRROUTER0_OUTL_69
R5FSS0_CORE1_INTR_IN_326	326	R5FSS0_INTRROUTER0_OUTL_70
R5FSS0_CORE1_INTR_IN_327	327	R5FSS0_INTRROUTER0_OUTL_71
R5FSS0_CORE1_INTR_IN_328	328	R5FSS0_INTRROUTER0_OUTL_72
R5FSS0_CORE1_INTR_IN_329	329	R5FSS0_INTRROUTER0_OUTL_73
R5FSS0_CORE1_INTR_IN_330	330	R5FSS0_INTRROUTER0_OUTL_74
R5FSS0_CORE1_INTR_IN_331	331	R5FSS0_INTRROUTER0_OUTL_75
R5FSS0_CORE1_INTR_IN_332	332	R5FSS0_INTRROUTER0_OUTL_76

**Table 9-48. R5FSS0\_CORE1 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS0_CORE1_INTR_IN_333	333	R5FSS0_INTRROUTER0_OUTL_77
R5FSS0_CORE1_INTR_IN_334	334	R5FSS0_INTRROUTER0_OUTL_78
R5FSS0_CORE1_INTR_IN_335	335	R5FSS0_INTRROUTER0_OUTL_79
R5FSS0_CORE1_INTR_IN_336	336	R5FSS0_INTRROUTER0_OUTL_80
R5FSS0_CORE1_INTR_IN_337	337	R5FSS0_INTRROUTER0_OUTL_81
R5FSS0_CORE1_INTR_IN_338	338	R5FSS0_INTRROUTER0_OUTL_82
R5FSS0_CORE1_INTR_IN_339	339	R5FSS0_INTRROUTER0_OUTL_83
R5FSS0_CORE1_INTR_IN_340	340	R5FSS0_INTRROUTER0_OUTL_84
R5FSS0_CORE1_INTR_IN_341	341	R5FSS0_INTRROUTER0_OUTL_85
R5FSS0_CORE1_INTR_IN_342	342	R5FSS0_INTRROUTER0_OUTL_86
R5FSS0_CORE1_INTR_IN_343	343	R5FSS0_INTRROUTER0_OUTL_87
R5FSS0_CORE1_INTR_IN_344	344	R5FSS0_INTRROUTER0_OUTL_88
R5FSS0_CORE1_INTR_IN_345	345	R5FSS0_INTRROUTER0_OUTL_89
R5FSS0_CORE1_INTR_IN_346	346	R5FSS0_INTRROUTER0_OUTL_90
R5FSS0_CORE1_INTR_IN_347	347	R5FSS0_INTRROUTER0_OUTL_91
R5FSS0_CORE1_INTR_IN_348	348	R5FSS0_INTRROUTER0_OUTL_92
R5FSS0_CORE1_INTR_IN_349	349	R5FSS0_INTRROUTER0_OUTL_93
R5FSS0_CORE1_INTR_IN_350	350	R5FSS0_INTRROUTER0_OUTL_94
R5FSS0_CORE1_INTR_IN_351	351	R5FSS0_INTRROUTER0_OUTL_95
R5FSS0_CORE1_INTR_IN_352	352	R5FSS0_INTRROUTER0_OUTL_96
R5FSS0_CORE1_INTR_IN_353	353	R5FSS0_INTRROUTER0_OUTL_97
R5FSS0_CORE1_INTR_IN_354	354	R5FSS0_INTRROUTER0_OUTL_98
R5FSS0_CORE1_INTR_IN_355	355	R5FSS0_INTRROUTER0_OUTL_99
R5FSS0_CORE1_INTR_IN_356	356	R5FSS0_INTRROUTER0_OUTL_100
R5FSS0_CORE1_INTR_IN_357	357	R5FSS0_INTRROUTER0_OUTL_101
R5FSS0_CORE1_INTR_IN_358	358	R5FSS0_INTRROUTER0_OUTL_102
R5FSS0_CORE1_INTR_IN_359	359	R5FSS0_INTRROUTER0_OUTL_103
R5FSS0_CORE1_INTR_IN_360	360	R5FSS0_INTRROUTER0_OUTL_104
R5FSS0_CORE1_INTR_IN_361	361	R5FSS0_INTRROUTER0_OUTL_105
R5FSS0_CORE1_INTR_IN_362	362	R5FSS0_INTRROUTER0_OUTL_106
R5FSS0_CORE1_INTR_IN_363	363	R5FSS0_INTRROUTER0_OUTL_107
R5FSS0_CORE1_INTR_IN_364	364	R5FSS0_INTRROUTER0_OUTL_108
R5FSS0_CORE1_INTR_IN_365	365	R5FSS0_INTRROUTER0_OUTL_109
R5FSS0_CORE1_INTR_IN_366	366	R5FSS0_INTRROUTER0_OUTL_110
R5FSS0_CORE1_INTR_IN_367	367	R5FSS0_INTRROUTER0_OUTL_111
R5FSS0_CORE1_INTR_IN_368	368	R5FSS0_INTRROUTER0_OUTL_112
R5FSS0_CORE1_INTR_IN_369	369	R5FSS0_INTRROUTER0_OUTL_113
R5FSS0_CORE1_INTR_IN_370	370	R5FSS0_INTRROUTER0_OUTL_114
R5FSS0_CORE1_INTR_IN_371	371	R5FSS0_INTRROUTER0_OUTL_115
R5FSS0_CORE1_INTR_IN_372	372	R5FSS0_INTRROUTER0_OUTL_116
R5FSS0_CORE1_INTR_IN_373	373	R5FSS0_INTRROUTER0_OUTL_117
R5FSS0_CORE1_INTR_IN_374	374	R5FSS0_INTRROUTER0_OUTL_118
R5FSS0_CORE1_INTR_IN_375	375	R5FSS0_INTRROUTER0_OUTL_119
R5FSS0_CORE1_INTR_IN_376	376	R5FSS0_INTRROUTER0_OUTL_120
R5FSS0_CORE1_INTR_IN_377	377	R5FSS0_INTRROUTER0_OUTL_121

**Table 9-48. R5FSS0\_CORE1 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS0_CORE1_INTR_IN_378	378	R5FSS0_INTROUTER0_OUTL_122
R5FSS0_CORE1_INTR_IN_379	379	R5FSS0_INTROUTER0_OUTL_123
R5FSS0_CORE1_INTR_IN_380	380	R5FSS0_INTROUTER0_OUTL_124
R5FSS0_CORE1_INTR_IN_381	381	R5FSS0_INTROUTER0_OUTL_125
R5FSS0_CORE1_INTR_IN_382	382	R5FSS0_INTROUTER0_OUTL_126
R5FSS0_CORE1_INTR_IN_383	383	R5FSS0_INTROUTER0_OUTL_127
R5FSS0_CORE1_INTR_IN_384	384	R5FSS0_INTROUTER0_OUTL_128
R5FSS0_CORE1_INTR_IN_385	385	R5FSS0_INTROUTER0_OUTL_129
R5FSS0_CORE1_INTR_IN_386	386	R5FSS0_INTROUTER0_OUTL_130
R5FSS0_CORE1_INTR_IN_387	387	R5FSS0_INTROUTER0_OUTL_131
R5FSS0_CORE1_INTR_IN_388	388	R5FSS0_INTROUTER0_OUTL_132
R5FSS0_CORE1_INTR_IN_389	389	R5FSS0_INTROUTER0_OUTL_133
R5FSS0_CORE1_INTR_IN_390	390	R5FSS0_INTROUTER0_OUTL_134
R5FSS0_CORE1_INTR_IN_391	391	R5FSS0_INTROUTER0_OUTL_135
R5FSS0_CORE1_INTR_IN_392	392	R5FSS0_INTROUTER0_OUTL_136
R5FSS0_CORE1_INTR_IN_393	393	R5FSS0_INTROUTER0_OUTL_137
R5FSS0_CORE1_INTR_IN_394	394	R5FSS0_INTROUTER0_OUTL_138
R5FSS0_CORE1_INTR_IN_395	395	R5FSS0_INTROUTER0_OUTL_139
R5FSS0_CORE1_INTR_IN_396	396	R5FSS0_INTROUTER0_OUTL_140
R5FSS0_CORE1_INTR_IN_397	397	R5FSS0_INTROUTER0_OUTL_141
R5FSS0_CORE1_INTR_IN_398	398	R5FSS0_INTROUTER0_OUTL_142
R5FSS0_CORE1_INTR_IN_399	399	R5FSS0_INTROUTER0_OUTL_143
R5FSS0_CORE1_INTR_IN_400	400	R5FSS0_INTROUTER0_OUTL_144
R5FSS0_CORE1_INTR_IN_401	401	R5FSS0_INTROUTER0_OUTL_145
R5FSS0_CORE1_INTR_IN_402	402	R5FSS0_INTROUTER0_OUTL_146
R5FSS0_CORE1_INTR_IN_403	403	R5FSS0_INTROUTER0_OUTL_147
R5FSS0_CORE1_INTR_IN_404	404	R5FSS0_INTROUTER0_OUTL_148
R5FSS0_CORE1_INTR_IN_405	405	R5FSS0_INTROUTER0_OUTL_149
R5FSS0_CORE1_INTR_IN_406	406	R5FSS0_INTROUTER0_OUTL_150
R5FSS0_CORE1_INTR_IN_407	407	R5FSS0_INTROUTER0_OUTL_151
R5FSS0_CORE1_INTR_IN_408	408	R5FSS0_INTROUTER0_OUTL_152
R5FSS0_CORE1_INTR_IN_409	409	R5FSS0_INTROUTER0_OUTL_153
R5FSS0_CORE1_INTR_IN_410	410	R5FSS0_INTROUTER0_OUTL_154
R5FSS0_CORE1_INTR_IN_411	411	R5FSS0_INTROUTER0_OUTL_155
R5FSS0_CORE1_INTR_IN_412	412	R5FSS0_INTROUTER0_OUTL_156
R5FSS0_CORE1_INTR_IN_413	413	R5FSS0_INTROUTER0_OUTL_157
R5FSS0_CORE1_INTR_IN_414	414	R5FSS0_INTROUTER0_OUTL_158
R5FSS0_CORE1_INTR_IN_415	415	R5FSS0_INTROUTER0_OUTL_159
R5FSS0_CORE1_INTR_IN_416	416	R5FSS0_INTROUTER0_OUTL_160
R5FSS0_CORE1_INTR_IN_417	417	R5FSS0_INTROUTER0_OUTL_161
R5FSS0_CORE1_INTR_IN_418	418	R5FSS0_INTROUTER0_OUTL_162
R5FSS0_CORE1_INTR_IN_419	419	R5FSS0_INTROUTER0_OUTL_163
R5FSS0_CORE1_INTR_IN_420	420	R5FSS0_INTROUTER0_OUTL_164
R5FSS0_CORE1_INTR_IN_421	421	R5FSS0_INTROUTER0_OUTL_165
R5FSS0_CORE1_INTR_IN_422	422	R5FSS0_INTROUTER0_OUTL_166

**Table 9-48. R5FSS0\_CORE1 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS0_CORE1_INTR_IN_423	423	R5FSS0_INTROUTER0_OUTL_167
R5FSS0_CORE1_INTR_IN_424	424	R5FSS0_INTROUTER0_OUTL_168
R5FSS0_CORE1_INTR_IN_425	425	R5FSS0_INTROUTER0_OUTL_169
R5FSS0_CORE1_INTR_IN_426	426	R5FSS0_INTROUTER0_OUTL_170
R5FSS0_CORE1_INTR_IN_427	427	R5FSS0_INTROUTER0_OUTL_171
R5FSS0_CORE1_INTR_IN_428	428	R5FSS0_INTROUTER0_OUTL_172
R5FSS0_CORE1_INTR_IN_429	429	R5FSS0_INTROUTER0_OUTL_173
R5FSS0_CORE1_INTR_IN_430	430	R5FSS0_INTROUTER0_OUTL_174
R5FSS0_CORE1_INTR_IN_431	431	R5FSS0_INTROUTER0_OUTL_175
R5FSS0_CORE1_INTR_IN_432	432	R5FSS0_INTROUTER0_OUTL_176
R5FSS0_CORE1_INTR_IN_433	433	R5FSS0_INTROUTER0_OUTL_177
R5FSS0_CORE1_INTR_IN_434	434	R5FSS0_INTROUTER0_OUTL_178
R5FSS0_CORE1_INTR_IN_435	435	R5FSS0_INTROUTER0_OUTL_179
R5FSS0_CORE1_INTR_IN_436	436	R5FSS0_INTROUTER0_OUTL_180
R5FSS0_CORE1_INTR_IN_437	437	R5FSS0_INTROUTER0_OUTL_181
R5FSS0_CORE1_INTR_IN_438	438	R5FSS0_INTROUTER0_OUTL_182
R5FSS0_CORE1_INTR_IN_439	439	R5FSS0_INTROUTER0_OUTL_183
R5FSS0_CORE1_INTR_IN_440	440	R5FSS0_INTROUTER0_OUTL_184
R5FSS0_CORE1_INTR_IN_441	441	R5FSS0_INTROUTER0_OUTL_185
R5FSS0_CORE1_INTR_IN_442	442	R5FSS0_INTROUTER0_OUTL_186
R5FSS0_CORE1_INTR_IN_443	443	R5FSS0_INTROUTER0_OUTL_187
R5FSS0_CORE1_INTR_IN_444	444	R5FSS0_INTROUTER0_OUTL_188
R5FSS0_CORE1_INTR_IN_445	445	R5FSS0_INTROUTER0_OUTL_189
R5FSS0_CORE1_INTR_IN_446	446	R5FSS0_INTROUTER0_OUTL_190
R5FSS0_CORE1_INTR_IN_447	447	R5FSS0_INTROUTER0_OUTL_191
R5FSS0_CORE1_INTR_IN_448	448	R5FSS0_INTROUTER0_OUTL_192
R5FSS0_CORE1_INTR_IN_449	449	R5FSS0_INTROUTER0_OUTL_193
R5FSS0_CORE1_INTR_IN_450	450	R5FSS0_INTROUTER0_OUTL_194
R5FSS0_CORE1_INTR_IN_451	451	R5FSS0_INTROUTER0_OUTL_195
R5FSS0_CORE1_INTR_IN_452	452	R5FSS0_INTROUTER0_OUTL_196
R5FSS0_CORE1_INTR_IN_453	453	R5FSS0_INTROUTER0_OUTL_197
R5FSS0_CORE1_INTR_IN_454	454	R5FSS0_INTROUTER0_OUTL_198
R5FSS0_CORE1_INTR_IN_455	455	R5FSS0_INTROUTER0_OUTL_199
R5FSS0_CORE1_INTR_IN_456	456	R5FSS0_INTROUTER0_OUTL_200
R5FSS0_CORE1_INTR_IN_457	457	R5FSS0_INTROUTER0_OUTL_201
R5FSS0_CORE1_INTR_IN_458	458	R5FSS0_INTROUTER0_OUTL_202
R5FSS0_CORE1_INTR_IN_459	459	R5FSS0_INTROUTER0_OUTL_203
R5FSS0_CORE1_INTR_IN_460	460	R5FSS0_INTROUTER0_OUTL_204
R5FSS0_CORE1_INTR_IN_461	461	R5FSS0_INTROUTER0_OUTL_205
R5FSS0_CORE1_INTR_IN_462	462	R5FSS0_INTROUTER0_OUTL_206
R5FSS0_CORE1_INTR_IN_463	463	R5FSS0_INTROUTER0_OUTL_207
R5FSS0_CORE1_INTR_IN_464	464	R5FSS0_INTROUTER0_OUTL_208
R5FSS0_CORE1_INTR_IN_465	465	R5FSS0_INTROUTER0_OUTL_209
R5FSS0_CORE1_INTR_IN_466	466	R5FSS0_INTROUTER0_OUTL_210
R5FSS0_CORE1_INTR_IN_467	467	R5FSS0_INTROUTER0_OUTL_211

**Table 9-48. R5FSS0\_CORE1 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS0_CORE1_INTR_IN_468	468	R5FSS0_INTROUTER0_OUTL_212
R5FSS0_CORE1_INTR_IN_469	469	R5FSS0_INTROUTER0_OUTL_213
R5FSS0_CORE1_INTR_IN_470	470	R5FSS0_INTROUTER0_OUTL_214
R5FSS0_CORE1_INTR_IN_471	471	R5FSS0_INTROUTER0_OUTL_215
R5FSS0_CORE1_INTR_IN_472	472	R5FSS0_INTROUTER0_OUTL_216
R5FSS0_CORE1_INTR_IN_473	473	R5FSS0_INTROUTER0_OUTL_217
R5FSS0_CORE1_INTR_IN_474	474	R5FSS0_INTROUTER0_OUTL_218
R5FSS0_CORE1_INTR_IN_475	475	R5FSS0_INTROUTER0_OUTL_219
R5FSS0_CORE1_INTR_IN_476	476	R5FSS0_INTROUTER0_OUTL_220
R5FSS0_CORE1_INTR_IN_477	477	R5FSS0_INTROUTER0_OUTL_221
R5FSS0_CORE1_INTR_IN_478	478	R5FSS0_INTROUTER0_OUTL_222
R5FSS0_CORE1_INTR_IN_479	479	R5FSS0_INTROUTER0_OUTL_223
R5FSS0_CORE1_INTR_IN_480	480	R5FSS0_INTROUTER0_OUTL_224
R5FSS0_CORE1_INTR_IN_481	481	R5FSS0_INTROUTER0_OUTL_225
R5FSS0_CORE1_INTR_IN_482	482	R5FSS0_INTROUTER0_OUTL_226
R5FSS0_CORE1_INTR_IN_483	483	R5FSS0_INTROUTER0_OUTL_227
R5FSS0_CORE1_INTR_IN_484	484	R5FSS0_INTROUTER0_OUTL_228
R5FSS0_CORE1_INTR_IN_485	485	R5FSS0_INTROUTER0_OUTL_229
R5FSS0_CORE1_INTR_IN_486	486	R5FSS0_INTROUTER0_OUTL_230
R5FSS0_CORE1_INTR_IN_487	487	R5FSS0_INTROUTER0_OUTL_231
R5FSS0_CORE1_INTR_IN_488	488	R5FSS0_INTROUTER0_OUTL_232
R5FSS0_CORE1_INTR_IN_489	489	R5FSS0_INTROUTER0_OUTL_233
R5FSS0_CORE1_INTR_IN_490	490	R5FSS0_INTROUTER0_OUTL_234
R5FSS0_CORE1_INTR_IN_491	491	R5FSS0_INTROUTER0_OUTL_235
R5FSS0_CORE1_INTR_IN_492	492	R5FSS0_INTROUTER0_OUTL_236
R5FSS0_CORE1_INTR_IN_493	493	R5FSS0_INTROUTER0_OUTL_237
R5FSS0_CORE1_INTR_IN_494	494	R5FSS0_INTROUTER0_OUTL_238
R5FSS0_CORE1_INTR_IN_495	495	R5FSS0_INTROUTER0_OUTL_239
R5FSS0_CORE1_INTR_IN_496	496	R5FSS0_INTROUTER0_OUTL_240
R5FSS0_CORE1_INTR_IN_497	497	R5FSS0_INTROUTER0_OUTL_241
R5FSS0_CORE1_INTR_IN_498	498	R5FSS0_INTROUTER0_OUTL_242
R5FSS0_CORE1_INTR_IN_499	499	R5FSS0_INTROUTER0_OUTL_243
R5FSS0_CORE1_INTR_IN_500	500	R5FSS0_INTROUTER0_OUTL_244
R5FSS0_CORE1_INTR_IN_501	501	R5FSS0_INTROUTER0_OUTL_245
R5FSS0_CORE1_INTR_IN_502	502	R5FSS0_INTROUTER0_OUTL_246
R5FSS0_CORE1_INTR_IN_503	503	R5FSS0_INTROUTER0_OUTL_247
R5FSS0_CORE1_INTR_IN_504	504	R5FSS0_INTROUTER0_OUTL_248
R5FSS0_CORE1_INTR_IN_505	505	R5FSS0_INTROUTER0_OUTL_249
R5FSS0_CORE1_INTR_IN_506	506	R5FSS0_INTROUTER0_OUTL_250
R5FSS0_CORE1_INTR_IN_507	507	R5FSS0_INTROUTER0_OUTL_251
R5FSS0_CORE1_INTR_IN_508	508	R5FSS0_INTROUTER0_OUTL_252
R5FSS0_CORE1_INTR_IN_509	509	R5FSS0_INTROUTER0_OUTL_253
R5FSS0_CORE1_INTR_IN_510	510	R5FSS0_INTROUTER0_OUTL_254
R5FSS0_CORE1_INTR_IN_511	511	R5FSS0_INTROUTER0_OUTL_255

#### 9.4.3.4 R5FSS1\_CORE0 Interrupt Map

Table 9-47 shows the mapping of events to the R5FSS1\_CORE0. The R5FSS1 VIM supports both R5FSS1 cores.

The R5FSS1\_CORE0 events are the events used by both processors when operating in lockstep mode.

**Table 9-49. R5FSS1\_CORE0 Interrupt Map**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS1_CORE0_INTR_IN_0	0	CTRL_MMR0_IPC_SET18_IPC_SET_IPCFG_0
R5FSS1_CORE0_INTR_IN_1	1	CTRL_MMR0_IPC_SET19_IPC_SET_IPCFG_0
R5FSS1_CORE0_INTR_IN_2	2	RTI30_INTR_WWD_0
R5FSS1_CORE0_INTR_IN_3	3	RTI31_INTR_WWD_0
R5FSS1_CORE0_INTR_IN_4	4	R5FSS1_COMMRX_LEVEL_0_0
R5FSS1_CORE0_INTR_IN_5	5	R5FSS1_COMMTX_LEVEL_0_0
R5FSS1_CORE0_INTR_IN_6	6	R5FSS1_CORE0_VALFIQ_0
R5FSS1_CORE0_INTR_IN_7	7	R5FSS1_CORE0_VALIRQ_0
R5FSS1_CORE0_INTR_IN_8	8	R5FSS1_CORE0_CTI_0
R5FSS1_CORE0_INTR_IN_9	9	R5FSS1_CORE1_CTI_0
R5FSS1_CORE0_INTR_IN_10	10	ESM0_ESM_INT_LOW_LVL_0
R5FSS1_CORE0_INTR_IN_11	11	ESM0_ESM_INT_HI_LVL_0
R5FSS1_CORE0_INTR_IN_12	12	ESM0_ESM_INT_CFG_LVL_0
R5FSS1_CORE0_INTR_IN_13	13	GLUELOGIC_EXT_INTN_GLUE_EXT_INT_LVL_0
R5FSS1_CORE0_INTR_IN_16	16	R5FSS1_CORE0_EXP_INTR_0
R5FSS1_CORE0_INTR_IN_17	17	R5FSS1_CORE1_EXP_INTR_0
R5FSS1_CORE0_INTR_IN_32	32	DMPAC0_INTD_0_SYSTEM_INTR_LEVEL_0
R5FSS1_CORE0_INTR_IN_33	33	DMPAC0_INTD_0_SYSTEM_INTR_LEVEL_1
R5FSS1_CORE0_INTR_IN_34	34	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_0
R5FSS1_CORE0_INTR_IN_35	35	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_1
R5FSS1_CORE0_INTR_IN_36	36	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_2
R5FSS1_CORE0_INTR_IN_37	37	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_3
R5FSS1_CORE0_INTR_IN_38	38	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_4
R5FSS1_CORE0_INTR_IN_39	39	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_5
R5FSS1_CORE0_INTR_IN_40	40	GPU0_MISC_0_IRQ_0
R5FSS1_CORE0_INTR_IN_48	48	GLUELOGIC_GPU_GPIO_REQACK_GLUE_GPU_GPIO_REQINT_LVL_0
R5FSS1_CORE0_INTR_IN_49	49	GLUELOGIC_GPU_GPIO_REQACK_GLUE_GPU_GPIO_ACKINT_LVL_0
R5FSS1_CORE0_INTR_IN_52	52	DSS0_DSS_INST0_DISPC_FUNC_IRQ_PROC0_0
R5FSS1_CORE0_INTR_IN_53	53	DSS0_DSS_INST0_DISPC_FUNC_IRQ_PROC1_0
R5FSS1_CORE0_INTR_IN_54	54	DSS0_DSS_INST0_DISPC_SAFETY_ERROR_IRQ_PROC0_0
R5FSS1_CORE0_INTR_IN_55	55	DSS0_DSS_INST0_DISPC_SAFETY_ERROR_IRQ_PROC1_0
R5FSS1_CORE0_INTR_IN_56	56	DSS0_DSS_INST0_DISPC_SECURE_IRQ_PROC0_0
R5FSS1_CORE0_INTR_IN_57	57	DSS0_DSS_INST0_DISPC_SECURE_IRQ_PROC1_0
R5FSS1_CORE0_INTR_IN_64	64	DSS_EDP0_INTR_0
R5FSS1_CORE0_INTR_IN_65	65	DSS_EDP0_INTR_1
R5FSS1_CORE0_INTR_IN_66	66	DSS_EDP0_INTR_2
R5FSS1_CORE0_INTR_IN_67	67	DSS_EDP0_INTR_3
R5FSS1_CORE0_INTR_IN_76	76	DSS_DSI0_DSI_0_FUNC_INTR_0
R5FSS1_CORE0_INTR_IN_78	78	CSI_RX_IF0_CSI_ERR_IRQ_0
R5FSS1_CORE0_INTR_IN_79	79	CSI_RX_IF0_CSI_IRQ_0



**Table 9-49. R5FSS1\_CORE0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS1_CORE0_INTR_IN_80	80	CSI_RX_IF0_CSI_LEVEL_0
R5FSS1_CORE0_INTR_IN_81	81	CSI_RX_IF1_CSI_ERR_IRQ_0
R5FSS1_CORE0_INTR_IN_82	82	CSI_RX_IF1_CSI_IRQ_0
R5FSS1_CORE0_INTR_IN_83	83	CSI_RX_IF1_CSI_LEVEL_0
R5FSS1_CORE0_INTR_IN_87	87	DECODER0_IRQ_0
R5FSS1_CORE0_INTR_IN_88	88	ENCODER0_IRQ_0
R5FSS1_CORE0_INTR_IN_96	96	CPSW0_STAT_PEND_0
R5FSS1_CORE0_INTR_IN_97	97	CPSW0_MDIO_PEND_0
R5FSS1_CORE0_INTR_IN_98	98	CPSW0_EVTN_PEND_0
R5FSS1_CORE0_INTR_IN_99	99	MMCS00_EMMCSS_INTR_0
R5FSS1_CORE0_INTR_IN_104	104	EHRPWM0_EPWM_ETINT_0
R5FSS1_CORE0_INTR_IN_105	105	EHRPWM1_EPWM_ETINT_0
R5FSS1_CORE0_INTR_IN_106	106	EHRPWM2_EPWM_ETINT_0
R5FSS1_CORE0_INTR_IN_107	107	EHRPWM3_EPWM_ETINT_0
R5FSS1_CORE0_INTR_IN_108	108	EHRPWM4_EPWM_ETINT_0
R5FSS1_CORE0_INTR_IN_109	109	EHRPWM5_EPWM_ETINT_0
R5FSS1_CORE0_INTR_IN_110	110	EHRPWM0_EPWM_TRIPZINT_0
R5FSS1_CORE0_INTR_IN_111	111	EHRPWM1_EPWM_TRIPZINT_0
R5FSS1_CORE0_INTR_IN_112	112	EHRPWM2_EPWM_TRIPZINT_0
R5FSS1_CORE0_INTR_IN_113	113	EHRPWM3_EPWM_TRIPZINT_0
R5FSS1_CORE0_INTR_IN_114	114	EHRPWM4_EPWM_TRIPZINT_0
R5FSS1_CORE0_INTR_IN_115	115	EHRPWM5_EPWM_TRIPZINT_0
R5FSS1_CORE0_INTR_IN_116	116	EQEP0_EQEP_INT_0
R5FSS1_CORE0_INTR_IN_117	117	EQEP1_EQEP_INT_0
R5FSS1_CORE0_INTR_IN_118	118	EQEP2_EQEP_INT_0
R5FSS1_CORE0_INTR_IN_120	120	MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS1_CORE0_INTR_IN_121	121	MCAN0_MCANSS_MCAN_LVL_INT_0
R5FSS1_CORE0_INTR_IN_122	122	MCAN0_MCANSS_MCAN_LVL_INT_1
R5FSS1_CORE0_INTR_IN_123	123	MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS1_CORE0_INTR_IN_124	124	MCAN1_MCANSS_MCAN_LVL_INT_0
R5FSS1_CORE0_INTR_IN_125	125	MCAN1_MCANSS_MCAN_LVL_INT_1
R5FSS1_CORE0_INTR_IN_126	126	MCAN2_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS1_CORE0_INTR_IN_127	127	MCAN2_MCANSS_MCAN_LVL_INT_0
R5FSS1_CORE0_INTR_IN_128	128	MCAN2_MCANSS_MCAN_LVL_INT_1
R5FSS1_CORE0_INTR_IN_129	129	MCAN3_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS1_CORE0_INTR_IN_130	130	MCAN3_MCANSS_MCAN_LVL_INT_0
R5FSS1_CORE0_INTR_IN_131	131	MCAN3_MCANSS_MCAN_LVL_INT_1
R5FSS1_CORE0_INTR_IN_132	132	MCASP0_REC_INTR_PEND_0
R5FSS1_CORE0_INTR_IN_133	133	MCASP0_XMIT_INTR_PEND_0
R5FSS1_CORE0_INTR_IN_134	134	MCASP1_REC_INTR_PEND_0
R5FSS1_CORE0_INTR_IN_135	135	MCASP1_XMIT_INTR_PEND_0
R5FSS1_CORE0_INTR_IN_136	136	PCIE0_PCIE_LEGACY_PULSE_0
R5FSS1_CORE0_INTR_IN_137	137	PCIE0_PCIE_DOWNSTREAM_PULSE_0
R5FSS1_CORE0_INTR_IN_138	138	PCIE0_PCIE_FLR_PULSE_0
R5FSS1_CORE0_INTR_IN_139	139	PCIE0_PCIE_PHY_LEVEL_0



**Table 9-49. R5FSS1\_CORE0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS1_CORE0_INTR_IN_140	140	PCIE0_PCIE_LOCAL_LEVEL_0
R5FSS1_CORE0_INTR_IN_141	141	PCIE0_PCIE_ERROR_PULSE_0
R5FSS1_CORE0_INTR_IN_142	142	PCIE0_PCIE_LINK_STATE_PULSE_0
R5FSS1_CORE0_INTR_IN_143	143	PCIE0_PCIE_PWR_STATE_PULSE_0
R5FSS1_CORE0_INTR_IN_144	144	PCIE0_PCIE_PTM_VALID_PULSE_0
R5FSS1_CORE0_INTR_IN_145	145	PCIE0_PCIE_HOT_RESET_PULSE_0
R5FSS1_CORE0_INTR_IN_146	146	PCIE0_PCIE_CPTS_PEND_0
R5FSS1_CORE0_INTR_IN_150	150	I2C0_POINTRPEND_0
R5FSS1_CORE0_INTR_IN_151	151	I2C1_POINTRPEND_0
R5FSS1_CORE0_INTR_IN_152	152	MCSPi0_INTR_SPI_0
R5FSS1_CORE0_INTR_IN_153	153	MCSPi1_INTR_SPI_0
R5FSS1_CORE0_INTR_IN_154	154	MLB0_MLBSS_MLB_INT_0
R5FSS1_CORE0_INTR_IN_155	155	MLB0_MLBSS_MLB_AHB_INT_0
R5FSS1_CORE0_INTR_IN_156	156	MLB0_MLBSS_MLB_AHB_INT_1
R5FSS1_CORE0_INTR_IN_158	158	UART0_USART_IRQ_0
R5FSS1_CORE0_INTR_IN_159	159	UART1_USART_IRQ_0
R5FSS1_CORE0_INTR_IN_160	160	UART2_USART_IRQ_0
R5FSS1_CORE0_INTR_IN_168	168	TIMER12_INTR_PEND_0
R5FSS1_CORE0_INTR_IN_169	169	TIMER13_INTR_PEND_0
R5FSS1_CORE0_INTR_IN_170	170	TIMER14_INTR_PEND_0
R5FSS1_CORE0_INTR_IN_171	171	TIMER15_INTR_PEND_0
R5FSS1_CORE0_INTR_IN_172	172	TIMER16_INTR_PEND_0
R5FSS1_CORE0_INTR_IN_173	173	TIMER17_INTR_PEND_0
R5FSS1_CORE0_INTR_IN_174	174	TIMER18_INTR_PEND_0
R5FSS1_CORE0_INTR_IN_175	175	TIMER19_INTR_PEND_0
R5FSS1_CORE0_INTR_IN_176	176	GPIOMUX_INTRTR0_OUTP_16
R5FSS1_CORE0_INTR_IN_177	177	GPIOMUX_INTRTR0_OUTP_17
R5FSS1_CORE0_INTR_IN_178	178	GPIOMUX_INTRTR0_OUTP_18
R5FSS1_CORE0_INTR_IN_179	179	GPIOMUX_INTRTR0_OUTP_19
R5FSS1_CORE0_INTR_IN_180	180	GPIOMUX_INTRTR0_OUTP_20
R5FSS1_CORE0_INTR_IN_181	181	GPIOMUX_INTRTR0_OUTP_21
R5FSS1_CORE0_INTR_IN_182	182	GPIOMUX_INTRTR0_OUTP_22
R5FSS1_CORE0_INTR_IN_183	183	GPIOMUX_INTRTR0_OUTP_23
R5FSS1_CORE0_INTR_IN_184	184	GPIOMUX_INTRTR0_OUTP_24
R5FSS1_CORE0_INTR_IN_185	185	GPIOMUX_INTRTR0_OUTP_25
R5FSS1_CORE0_INTR_IN_186	186	GPIOMUX_INTRTR0_OUTP_26
R5FSS1_CORE0_INTR_IN_187	187	GPIOMUX_INTRTR0_OUTP_27
R5FSS1_CORE0_INTR_IN_188	188	GPIOMUX_INTRTR0_OUTP_28
R5FSS1_CORE0_INTR_IN_189	189	GPIOMUX_INTRTR0_OUTP_29
R5FSS1_CORE0_INTR_IN_190	190	GPIOMUX_INTRTR0_OUTP_30
R5FSS1_CORE0_INTR_IN_191	191	GPIOMUX_INTRTR0_OUTP_31
R5FSS1_CORE0_INTR_IN_192	192	MCAN4_MCANSS_MCAN_LVL_INT_0
R5FSS1_CORE0_INTR_IN_193	193	MCAN4_MCANSS_MCAN_LVL_INT_1
R5FSS1_CORE0_INTR_IN_194	194	MCAN4_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS1_CORE0_INTR_IN_195	195	MCAN5_MCANSS_MCAN_LVL_INT_0

**Table 9-49. R5FSS1\_CORE0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS1_CORE0_INTR_IN_196	196	MCAN5_MCANSS_MCAN_LVL_INT_1
R5FSS1_CORE0_INTR_IN_197	197	MCAN5_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS1_CORE0_INTR_IN_198	198	MCAN6_MCANSS_MCAN_LVL_INT_0
R5FSS1_CORE0_INTR_IN_199	199	MCAN6_MCANSS_MCAN_LVL_INT_1
R5FSS1_CORE0_INTR_IN_200	200	MCAN6_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS1_CORE0_INTR_IN_201	201	MCAN7_MCANSS_MCAN_LVL_INT_0
R5FSS1_CORE0_INTR_IN_202	202	MCAN7_MCANSS_MCAN_LVL_INT_1
R5FSS1_CORE0_INTR_IN_203	203	MCAN7_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS1_CORE0_INTR_IN_204	204	MCAN8_MCANSS_MCAN_LVL_INT_0
R5FSS1_CORE0_INTR_IN_205	205	MCAN8_MCANSS_MCAN_LVL_INT_1
R5FSS1_CORE0_INTR_IN_206	206	MCAN8_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS1_CORE0_INTR_IN_207	207	MCAN9_MCANSS_MCAN_LVL_INT_0
R5FSS1_CORE0_INTR_IN_208	208	MCAN9_MCANSS_MCAN_LVL_INT_1
R5FSS1_CORE0_INTR_IN_209	209	MCAN9_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS1_CORE0_INTR_IN_210	210	MCAN10_MCANSS_MCAN_LVL_INT_0
R5FSS1_CORE0_INTR_IN_211	211	MCAN10_MCANSS_MCAN_LVL_INT_1
R5FSS1_CORE0_INTR_IN_212	212	MCAN10_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS1_CORE0_INTR_IN_213	213	MCAN11_MCANSS_MCAN_LVL_INT_0
R5FSS1_CORE0_INTR_IN_214	214	MCAN11_MCANSS_MCAN_LVL_INT_1
R5FSS1_CORE0_INTR_IN_215	215	MCAN11_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS1_CORE0_INTR_IN_216	216	MCAN12_MCANSS_MCAN_LVL_INT_0
R5FSS1_CORE0_INTR_IN_217	217	MCAN12_MCANSS_MCAN_LVL_INT_1
R5FSS1_CORE0_INTR_IN_218	218	MCAN12_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS1_CORE0_INTR_IN_219	219	MCAN13_MCANSS_MCAN_LVL_INT_0
R5FSS1_CORE0_INTR_IN_220	220	MCAN13_MCANSS_MCAN_LVL_INT_1
R5FSS1_CORE0_INTR_IN_221	221	MCAN13_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS1_CORE0_INTR_IN_224	224	NAVSS0_INTR_ROUTER_0_OUTL_INTR_256
R5FSS1_CORE0_INTR_IN_225	225	NAVSS0_INTR_ROUTER_0_OUTL_INTR_257
R5FSS1_CORE0_INTR_IN_226	226	NAVSS0_INTR_ROUTER_0_OUTL_INTR_258
R5FSS1_CORE0_INTR_IN_227	227	NAVSS0_INTR_ROUTER_0_OUTL_INTR_259
R5FSS1_CORE0_INTR_IN_228	228	NAVSS0_INTR_ROUTER_0_OUTL_INTR_260
R5FSS1_CORE0_INTR_IN_229	229	NAVSS0_INTR_ROUTER_0_OUTL_INTR_261
R5FSS1_CORE0_INTR_IN_230	230	NAVSS0_INTR_ROUTER_0_OUTL_INTR_262
R5FSS1_CORE0_INTR_IN_231	231	NAVSS0_INTR_ROUTER_0_OUTL_INTR_263
R5FSS1_CORE0_INTR_IN_232	232	NAVSS0_INTR_ROUTER_0_OUTL_INTR_264
R5FSS1_CORE0_INTR_IN_233	233	NAVSS0_INTR_ROUTER_0_OUTL_INTR_265
R5FSS1_CORE0_INTR_IN_234	234	NAVSS0_INTR_ROUTER_0_OUTL_INTR_266
R5FSS1_CORE0_INTR_IN_235	235	NAVSS0_INTR_ROUTER_0_OUTL_INTR_267
R5FSS1_CORE0_INTR_IN_236	236	NAVSS0_INTR_ROUTER_0_OUTL_INTR_268
R5FSS1_CORE0_INTR_IN_237	237	NAVSS0_INTR_ROUTER_0_OUTL_INTR_269
R5FSS1_CORE0_INTR_IN_238	238	NAVSS0_INTR_ROUTER_0_OUTL_INTR_270
R5FSS1_CORE0_INTR_IN_239	239	NAVSS0_INTR_ROUTER_0_OUTL_INTR_271
R5FSS1_CORE0_INTR_IN_240	240	NAVSS0_INTR_ROUTER_0_OUTL_INTR_272
R5FSS1_CORE0_INTR_IN_241	241	NAVSS0_INTR_ROUTER_0_OUTL_INTR_273
R5FSS1_CORE0_INTR_IN_242	242	NAVSS0_INTR_ROUTER_0_OUTL_INTR_274

**Table 9-49. R5FSS1\_CORE0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS1_CORE0_INTR_IN_243	243	NAVSS0_INTR_ROUTER_0_OUTL_INTR_275
R5FSS1_CORE0_INTR_IN_244	244	NAVSS0_INTR_ROUTER_0_OUTL_INTR_276
R5FSS1_CORE0_INTR_IN_245	245	NAVSS0_INTR_ROUTER_0_OUTL_INTR_277
R5FSS1_CORE0_INTR_IN_246	246	NAVSS0_INTR_ROUTER_0_OUTL_INTR_278
R5FSS1_CORE0_INTR_IN_247	247	NAVSS0_INTR_ROUTER_0_OUTL_INTR_279
R5FSS1_CORE0_INTR_IN_248	248	NAVSS0_INTR_ROUTER_0_OUTL_INTR_280
R5FSS1_CORE0_INTR_IN_249	249	NAVSS0_INTR_ROUTER_0_OUTL_INTR_281
R5FSS1_CORE0_INTR_IN_250	250	NAVSS0_INTR_ROUTER_0_OUTL_INTR_282
R5FSS1_CORE0_INTR_IN_251	251	NAVSS0_INTR_ROUTER_0_OUTL_INTR_283
R5FSS1_CORE0_INTR_IN_252	252	NAVSS0_INTR_ROUTER_0_OUTL_INTR_284
R5FSS1_CORE0_INTR_IN_253	253	NAVSS0_INTR_ROUTER_0_OUTL_INTR_285
R5FSS1_CORE0_INTR_IN_254	254	NAVSS0_INTR_ROUTER_0_OUTL_INTR_286
R5FSS1_CORE0_INTR_IN_255	255	NAVSS0_INTR_ROUTER_0_OUTL_INTR_287
R5FSS1_CORE0_INTR_IN_256	256	R5FSS1_INTROUTER0_OUTL_0
R5FSS1_CORE0_INTR_IN_257	257	R5FSS1_INTROUTER0_OUTL_1
R5FSS1_CORE0_INTR_IN_258	258	R5FSS1_INTROUTER0_OUTL_2
R5FSS1_CORE0_INTR_IN_259	259	R5FSS1_INTROUTER0_OUTL_3
R5FSS1_CORE0_INTR_IN_260	260	R5FSS1_INTROUTER0_OUTL_4
R5FSS1_CORE0_INTR_IN_261	261	R5FSS1_INTROUTER0_OUTL_5
R5FSS1_CORE0_INTR_IN_262	262	R5FSS1_INTROUTER0_OUTL_6
R5FSS1_CORE0_INTR_IN_263	263	R5FSS1_INTROUTER0_OUTL_7
R5FSS1_CORE0_INTR_IN_264	264	R5FSS1_INTROUTER0_OUTL_8
R5FSS1_CORE0_INTR_IN_265	265	R5FSS1_INTROUTER0_OUTL_9
R5FSS1_CORE0_INTR_IN_266	266	R5FSS1_INTROUTER0_OUTL_10
R5FSS1_CORE0_INTR_IN_267	267	R5FSS1_INTROUTER0_OUTL_11
R5FSS1_CORE0_INTR_IN_268	268	R5FSS1_INTROUTER0_OUTL_12
R5FSS1_CORE0_INTR_IN_269	269	R5FSS1_INTROUTER0_OUTL_13
R5FSS1_CORE0_INTR_IN_270	270	R5FSS1_INTROUTER0_OUTL_14
R5FSS1_CORE0_INTR_IN_271	271	R5FSS1_INTROUTER0_OUTL_15
R5FSS1_CORE0_INTR_IN_272	272	R5FSS1_INTROUTER0_OUTL_16
R5FSS1_CORE0_INTR_IN_273	273	R5FSS1_INTROUTER0_OUTL_17
R5FSS1_CORE0_INTR_IN_274	274	R5FSS1_INTROUTER0_OUTL_18
R5FSS1_CORE0_INTR_IN_275	275	R5FSS1_INTROUTER0_OUTL_19
R5FSS1_CORE0_INTR_IN_276	276	R5FSS1_INTROUTER0_OUTL_20
R5FSS1_CORE0_INTR_IN_277	277	R5FSS1_INTROUTER0_OUTL_21
R5FSS1_CORE0_INTR_IN_278	278	R5FSS1_INTROUTER0_OUTL_22
R5FSS1_CORE0_INTR_IN_279	279	R5FSS1_INTROUTER0_OUTL_23
R5FSS1_CORE0_INTR_IN_280	280	R5FSS1_INTROUTER0_OUTL_24
R5FSS1_CORE0_INTR_IN_281	281	R5FSS1_INTROUTER0_OUTL_25
R5FSS1_CORE0_INTR_IN_282	282	R5FSS1_INTROUTER0_OUTL_26
R5FSS1_CORE0_INTR_IN_283	283	R5FSS1_INTROUTER0_OUTL_27
R5FSS1_CORE0_INTR_IN_284	284	R5FSS1_INTROUTER0_OUTL_28
R5FSS1_CORE0_INTR_IN_285	285	R5FSS1_INTROUTER0_OUTL_29
R5FSS1_CORE0_INTR_IN_286	286	R5FSS1_INTROUTER0_OUTL_30
R5FSS1_CORE0_INTR_IN_287	287	R5FSS1_INTROUTER0_OUTL_31

**Table 9-49. R5FSS1\_CORE0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS1_CORE0_INTR_IN_288	288	R5FSS1_INTROUTER0_OUTL_32
R5FSS1_CORE0_INTR_IN_289	289	R5FSS1_INTROUTER0_OUTL_33
R5FSS1_CORE0_INTR_IN_290	290	R5FSS1_INTROUTER0_OUTL_34
R5FSS1_CORE0_INTR_IN_291	291	R5FSS1_INTROUTER0_OUTL_35
R5FSS1_CORE0_INTR_IN_292	292	R5FSS1_INTROUTER0_OUTL_36
R5FSS1_CORE0_INTR_IN_293	293	R5FSS1_INTROUTER0_OUTL_37
R5FSS1_CORE0_INTR_IN_294	294	R5FSS1_INTROUTER0_OUTL_38
R5FSS1_CORE0_INTR_IN_295	295	R5FSS1_INTROUTER0_OUTL_39
R5FSS1_CORE0_INTR_IN_296	296	R5FSS1_INTROUTER0_OUTL_40
R5FSS1_CORE0_INTR_IN_297	297	R5FSS1_INTROUTER0_OUTL_41
R5FSS1_CORE0_INTR_IN_298	298	R5FSS1_INTROUTER0_OUTL_42
R5FSS1_CORE0_INTR_IN_299	299	R5FSS1_INTROUTER0_OUTL_43
R5FSS1_CORE0_INTR_IN_300	300	R5FSS1_INTROUTER0_OUTL_44
R5FSS1_CORE0_INTR_IN_301	301	R5FSS1_INTROUTER0_OUTL_45
R5FSS1_CORE0_INTR_IN_302	302	R5FSS1_INTROUTER0_OUTL_46
R5FSS1_CORE0_INTR_IN_303	303	R5FSS1_INTROUTER0_OUTL_47
R5FSS1_CORE0_INTR_IN_304	304	R5FSS1_INTROUTER0_OUTL_48
R5FSS1_CORE0_INTR_IN_305	305	R5FSS1_INTROUTER0_OUTL_49
R5FSS1_CORE0_INTR_IN_306	306	R5FSS1_INTROUTER0_OUTL_50
R5FSS1_CORE0_INTR_IN_307	307	R5FSS1_INTROUTER0_OUTL_51
R5FSS1_CORE0_INTR_IN_308	308	R5FSS1_INTROUTER0_OUTL_52
R5FSS1_CORE0_INTR_IN_309	309	R5FSS1_INTROUTER0_OUTL_53
R5FSS1_CORE0_INTR_IN_310	310	R5FSS1_INTROUTER0_OUTL_54
R5FSS1_CORE0_INTR_IN_311	311	R5FSS1_INTROUTER0_OUTL_55
R5FSS1_CORE0_INTR_IN_312	312	R5FSS1_INTROUTER0_OUTL_56
R5FSS1_CORE0_INTR_IN_313	313	R5FSS1_INTROUTER0_OUTL_57
R5FSS1_CORE0_INTR_IN_314	314	R5FSS1_INTROUTER0_OUTL_58
R5FSS1_CORE0_INTR_IN_315	315	R5FSS1_INTROUTER0_OUTL_59
R5FSS1_CORE0_INTR_IN_316	316	R5FSS1_INTROUTER0_OUTL_60
R5FSS1_CORE0_INTR_IN_317	317	R5FSS1_INTROUTER0_OUTL_61
R5FSS1_CORE0_INTR_IN_318	318	R5FSS1_INTROUTER0_OUTL_62
R5FSS1_CORE0_INTR_IN_319	319	R5FSS1_INTROUTER0_OUTL_63
R5FSS1_CORE0_INTR_IN_320	320	R5FSS1_INTROUTER0_OUTL_64
R5FSS1_CORE0_INTR_IN_321	321	R5FSS1_INTROUTER0_OUTL_65
R5FSS1_CORE0_INTR_IN_322	322	R5FSS1_INTROUTER0_OUTL_66
R5FSS1_CORE0_INTR_IN_323	323	R5FSS1_INTROUTER0_OUTL_67
R5FSS1_CORE0_INTR_IN_324	324	R5FSS1_INTROUTER0_OUTL_68
R5FSS1_CORE0_INTR_IN_325	325	R5FSS1_INTROUTER0_OUTL_69
R5FSS1_CORE0_INTR_IN_326	326	R5FSS1_INTROUTER0_OUTL_70
R5FSS1_CORE0_INTR_IN_327	327	R5FSS1_INTROUTER0_OUTL_71
R5FSS1_CORE0_INTR_IN_328	328	R5FSS1_INTROUTER0_OUTL_72
R5FSS1_CORE0_INTR_IN_329	329	R5FSS1_INTROUTER0_OUTL_73
R5FSS1_CORE0_INTR_IN_330	330	R5FSS1_INTROUTER0_OUTL_74
R5FSS1_CORE0_INTR_IN_331	331	R5FSS1_INTROUTER0_OUTL_75
R5FSS1_CORE0_INTR_IN_332	332	R5FSS1_INTROUTER0_OUTL_76

**Table 9-49. R5FSS1\_CORE0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS1_CORE0_INTR_IN_333	333	R5FSS1_INTROUTER0_OUTL_77
R5FSS1_CORE0_INTR_IN_334	334	R5FSS1_INTROUTER0_OUTL_78
R5FSS1_CORE0_INTR_IN_335	335	R5FSS1_INTROUTER0_OUTL_79
R5FSS1_CORE0_INTR_IN_336	336	R5FSS1_INTROUTER0_OUTL_80
R5FSS1_CORE0_INTR_IN_337	337	R5FSS1_INTROUTER0_OUTL_81
R5FSS1_CORE0_INTR_IN_338	338	R5FSS1_INTROUTER0_OUTL_82
R5FSS1_CORE0_INTR_IN_339	339	R5FSS1_INTROUTER0_OUTL_83
R5FSS1_CORE0_INTR_IN_340	340	R5FSS1_INTROUTER0_OUTL_84
R5FSS1_CORE0_INTR_IN_341	341	R5FSS1_INTROUTER0_OUTL_85
R5FSS1_CORE0_INTR_IN_342	342	R5FSS1_INTROUTER0_OUTL_86
R5FSS1_CORE0_INTR_IN_343	343	R5FSS1_INTROUTER0_OUTL_87
R5FSS1_CORE0_INTR_IN_344	344	R5FSS1_INTROUTER0_OUTL_88
R5FSS1_CORE0_INTR_IN_345	345	R5FSS1_INTROUTER0_OUTL_89
R5FSS1_CORE0_INTR_IN_346	346	R5FSS1_INTROUTER0_OUTL_90
R5FSS1_CORE0_INTR_IN_347	347	R5FSS1_INTROUTER0_OUTL_91
R5FSS1_CORE0_INTR_IN_348	348	R5FSS1_INTROUTER0_OUTL_92
R5FSS1_CORE0_INTR_IN_349	349	R5FSS1_INTROUTER0_OUTL_93
R5FSS1_CORE0_INTR_IN_350	350	R5FSS1_INTROUTER0_OUTL_94
R5FSS1_CORE0_INTR_IN_351	351	R5FSS1_INTROUTER0_OUTL_95
R5FSS1_CORE0_INTR_IN_352	352	R5FSS1_INTROUTER0_OUTL_96
R5FSS1_CORE0_INTR_IN_353	353	R5FSS1_INTROUTER0_OUTL_97
R5FSS1_CORE0_INTR_IN_354	354	R5FSS1_INTROUTER0_OUTL_98
R5FSS1_CORE0_INTR_IN_355	355	R5FSS1_INTROUTER0_OUTL_99
R5FSS1_CORE0_INTR_IN_356	356	R5FSS1_INTROUTER0_OUTL_100
R5FSS1_CORE0_INTR_IN_357	357	R5FSS1_INTROUTER0_OUTL_101
R5FSS1_CORE0_INTR_IN_358	358	R5FSS1_INTROUTER0_OUTL_102
R5FSS1_CORE0_INTR_IN_359	359	R5FSS1_INTROUTER0_OUTL_103
R5FSS1_CORE0_INTR_IN_360	360	R5FSS1_INTROUTER0_OUTL_104
R5FSS1_CORE0_INTR_IN_361	361	R5FSS1_INTROUTER0_OUTL_105
R5FSS1_CORE0_INTR_IN_362	362	R5FSS1_INTROUTER0_OUTL_106
R5FSS1_CORE0_INTR_IN_363	363	R5FSS1_INTROUTER0_OUTL_107
R5FSS1_CORE0_INTR_IN_364	364	R5FSS1_INTROUTER0_OUTL_108
R5FSS1_CORE0_INTR_IN_365	365	R5FSS1_INTROUTER0_OUTL_109
R5FSS1_CORE0_INTR_IN_366	366	R5FSS1_INTROUTER0_OUTL_110
R5FSS1_CORE0_INTR_IN_367	367	R5FSS1_INTROUTER0_OUTL_111
R5FSS1_CORE0_INTR_IN_368	368	R5FSS1_INTROUTER0_OUTL_112
R5FSS1_CORE0_INTR_IN_369	369	R5FSS1_INTROUTER0_OUTL_113
R5FSS1_CORE0_INTR_IN_370	370	R5FSS1_INTROUTER0_OUTL_114
R5FSS1_CORE0_INTR_IN_371	371	R5FSS1_INTROUTER0_OUTL_115
R5FSS1_CORE0_INTR_IN_372	372	R5FSS1_INTROUTER0_OUTL_116
R5FSS1_CORE0_INTR_IN_373	373	R5FSS1_INTROUTER0_OUTL_117
R5FSS1_CORE0_INTR_IN_374	374	R5FSS1_INTROUTER0_OUTL_118
R5FSS1_CORE0_INTR_IN_375	375	R5FSS1_INTROUTER0_OUTL_119
R5FSS1_CORE0_INTR_IN_376	376	R5FSS1_INTROUTER0_OUTL_120
R5FSS1_CORE0_INTR_IN_377	377	R5FSS1_INTROUTER0_OUTL_121

**Table 9-49. R5FSS1\_CORE0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS1_CORE0_INTR_IN_378	378	R5FSS1_INTROUTER0_OUTL_122
R5FSS1_CORE0_INTR_IN_379	379	R5FSS1_INTROUTER0_OUTL_123
R5FSS1_CORE0_INTR_IN_380	380	R5FSS1_INTROUTER0_OUTL_124
R5FSS1_CORE0_INTR_IN_381	381	R5FSS1_INTROUTER0_OUTL_125
R5FSS1_CORE0_INTR_IN_382	382	R5FSS1_INTROUTER0_OUTL_126
R5FSS1_CORE0_INTR_IN_383	383	R5FSS1_INTROUTER0_OUTL_127
R5FSS1_CORE0_INTR_IN_384	384	R5FSS1_INTROUTER0_OUTL_128
R5FSS1_CORE0_INTR_IN_385	385	R5FSS1_INTROUTER0_OUTL_129
R5FSS1_CORE0_INTR_IN_386	386	R5FSS1_INTROUTER0_OUTL_130
R5FSS1_CORE0_INTR_IN_387	387	R5FSS1_INTROUTER0_OUTL_131
R5FSS1_CORE0_INTR_IN_388	388	R5FSS1_INTROUTER0_OUTL_132
R5FSS1_CORE0_INTR_IN_389	389	R5FSS1_INTROUTER0_OUTL_133
R5FSS1_CORE0_INTR_IN_390	390	R5FSS1_INTROUTER0_OUTL_134
R5FSS1_CORE0_INTR_IN_391	391	R5FSS1_INTROUTER0_OUTL_135
R5FSS1_CORE0_INTR_IN_392	392	R5FSS1_INTROUTER0_OUTL_136
R5FSS1_CORE0_INTR_IN_393	393	R5FSS1_INTROUTER0_OUTL_137
R5FSS1_CORE0_INTR_IN_394	394	R5FSS1_INTROUTER0_OUTL_138
R5FSS1_CORE0_INTR_IN_395	395	R5FSS1_INTROUTER0_OUTL_139
R5FSS1_CORE0_INTR_IN_396	396	R5FSS1_INTROUTER0_OUTL_140
R5FSS1_CORE0_INTR_IN_397	397	R5FSS1_INTROUTER0_OUTL_141
R5FSS1_CORE0_INTR_IN_398	398	R5FSS1_INTROUTER0_OUTL_142
R5FSS1_CORE0_INTR_IN_399	399	R5FSS1_INTROUTER0_OUTL_143
R5FSS1_CORE0_INTR_IN_400	400	R5FSS1_INTROUTER0_OUTL_144
R5FSS1_CORE0_INTR_IN_401	401	R5FSS1_INTROUTER0_OUTL_145
R5FSS1_CORE0_INTR_IN_402	402	R5FSS1_INTROUTER0_OUTL_146
R5FSS1_CORE0_INTR_IN_403	403	R5FSS1_INTROUTER0_OUTL_147
R5FSS1_CORE0_INTR_IN_404	404	R5FSS1_INTROUTER0_OUTL_148
R5FSS1_CORE0_INTR_IN_405	405	R5FSS1_INTROUTER0_OUTL_149
R5FSS1_CORE0_INTR_IN_406	406	R5FSS1_INTROUTER0_OUTL_150
R5FSS1_CORE0_INTR_IN_407	407	R5FSS1_INTROUTER0_OUTL_151
R5FSS1_CORE0_INTR_IN_408	408	R5FSS1_INTROUTER0_OUTL_152
R5FSS1_CORE0_INTR_IN_409	409	R5FSS1_INTROUTER0_OUTL_153
R5FSS1_CORE0_INTR_IN_410	410	R5FSS1_INTROUTER0_OUTL_154
R5FSS1_CORE0_INTR_IN_411	411	R5FSS1_INTROUTER0_OUTL_155
R5FSS1_CORE0_INTR_IN_412	412	R5FSS1_INTROUTER0_OUTL_156
R5FSS1_CORE0_INTR_IN_413	413	R5FSS1_INTROUTER0_OUTL_157
R5FSS1_CORE0_INTR_IN_414	414	R5FSS1_INTROUTER0_OUTL_158
R5FSS1_CORE0_INTR_IN_415	415	R5FSS1_INTROUTER0_OUTL_159
R5FSS1_CORE0_INTR_IN_416	416	R5FSS1_INTROUTER0_OUTL_160
R5FSS1_CORE0_INTR_IN_417	417	R5FSS1_INTROUTER0_OUTL_161
R5FSS1_CORE0_INTR_IN_418	418	R5FSS1_INTROUTER0_OUTL_162
R5FSS1_CORE0_INTR_IN_419	419	R5FSS1_INTROUTER0_OUTL_163
R5FSS1_CORE0_INTR_IN_420	420	R5FSS1_INTROUTER0_OUTL_164
R5FSS1_CORE0_INTR_IN_421	421	R5FSS1_INTROUTER0_OUTL_165
R5FSS1_CORE0_INTR_IN_422	422	R5FSS1_INTROUTER0_OUTL_166

**Table 9-49. R5FSS1\_CORE0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS1_CORE0_INTR_IN_423	423	R5FSS1_INTROUTER0_OUTL_167
R5FSS1_CORE0_INTR_IN_424	424	R5FSS1_INTROUTER0_OUTL_168
R5FSS1_CORE0_INTR_IN_425	425	R5FSS1_INTROUTER0_OUTL_169
R5FSS1_CORE0_INTR_IN_426	426	R5FSS1_INTROUTER0_OUTL_170
R5FSS1_CORE0_INTR_IN_427	427	R5FSS1_INTROUTER0_OUTL_171
R5FSS1_CORE0_INTR_IN_428	428	R5FSS1_INTROUTER0_OUTL_172
R5FSS1_CORE0_INTR_IN_429	429	R5FSS1_INTROUTER0_OUTL_173
R5FSS1_CORE0_INTR_IN_430	430	R5FSS1_INTROUTER0_OUTL_174
R5FSS1_CORE0_INTR_IN_431	431	R5FSS1_INTROUTER0_OUTL_175
R5FSS1_CORE0_INTR_IN_432	432	R5FSS1_INTROUTER0_OUTL_176
R5FSS1_CORE0_INTR_IN_433	433	R5FSS1_INTROUTER0_OUTL_177
R5FSS1_CORE0_INTR_IN_434	434	R5FSS1_INTROUTER0_OUTL_178
R5FSS1_CORE0_INTR_IN_435	435	R5FSS1_INTROUTER0_OUTL_179
R5FSS1_CORE0_INTR_IN_436	436	R5FSS1_INTROUTER0_OUTL_180
R5FSS1_CORE0_INTR_IN_437	437	R5FSS1_INTROUTER0_OUTL_181
R5FSS1_CORE0_INTR_IN_438	438	R5FSS1_INTROUTER0_OUTL_182
R5FSS1_CORE0_INTR_IN_439	439	R5FSS1_INTROUTER0_OUTL_183
R5FSS1_CORE0_INTR_IN_440	440	R5FSS1_INTROUTER0_OUTL_184
R5FSS1_CORE0_INTR_IN_441	441	R5FSS1_INTROUTER0_OUTL_185
R5FSS1_CORE0_INTR_IN_442	442	R5FSS1_INTROUTER0_OUTL_186
R5FSS1_CORE0_INTR_IN_443	443	R5FSS1_INTROUTER0_OUTL_187
R5FSS1_CORE0_INTR_IN_444	444	R5FSS1_INTROUTER0_OUTL_188
R5FSS1_CORE0_INTR_IN_445	445	R5FSS1_INTROUTER0_OUTL_189
R5FSS1_CORE0_INTR_IN_446	446	R5FSS1_INTROUTER0_OUTL_190
R5FSS1_CORE0_INTR_IN_447	447	R5FSS1_INTROUTER0_OUTL_191
R5FSS1_CORE0_INTR_IN_448	448	R5FSS1_INTROUTER0_OUTL_192
R5FSS1_CORE0_INTR_IN_449	449	R5FSS1_INTROUTER0_OUTL_193
R5FSS1_CORE0_INTR_IN_450	450	R5FSS1_INTROUTER0_OUTL_194
R5FSS1_CORE0_INTR_IN_451	451	R5FSS1_INTROUTER0_OUTL_195
R5FSS1_CORE0_INTR_IN_452	452	R5FSS1_INTROUTER0_OUTL_196
R5FSS1_CORE0_INTR_IN_453	453	R5FSS1_INTROUTER0_OUTL_197
R5FSS1_CORE0_INTR_IN_454	454	R5FSS1_INTROUTER0_OUTL_198
R5FSS1_CORE0_INTR_IN_455	455	R5FSS1_INTROUTER0_OUTL_199
R5FSS1_CORE0_INTR_IN_456	456	R5FSS1_INTROUTER0_OUTL_200
R5FSS1_CORE0_INTR_IN_457	457	R5FSS1_INTROUTER0_OUTL_201
R5FSS1_CORE0_INTR_IN_458	458	R5FSS1_INTROUTER0_OUTL_202
R5FSS1_CORE0_INTR_IN_459	459	R5FSS1_INTROUTER0_OUTL_203
R5FSS1_CORE0_INTR_IN_460	460	R5FSS1_INTROUTER0_OUTL_204
R5FSS1_CORE0_INTR_IN_461	461	R5FSS1_INTROUTER0_OUTL_205
R5FSS1_CORE0_INTR_IN_462	462	R5FSS1_INTROUTER0_OUTL_206
R5FSS1_CORE0_INTR_IN_463	463	R5FSS1_INTROUTER0_OUTL_207
R5FSS1_CORE0_INTR_IN_464	464	R5FSS1_INTROUTER0_OUTL_208
R5FSS1_CORE0_INTR_IN_465	465	R5FSS1_INTROUTER0_OUTL_209
R5FSS1_CORE0_INTR_IN_466	466	R5FSS1_INTROUTER0_OUTL_210
R5FSS1_CORE0_INTR_IN_467	467	R5FSS1_INTROUTER0_OUTL_211



**Table 9-49. R5FSS1\_CORE0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS1_CORE0_INTR_IN_468	468	R5FSS1_INTRROUTER0_OUTL_212
R5FSS1_CORE0_INTR_IN_469	469	R5FSS1_INTRROUTER0_OUTL_213
R5FSS1_CORE0_INTR_IN_470	470	R5FSS1_INTRROUTER0_OUTL_214
R5FSS1_CORE0_INTR_IN_471	471	R5FSS1_INTRROUTER0_OUTL_215
R5FSS1_CORE0_INTR_IN_472	472	R5FSS1_INTRROUTER0_OUTL_216
R5FSS1_CORE0_INTR_IN_473	473	R5FSS1_INTRROUTER0_OUTL_217
R5FSS1_CORE0_INTR_IN_474	474	R5FSS1_INTRROUTER0_OUTL_218
R5FSS1_CORE0_INTR_IN_475	475	R5FSS1_INTRROUTER0_OUTL_219
R5FSS1_CORE0_INTR_IN_476	476	R5FSS1_INTRROUTER0_OUTL_220
R5FSS1_CORE0_INTR_IN_477	477	R5FSS1_INTRROUTER0_OUTL_221
R5FSS1_CORE0_INTR_IN_478	478	R5FSS1_INTRROUTER0_OUTL_222
R5FSS1_CORE0_INTR_IN_479	479	R5FSS1_INTRROUTER0_OUTL_223
R5FSS1_CORE0_INTR_IN_480	480	R5FSS1_INTRROUTER0_OUTL_224
R5FSS1_CORE0_INTR_IN_481	481	R5FSS1_INTRROUTER0_OUTL_225
R5FSS1_CORE0_INTR_IN_482	482	R5FSS1_INTRROUTER0_OUTL_226
R5FSS1_CORE0_INTR_IN_483	483	R5FSS1_INTRROUTER0_OUTL_227
R5FSS1_CORE0_INTR_IN_484	484	R5FSS1_INTRROUTER0_OUTL_228
R5FSS1_CORE0_INTR_IN_485	485	R5FSS1_INTRROUTER0_OUTL_229
R5FSS1_CORE0_INTR_IN_486	486	R5FSS1_INTRROUTER0_OUTL_230
R5FSS1_CORE0_INTR_IN_487	487	R5FSS1_INTRROUTER0_OUTL_231
R5FSS1_CORE0_INTR_IN_488	488	R5FSS1_INTRROUTER0_OUTL_232
R5FSS1_CORE0_INTR_IN_489	489	R5FSS1_INTRROUTER0_OUTL_233
R5FSS1_CORE0_INTR_IN_490	490	R5FSS1_INTRROUTER0_OUTL_234
R5FSS1_CORE0_INTR_IN_491	491	R5FSS1_INTRROUTER0_OUTL_235
R5FSS1_CORE0_INTR_IN_492	492	R5FSS1_INTRROUTER0_OUTL_236
R5FSS1_CORE0_INTR_IN_493	493	R5FSS1_INTRROUTER0_OUTL_237
R5FSS1_CORE0_INTR_IN_494	494	R5FSS1_INTRROUTER0_OUTL_238
R5FSS1_CORE0_INTR_IN_495	495	R5FSS1_INTRROUTER0_OUTL_239
R5FSS1_CORE0_INTR_IN_496	496	R5FSS1_INTRROUTER0_OUTL_240
R5FSS1_CORE0_INTR_IN_497	497	R5FSS1_INTRROUTER0_OUTL_241
R5FSS1_CORE0_INTR_IN_498	498	R5FSS1_INTRROUTER0_OUTL_242
R5FSS1_CORE0_INTR_IN_499	499	R5FSS1_INTRROUTER0_OUTL_243
R5FSS1_CORE0_INTR_IN_500	500	R5FSS1_INTRROUTER0_OUTL_244
R5FSS1_CORE0_INTR_IN_501	501	R5FSS1_INTRROUTER0_OUTL_245
R5FSS1_CORE0_INTR_IN_502	502	R5FSS1_INTRROUTER0_OUTL_246
R5FSS1_CORE0_INTR_IN_503	503	R5FSS1_INTRROUTER0_OUTL_247
R5FSS1_CORE0_INTR_IN_504	504	R5FSS1_INTRROUTER0_OUTL_248
R5FSS1_CORE0_INTR_IN_505	505	R5FSS1_INTRROUTER0_OUTL_249
R5FSS1_CORE0_INTR_IN_506	506	R5FSS1_INTRROUTER0_OUTL_250
R5FSS1_CORE0_INTR_IN_507	507	R5FSS1_INTRROUTER0_OUTL_251
R5FSS1_CORE0_INTR_IN_508	508	R5FSS1_INTRROUTER0_OUTL_252
R5FSS1_CORE0_INTR_IN_509	509	R5FSS1_INTRROUTER0_OUTL_253
R5FSS1_CORE0_INTR_IN_510	510	R5FSS1_INTRROUTER0_OUTL_254
R5FSS1_CORE0_INTR_IN_511	511	R5FSS1_INTRROUTER0_OUTL_255



#### 9.4.3.5 R5FSS1\_CORE1 Interrupt Map

Table 9-48 shows the mapping of events to the R5FSS1\_CORE1. The R5FSS1 VIM supports both R5FSS1 cores.

The R5FSS1\_CORE1 events are not used when operating in lockstep mode.

**Table 9-50. R5FSS1\_CORE1 Interrupt Map**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS1_CORE1_INTR_IN_0	0	CTRL_MMR0_IPC_SET18_IPC_SET_IPCFG_0
R5FSS1_CORE1_INTR_IN_1	1	CTRL_MMR0_IPC_SET19_IPC_SET_IPCFG_0
R5FSS1_CORE1_INTR_IN_2	2	RTI30_INTR_WWD_0
R5FSS1_CORE1_INTR_IN_3	3	RTI31_INTR_WWD_0
R5FSS1_CORE1_INTR_IN_4	4	R5FSS1_COMMRX_LEVEL_1_0
R5FSS1_CORE1_INTR_IN_5	5	R5FSS1_COMMTX_LEVEL_1_0
R5FSS1_CORE1_INTR_IN_6	6	R5FSS1_CORE1_VALFIQ_0
R5FSS1_CORE1_INTR_IN_7	7	R5FSS1_CORE1_VALIRQ_0
R5FSS1_CORE1_INTR_IN_8	8	R5FSS1_CORE0_CTI_0
R5FSS1_CORE1_INTR_IN_9	9	R5FSS1_CORE1_CTI_0
R5FSS1_CORE1_INTR_IN_10	10	ESM0_ESM_INT_LOW_LVL_0
R5FSS1_CORE1_INTR_IN_11	11	ESM0_ESM_INT_HI_LVL_0
R5FSS1_CORE1_INTR_IN_12	12	ESM0_ESM_INT_CFG_LVL_0
R5FSS1_CORE1_INTR_IN_13	13	GLUELOGIC_EXT_INTN_GLUE_EXT_INT_LVL_0
R5FSS1_CORE1_INTR_IN_16	16	R5FSS1_CORE0_EXP_INTR_0
R5FSS1_CORE1_INTR_IN_17	17	R5FSS1_CORE1_EXP_INTR_0
R5FSS1_CORE1_INTR_IN_32	32	DMPAC0_INTD_0_SYSTEM_INTR_LEVEL_0
R5FSS1_CORE1_INTR_IN_33	33	DMPAC0_INTD_0_SYSTEM_INTR_LEVEL_1
R5FSS1_CORE1_INTR_IN_34	34	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_0
R5FSS1_CORE1_INTR_IN_35	35	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_1
R5FSS1_CORE1_INTR_IN_36	36	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_2
R5FSS1_CORE1_INTR_IN_37	37	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_3
R5FSS1_CORE1_INTR_IN_38	38	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_4
R5FSS1_CORE1_INTR_IN_39	39	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_5
R5FSS1_CORE1_INTR_IN_40	40	GPU0_MISC_0_IRQ_0
R5FSS1_CORE1_INTR_IN_48	48	GLUELOGIC_GPU_GPIO_REQACK_GLUE_GPU_GPIO_REQINT_LVL_0
R5FSS1_CORE1_INTR_IN_49	49	GLUELOGIC_GPU_GPIO_REQACK_GLUE_GPU_GPIO_ACKINT_LVL_0
R5FSS1_CORE1_INTR_IN_52	52	DSS0_DSS_INST0_DISPC_FUNC_IRQ_PROC0_0
R5FSS1_CORE1_INTR_IN_53	53	DSS0_DSS_INST0_DISPC_FUNC_IRQ_PROC1_0
R5FSS1_CORE1_INTR_IN_54	54	DSS0_DSS_INST0_DISPC_SAFETY_ERROR_IRQ_PROC0_0
R5FSS1_CORE1_INTR_IN_55	55	DSS0_DSS_INST0_DISPC_SAFETY_ERROR_IRQ_PROC1_0
R5FSS1_CORE1_INTR_IN_56	56	DSS0_DSS_INST0_DISPC_SECURE_IRQ_PROC0_0
R5FSS1_CORE1_INTR_IN_57	57	DSS0_DSS_INST0_DISPC_SECURE_IRQ_PROC1_0
R5FSS1_CORE1_INTR_IN_64	64	DSS_EDP0_INTR_0
R5FSS1_CORE1_INTR_IN_65	65	DSS_EDP0_INTR_1
R5FSS1_CORE1_INTR_IN_66	66	DSS_EDP0_INTR_2
R5FSS1_CORE1_INTR_IN_67	67	DSS_EDP0_INTR_3
R5FSS1_CORE1_INTR_IN_76	76	DSS_DSI0_DSI_0_FUNC_INTR_0
R5FSS1_CORE1_INTR_IN_78	78	CSI_RX_IF0_CSI_ERR_IRQ_0
R5FSS1_CORE1_INTR_IN_79	79	CSI_RX_IF0_CSI_IRQ_0

**Table 9-50. R5FSS1\_CORE1 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS1_CORE1_INTR_IN_80	80	CSI_RX_IF0_CSI_LEVEL_0
R5FSS1_CORE1_INTR_IN_81	81	CSI_RX_IF1_CSI_ERR_IRQ_0
R5FSS1_CORE1_INTR_IN_82	82	CSI_RX_IF1_CSI_IRQ_0
R5FSS1_CORE1_INTR_IN_83	83	CSI_RX_IF1_CSI_LEVEL_0
R5FSS1_CORE1_INTR_IN_87	87	DECODER0_IRQ_0
R5FSS1_CORE1_INTR_IN_88	88	ENCODER0_IRQ_0
R5FSS1_CORE1_INTR_IN_96	96	CPSW0_STAT_PEND_0
R5FSS1_CORE1_INTR_IN_97	97	CPSW0_MDIO_PEND_0
R5FSS1_CORE1_INTR_IN_98	98	CPSW0_EVTN_PEND_0
R5FSS1_CORE1_INTR_IN_99	99	MMCS00_EMMCSS_INTR_0
R5FSS1_CORE1_INTR_IN_104	104	EHRPWM0_EPWM_ETINT_0
R5FSS1_CORE1_INTR_IN_105	105	EHRPWM1_EPWM_ETINT_0
R5FSS1_CORE1_INTR_IN_106	106	EHRPWM2_EPWM_ETINT_0
R5FSS1_CORE1_INTR_IN_107	107	EHRPWM3_EPWM_ETINT_0
R5FSS1_CORE1_INTR_IN_108	108	EHRPWM4_EPWM_ETINT_0
R5FSS1_CORE1_INTR_IN_109	109	EHRPWM5_EPWM_ETINT_0
R5FSS1_CORE1_INTR_IN_110	110	EHRPWM0_EPWM_TRIPZINT_0
R5FSS1_CORE1_INTR_IN_111	111	EHRPWM1_EPWM_TRIPZINT_0
R5FSS1_CORE1_INTR_IN_112	112	EHRPWM2_EPWM_TRIPZINT_0
R5FSS1_CORE1_INTR_IN_113	113	EHRPWM3_EPWM_TRIPZINT_0
R5FSS1_CORE1_INTR_IN_114	114	EHRPWM4_EPWM_TRIPZINT_0
R5FSS1_CORE1_INTR_IN_115	115	EHRPWM5_EPWM_TRIPZINT_0
R5FSS1_CORE1_INTR_IN_116	116	EQEP0_EQEP_INT_0
R5FSS1_CORE1_INTR_IN_117	117	EQEP1_EQEP_INT_0
R5FSS1_CORE1_INTR_IN_118	118	EQEP2_EQEP_INT_0
R5FSS1_CORE1_INTR_IN_120	120	MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS1_CORE1_INTR_IN_121	121	MCAN0_MCANSS_MCAN_LVL_INT_0
R5FSS1_CORE1_INTR_IN_122	122	MCAN0_MCANSS_MCAN_LVL_INT_1
R5FSS1_CORE1_INTR_IN_123	123	MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS1_CORE1_INTR_IN_124	124	MCAN1_MCANSS_MCAN_LVL_INT_0
R5FSS1_CORE1_INTR_IN_125	125	MCAN1_MCANSS_MCAN_LVL_INT_1
R5FSS1_CORE1_INTR_IN_126	126	MCAN2_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS1_CORE1_INTR_IN_127	127	MCAN2_MCANSS_MCAN_LVL_INT_0
R5FSS1_CORE1_INTR_IN_128	128	MCAN2_MCANSS_MCAN_LVL_INT_1
R5FSS1_CORE1_INTR_IN_129	129	MCAN3_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS1_CORE1_INTR_IN_130	130	MCAN3_MCANSS_MCAN_LVL_INT_0
R5FSS1_CORE1_INTR_IN_131	131	MCAN3_MCANSS_MCAN_LVL_INT_1
R5FSS1_CORE1_INTR_IN_132	132	MCASP0_REC_INTR_PEND_0
R5FSS1_CORE1_INTR_IN_133	133	MCASP0_XMIT_INTR_PEND_0
R5FSS1_CORE1_INTR_IN_134	134	MCASP1_REC_INTR_PEND_0
R5FSS1_CORE1_INTR_IN_135	135	MCASP1_XMIT_INTR_PEND_0
R5FSS1_CORE1_INTR_IN_136	136	PCIE0_PCIE_LEGACY_PULSE_0
R5FSS1_CORE1_INTR_IN_137	137	PCIE0_PCIE_DOWNSTREAM_PULSE_0
R5FSS1_CORE1_INTR_IN_138	138	PCIE0_PCIE_FLR_PULSE_0
R5FSS1_CORE1_INTR_IN_139	139	PCIE0_PCIE_PHY_LEVEL_0

**Table 9-50. R5FSS1\_CORE1 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS1_CORE1_INTR_IN_140	140	PCIE0_PCIE_LOCAL_LEVEL_0
R5FSS1_CORE1_INTR_IN_141	141	PCIE0_PCIE_ERROR_PULSE_0
R5FSS1_CORE1_INTR_IN_142	142	PCIE0_PCIE_LINK_STATE_PULSE_0
R5FSS1_CORE1_INTR_IN_143	143	PCIE0_PCIE_PWR_STATE_PULSE_0
R5FSS1_CORE1_INTR_IN_144	144	PCIE0_PCIE_PTM_VALID_PULSE_0
R5FSS1_CORE1_INTR_IN_145	145	PCIE0_PCIE_HOT_RESET_PULSE_0
R5FSS1_CORE1_INTR_IN_146	146	PCIE0_PCIE_CPTS_PEND_0
R5FSS1_CORE1_INTR_IN_150	150	I2C0_POINTRPEND_0
R5FSS1_CORE1_INTR_IN_151	151	I2C1_POINTRPEND_0
R5FSS1_CORE1_INTR_IN_152	152	MCSPi0_INTR_SPI_0
R5FSS1_CORE1_INTR_IN_153	153	MCSPi1_INTR_SPI_0
R5FSS1_CORE1_INTR_IN_154	154	MLB0_MLBSS_MLB_INT_0
R5FSS1_CORE1_INTR_IN_155	155	MLB0_MLBSS_MLB_AHB_INT_0
R5FSS1_CORE1_INTR_IN_156	156	MLB0_MLBSS_MLB_AHB_INT_1
R5FSS1_CORE1_INTR_IN_158	158	UART0_USART_IRQ_0
R5FSS1_CORE1_INTR_IN_159	159	UART1_USART_IRQ_0
R5FSS1_CORE1_INTR_IN_160	160	UART2_USART_IRQ_0
R5FSS1_CORE1_INTR_IN_168	168	TIMER12_INTR_PEND_0
R5FSS1_CORE1_INTR_IN_169	169	TIMER13_INTR_PEND_0
R5FSS1_CORE1_INTR_IN_170	170	TIMER14_INTR_PEND_0
R5FSS1_CORE1_INTR_IN_171	171	TIMER15_INTR_PEND_0
R5FSS1_CORE1_INTR_IN_172	172	TIMER16_INTR_PEND_0
R5FSS1_CORE1_INTR_IN_173	173	TIMER17_INTR_PEND_0
R5FSS1_CORE1_INTR_IN_174	174	TIMER18_INTR_PEND_0
R5FSS1_CORE1_INTR_IN_175	175	TIMER19_INTR_PEND_0
R5FSS1_CORE1_INTR_IN_176	176	GPIOMUX_INTRTR0_OUTP_16
R5FSS1_CORE1_INTR_IN_177	177	GPIOMUX_INTRTR0_OUTP_17
R5FSS1_CORE1_INTR_IN_178	178	GPIOMUX_INTRTR0_OUTP_18
R5FSS1_CORE1_INTR_IN_179	179	GPIOMUX_INTRTR0_OUTP_19
R5FSS1_CORE1_INTR_IN_180	180	GPIOMUX_INTRTR0_OUTP_20
R5FSS1_CORE1_INTR_IN_181	181	GPIOMUX_INTRTR0_OUTP_21
R5FSS1_CORE1_INTR_IN_182	182	GPIOMUX_INTRTR0_OUTP_22
R5FSS1_CORE1_INTR_IN_183	183	GPIOMUX_INTRTR0_OUTP_23
R5FSS1_CORE1_INTR_IN_184	184	GPIOMUX_INTRTR0_OUTP_24
R5FSS1_CORE1_INTR_IN_185	185	GPIOMUX_INTRTR0_OUTP_25
R5FSS1_CORE1_INTR_IN_186	186	GPIOMUX_INTRTR0_OUTP_26
R5FSS1_CORE1_INTR_IN_187	187	GPIOMUX_INTRTR0_OUTP_27
R5FSS1_CORE1_INTR_IN_188	188	GPIOMUX_INTRTR0_OUTP_28
R5FSS1_CORE1_INTR_IN_189	189	GPIOMUX_INTRTR0_OUTP_29
R5FSS1_CORE1_INTR_IN_190	190	GPIOMUX_INTRTR0_OUTP_30
R5FSS1_CORE1_INTR_IN_191	191	GPIOMUX_INTRTR0_OUTP_31
R5FSS1_CORE1_INTR_IN_192	192	MCAN4_MCANSS_MCAN_LVL_INT_0
R5FSS1_CORE1_INTR_IN_193	193	MCAN4_MCANSS_MCAN_LVL_INT_1
R5FSS1_CORE1_INTR_IN_194	194	MCAN4_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS1_CORE1_INTR_IN_195	195	MCAN5_MCANSS_MCAN_LVL_INT_0

**Table 9-50. R5FSS1\_CORE1 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS1_CORE1_INTR_IN_196	196	MCAN5_MCANSS_MCAN_LVL_INT_1
R5FSS1_CORE1_INTR_IN_197	197	MCAN5_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS1_CORE1_INTR_IN_198	198	MCAN6_MCANSS_MCAN_LVL_INT_0
R5FSS1_CORE1_INTR_IN_199	199	MCAN6_MCANSS_MCAN_LVL_INT_1
R5FSS1_CORE1_INTR_IN_200	200	MCAN6_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS1_CORE1_INTR_IN_201	201	MCAN7_MCANSS_MCAN_LVL_INT_0
R5FSS1_CORE1_INTR_IN_202	202	MCAN7_MCANSS_MCAN_LVL_INT_1
R5FSS1_CORE1_INTR_IN_203	203	MCAN7_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS1_CORE1_INTR_IN_204	204	MCAN8_MCANSS_MCAN_LVL_INT_0
R5FSS1_CORE1_INTR_IN_205	205	MCAN8_MCANSS_MCAN_LVL_INT_1
R5FSS1_CORE1_INTR_IN_206	206	MCAN8_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS1_CORE1_INTR_IN_207	207	MCAN9_MCANSS_MCAN_LVL_INT_0
R5FSS1_CORE1_INTR_IN_208	208	MCAN9_MCANSS_MCAN_LVL_INT_1
R5FSS1_CORE1_INTR_IN_209	209	MCAN9_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS1_CORE1_INTR_IN_210	210	MCAN10_MCANSS_MCAN_LVL_INT_0
R5FSS1_CORE1_INTR_IN_211	211	MCAN10_MCANSS_MCAN_LVL_INT_1
R5FSS1_CORE1_INTR_IN_212	212	MCAN10_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS1_CORE1_INTR_IN_213	213	MCAN11_MCANSS_MCAN_LVL_INT_0
R5FSS1_CORE1_INTR_IN_214	214	MCAN11_MCANSS_MCAN_LVL_INT_1
R5FSS1_CORE1_INTR_IN_215	215	MCAN11_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS1_CORE1_INTR_IN_216	216	MCAN12_MCANSS_MCAN_LVL_INT_0
R5FSS1_CORE1_INTR_IN_217	217	MCAN12_MCANSS_MCAN_LVL_INT_1
R5FSS1_CORE1_INTR_IN_218	218	MCAN12_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS1_CORE1_INTR_IN_219	219	MCAN13_MCANSS_MCAN_LVL_INT_0
R5FSS1_CORE1_INTR_IN_220	220	MCAN13_MCANSS_MCAN_LVL_INT_1
R5FSS1_CORE1_INTR_IN_221	221	MCAN13_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
R5FSS1_CORE1_INTR_IN_224	224	NAVSS0_INTR_ROUTER_0_OUTL_INTR_288
R5FSS1_CORE1_INTR_IN_225	225	NAVSS0_INTR_ROUTER_0_OUTL_INTR_289
R5FSS1_CORE1_INTR_IN_226	226	NAVSS0_INTR_ROUTER_0_OUTL_INTR_290
R5FSS1_CORE1_INTR_IN_227	227	NAVSS0_INTR_ROUTER_0_OUTL_INTR_291
R5FSS1_CORE1_INTR_IN_228	228	NAVSS0_INTR_ROUTER_0_OUTL_INTR_292
R5FSS1_CORE1_INTR_IN_229	229	NAVSS0_INTR_ROUTER_0_OUTL_INTR_293
R5FSS1_CORE1_INTR_IN_230	230	NAVSS0_INTR_ROUTER_0_OUTL_INTR_294
R5FSS1_CORE1_INTR_IN_231	231	NAVSS0_INTR_ROUTER_0_OUTL_INTR_295
R5FSS1_CORE1_INTR_IN_232	232	NAVSS0_INTR_ROUTER_0_OUTL_INTR_296
R5FSS1_CORE1_INTR_IN_233	233	NAVSS0_INTR_ROUTER_0_OUTL_INTR_297
R5FSS1_CORE1_INTR_IN_234	234	NAVSS0_INTR_ROUTER_0_OUTL_INTR_298
R5FSS1_CORE1_INTR_IN_235	235	NAVSS0_INTR_ROUTER_0_OUTL_INTR_299
R5FSS1_CORE1_INTR_IN_236	236	NAVSS0_INTR_ROUTER_0_OUTL_INTR_300
R5FSS1_CORE1_INTR_IN_237	237	NAVSS0_INTR_ROUTER_0_OUTL_INTR_301
R5FSS1_CORE1_INTR_IN_238	238	NAVSS0_INTR_ROUTER_0_OUTL_INTR_302
R5FSS1_CORE1_INTR_IN_239	239	NAVSS0_INTR_ROUTER_0_OUTL_INTR_303
R5FSS1_CORE1_INTR_IN_240	240	NAVSS0_INTR_ROUTER_0_OUTL_INTR_304
R5FSS1_CORE1_INTR_IN_241	241	NAVSS0_INTR_ROUTER_0_OUTL_INTR_305
R5FSS1_CORE1_INTR_IN_242	242	NAVSS0_INTR_ROUTER_0_OUTL_INTR_306

**Table 9-50. R5FSS1\_CORE1 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS1_CORE1_INTR_IN_243	243	NAVSS0_INTR_ROUTER_0_OUTL_INTR_307
R5FSS1_CORE1_INTR_IN_244	244	NAVSS0_INTR_ROUTER_0_OUTL_INTR_308
R5FSS1_CORE1_INTR_IN_245	245	NAVSS0_INTR_ROUTER_0_OUTL_INTR_309
R5FSS1_CORE1_INTR_IN_246	246	NAVSS0_INTR_ROUTER_0_OUTL_INTR_310
R5FSS1_CORE1_INTR_IN_247	247	NAVSS0_INTR_ROUTER_0_OUTL_INTR_311
R5FSS1_CORE1_INTR_IN_248	248	NAVSS0_INTR_ROUTER_0_OUTL_INTR_312
R5FSS1_CORE1_INTR_IN_249	249	NAVSS0_INTR_ROUTER_0_OUTL_INTR_313
R5FSS1_CORE1_INTR_IN_250	250	NAVSS0_INTR_ROUTER_0_OUTL_INTR_314
R5FSS1_CORE1_INTR_IN_251	251	NAVSS0_INTR_ROUTER_0_OUTL_INTR_315
R5FSS1_CORE1_INTR_IN_252	252	NAVSS0_INTR_ROUTER_0_OUTL_INTR_316
R5FSS1_CORE1_INTR_IN_253	253	NAVSS0_INTR_ROUTER_0_OUTL_INTR_317
R5FSS1_CORE1_INTR_IN_254	254	NAVSS0_INTR_ROUTER_0_OUTL_INTR_318
R5FSS1_CORE1_INTR_IN_255	255	NAVSS0_INTR_ROUTER_0_OUTL_INTR_319
R5FSS1_CORE1_INTR_IN_256	256	R5FSS1_INTROUTER0_OUTL_0
R5FSS1_CORE1_INTR_IN_257	257	R5FSS1_INTROUTER0_OUTL_1
R5FSS1_CORE1_INTR_IN_258	258	R5FSS1_INTROUTER0_OUTL_2
R5FSS1_CORE1_INTR_IN_259	259	R5FSS1_INTROUTER0_OUTL_3
R5FSS1_CORE1_INTR_IN_260	260	R5FSS1_INTROUTER0_OUTL_4
R5FSS1_CORE1_INTR_IN_261	261	R5FSS1_INTROUTER0_OUTL_5
R5FSS1_CORE1_INTR_IN_262	262	R5FSS1_INTROUTER0_OUTL_6
R5FSS1_CORE1_INTR_IN_263	263	R5FSS1_INTROUTER0_OUTL_7
R5FSS1_CORE1_INTR_IN_264	264	R5FSS1_INTROUTER0_OUTL_8
R5FSS1_CORE1_INTR_IN_265	265	R5FSS1_INTROUTER0_OUTL_9
R5FSS1_CORE1_INTR_IN_266	266	R5FSS1_INTROUTER0_OUTL_10
R5FSS1_CORE1_INTR_IN_267	267	R5FSS1_INTROUTER0_OUTL_11
R5FSS1_CORE1_INTR_IN_268	268	R5FSS1_INTROUTER0_OUTL_12
R5FSS1_CORE1_INTR_IN_269	269	R5FSS1_INTROUTER0_OUTL_13
R5FSS1_CORE1_INTR_IN_270	270	R5FSS1_INTROUTER0_OUTL_14
R5FSS1_CORE1_INTR_IN_271	271	R5FSS1_INTROUTER0_OUTL_15
R5FSS1_CORE1_INTR_IN_272	272	R5FSS1_INTROUTER0_OUTL_16
R5FSS1_CORE1_INTR_IN_273	273	R5FSS1_INTROUTER0_OUTL_17
R5FSS1_CORE1_INTR_IN_274	274	R5FSS1_INTROUTER0_OUTL_18
R5FSS1_CORE1_INTR_IN_275	275	R5FSS1_INTROUTER0_OUTL_19
R5FSS1_CORE1_INTR_IN_276	276	R5FSS1_INTROUTER0_OUTL_20
R5FSS1_CORE1_INTR_IN_277	277	R5FSS1_INTROUTER0_OUTL_21
R5FSS1_CORE1_INTR_IN_278	278	R5FSS1_INTROUTER0_OUTL_22
R5FSS1_CORE1_INTR_IN_279	279	R5FSS1_INTROUTER0_OUTL_23
R5FSS1_CORE1_INTR_IN_280	280	R5FSS1_INTROUTER0_OUTL_24
R5FSS1_CORE1_INTR_IN_281	281	R5FSS1_INTROUTER0_OUTL_25
R5FSS1_CORE1_INTR_IN_282	282	R5FSS1_INTROUTER0_OUTL_26
R5FSS1_CORE1_INTR_IN_283	283	R5FSS1_INTROUTER0_OUTL_27
R5FSS1_CORE1_INTR_IN_284	284	R5FSS1_INTROUTER0_OUTL_28
R5FSS1_CORE1_INTR_IN_285	285	R5FSS1_INTROUTER0_OUTL_29
R5FSS1_CORE1_INTR_IN_286	286	R5FSS1_INTROUTER0_OUTL_30
R5FSS1_CORE1_INTR_IN_287	287	R5FSS1_INTROUTER0_OUTL_31

**Table 9-50. R5FSS1\_CORE1 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS1_CORE1_INTR_IN_288	288	R5FSS1_INTRROUTER0_OUTL_32
R5FSS1_CORE1_INTR_IN_289	289	R5FSS1_INTRROUTER0_OUTL_33
R5FSS1_CORE1_INTR_IN_290	290	R5FSS1_INTRROUTER0_OUTL_34
R5FSS1_CORE1_INTR_IN_291	291	R5FSS1_INTRROUTER0_OUTL_35
R5FSS1_CORE1_INTR_IN_292	292	R5FSS1_INTRROUTER0_OUTL_36
R5FSS1_CORE1_INTR_IN_293	293	R5FSS1_INTRROUTER0_OUTL_37
R5FSS1_CORE1_INTR_IN_294	294	R5FSS1_INTRROUTER0_OUTL_38
R5FSS1_CORE1_INTR_IN_295	295	R5FSS1_INTRROUTER0_OUTL_39
R5FSS1_CORE1_INTR_IN_296	296	R5FSS1_INTRROUTER0_OUTL_40
R5FSS1_CORE1_INTR_IN_297	297	R5FSS1_INTRROUTER0_OUTL_41
R5FSS1_CORE1_INTR_IN_298	298	R5FSS1_INTRROUTER0_OUTL_42
R5FSS1_CORE1_INTR_IN_299	299	R5FSS1_INTRROUTER0_OUTL_43
R5FSS1_CORE1_INTR_IN_300	300	R5FSS1_INTRROUTER0_OUTL_44
R5FSS1_CORE1_INTR_IN_301	301	R5FSS1_INTRROUTER0_OUTL_45
R5FSS1_CORE1_INTR_IN_302	302	R5FSS1_INTRROUTER0_OUTL_46
R5FSS1_CORE1_INTR_IN_303	303	R5FSS1_INTRROUTER0_OUTL_47
R5FSS1_CORE1_INTR_IN_304	304	R5FSS1_INTRROUTER0_OUTL_48
R5FSS1_CORE1_INTR_IN_305	305	R5FSS1_INTRROUTER0_OUTL_49
R5FSS1_CORE1_INTR_IN_306	306	R5FSS1_INTRROUTER0_OUTL_50
R5FSS1_CORE1_INTR_IN_307	307	R5FSS1_INTRROUTER0_OUTL_51
R5FSS1_CORE1_INTR_IN_308	308	R5FSS1_INTRROUTER0_OUTL_52
R5FSS1_CORE1_INTR_IN_309	309	R5FSS1_INTRROUTER0_OUTL_53
R5FSS1_CORE1_INTR_IN_310	310	R5FSS1_INTRROUTER0_OUTL_54
R5FSS1_CORE1_INTR_IN_311	311	R5FSS1_INTRROUTER0_OUTL_55
R5FSS1_CORE1_INTR_IN_312	312	R5FSS1_INTRROUTER0_OUTL_56
R5FSS1_CORE1_INTR_IN_313	313	R5FSS1_INTRROUTER0_OUTL_57
R5FSS1_CORE1_INTR_IN_314	314	R5FSS1_INTRROUTER0_OUTL_58
R5FSS1_CORE1_INTR_IN_315	315	R5FSS1_INTRROUTER0_OUTL_59
R5FSS1_CORE1_INTR_IN_316	316	R5FSS1_INTRROUTER0_OUTL_60
R5FSS1_CORE1_INTR_IN_317	317	R5FSS1_INTRROUTER0_OUTL_61
R5FSS1_CORE1_INTR_IN_318	318	R5FSS1_INTRROUTER0_OUTL_62
R5FSS1_CORE1_INTR_IN_319	319	R5FSS1_INTRROUTER0_OUTL_63
R5FSS1_CORE1_INTR_IN_320	320	R5FSS1_INTRROUTER0_OUTL_64
R5FSS1_CORE1_INTR_IN_321	321	R5FSS1_INTRROUTER0_OUTL_65
R5FSS1_CORE1_INTR_IN_322	322	R5FSS1_INTRROUTER0_OUTL_66
R5FSS1_CORE1_INTR_IN_323	323	R5FSS1_INTRROUTER0_OUTL_67
R5FSS1_CORE1_INTR_IN_324	324	R5FSS1_INTRROUTER0_OUTL_68
R5FSS1_CORE1_INTR_IN_325	325	R5FSS1_INTRROUTER0_OUTL_69
R5FSS1_CORE1_INTR_IN_326	326	R5FSS1_INTRROUTER0_OUTL_70
R5FSS1_CORE1_INTR_IN_327	327	R5FSS1_INTRROUTER0_OUTL_71
R5FSS1_CORE1_INTR_IN_328	328	R5FSS1_INTRROUTER0_OUTL_72
R5FSS1_CORE1_INTR_IN_329	329	R5FSS1_INTRROUTER0_OUTL_73
R5FSS1_CORE1_INTR_IN_330	330	R5FSS1_INTRROUTER0_OUTL_74
R5FSS1_CORE1_INTR_IN_331	331	R5FSS1_INTRROUTER0_OUTL_75
R5FSS1_CORE1_INTR_IN_332	332	R5FSS1_INTRROUTER0_OUTL_76

**Table 9-50. R5FSS1\_CORE1 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS1_CORE1_INTR_IN_333	333	R5FSS1_INTRROUTER0_OUTL_77
R5FSS1_CORE1_INTR_IN_334	334	R5FSS1_INTRROUTER0_OUTL_78
R5FSS1_CORE1_INTR_IN_335	335	R5FSS1_INTRROUTER0_OUTL_79
R5FSS1_CORE1_INTR_IN_336	336	R5FSS1_INTRROUTER0_OUTL_80
R5FSS1_CORE1_INTR_IN_337	337	R5FSS1_INTRROUTER0_OUTL_81
R5FSS1_CORE1_INTR_IN_338	338	R5FSS1_INTRROUTER0_OUTL_82
R5FSS1_CORE1_INTR_IN_339	339	R5FSS1_INTRROUTER0_OUTL_83
R5FSS1_CORE1_INTR_IN_340	340	R5FSS1_INTRROUTER0_OUTL_84
R5FSS1_CORE1_INTR_IN_341	341	R5FSS1_INTRROUTER0_OUTL_85
R5FSS1_CORE1_INTR_IN_342	342	R5FSS1_INTRROUTER0_OUTL_86
R5FSS1_CORE1_INTR_IN_343	343	R5FSS1_INTRROUTER0_OUTL_87
R5FSS1_CORE1_INTR_IN_344	344	R5FSS1_INTRROUTER0_OUTL_88
R5FSS1_CORE1_INTR_IN_345	345	R5FSS1_INTRROUTER0_OUTL_89
R5FSS1_CORE1_INTR_IN_346	346	R5FSS1_INTRROUTER0_OUTL_90
R5FSS1_CORE1_INTR_IN_347	347	R5FSS1_INTRROUTER0_OUTL_91
R5FSS1_CORE1_INTR_IN_348	348	R5FSS1_INTRROUTER0_OUTL_92
R5FSS1_CORE1_INTR_IN_349	349	R5FSS1_INTRROUTER0_OUTL_93
R5FSS1_CORE1_INTR_IN_350	350	R5FSS1_INTRROUTER0_OUTL_94
R5FSS1_CORE1_INTR_IN_351	351	R5FSS1_INTRROUTER0_OUTL_95
R5FSS1_CORE1_INTR_IN_352	352	R5FSS1_INTRROUTER0_OUTL_96
R5FSS1_CORE1_INTR_IN_353	353	R5FSS1_INTRROUTER0_OUTL_97
R5FSS1_CORE1_INTR_IN_354	354	R5FSS1_INTRROUTER0_OUTL_98
R5FSS1_CORE1_INTR_IN_355	355	R5FSS1_INTRROUTER0_OUTL_99
R5FSS1_CORE1_INTR_IN_356	356	R5FSS1_INTRROUTER0_OUTL_100
R5FSS1_CORE1_INTR_IN_357	357	R5FSS1_INTRROUTER0_OUTL_101
R5FSS1_CORE1_INTR_IN_358	358	R5FSS1_INTRROUTER0_OUTL_102
R5FSS1_CORE1_INTR_IN_359	359	R5FSS1_INTRROUTER0_OUTL_103
R5FSS1_CORE1_INTR_IN_360	360	R5FSS1_INTRROUTER0_OUTL_104
R5FSS1_CORE1_INTR_IN_361	361	R5FSS1_INTRROUTER0_OUTL_105
R5FSS1_CORE1_INTR_IN_362	362	R5FSS1_INTRROUTER0_OUTL_106
R5FSS1_CORE1_INTR_IN_363	363	R5FSS1_INTRROUTER0_OUTL_107
R5FSS1_CORE1_INTR_IN_364	364	R5FSS1_INTRROUTER0_OUTL_108
R5FSS1_CORE1_INTR_IN_365	365	R5FSS1_INTRROUTER0_OUTL_109
R5FSS1_CORE1_INTR_IN_366	366	R5FSS1_INTRROUTER0_OUTL_110
R5FSS1_CORE1_INTR_IN_367	367	R5FSS1_INTRROUTER0_OUTL_111
R5FSS1_CORE1_INTR_IN_368	368	R5FSS1_INTRROUTER0_OUTL_112
R5FSS1_CORE1_INTR_IN_369	369	R5FSS1_INTRROUTER0_OUTL_113
R5FSS1_CORE1_INTR_IN_370	370	R5FSS1_INTRROUTER0_OUTL_114
R5FSS1_CORE1_INTR_IN_371	371	R5FSS1_INTRROUTER0_OUTL_115
R5FSS1_CORE1_INTR_IN_372	372	R5FSS1_INTRROUTER0_OUTL_116
R5FSS1_CORE1_INTR_IN_373	373	R5FSS1_INTRROUTER0_OUTL_117
R5FSS1_CORE1_INTR_IN_374	374	R5FSS1_INTRROUTER0_OUTL_118
R5FSS1_CORE1_INTR_IN_375	375	R5FSS1_INTRROUTER0_OUTL_119
R5FSS1_CORE1_INTR_IN_376	376	R5FSS1_INTRROUTER0_OUTL_120
R5FSS1_CORE1_INTR_IN_377	377	R5FSS1_INTRROUTER0_OUTL_121



**Table 9-50. R5FSS1\_CORE1 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS1_CORE1_INTR_IN_378	378	R5FSS1_INTRROUTER0_OUTL_122
R5FSS1_CORE1_INTR_IN_379	379	R5FSS1_INTRROUTER0_OUTL_123
R5FSS1_CORE1_INTR_IN_380	380	R5FSS1_INTRROUTER0_OUTL_124
R5FSS1_CORE1_INTR_IN_381	381	R5FSS1_INTRROUTER0_OUTL_125
R5FSS1_CORE1_INTR_IN_382	382	R5FSS1_INTRROUTER0_OUTL_126
R5FSS1_CORE1_INTR_IN_383	383	R5FSS1_INTRROUTER0_OUTL_127
R5FSS1_CORE1_INTR_IN_384	384	R5FSS1_INTRROUTER0_OUTL_128
R5FSS1_CORE1_INTR_IN_385	385	R5FSS1_INTRROUTER0_OUTL_129
R5FSS1_CORE1_INTR_IN_386	386	R5FSS1_INTRROUTER0_OUTL_130
R5FSS1_CORE1_INTR_IN_387	387	R5FSS1_INTRROUTER0_OUTL_131
R5FSS1_CORE1_INTR_IN_388	388	R5FSS1_INTRROUTER0_OUTL_132
R5FSS1_CORE1_INTR_IN_389	389	R5FSS1_INTRROUTER0_OUTL_133
R5FSS1_CORE1_INTR_IN_390	390	R5FSS1_INTRROUTER0_OUTL_134
R5FSS1_CORE1_INTR_IN_391	391	R5FSS1_INTRROUTER0_OUTL_135
R5FSS1_CORE1_INTR_IN_392	392	R5FSS1_INTRROUTER0_OUTL_136
R5FSS1_CORE1_INTR_IN_393	393	R5FSS1_INTRROUTER0_OUTL_137
R5FSS1_CORE1_INTR_IN_394	394	R5FSS1_INTRROUTER0_OUTL_138
R5FSS1_CORE1_INTR_IN_395	395	R5FSS1_INTRROUTER0_OUTL_139
R5FSS1_CORE1_INTR_IN_396	396	R5FSS1_INTRROUTER0_OUTL_140
R5FSS1_CORE1_INTR_IN_397	397	R5FSS1_INTRROUTER0_OUTL_141
R5FSS1_CORE1_INTR_IN_398	398	R5FSS1_INTRROUTER0_OUTL_142
R5FSS1_CORE1_INTR_IN_399	399	R5FSS1_INTRROUTER0_OUTL_143
R5FSS1_CORE1_INTR_IN_400	400	R5FSS1_INTRROUTER0_OUTL_144
R5FSS1_CORE1_INTR_IN_401	401	R5FSS1_INTRROUTER0_OUTL_145
R5FSS1_CORE1_INTR_IN_402	402	R5FSS1_INTRROUTER0_OUTL_146
R5FSS1_CORE1_INTR_IN_403	403	R5FSS1_INTRROUTER0_OUTL_147
R5FSS1_CORE1_INTR_IN_404	404	R5FSS1_INTRROUTER0_OUTL_148
R5FSS1_CORE1_INTR_IN_405	405	R5FSS1_INTRROUTER0_OUTL_149
R5FSS1_CORE1_INTR_IN_406	406	R5FSS1_INTRROUTER0_OUTL_150
R5FSS1_CORE1_INTR_IN_407	407	R5FSS1_INTRROUTER0_OUTL_151
R5FSS1_CORE1_INTR_IN_408	408	R5FSS1_INTRROUTER0_OUTL_152
R5FSS1_CORE1_INTR_IN_409	409	R5FSS1_INTRROUTER0_OUTL_153
R5FSS1_CORE1_INTR_IN_410	410	R5FSS1_INTRROUTER0_OUTL_154
R5FSS1_CORE1_INTR_IN_411	411	R5FSS1_INTRROUTER0_OUTL_155
R5FSS1_CORE1_INTR_IN_412	412	R5FSS1_INTRROUTER0_OUTL_156
R5FSS1_CORE1_INTR_IN_413	413	R5FSS1_INTRROUTER0_OUTL_157
R5FSS1_CORE1_INTR_IN_414	414	R5FSS1_INTRROUTER0_OUTL_158
R5FSS1_CORE1_INTR_IN_415	415	R5FSS1_INTRROUTER0_OUTL_159
R5FSS1_CORE1_INTR_IN_416	416	R5FSS1_INTRROUTER0_OUTL_160
R5FSS1_CORE1_INTR_IN_417	417	R5FSS1_INTRROUTER0_OUTL_161
R5FSS1_CORE1_INTR_IN_418	418	R5FSS1_INTRROUTER0_OUTL_162
R5FSS1_CORE1_INTR_IN_419	419	R5FSS1_INTRROUTER0_OUTL_163
R5FSS1_CORE1_INTR_IN_420	420	R5FSS1_INTRROUTER0_OUTL_164
R5FSS1_CORE1_INTR_IN_421	421	R5FSS1_INTRROUTER0_OUTL_165
R5FSS1_CORE1_INTR_IN_422	422	R5FSS1_INTRROUTER0_OUTL_166



**Table 9-50. R5FSS1\_CORE1 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS1_CORE1_INTR_IN_423	423	R5FSS1_INTRROUTER0_OUTL_167
R5FSS1_CORE1_INTR_IN_424	424	R5FSS1_INTRROUTER0_OUTL_168
R5FSS1_CORE1_INTR_IN_425	425	R5FSS1_INTRROUTER0_OUTL_169
R5FSS1_CORE1_INTR_IN_426	426	R5FSS1_INTRROUTER0_OUTL_170
R5FSS1_CORE1_INTR_IN_427	427	R5FSS1_INTRROUTER0_OUTL_171
R5FSS1_CORE1_INTR_IN_428	428	R5FSS1_INTRROUTER0_OUTL_172
R5FSS1_CORE1_INTR_IN_429	429	R5FSS1_INTRROUTER0_OUTL_173
R5FSS1_CORE1_INTR_IN_430	430	R5FSS1_INTRROUTER0_OUTL_174
R5FSS1_CORE1_INTR_IN_431	431	R5FSS1_INTRROUTER0_OUTL_175
R5FSS1_CORE1_INTR_IN_432	432	R5FSS1_INTRROUTER0_OUTL_176
R5FSS1_CORE1_INTR_IN_433	433	R5FSS1_INTRROUTER0_OUTL_177
R5FSS1_CORE1_INTR_IN_434	434	R5FSS1_INTRROUTER0_OUTL_178
R5FSS1_CORE1_INTR_IN_435	435	R5FSS1_INTRROUTER0_OUTL_179
R5FSS1_CORE1_INTR_IN_436	436	R5FSS1_INTRROUTER0_OUTL_180
R5FSS1_CORE1_INTR_IN_437	437	R5FSS1_INTRROUTER0_OUTL_181
R5FSS1_CORE1_INTR_IN_438	438	R5FSS1_INTRROUTER0_OUTL_182
R5FSS1_CORE1_INTR_IN_439	439	R5FSS1_INTRROUTER0_OUTL_183
R5FSS1_CORE1_INTR_IN_440	440	R5FSS1_INTRROUTER0_OUTL_184
R5FSS1_CORE1_INTR_IN_441	441	R5FSS1_INTRROUTER0_OUTL_185
R5FSS1_CORE1_INTR_IN_442	442	R5FSS1_INTRROUTER0_OUTL_186
R5FSS1_CORE1_INTR_IN_443	443	R5FSS1_INTRROUTER0_OUTL_187
R5FSS1_CORE1_INTR_IN_444	444	R5FSS1_INTRROUTER0_OUTL_188
R5FSS1_CORE1_INTR_IN_445	445	R5FSS1_INTRROUTER0_OUTL_189
R5FSS1_CORE1_INTR_IN_446	446	R5FSS1_INTRROUTER0_OUTL_190
R5FSS1_CORE1_INTR_IN_447	447	R5FSS1_INTRROUTER0_OUTL_191
R5FSS1_CORE1_INTR_IN_448	448	R5FSS1_INTRROUTER0_OUTL_192
R5FSS1_CORE1_INTR_IN_449	449	R5FSS1_INTRROUTER0_OUTL_193
R5FSS1_CORE1_INTR_IN_450	450	R5FSS1_INTRROUTER0_OUTL_194
R5FSS1_CORE1_INTR_IN_451	451	R5FSS1_INTRROUTER0_OUTL_195
R5FSS1_CORE1_INTR_IN_452	452	R5FSS1_INTRROUTER0_OUTL_196
R5FSS1_CORE1_INTR_IN_453	453	R5FSS1_INTRROUTER0_OUTL_197
R5FSS1_CORE1_INTR_IN_454	454	R5FSS1_INTRROUTER0_OUTL_198
R5FSS1_CORE1_INTR_IN_455	455	R5FSS1_INTRROUTER0_OUTL_199
R5FSS1_CORE1_INTR_IN_456	456	R5FSS1_INTRROUTER0_OUTL_200
R5FSS1_CORE1_INTR_IN_457	457	R5FSS1_INTRROUTER0_OUTL_201
R5FSS1_CORE1_INTR_IN_458	458	R5FSS1_INTRROUTER0_OUTL_202
R5FSS1_CORE1_INTR_IN_459	459	R5FSS1_INTRROUTER0_OUTL_203
R5FSS1_CORE1_INTR_IN_460	460	R5FSS1_INTRROUTER0_OUTL_204
R5FSS1_CORE1_INTR_IN_461	461	R5FSS1_INTRROUTER0_OUTL_205
R5FSS1_CORE1_INTR_IN_462	462	R5FSS1_INTRROUTER0_OUTL_206
R5FSS1_CORE1_INTR_IN_463	463	R5FSS1_INTRROUTER0_OUTL_207
R5FSS1_CORE1_INTR_IN_464	464	R5FSS1_INTRROUTER0_OUTL_208
R5FSS1_CORE1_INTR_IN_465	465	R5FSS1_INTRROUTER0_OUTL_209
R5FSS1_CORE1_INTR_IN_466	466	R5FSS1_INTRROUTER0_OUTL_210
R5FSS1_CORE1_INTR_IN_467	467	R5FSS1_INTRROUTER0_OUTL_211

**Table 9-50. R5FSS1\_CORE1 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
R5FSS1_CORE1_INTR_IN_468	468	R5FSS1_INTRROUTER0_OUTL_212
R5FSS1_CORE1_INTR_IN_469	469	R5FSS1_INTRROUTER0_OUTL_213
R5FSS1_CORE1_INTR_IN_470	470	R5FSS1_INTRROUTER0_OUTL_214
R5FSS1_CORE1_INTR_IN_471	471	R5FSS1_INTRROUTER0_OUTL_215
R5FSS1_CORE1_INTR_IN_472	472	R5FSS1_INTRROUTER0_OUTL_216
R5FSS1_CORE1_INTR_IN_473	473	R5FSS1_INTRROUTER0_OUTL_217
R5FSS1_CORE1_INTR_IN_474	474	R5FSS1_INTRROUTER0_OUTL_218
R5FSS1_CORE1_INTR_IN_475	475	R5FSS1_INTRROUTER0_OUTL_219
R5FSS1_CORE1_INTR_IN_476	476	R5FSS1_INTRROUTER0_OUTL_220
R5FSS1_CORE1_INTR_IN_477	477	R5FSS1_INTRROUTER0_OUTL_221
R5FSS1_CORE1_INTR_IN_478	478	R5FSS1_INTRROUTER0_OUTL_222
R5FSS1_CORE1_INTR_IN_479	479	R5FSS1_INTRROUTER0_OUTL_223
R5FSS1_CORE1_INTR_IN_480	480	R5FSS1_INTRROUTER0_OUTL_224
R5FSS1_CORE1_INTR_IN_481	481	R5FSS1_INTRROUTER0_OUTL_225
R5FSS1_CORE1_INTR_IN_482	482	R5FSS1_INTRROUTER0_OUTL_226
R5FSS1_CORE1_INTR_IN_483	483	R5FSS1_INTRROUTER0_OUTL_227
R5FSS1_CORE1_INTR_IN_484	484	R5FSS1_INTRROUTER0_OUTL_228
R5FSS1_CORE1_INTR_IN_485	485	R5FSS1_INTRROUTER0_OUTL_229
R5FSS1_CORE1_INTR_IN_486	486	R5FSS1_INTRROUTER0_OUTL_230
R5FSS1_CORE1_INTR_IN_487	487	R5FSS1_INTRROUTER0_OUTL_231
R5FSS1_CORE1_INTR_IN_488	488	R5FSS1_INTRROUTER0_OUTL_232
R5FSS1_CORE1_INTR_IN_489	489	R5FSS1_INTRROUTER0_OUTL_233
R5FSS1_CORE1_INTR_IN_490	490	R5FSS1_INTRROUTER0_OUTL_234
R5FSS1_CORE1_INTR_IN_491	491	R5FSS1_INTRROUTER0_OUTL_235
R5FSS1_CORE1_INTR_IN_492	492	R5FSS1_INTRROUTER0_OUTL_236
R5FSS1_CORE1_INTR_IN_493	493	R5FSS1_INTRROUTER0_OUTL_237
R5FSS1_CORE1_INTR_IN_494	494	R5FSS1_INTRROUTER0_OUTL_238
R5FSS1_CORE1_INTR_IN_495	495	R5FSS1_INTRROUTER0_OUTL_239
R5FSS1_CORE1_INTR_IN_496	496	R5FSS1_INTRROUTER0_OUTL_240
R5FSS1_CORE1_INTR_IN_497	497	R5FSS1_INTRROUTER0_OUTL_241
R5FSS1_CORE1_INTR_IN_498	498	R5FSS1_INTRROUTER0_OUTL_242
R5FSS1_CORE1_INTR_IN_499	499	R5FSS1_INTRROUTER0_OUTL_243
R5FSS1_CORE1_INTR_IN_500	500	R5FSS1_INTRROUTER0_OUTL_244
R5FSS1_CORE1_INTR_IN_501	501	R5FSS1_INTRROUTER0_OUTL_245
R5FSS1_CORE1_INTR_IN_502	502	R5FSS1_INTRROUTER0_OUTL_246
R5FSS1_CORE1_INTR_IN_503	503	R5FSS1_INTRROUTER0_OUTL_247
R5FSS1_CORE1_INTR_IN_504	504	R5FSS1_INTRROUTER0_OUTL_248
R5FSS1_CORE1_INTR_IN_505	505	R5FSS1_INTRROUTER0_OUTL_249
R5FSS1_CORE1_INTR_IN_506	506	R5FSS1_INTRROUTER0_OUTL_250
R5FSS1_CORE1_INTR_IN_507	507	R5FSS1_INTRROUTER0_OUTL_251
R5FSS1_CORE1_INTR_IN_508	508	R5FSS1_INTRROUTER0_OUTL_252
R5FSS1_CORE1_INTR_IN_509	509	R5FSS1_INTRROUTER0_OUTL_253
R5FSS1_CORE1_INTR_IN_510	510	R5FSS1_INTRROUTER0_OUTL_254
R5FSS1_CORE1_INTR_IN_511	511	R5FSS1_INTRROUTER0_OUTL_255

#### 9.4.3.6 R5FSS0\_INTRTR0 Interrupt Map

Table 9-51 lists all the interrupts that are mapped to the R5FSS0\_INTRTR0 inputs, along with the corresponding register programming values used for mux control. Any R5FSS0\_INTRTR0 interrupt input that is not listed in this table should be considered as unused.

**Table 9-51. R5FSS0\_INTRTR0 Interrupt Map**

Interrupt Input Line	Peripheral Interrupt	R5FSS0_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
RESERVED	RESERVED	0 <sup>(1)</sup>
R5FSS0_INTRTR0_IN_0	USB1_OTGIRQ_0	1
R5FSS0_INTRTR0_IN_1	USB1_IRQ_0	2
R5FSS0_INTRTR0_IN_2	USB1_IRQ_1	3
R5FSS0_INTRTR0_IN_3	USB1_IRQ_2	4
R5FSS0_INTRTR0_IN_4	USB1_IRQ_3	5
R5FSS0_INTRTR0_IN_5	USB1_IRQ_4	6
R5FSS0_INTRTR0_IN_6	USB1_IRQ_5	7
R5FSS0_INTRTR0_IN_7	USB1_IRQ_6	8
R5FSS0_INTRTR0_IN_8	USB1_IRQ_7	9
R5FSS0_INTRTR0_IN_18	PCIE2_PCIE_LEGACY_PULSE_0	19
R5FSS0_INTRTR0_IN_19	PCIE2_PCIE_DOWNSTREAM_PULSE_0	20
R5FSS0_INTRTR0_IN_20	PCIE2_PCIE_FLR_PULSE_0	21
R5FSS0_INTRTR0_IN_21	PCIE2_PCIE_PHY_LEVEL_0	22
R5FSS0_INTRTR0_IN_22	PCIE2_PCIE_LOCAL_LEVEL_0	23
R5FSS0_INTRTR0_IN_23	PCIE2_PCIE_ERROR_PULSE_0	24
R5FSS0_INTRTR0_IN_24	PCIE2_PCIE_LINK_STATE_PULSE_0	25
R5FSS0_INTRTR0_IN_25	PCIE2_PCIE_PWR_STATE_PULSE_0	26
R5FSS0_INTRTR0_IN_26	PCIE2_PCIE_PTM_VALID_PULSE_0	27
R5FSS0_INTRTR0_IN_27	PCIE2_PCIE_HOT_RESET_PULSE_0	28
R5FSS0_INTRTR0_IN_28	PCIE2_PCIE_CPTS_PEND_0	29
R5FSS0_INTRTR0_IN_29	PRU_ICSSG0_PR1_HOST_INTR_PEND_0	30
R5FSS0_INTRTR0_IN_30	PRU_ICSSG0_PR1_HOST_INTR_PEND_1	31
R5FSS0_INTRTR0_IN_31	PRU_ICSSG0_PR1_HOST_INTR_PEND_2	32
R5FSS0_INTRTR0_IN_32	PRU_ICSSG0_PR1_HOST_INTR_PEND_3	33
R5FSS0_INTRTR0_IN_33	PRU_ICSSG0_PR1_HOST_INTR_PEND_4	34
R5FSS0_INTRTR0_IN_34	PRU_ICSSG0_PR1_HOST_INTR_PEND_5	35
R5FSS0_INTRTR0_IN_35	PRU_ICSSG0_PR1_HOST_INTR_PEND_6	36
R5FSS0_INTRTR0_IN_36	PRU_ICSSG0_PR1_HOST_INTR_PEND_7	37
R5FSS0_INTRTR0_IN_37	PRU_ICSSG1_PR1_HOST_INTR_PEND_0	38
R5FSS0_INTRTR0_IN_38	PRU_ICSSG1_PR1_HOST_INTR_PEND_1	39
R5FSS0_INTRTR0_IN_39	PRU_ICSSG1_PR1_HOST_INTR_PEND_2	40
R5FSS0_INTRTR0_IN_40	PRU_ICSSG1_PR1_HOST_INTR_PEND_3	41
R5FSS0_INTRTR0_IN_41	PRU_ICSSG1_PR1_HOST_INTR_PEND_4	42
R5FSS0_INTRTR0_IN_42	PRU_ICSSG1_PR1_HOST_INTR_PEND_5	43
R5FSS0_INTRTR0_IN_43	PRU_ICSSG1_PR1_HOST_INTR_PEND_6	44
R5FSS0_INTRTR0_IN_44	PRU_ICSSG1_PR1_HOST_INTR_PEND_7	45
R5FSS0_INTRTR0_IN_45	PRU_ICSSG0_PR1_TX_SOF_INTR_REQ_0	46
R5FSS0_INTRTR0_IN_46	PRU_ICSSG0_PR1_TX_SOF_INTR_REQ_1	47
R5FSS0_INTRTR0_IN_47	PRU_ICSSG0_PR1_RX_SOF_INTR_REQ_0	48

**Table 9-51. R5FSS0\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	R5FSS0_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
R5FSS0_INTRTR0_IN_48	PRU_ICSSG0_PR1_RX_SOF_INTR_REQ_1	49
R5FSS0_INTRTR0_IN_49	PRU_ICSSG1_PR1_TX_SOF_INTR_REQ_0	50
R5FSS0_INTRTR0_IN_50	PRU_ICSSG1_PR1_TX_SOF_INTR_REQ_1	51
R5FSS0_INTRTR0_IN_51	PRU_ICSSG1_PR1_RX_SOF_INTR_REQ_0	52
R5FSS0_INTRTR0_IN_52	PRU_ICSSG1_PR1_RX_SOF_INTR_REQ_1	53
R5FSS0_INTRTR0_IN_53	PCIE3_PCIE_LEGACY_PULSE_0	54
R5FSS0_INTRTR0_IN_54	PCIE3_PCIE_DOWNSTREAM_PULSE_0	55
R5FSS0_INTRTR0_IN_55	PCIE3_PCIE_FLR_PULSE_0	56
R5FSS0_INTRTR0_IN_56	PCIE3_PCIE_PHY_LEVEL_0	57
R5FSS0_INTRTR0_IN_57	PCIE3_PCIE_LOCAL_LEVEL_0	58
R5FSS0_INTRTR0_IN_58	PCIE3_PCIE_ERROR_PULSE_0	59
R5FSS0_INTRTR0_IN_59	PCIE3_PCIE_LINK_STATE_PULSE_0	60
R5FSS0_INTRTR0_IN_60	PCIE3_PCIE_PWR_STATE_PULSE_0	61
R5FSS0_INTRTR0_IN_61	PCIE3_PCIE_PTM_VALID_PULSE_0	62
R5FSS0_INTRTR0_IN_62	PCIE3_PCIE_HOT_RESET_PULSE_0	63
R5FSS0_INTRTR0_IN_63	PCIE3_PCIE_CPTS_PEND_0	64
R5FSS0_INTRTR0_IN_64	USB1_HOST_SYSTEM_ERROR_0	65
R5FSS0_INTRTR0_IN_67	GPIOMUX_INTRTR0_OUTP_0	68
R5FSS0_INTRTR0_IN_68	GPIOMUX_INTRTR0_OUTP_1	69
R5FSS0_INTRTR0_IN_69	GPIOMUX_INTRTR0_OUTP_2	70
R5FSS0_INTRTR0_IN_70	GPIOMUX_INTRTR0_OUTP_3	71
R5FSS0_INTRTR0_IN_71	GPIOMUX_INTRTR0_OUTP_4	72
R5FSS0_INTRTR0_IN_72	GPIOMUX_INTRTR0_OUTP_5	73
R5FSS0_INTRTR0_IN_73	GPIOMUX_INTRTR0_OUTP_6	74
R5FSS0_INTRTR0_IN_74	GPIOMUX_INTRTR0_OUTP_7	75
R5FSS0_INTRTR0_IN_75	GPIOMUX_INTRTR0_OUTP_8	76
R5FSS0_INTRTR0_IN_76	GPIOMUX_INTRTR0_OUTP_9	77
R5FSS0_INTRTR0_IN_77	GPIOMUX_INTRTR0_OUTP_10	78
R5FSS0_INTRTR0_IN_78	GPIOMUX_INTRTR0_OUTP_11	79
R5FSS0_INTRTR0_IN_79	GPIOMUX_INTRTR0_OUTP_12	80
R5FSS0_INTRTR0_IN_80	GPIOMUX_INTRTR0_OUTP_13	81
R5FSS0_INTRTR0_IN_81	GPIOMUX_INTRTR0_OUTP_14	82
R5FSS0_INTRTR0_IN_82	GPIOMUX_INTRTR0_OUTP_15	83
R5FSS0_INTRTR0_IN_83	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_0	84
R5FSS0_INTRTR0_IN_84	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_1	85
R5FSS0_INTRTR0_IN_85	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_2	86
R5FSS0_INTRTR0_IN_86	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_3	87
R5FSS0_INTRTR0_IN_87	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_4	88
R5FSS0_INTRTR0_IN_88	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_5	89
R5FSS0_INTRTR0_IN_89	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_6	90
R5FSS0_INTRTR0_IN_90	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_7	91
R5FSS0_INTRTR0_IN_91	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_8	92
R5FSS0_INTRTR0_IN_92	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_9	93
R5FSS0_INTRTR0_IN_93	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_10	94

**Table 9-51. R5FSS0\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	R5FSS0_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
R5FSS0_INTRTR0_IN_94	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_11	95
R5FSS0_INTRTR0_IN_95	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_12	96
R5FSS0_INTRTR0_IN_96	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_13	97
R5FSS0_INTRTR0_IN_97	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_14	98
R5FSS0_INTRTR0_IN_98	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_15	99
R5FSS0_INTRTR0_IN_99	CMPEVENT_INTRTR0_OUTP_16	100
R5FSS0_INTRTR0_IN_100	CMPEVENT_INTRTR0_OUTP_17	101
R5FSS0_INTRTR0_IN_101	CMPEVENT_INTRTR0_OUTP_18	102
R5FSS0_INTRTR0_IN_102	CMPEVENT_INTRTR0_OUTP_19	103
R5FSS0_INTRTR0_IN_103	CMPEVENT_INTRTR0_OUTP_20	104
R5FSS0_INTRTR0_IN_104	CMPEVENT_INTRTR0_OUTP_21	105
R5FSS0_INTRTR0_IN_105	CMPEVENT_INTRTR0_OUTP_22	106
R5FSS0_INTRTR0_IN_106	CMPEVENT_INTRTR0_OUTP_23	107
R5FSS0_INTRTR0_IN_107	CMPEVENT_INTRTR0_OUTP_24	108
R5FSS0_INTRTR0_IN_108	CMPEVENT_INTRTR0_OUTP_25	109
R5FSS0_INTRTR0_IN_109	CMPEVENT_INTRTR0_OUTP_26	110
R5FSS0_INTRTR0_IN_110	CMPEVENT_INTRTR0_OUTP_27	111
R5FSS0_INTRTR0_IN_111	CMPEVENT_INTRTR0_OUTP_28	112
R5FSS0_INTRTR0_IN_112	CMPEVENT_INTRTR0_OUTP_29	113
R5FSS0_INTRTR0_IN_113	CMPEVENT_INTRTR0_OUTP_30	114
R5FSS0_INTRTR0_IN_114	CMPEVENT_INTRTR0_OUTP_31	115
R5FSS0_INTRTR0_IN_115	MCU_ADC0_GEN_LEVEL_0	116
R5FSS0_INTRTR0_IN_116	MCU_ADC1_GEN_LEVEL_0	117
R5FSS0_INTRTR0_IN_117	MCU_CPSW0_STAT_PEND_0	118
R5FSS0_INTRTR0_IN_118	MCU_CPSW0_MDIO_PEND_0	119
R5FSS0_INTRTR0_IN_119	MCU_CPSW0_EVNT_PEND_0	120
R5FSS0_INTRTR0_IN_120	MCU_DCC0_INTR_DONE_LEVEL_0	121
R5FSS0_INTRTR0_IN_121	MCU_DCC1_INTR_DONE_LEVEL_0	122
R5FSS0_INTRTR0_IN_122	MCU_DCC2_INTR_DONE_LEVEL_0	123
R5FSS0_INTRTR0_IN_123	MCU_TIMER0_INTR_PEND_0	124
R5FSS0_INTRTR0_IN_124	MCU_TIMER1_INTR_PEND_0	125
R5FSS0_INTRTR0_IN_125	MCU_TIMER2_INTR_PEND_0	126
R5FSS0_INTRTR0_IN_126	MCU_TIMER3_INTR_PEND_0	127
R5FSS0_INTRTR0_IN_127	MCU_TIMER4_INTR_PEND_0	128
R5FSS0_INTRTR0_IN_128	MCU_TIMER5_INTR_PEND_0	129
R5FSS0_INTRTR0_IN_129	MCU_TIMER6_INTR_PEND_0	130
R5FSS0_INTRTR0_IN_130	MCU_TIMER7_INTR_PEND_0	131
R5FSS0_INTRTR0_IN_131	MCU_TIMER8_INTR_PEND_0	132
R5FSS0_INTRTR0_IN_132	MCU_TIMER9_INTR_PEND_0	133
R5FSS0_INTRTR0_IN_133	MCU_I2C0_POINTRPEND_0	134
R5FSS0_INTRTR0_IN_134	MCU_I2C1_POINTRPEND_0	135
R5FSS0_INTRTR0_IN_135	MCU_MCSPI0_INTR_SPI_0	136
R5FSS0_INTRTR0_IN_136	MCU_MCSPI1_INTR_SPI_0	137
R5FSS0_INTRTR0_IN_137	MCU_MCSPI2_INTR_SPI_0	138

**Table 9-51. R5FSS0\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	R5FSS0_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
R5FSS0_INTRTR0_IN_138	MCU_UART0_USART_IRQ_0	139
R5FSS0_INTRTR0_IN_139	MCU_I3C0_I3C_INT_0	140
R5FSS0_INTRTR0_IN_140	MCU_I3C1_I3C_INT_0	141
R5FSS0_INTRTR0_IN_141	MCU_FSS0_OSPI_0_OSPI_LVL_INTR_0	142
R5FSS0_INTRTR0_IN_142	MCU_FSS0_OSPI_1_OSPI_LVL_INTR_0	143
R5FSS0_INTRTR0_IN_143	MCU_FSS0_HYPERBUS1P0_0_HPB_INTR_0	144
R5FSS0_INTRTR0_IN_144	MCU_FSS0_FSAS_0_OTFE_INTR_ERR_PEND_0	145
R5FSS0_INTRTR0_IN_145	MCU_FSS0_FSAS_0_ECC_INTR_ERR_PEND_0	146
R5FSS0_INTRTR0_IN_146	MCU_SA2_UL0_SA_UL_PKA_0	147
R5FSS0_INTRTR0_IN_147	MCU_SA2_UL0_SA_UL_TRNG_0	148
R5FSS0_INTRTR0_IN_148	MCU_ESM0_ESM_INT_LOW_LVL_0	149
R5FSS0_INTRTR0_IN_149	MCU_ESM0_ESM_INT_HI_LVL_0	150
R5FSS0_INTRTR0_IN_150	MCU_ESM0_ESM_INT_CFG_LVL_0	151
R5FSS0_INTRTR0_IN_151	MCU_CTRL_MMR0_ACCESS_ERR_0	152
R5FSS0_INTRTR0_IN_152	MCU_CBASS0_LPSC_MCU_COMMON_ERR_INTR_0	153
R5FSS0_INTRTR0_IN_153	GLUELOGIC_DBG_CBASS_INTR_OR_GLUE_DBG_CBASS_AGG_ERR_INTR_0	154
R5FSS0_INTRTR0_IN_154	GLUELOGIC_FW_CBASS_INTR_OR_GLUE_FW_CBASS_AGG_ERR_INTR_0	155
R5FSS0_INTRTR0_IN_156	WKUP_CBASS0_LPSC_WKUP_COMMON_ERR_INTR_0	157
R5FSS0_INTRTR0_IN_157	WKUP_I2C0_POINTRPEND_0	158
R5FSS0_INTRTR0_IN_158	WKUP_UART0_USART_IRQ_0	159
R5FSS0_INTRTR0_IN_159	WKUP_GPIOMUX_INTRTR0_OUTP_16	160
R5FSS0_INTRTR0_IN_160	WKUP_GPIOMUX_INTRTR0_OUTP_17	161
R5FSS0_INTRTR0_IN_161	WKUP_GPIOMUX_INTRTR0_OUTP_18	162
R5FSS0_INTRTR0_IN_162	WKUP_GPIOMUX_INTRTR0_OUTP_19	163
R5FSS0_INTRTR0_IN_163	WKUP_GPIOMUX_INTRTR0_OUTP_20	164
R5FSS0_INTRTR0_IN_164	WKUP_GPIOMUX_INTRTR0_OUTP_21	165
R5FSS0_INTRTR0_IN_165	WKUP_GPIOMUX_INTRTR0_OUTP_22	166
R5FSS0_INTRTR0_IN_166	WKUP_GPIOMUX_INTRTR0_OUTP_23	167
R5FSS0_INTRTR0_IN_167	WKUP_GPIOMUX_INTRTR0_OUTP_24	168
R5FSS0_INTRTR0_IN_168	WKUP_GPIOMUX_INTRTR0_OUTP_25	169
R5FSS0_INTRTR0_IN_169	WKUP_GPIOMUX_INTRTR0_OUTP_26	170
R5FSS0_INTRTR0_IN_170	WKUP_GPIOMUX_INTRTR0_OUTP_27	171
R5FSS0_INTRTR0_IN_171	WKUP_GPIOMUX_INTRTR0_OUTP_28	172
R5FSS0_INTRTR0_IN_172	WKUP_GPIOMUX_INTRTR0_OUTP_29	173
R5FSS0_INTRTR0_IN_173	WKUP_GPIOMUX_INTRTR0_OUTP_30	174
R5FSS0_INTRTR0_IN_174	WKUP_GPIOMUX_INTRTR0_OUTP_31	175
R5FSS0_INTRTR0_IN_175	CCDEBUGSS0_AQCMPINTR_LEVEL_0	176
R5FSS0_INTRTR0_IN_176	DEBUGSS0_AQCMPINTR_LEVEL_0	177
R5FSS0_INTRTR0_IN_177	DEBUGSS1_AQCMPINTR_LEVEL_0	178
R5FSS0_INTRTR0_IN_178	C66DEBUGSS0_AQCMPINTR_LEVEL_0	179
R5FSS0_INTRTR0_IN_179	DEBUGSS0_CTM_LEVEL_0	180
R5FSS0_INTRTR0_IN_180	DEBUGSS1_CTM_LEVEL_0	181
R5FSS0_INTRTR0_IN_181	C66DEBUGSS1_AQCMPINTR_LEVEL_0	182

**Table 9-51. R5FSS0\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	R5FSS0_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
R5FSS0_INTRTR0_IN_182	I2C2_POINTRPEND_0	183
R5FSS0_INTRTR0_IN_183	I2C3_POINTRPEND_0	184
R5FSS0_INTRTR0_IN_184	I2C4_POINTRPEND_0	185
R5FSS0_INTRTR0_IN_185	I2C5_POINTRPEND_0	186
R5FSS0_INTRTR0_IN_186	I2C6_POINTRPEND_0	187
R5FSS0_INTRTR0_IN_187	UART3_USART_IRQ_0	188
R5FSS0_INTRTR0_IN_188	UART4_USART_IRQ_0	189
R5FSS0_INTRTR0_IN_189	UART5_USART_IRQ_0	190
R5FSS0_INTRTR0_IN_190	UART6_USART_IRQ_0	191
R5FSS0_INTRTR0_IN_191	UART7_USART_IRQ_0	192
R5FSS0_INTRTR0_IN_192	UART8_USART_IRQ_0	193
R5FSS0_INTRTR0_IN_193	UART9_USART_IRQ_0	194
R5FSS0_INTRTR0_IN_194	MCSP12_INTR_SPI_0	195
R5FSS0_INTRTR0_IN_195	MCSP13_INTR_SPI_0	196
R5FSS0_INTRTR0_IN_196	MCSP14_INTR_SPI_0	197
R5FSS0_INTRTR0_IN_197	MCSP15_INTR_SPI_0	198
R5FSS0_INTRTR0_IN_198	MCSP16_INTR_SPI_0	199
R5FSS0_INTRTR0_IN_199	MCSP17_INTR_SPI_0	200
R5FSS0_INTRTR0_IN_200	I3C0_I3C_INT_0	201
R5FSS0_INTRTR0_IN_202	AASRC0_ERR_LEVEL_0	203
R5FSS0_INTRTR0_IN_203	AASRC0_INFIFO_LEVEL_0	204
R5FSS0_INTRTR0_IN_204	AASRC0_INGROUP_LEVEL_0	205
R5FSS0_INTRTR0_IN_205	AASRC0_OUTFIFO_LEVEL_0	206
R5FSS0_INTRTR0_IN_206	AASRC0_OUTGROUP_LEVEL_0	207
R5FSS0_INTRTR0_IN_207	MCASP2_XMIT_INTR_PEND_0	208
R5FSS0_INTRTR0_IN_208	MCASP2_REC_INTR_PEND_0	209
R5FSS0_INTRTR0_IN_209	MCASP3_XMIT_INTR_PEND_0	210
R5FSS0_INTRTR0_IN_210	MCASP3_REC_INTR_PEND_0	211
R5FSS0_INTRTR0_IN_211	MCASP4_XMIT_INTR_PEND_0	212
R5FSS0_INTRTR0_IN_212	MCASP4_REC_INTR_PEND_0	213
R5FSS0_INTRTR0_IN_213	MCASP5_XMIT_INTR_PEND_0	214
R5FSS0_INTRTR0_IN_214	MCASP5_REC_INTR_PEND_0	215
R5FSS0_INTRTR0_IN_215	MCASP6_XMIT_INTR_PEND_0	216
R5FSS0_INTRTR0_IN_216	MCASP6_REC_INTR_PEND_0	217
R5FSS0_INTRTR0_IN_217	MCASP7_XMIT_INTR_PEND_0	218
R5FSS0_INTRTR0_IN_218	MCASP7_REC_INTR_PEND_0	219
R5FSS0_INTRTR0_IN_219	MCASP8_XMIT_INTR_PEND_0	220
R5FSS0_INTRTR0_IN_220	MCASP8_REC_INTR_PEND_0	221
R5FSS0_INTRTR0_IN_221	MCASP9_XMIT_INTR_PEND_0	222
R5FSS0_INTRTR0_IN_222	MCASP9_REC_INTR_PEND_0	223
R5FSS0_INTRTR0_IN_223	MCASP10_XMIT_INTR_PEND_0	224
R5FSS0_INTRTR0_IN_224	MCASP10_REC_INTR_PEND_0	225
R5FSS0_INTRTR0_IN_225	MCASP11_XMIT_INTR_PEND_0	226
R5FSS0_INTRTR0_IN_226	MCASP11_REC_INTR_PEND_0	228



**Table 9-51. R5FSS0\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	R5FSS0_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
R5FSS0_INTRTR0_IN_228	GPMC0_GPMC_SINTERRUPT_0	229
R5FSS0_INTRTR0_IN_229	ELM0_ELM_POROCPSINTERRUPT_LVL_0	230
R5FSS0_INTRTR0_IN_230	USB0_OTGIRQ_0	231
R5FSS0_INTRTR0_IN_231	USB0_IRQ_0	232
R5FSS0_INTRTR0_IN_232	USB0_IRQ_1	233
R5FSS0_INTRTR0_IN_233	USB0_IRQ_2	234
R5FSS0_INTRTR0_IN_234	USB0_IRQ_3	235
R5FSS0_INTRTR0_IN_235	USB0_IRQ_4	236
R5FSS0_INTRTR0_IN_236	USB0_IRQ_5	237
R5FSS0_INTRTR0_IN_237	USB0_IRQ_6	238
R5FSS0_INTRTR0_IN_238	USB0_IRQ_7	239
R5FSS0_INTRTR0_IN_239	TIMER0_INTR_PEND_0	240
R5FSS0_INTRTR0_IN_240	TIMER1_INTR_PEND_0	241
R5FSS0_INTRTR0_IN_241	TIMER2_INTR_PEND_0	242
R5FSS0_INTRTR0_IN_242	TIMER3_INTR_PEND_0	243
R5FSS0_INTRTR0_IN_243	TIMER4_INTR_PEND_0	244
R5FSS0_INTRTR0_IN_244	TIMER5_INTR_PEND_0	245
R5FSS0_INTRTR0_IN_245	TIMER6_INTR_PEND_0	246
R5FSS0_INTRTR0_IN_246	TIMER7_INTR_PEND_0	247
R5FSS0_INTRTR0_IN_247	TIMER8_INTR_PEND_0	248
R5FSS0_INTRTR0_IN_248	TIMER9_INTR_PEND_0	249
R5FSS0_INTRTR0_IN_249	TIMER10_INTR_PEND_0	250
R5FSS0_INTRTR0_IN_250	TIMER11_INTR_PEND_0	251
R5FSS0_INTRTR0_IN_251	PCIE1_PCIE_LEGACY_PULSE_0	252
R5FSS0_INTRTR0_IN_252	PCIE1_PCIE_DOWNSTREAM_PULSE_0	253
R5FSS0_INTRTR0_IN_253	PCIE1_PCIE_FLR_PULSE_0	254
R5FSS0_INTRTR0_IN_254	PCIE1_PCIE_PHY_LEVEL_0	255
R5FSS0_INTRTR0_IN_255	PCIE1_PCIE_LOCAL_LEVEL_0	256
R5FSS0_INTRTR0_IN_256	PCIE1_PCIE_ERROR_PULSE_0	257
R5FSS0_INTRTR0_IN_257	PCIE1_PCIE_LINK_STATE_PULSE_0	258
R5FSS0_INTRTR0_IN_258	PCIE1_PCIE_PWR_STATE_PULSE_0	259
R5FSS0_INTRTR0_IN_259	PCIE1_PCIE_PTM_VALID_PULSE_0	260
R5FSS0_INTRTR0_IN_260	PCIE1_PCIE_HOT_RESET_PULSE_0	261
R5FSS0_INTRTR0_IN_261	PCIE1_PCIE_CPTS_PEND_0	262
R5FSS0_INTRTR0_IN_263	DDR0_DDRSS_CONTROLLER_0	264
R5FSS0_INTRTR0_IN_264	DDR0_DDRSS_V2A_OTHER_ERR_LVL_0	265
R5FSS0_INTRTR0_IN_265	DDR0_DDRSS_HS_PHY_GLOBAL_ERROR_0	266
R5FSS0_INTRTR0_IN_266	DDR0_DDRSS_PLL_FREQ_CHANGE_REQ_0	267
R5FSS0_INTRTR0_IN_267	CSI_TX_IF0_CSI_INTERRUPT_0	268
R5FSS0_INTRTR0_IN_268	CSI_TX_IF0_CSI_LEVEL_0	269
R5FSS0_INTRTR0_IN_275	VPFE0_CCDC_INTR_PEND_0	276
R5FSS0_INTRTR0_IN_276	VPFE0_RAT_EXP_INTR_0	277
R5FSS0_INTRTR0_IN_278	WKUP_DMSC0_RAT_0_EXP_INTR_0	279
R5FSS0_INTRTR0_IN_279	DCC0_INTR_DONE_LEVEL_0	280



**Table 9-51. R5FSS0\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	R5FSS0_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
R5FSS0_INTRTR0_IN_280	DCC1_INTR_DONE_LEVEL_0	281
R5FSS0_INTRTR0_IN_281	DCC2_INTR_DONE_LEVEL_0	282
R5FSS0_INTRTR0_IN_282	DCC3_INTR_DONE_LEVEL_0	283
R5FSS0_INTRTR0_IN_283	DCC4_INTR_DONE_LEVEL_0	284
R5FSS0_INTRTR0_IN_284	DCC5_INTR_DONE_LEVEL_0	285
R5FSS0_INTRTR0_IN_285	DCC6_INTR_DONE_LEVEL_0	286
R5FSS0_INTRTR0_IN_286	DCC7_INTR_DONE_LEVEL_0	287
R5FSS0_INTRTR0_IN_287	CMPEVENT_INTRTR0_OUTP_0	288
R5FSS0_INTRTR0_IN_288	CMPEVENT_INTRTR0_OUTP_1	289
R5FSS0_INTRTR0_IN_289	CMPEVENT_INTRTR0_OUTP_2	290
R5FSS0_INTRTR0_IN_290	CMPEVENT_INTRTR0_OUTP_3	291
R5FSS0_INTRTR0_IN_291	CMPEVENT_INTRTR0_OUTP_4	292
R5FSS0_INTRTR0_IN_292	CMPEVENT_INTRTR0_OUTP_5	293
R5FSS0_INTRTR0_IN_293	CMPEVENT_INTRTR0_OUTP_6	294
R5FSS0_INTRTR0_IN_294	CMPEVENT_INTRTR0_OUTP_7	295
R5FSS0_INTRTR0_IN_295	CMPEVENT_INTRTR0_OUTP_8	296
R5FSS0_INTRTR0_IN_296	CMPEVENT_INTRTR0_OUTP_9	297
R5FSS0_INTRTR0_IN_297	CMPEVENT_INTRTR0_OUTP_10	298
R5FSS0_INTRTR0_IN_298	CMPEVENT_INTRTR0_OUTP_11	299
R5FSS0_INTRTR0_IN_299	CMPEVENT_INTRTR0_OUTP_12	300
R5FSS0_INTRTR0_IN_300	CMPEVENT_INTRTR0_OUTP_13	301
R5FSS0_INTRTR0_IN_301	CMPEVENT_INTRTR0_OUTP_14	302
R5FSS0_INTRTR0_IN_302	CMPEVENT_INTRTR0_OUTP_15	303
R5FSS0_INTRTR0_IN_303	DCC8_INTR_DONE_LEVEL_0	304
R5FSS0_INTRTR0_IN_304	DCC9_INTR_DONE_LEVEL_0	305
R5FSS0_INTRTR0_IN_305	DCC10_INTR_DONE_LEVEL_0	306
R5FSS0_INTRTR0_IN_306	DCC11_INTR_DONE_LEVEL_0	307
R5FSS0_INTRTR0_IN_307	DCC12_INTR_DONE_LEVEL_0	308
R5FSS0_INTRTR0_IN_309	MMCS11_EMMCSDSS_INTR_0	310
R5FSS0_INTRTR0_IN_310	MMCS12_EMMCSDSS_INTR_0	311
R5FSS0_INTRTR0_IN_311	UFS0_UFS_INTR_0	312
R5FSS0_INTRTR0_IN_313	SA2_UL0_SA_UL_PKA_0	314
R5FSS0_INTRTR0_IN_314	SA2_UL0_SA_UL_TRNG_0	315
R5FSS0_INTRTR0_IN_315	ECAP0_ECAP_INT_0	316
R5FSS0_INTRTR0_IN_316	ECAP1_ECAP_INT_0	317
R5FSS0_INTRTR0_IN_317	ECAP2_ECAP_INT_0	318
R5FSS0_INTRTR0_IN_318	GLUELOGIC_SOCA_INT_GLUE_SOCA_INT_0	319
R5FSS0_INTRTR0_IN_319	GLUELOGIC_SOCB_INT_GLUE_SOCB_INT_0	320
R5FSS0_INTRTR0_IN_322	USB0_HOST_SYSTEM_ERROR_0	323
R5FSS0_INTRTR0_IN_324	CBASS_INFRA0_DEFAULT_ERR_INTR_0	325
R5FSS0_INTRTR0_IN_326	CTRL_MMR0_ACCESS_ERR_0	327
R5FSS0_INTRTR0_IN_327	WKUP_VTM0_THERM_LVL_GT_TH1_INTR_0	328
R5FSS0_INTRTR0_IN_328	WKUP_VTM0_THERM_LVL_GT_TH2_INTR_0	329
R5FSS0_INTRTR0_IN_329	WKUP_VTM0_THERM_LVL_LT_TH0_INTR_0	330

**Table 9-51. R5FSS0\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	R5FSS0_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
R5FSS0_INTRTR0_IN_331	COMPUTE_CLUSTER0_GIC_OUTPUT_WAKER_GIC_PWR0_WAKE_REQUEST_0	332
R5FSS0_INTRTR0_IN_332	COMPUTE_CLUSTER0_GIC_OUTPUT_WAKER_GIC_PWR0_WAKE_REQUEST_1	333
R5FSS0_INTRTR0_IN_335	MCU_MCAN0_MCANSS_MCAN_LVL_INT_0	336
R5FSS0_INTRTR0_IN_336	MCU_MCAN0_MCANSS_MCAN_LVL_INT_1	337
R5FSS0_INTRTR0_IN_337	MCU_MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	338
R5FSS0_INTRTR0_IN_338	MCU_MCAN1_MCANSS_MCAN_LVL_INT_0	339
R5FSS0_INTRTR0_IN_339	MCU_MCAN1_MCANSS_MCAN_LVL_INT_1	340
R5FSS0_INTRTR0_IN_340	MCU_MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	341

- (1) Do not use that setting. As it is the default one, it is strongly recommended to set the field to another value before enabling interrupt router operation.

#### 9.4.3.7 R5FSS1\_INTRTR0 Interrupt Map

Table 9-52 lists all the interrupts that are mapped to the R5FSS1\_INTRTR0 inputs, along with the corresponding register programming values used for mux control. Any R5FSS1\_INTRTR0 interrupt input that is not listed in this table should be considered as unused.

**Table 9-52. R5FSS1\_INTRTR0 Interrupt Map**

Interrupt Input Line	Peripheral Interrupt	R5FSS1_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
RESERVED	RESERVED	0 <sup>(1)</sup>
R5FSS1_INTRTR0_IN_0	USB1_OTGIRQ_0	1
R5FSS1_INTRTR0_IN_1	USB1_IRQ_0	2
R5FSS1_INTRTR0_IN_2	USB1_IRQ_1	3
R5FSS1_INTRTR0_IN_3	USB1_IRQ_2	4
R5FSS1_INTRTR0_IN_4	USB1_IRQ_3	5
R5FSS1_INTRTR0_IN_5	USB1_IRQ_4	6
R5FSS1_INTRTR0_IN_6	USB1_IRQ_5	7
R5FSS1_INTRTR0_IN_7	USB1_IRQ_6	8
R5FSS1_INTRTR0_IN_8	USB1_IRQ_7	9
R5FSS1_INTRTR0_IN_18	PCIE2_PCIE_LEGACY_PULSE_0	19
R5FSS1_INTRTR0_IN_19	PCIE2_PCIE_DOWNSTREAM_PULSE_0	20
R5FSS1_INTRTR0_IN_20	PCIE2_PCIE_FLR_PULSE_0	21
R5FSS1_INTRTR0_IN_21	PCIE2_PCIE_PHY_LEVEL_0	22
R5FSS1_INTRTR0_IN_22	PCIE2_PCIE_LOCAL_LEVEL_0	23
R5FSS1_INTRTR0_IN_23	PCIE2_PCIE_ERROR_PULSE_0	24
R5FSS1_INTRTR0_IN_24	PCIE2_PCIE_LINK_STATE_PULSE_0	25
R5FSS1_INTRTR0_IN_25	PCIE2_PCIE_PWR_STATE_PULSE_0	26
R5FSS1_INTRTR0_IN_26	PCIE2_PCIE_PTM_VALID_PULSE_0	27
R5FSS1_INTRTR0_IN_27	PCIE2_PCIE_HOT_RESET_PULSE_0	28
R5FSS1_INTRTR0_IN_28	PCIE2_PCIE_CPTS_PEND_0	29
R5FSS1_INTRTR0_IN_29	PRU_ICSSG0_PR1_HOST_INTR_PEND_0	30
R5FSS1_INTRTR0_IN_30	PRU_ICSSG0_PR1_HOST_INTR_PEND_1	31
R5FSS1_INTRTR0_IN_31	PRU_ICSSG0_PR1_HOST_INTR_PEND_2	32
R5FSS1_INTRTR0_IN_32	PRU_ICSSG0_PR1_HOST_INTR_PEND_3	33
R5FSS1_INTRTR0_IN_33	PRU_ICSSG0_PR1_HOST_INTR_PEND_4	34
R5FSS1_INTRTR0_IN_34	PRU_ICSSG0_PR1_HOST_INTR_PEND_5	35
R5FSS1_INTRTR0_IN_35	PRU_ICSSG0_PR1_HOST_INTR_PEND_6	36
R5FSS1_INTRTR0_IN_36	PRU_ICSSG0_PR1_HOST_INTR_PEND_7	37
R5FSS1_INTRTR0_IN_37	PRU_ICSSG1_PR1_HOST_INTR_PEND_0	38
R5FSS1_INTRTR0_IN_38	PRU_ICSSG1_PR1_HOST_INTR_PEND_1	39
R5FSS1_INTRTR0_IN_39	PRU_ICSSG1_PR1_HOST_INTR_PEND_2	40
R5FSS1_INTRTR0_IN_40	PRU_ICSSG1_PR1_HOST_INTR_PEND_3	41
R5FSS1_INTRTR0_IN_41	PRU_ICSSG1_PR1_HOST_INTR_PEND_4	42
R5FSS1_INTRTR0_IN_42	PRU_ICSSG1_PR1_HOST_INTR_PEND_5	43
R5FSS1_INTRTR0_IN_43	PRU_ICSSG1_PR1_HOST_INTR_PEND_6	44
R5FSS1_INTRTR0_IN_44	PRU_ICSSG1_PR1_HOST_INTR_PEND_7	45
R5FSS1_INTRTR0_IN_45	PRU_ICSSG0_PR1_TX_SOF_INTR_REQ_0	46
R5FSS1_INTRTR0_IN_46	PRU_ICSSG0_PR1_TX_SOF_INTR_REQ_1	47
R5FSS1_INTRTR0_IN_47	PRU_ICSSG0_PR1_RX_SOF_INTR_REQ_0	48

**Table 9-52. R5FSS1\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	R5FSS1_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
R5FSS1_INTRTR0_IN_48	PRU_ICSSG0_PR1_RX_SOF_INTR_REQ_1	49
R5FSS1_INTRTR0_IN_49	PRU_ICSSG1_PR1_TX_SOF_INTR_REQ_0	50
R5FSS1_INTRTR0_IN_50	PRU_ICSSG1_PR1_TX_SOF_INTR_REQ_1	51
R5FSS1_INTRTR0_IN_51	PRU_ICSSG1_PR1_RX_SOF_INTR_REQ_0	52
R5FSS1_INTRTR0_IN_52	PRU_ICSSG1_PR1_RX_SOF_INTR_REQ_1	53
R5FSS1_INTRTR0_IN_53	PCIE3_PCIE_LEGACY_PULSE_0	54
R5FSS1_INTRTR0_IN_54	PCIE3_PCIE_DOWNSTREAM_PULSE_0	55
R5FSS1_INTRTR0_IN_55	PCIE3_PCIE_FLR_PULSE_0	56
R5FSS1_INTRTR0_IN_56	PCIE3_PCIE_PHY_LEVEL_0	57
R5FSS1_INTRTR0_IN_57	PCIE3_PCIE_LOCAL_LEVEL_0	58
R5FSS1_INTRTR0_IN_58	PCIE3_PCIE_ERROR_PULSE_0	59
R5FSS1_INTRTR0_IN_59	PCIE3_PCIE_LINK_STATE_PULSE_0	60
R5FSS1_INTRTR0_IN_60	PCIE3_PCIE_PWR_STATE_PULSE_0	61
R5FSS1_INTRTR0_IN_61	PCIE3_PCIE_PTM_VALID_PULSE_0	62
R5FSS1_INTRTR0_IN_62	PCIE3_PCIE_HOT_RESET_PULSE_0	63
R5FSS1_INTRTR0_IN_63	PCIE3_PCIE_CPTS_PEND_0	64
R5FSS1_INTRTR0_IN_64	USB1_HOST_SYSTEM_ERROR_0	65
R5FSS1_INTRTR0_IN_67	GPIOMUX_INTRTR0_OUTP_0	68
R5FSS1_INTRTR0_IN_68	GPIOMUX_INTRTR0_OUTP_1	69
R5FSS1_INTRTR0_IN_69	GPIOMUX_INTRTR0_OUTP_2	70
R5FSS1_INTRTR0_IN_70	GPIOMUX_INTRTR0_OUTP_3	71
R5FSS1_INTRTR0_IN_71	GPIOMUX_INTRTR0_OUTP_4	72
R5FSS1_INTRTR0_IN_72	GPIOMUX_INTRTR0_OUTP_5	73
R5FSS1_INTRTR0_IN_73	GPIOMUX_INTRTR0_OUTP_6	74
R5FSS1_INTRTR0_IN_74	GPIOMUX_INTRTR0_OUTP_7	75
R5FSS1_INTRTR0_IN_75	GPIOMUX_INTRTR0_OUTP_8	76
R5FSS1_INTRTR0_IN_76	GPIOMUX_INTRTR0_OUTP_9	77
R5FSS1_INTRTR0_IN_77	GPIOMUX_INTRTR0_OUTP_10	78
R5FSS1_INTRTR0_IN_78	GPIOMUX_INTRTR0_OUTP_11	79
R5FSS1_INTRTR0_IN_79	GPIOMUX_INTRTR0_OUTP_12	80
R5FSS1_INTRTR0_IN_80	GPIOMUX_INTRTR0_OUTP_13	81
R5FSS1_INTRTR0_IN_81	GPIOMUX_INTRTR0_OUTP_14	82
R5FSS1_INTRTR0_IN_82	GPIOMUX_INTRTR0_OUTP_15	83
R5FSS1_INTRTR0_IN_83	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_0	84
R5FSS1_INTRTR0_IN_84	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_1	85
R5FSS1_INTRTR0_IN_85	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_2	86
R5FSS1_INTRTR0_IN_86	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_3	87
R5FSS1_INTRTR0_IN_87	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_4	88
R5FSS1_INTRTR0_IN_88	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_5	89
R5FSS1_INTRTR0_IN_89	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_6	90
R5FSS1_INTRTR0_IN_90	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_7	91
R5FSS1_INTRTR0_IN_91	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_8	92
R5FSS1_INTRTR0_IN_92	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_9	93
R5FSS1_INTRTR0_IN_93	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_10	94

**Table 9-52. R5FSS1\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	R5FSS1_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
R5FSS1_INTRTR0_IN_94	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_11	95
R5FSS1_INTRTR0_IN_95	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_12	96
R5FSS1_INTRTR0_IN_96	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_13	97
R5FSS1_INTRTR0_IN_97	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_14	98
R5FSS1_INTRTR0_IN_98	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_15	99
R5FSS1_INTRTR0_IN_99	CMPEVENT_INTRTR0_OUTP_16	100
R5FSS1_INTRTR0_IN_100	CMPEVENT_INTRTR0_OUTP_17	101
R5FSS1_INTRTR0_IN_101	CMPEVENT_INTRTR0_OUTP_18	102
R5FSS1_INTRTR0_IN_102	CMPEVENT_INTRTR0_OUTP_19	103
R5FSS1_INTRTR0_IN_103	CMPEVENT_INTRTR0_OUTP_20	104
R5FSS1_INTRTR0_IN_104	CMPEVENT_INTRTR0_OUTP_21	105
R5FSS1_INTRTR0_IN_105	CMPEVENT_INTRTR0_OUTP_22	106
R5FSS1_INTRTR0_IN_106	CMPEVENT_INTRTR0_OUTP_23	107
R5FSS1_INTRTR0_IN_107	CMPEVENT_INTRTR0_OUTP_24	108
R5FSS1_INTRTR0_IN_108	CMPEVENT_INTRTR0_OUTP_25	109
R5FSS1_INTRTR0_IN_109	CMPEVENT_INTRTR0_OUTP_26	110
R5FSS1_INTRTR0_IN_110	CMPEVENT_INTRTR0_OUTP_27	111
R5FSS1_INTRTR0_IN_111	CMPEVENT_INTRTR0_OUTP_28	112
R5FSS1_INTRTR0_IN_112	CMPEVENT_INTRTR0_OUTP_29	113
R5FSS1_INTRTR0_IN_113	CMPEVENT_INTRTR0_OUTP_30	114
R5FSS1_INTRTR0_IN_114	CMPEVENT_INTRTR0_OUTP_31	115
R5FSS1_INTRTR0_IN_115	MCU_ADC0_GEN_LEVEL_0	116
R5FSS1_INTRTR0_IN_116	MCU_ADC1_GEN_LEVEL_0	117
R5FSS1_INTRTR0_IN_117	MCU_CPSW0_STAT_PEND_0	118
R5FSS1_INTRTR0_IN_118	MCU_CPSW0_MDIO_PEND_0	119
R5FSS1_INTRTR0_IN_119	MCU_CPSW0_EVNT_PEND_0	120
R5FSS1_INTRTR0_IN_120	MCU_DCC0_INTR_DONE_LEVEL_0	121
R5FSS1_INTRTR0_IN_121	MCU_DCC1_INTR_DONE_LEVEL_0	122
R5FSS1_INTRTR0_IN_122	MCU_DCC2_INTR_DONE_LEVEL_0	123
R5FSS1_INTRTR0_IN_123	MCU_TIMER0_INTR_PEND_0	124
R5FSS1_INTRTR0_IN_124	MCU_TIMER1_INTR_PEND_0	125
R5FSS1_INTRTR0_IN_125	MCU_TIMER2_INTR_PEND_0	126
R5FSS1_INTRTR0_IN_126	MCU_TIMER3_INTR_PEND_0	127
R5FSS1_INTRTR0_IN_127	MCU_TIMER4_INTR_PEND_0	128
R5FSS1_INTRTR0_IN_128	MCU_TIMER5_INTR_PEND_0	129
R5FSS1_INTRTR0_IN_129	MCU_TIMER6_INTR_PEND_0	130
R5FSS1_INTRTR0_IN_130	MCU_TIMER7_INTR_PEND_0	131
R5FSS1_INTRTR0_IN_131	MCU_TIMER8_INTR_PEND_0	132
R5FSS1_INTRTR0_IN_132	MCU_TIMER9_INTR_PEND_0	133
R5FSS1_INTRTR0_IN_133	MCU_I2C0_POINTRPEND_0	134
R5FSS1_INTRTR0_IN_134	MCU_I2C1_POINTRPEND_0	135
R5FSS1_INTRTR0_IN_135	MCU_MCSPI0_INTR_SPI_0	136
R5FSS1_INTRTR0_IN_136	MCU_MCSPI1_INTR_SPI_0	137
R5FSS1_INTRTR0_IN_137	MCU_MCSPI2_INTR_SPI_0	138

**Table 9-52. R5FSS1\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	R5FSS1_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
R5FSS1_INTRTR0_IN_138	MCU_UART0_USART_IRQ_0	139
R5FSS1_INTRTR0_IN_139	MCU_I3C0_I3C_INT_0	140
R5FSS1_INTRTR0_IN_140	MCU_I3C1_I3C_INT_0	141
R5FSS1_INTRTR0_IN_141	MCU_FSS0_OSPI_0_OSPI_LVL_INTR_0	142
R5FSS1_INTRTR0_IN_142	MCU_FSS0_OSPI_1_OSPI_LVL_INTR_0	143
R5FSS1_INTRTR0_IN_143	MCU_FSS0_HYPERBUS1P0_0_HPB_INTR_0	144
R5FSS1_INTRTR0_IN_144	MCU_FSS0_FSAS_0_OTFE_INTR_ERR_PEND_0	145
R5FSS1_INTRTR0_IN_145	MCU_FSS0_FSAS_0_ECC_INTR_ERR_PEND_0	146
R5FSS1_INTRTR0_IN_146	MCU_SA2_UL0_SA_UL_PKA_0	147
R5FSS1_INTRTR0_IN_147	MCU_SA2_UL0_SA_UL_TRNG_0	148
R5FSS1_INTRTR0_IN_148	MCU_ESM0_ESM_INT_LOW_LVL_0	149
R5FSS1_INTRTR0_IN_149	MCU_ESM0_ESM_INT_HI_LVL_0	150
R5FSS1_INTRTR0_IN_150	MCU_ESM0_ESM_INT_CFG_LVL_0	151
R5FSS1_INTRTR0_IN_151	MCU_CTRL_MMR0_ACCESS_ERR_0	152
R5FSS1_INTRTR0_IN_152	MCU_CBASS0_LPSC_MCU_COMMON_ERR_INTR_0	153
R5FSS1_INTRTR0_IN_153	GLUELOGIC_DBG_CBASS_INTR_OR_GLUE_DBG_CBASS_AGG_ERR_INTR_0	154
R5FSS1_INTRTR0_IN_154	GLUELOGIC_FW_CBASS_INTR_OR_GLUE_FW_CBASS_AGG_ERR_INTR_0	155
R5FSS1_INTRTR0_IN_156	WKUP_CBASS0_LPSC_WKUP_COMMON_ERR_INTR_0	157
R5FSS1_INTRTR0_IN_157	WKUP_I2C0_POINTRPEND_0	158
R5FSS1_INTRTR0_IN_158	WKUP_UART0_USART_IRQ_0	159
R5FSS1_INTRTR0_IN_159	WKUP_GPIOMUX_INTRTR0_OUTP_16	160
R5FSS1_INTRTR0_IN_160	WKUP_GPIOMUX_INTRTR0_OUTP_17	161
R5FSS1_INTRTR0_IN_161	WKUP_GPIOMUX_INTRTR0_OUTP_18	162
R5FSS1_INTRTR0_IN_162	WKUP_GPIOMUX_INTRTR0_OUTP_19	163
R5FSS1_INTRTR0_IN_163	WKUP_GPIOMUX_INTRTR0_OUTP_20	164
R5FSS1_INTRTR0_IN_164	WKUP_GPIOMUX_INTRTR0_OUTP_21	165
R5FSS1_INTRTR0_IN_165	WKUP_GPIOMUX_INTRTR0_OUTP_22	166
R5FSS1_INTRTR0_IN_166	WKUP_GPIOMUX_INTRTR0_OUTP_23	167
R5FSS1_INTRTR0_IN_167	WKUP_GPIOMUX_INTRTR0_OUTP_24	168
R5FSS1_INTRTR0_IN_168	WKUP_GPIOMUX_INTRTR0_OUTP_25	169
R5FSS1_INTRTR0_IN_169	WKUP_GPIOMUX_INTRTR0_OUTP_26	170
R5FSS1_INTRTR0_IN_170	WKUP_GPIOMUX_INTRTR0_OUTP_27	171
R5FSS1_INTRTR0_IN_171	WKUP_GPIOMUX_INTRTR0_OUTP_28	172
R5FSS1_INTRTR0_IN_172	WKUP_GPIOMUX_INTRTR0_OUTP_29	173
R5FSS1_INTRTR0_IN_173	WKUP_GPIOMUX_INTRTR0_OUTP_30	174
R5FSS1_INTRTR0_IN_174	WKUP_GPIOMUX_INTRTR0_OUTP_31	175
R5FSS1_INTRTR0_IN_175	CCDEBUGSS0_AQCMPINTR_LEVEL_0	176
R5FSS1_INTRTR0_IN_176	DEBUGSS0_AQCMPINTR_LEVEL_0	177
R5FSS1_INTRTR0_IN_177	DEBUGSS1_AQCMPINTR_LEVEL_0	178
R5FSS1_INTRTR0_IN_178	C66DEBUGSS0_AQCMPINTR_LEVEL_0	179
R5FSS1_INTRTR0_IN_179	DEBUGSS0_CTM_LEVEL_0	180
R5FSS1_INTRTR0_IN_180	DEBUGSS1_CTM_LEVEL_0	181
R5FSS1_INTRTR0_IN_181	C66DEBUGSS1_AQCMPINTR_LEVEL_0	182

**Table 9-52. R5FSS1\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	R5FSS1_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
R5FSS1_INTRTR0_IN_182	I2C2_POINTRPEND_0	183
R5FSS1_INTRTR0_IN_183	I2C3_POINTRPEND_0	184
R5FSS1_INTRTR0_IN_184	I2C4_POINTRPEND_0	185
R5FSS1_INTRTR0_IN_185	I2C5_POINTRPEND_0	186
R5FSS1_INTRTR0_IN_186	I2C6_POINTRPEND_0	187
R5FSS1_INTRTR0_IN_187	UART3_USART_IRQ_0	188
R5FSS1_INTRTR0_IN_188	UART4_USART_IRQ_0	189
R5FSS1_INTRTR0_IN_189	UART5_USART_IRQ_0	190
R5FSS1_INTRTR0_IN_190	UART6_USART_IRQ_0	191
R5FSS1_INTRTR0_IN_191	UART7_USART_IRQ_0	192
R5FSS1_INTRTR0_IN_192	UART8_USART_IRQ_0	193
R5FSS1_INTRTR0_IN_193	UART9_USART_IRQ_0	194
R5FSS1_INTRTR0_IN_194	MCSP12_INTR_SPI_0	195
R5FSS1_INTRTR0_IN_195	MCSP13_INTR_SPI_0	196
R5FSS1_INTRTR0_IN_196	MCSP14_INTR_SPI_0	197
R5FSS1_INTRTR0_IN_197	MCSP15_INTR_SPI_0	198
R5FSS1_INTRTR0_IN_198	MCSP16_INTR_SPI_0	199
R5FSS1_INTRTR0_IN_199	MCSP17_INTR_SPI_0	200
R5FSS1_INTRTR0_IN_200	I3C0_I3C_INT_0	201
R5FSS1_INTRTR0_IN_202	AASRC0_ERR_LEVEL_0	203
R5FSS1_INTRTR0_IN_203	AASRC0_INFIFO_LEVEL_0	204
R5FSS1_INTRTR0_IN_204	AASRC0_INGROUP_LEVEL_0	205
R5FSS1_INTRTR0_IN_205	AASRC0_OUTFIFO_LEVEL_0	206
R5FSS1_INTRTR0_IN_206	AASRC0_OUTGROUP_LEVEL_0	207
R5FSS1_INTRTR0_IN_207	MCASP2_XMIT_INTR_PEND_0	208
R5FSS1_INTRTR0_IN_208	MCASP2_REC_INTR_PEND_0	209
R5FSS1_INTRTR0_IN_209	MCASP3_XMIT_INTR_PEND_0	210
R5FSS1_INTRTR0_IN_210	MCASP3_REC_INTR_PEND_0	211
R5FSS1_INTRTR0_IN_211	MCASP4_XMIT_INTR_PEND_0	212
R5FSS1_INTRTR0_IN_212	MCASP4_REC_INTR_PEND_0	213
R5FSS1_INTRTR0_IN_213	MCASP5_XMIT_INTR_PEND_0	214
R5FSS1_INTRTR0_IN_214	MCASP5_REC_INTR_PEND_0	215
R5FSS1_INTRTR0_IN_215	MCASP6_XMIT_INTR_PEND_0	216
R5FSS1_INTRTR0_IN_216	MCASP6_REC_INTR_PEND_0	217
R5FSS1_INTRTR0_IN_217	MCASP7_XMIT_INTR_PEND_0	218
R5FSS1_INTRTR0_IN_218	MCASP7_REC_INTR_PEND_0	219
R5FSS1_INTRTR0_IN_219	MCASP8_XMIT_INTR_PEND_0	220
R5FSS1_INTRTR0_IN_220	MCASP8_REC_INTR_PEND_0	221
R5FSS1_INTRTR0_IN_221	MCASP9_XMIT_INTR_PEND_0	222
R5FSS1_INTRTR0_IN_222	MCASP9_REC_INTR_PEND_0	223
R5FSS1_INTRTR0_IN_223	MCASP10_XMIT_INTR_PEND_0	224
R5FSS1_INTRTR0_IN_224	MCASP10_REC_INTR_PEND_0	225
R5FSS1_INTRTR0_IN_225	MCASP11_XMIT_INTR_PEND_0	226
R5FSS1_INTRTR0_IN_226	MCASP11_REC_INTR_PEND_0	228

**Table 9-52. R5FSS1\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	R5FSS1_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
R5FSS1_INTRTR0_IN_228	GPMC0_GPMC_SINTERRUPT_0	229
R5FSS1_INTRTR0_IN_229	ELM0_ELM_POROCPSINTERRUPT_LVL_0	230
R5FSS1_INTRTR0_IN_230	USB0_OTGIRQ_0	231
R5FSS1_INTRTR0_IN_231	USB0_IRQ_0	232
R5FSS1_INTRTR0_IN_232	USB0_IRQ_1	233
R5FSS1_INTRTR0_IN_233	USB0_IRQ_2	234
R5FSS1_INTRTR0_IN_234	USB0_IRQ_3	235
R5FSS1_INTRTR0_IN_235	USB0_IRQ_4	236
R5FSS1_INTRTR0_IN_236	USB0_IRQ_5	237
R5FSS1_INTRTR0_IN_237	USB0_IRQ_6	238
R5FSS1_INTRTR0_IN_238	USB0_IRQ_7	239
R5FSS1_INTRTR0_IN_239	TIMER0_INTR_PEND_0	240
R5FSS1_INTRTR0_IN_240	TIMER1_INTR_PEND_0	241
R5FSS1_INTRTR0_IN_241	TIMER2_INTR_PEND_0	242
R5FSS1_INTRTR0_IN_242	TIMER3_INTR_PEND_0	243
R5FSS1_INTRTR0_IN_243	TIMER4_INTR_PEND_0	244
R5FSS1_INTRTR0_IN_244	TIMER5_INTR_PEND_0	245
R5FSS1_INTRTR0_IN_245	TIMER6_INTR_PEND_0	246
R5FSS1_INTRTR0_IN_246	TIMER7_INTR_PEND_0	247
R5FSS1_INTRTR0_IN_247	TIMER8_INTR_PEND_0	248
R5FSS1_INTRTR0_IN_248	TIMER9_INTR_PEND_0	249
R5FSS1_INTRTR0_IN_249	TIMER10_INTR_PEND_0	250
R5FSS1_INTRTR0_IN_250	TIMER11_INTR_PEND_0	251
R5FSS1_INTRTR0_IN_251	PCIE1_PCIE_LEGACY_PULSE_0	252
R5FSS1_INTRTR0_IN_252	PCIE1_PCIE_DOWNSTREAM_PULSE_0	253
R5FSS1_INTRTR0_IN_253	PCIE1_PCIE_FLR_PULSE_0	254
R5FSS1_INTRTR0_IN_254	PCIE1_PCIE_PHY_LEVEL_0	255
R5FSS1_INTRTR0_IN_255	PCIE1_PCIE_LOCAL_LEVEL_0	256
R5FSS1_INTRTR0_IN_256	PCIE1_PCIE_ERROR_PULSE_0	257
R5FSS1_INTRTR0_IN_257	PCIE1_PCIE_LINK_STATE_PULSE_0	258
R5FSS1_INTRTR0_IN_258	PCIE1_PCIE_PWR_STATE_PULSE_0	259
R5FSS1_INTRTR0_IN_259	PCIE1_PCIE_PTM_VALID_PULSE_0	260
R5FSS1_INTRTR0_IN_260	PCIE1_PCIE_HOT_RESET_PULSE_0	261
R5FSS1_INTRTR0_IN_261	PCIE1_PCIE_CPTS_PEND_0	262
R5FSS1_INTRTR0_IN_263	DDR0_DDRSS_CONTROLLER_0	264
R5FSS1_INTRTR0_IN_264	DDR0_DDRSS_V2A_OTHER_ERR_LVL_0	265
R5FSS1_INTRTR0_IN_265	DDR0_DDRSS_HS_PHY_GLOBAL_ERROR_0	266
R5FSS1_INTRTR0_IN_266	DDR0_DDRSS_PLL_FREQ_CHANGE_REQ_0	267
R5FSS1_INTRTR0_IN_267	CSI_TX_IF0_CSI_INTERRUPT_0	268
R5FSS1_INTRTR0_IN_268	CSI_TX_IF0_CSI_LEVEL_0	269
R5FSS1_INTRTR0_IN_275	VPFE0_CCDC_INTR_PEND_0	276
R5FSS1_INTRTR0_IN_276	VPFE0_RAT_EXP_INTR_0	277
R5FSS1_INTRTR0_IN_278	WKUP_DMSC0_RAT_0_EXP_INTR_0	279
R5FSS1_INTRTR0_IN_279	DCC0_INTR_DONE_LEVEL_0	280



**Table 9-52. R5FSS1\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	R5FSS1_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
R5FSS1_INTRTR0_IN_280	DCC1_INTR_DONE_LEVEL_0	281
R5FSS1_INTRTR0_IN_281	DCC2_INTR_DONE_LEVEL_0	282
R5FSS1_INTRTR0_IN_282	DCC3_INTR_DONE_LEVEL_0	283
R5FSS1_INTRTR0_IN_283	DCC4_INTR_DONE_LEVEL_0	284
R5FSS1_INTRTR0_IN_284	DCC5_INTR_DONE_LEVEL_0	285
R5FSS1_INTRTR0_IN_285	DCC6_INTR_DONE_LEVEL_0	286
R5FSS1_INTRTR0_IN_286	DCC7_INTR_DONE_LEVEL_0	287
R5FSS1_INTRTR0_IN_287	CMPEVENT_INTRTR0_OUTP_0	288
R5FSS1_INTRTR0_IN_288	CMPEVENT_INTRTR0_OUTP_1	289
R5FSS1_INTRTR0_IN_289	CMPEVENT_INTRTR0_OUTP_2	290
R5FSS1_INTRTR0_IN_290	CMPEVENT_INTRTR0_OUTP_3	291
R5FSS1_INTRTR0_IN_291	CMPEVENT_INTRTR0_OUTP_4	292
R5FSS1_INTRTR0_IN_292	CMPEVENT_INTRTR0_OUTP_5	293
R5FSS1_INTRTR0_IN_293	CMPEVENT_INTRTR0_OUTP_6	294
R5FSS1_INTRTR0_IN_294	CMPEVENT_INTRTR0_OUTP_7	295
R5FSS1_INTRTR0_IN_295	CMPEVENT_INTRTR0_OUTP_8	296
R5FSS1_INTRTR0_IN_296	CMPEVENT_INTRTR0_OUTP_9	297
R5FSS1_INTRTR0_IN_297	CMPEVENT_INTRTR0_OUTP_10	298
R5FSS1_INTRTR0_IN_298	CMPEVENT_INTRTR0_OUTP_11	299
R5FSS1_INTRTR0_IN_299	CMPEVENT_INTRTR0_OUTP_12	300
R5FSS1_INTRTR0_IN_300	CMPEVENT_INTRTR0_OUTP_13	301
R5FSS1_INTRTR0_IN_301	CMPEVENT_INTRTR0_OUTP_14	302
R5FSS1_INTRTR0_IN_302	CMPEVENT_INTRTR0_OUTP_15	303
R5FSS1_INTRTR0_IN_303	DCC8_INTR_DONE_LEVEL_0	304
R5FSS1_INTRTR0_IN_304	DCC9_INTR_DONE_LEVEL_0	305
R5FSS1_INTRTR0_IN_305	DCC10_INTR_DONE_LEVEL_0	306
R5FSS1_INTRTR0_IN_306	DCC11_INTR_DONE_LEVEL_0	307
R5FSS1_INTRTR0_IN_307	DCC12_INTR_DONE_LEVEL_0	308
R5FSS1_INTRTR0_IN_309	MMCSD1_EMMCSDSS_INTR_0	310
R5FSS1_INTRTR0_IN_310	MMCSD2_EMMCSDSS_INTR_0	311
R5FSS1_INTRTR0_IN_311	UFS0_UFS_INTR_0	312
R5FSS1_INTRTR0_IN_313	SA2_UL0_SA_UL_PKA_0	314
R5FSS1_INTRTR0_IN_314	SA2_UL0_SA_UL_TRNG_0	315
R5FSS1_INTRTR0_IN_315	ECAP0_ECAP_INT_0	316
R5FSS1_INTRTR0_IN_316	ECAP1_ECAP_INT_0	317
R5FSS1_INTRTR0_IN_317	ECAP2_ECAP_INT_0	318
R5FSS1_INTRTR0_IN_318	GLUELOGIC_SOCA_INT_GLUE_SOCA_INT_0	319
R5FSS1_INTRTR0_IN_319	GLUELOGIC_SOCB_INT_GLUE_SOCB_INT_0	320
R5FSS1_INTRTR0_IN_322	USB0_HOST_SYSTEM_ERROR_0	323
R5FSS1_INTRTR0_IN_324	CBASS_INFRA0_DEFAULT_ERR_INTR_0	325
R5FSS1_INTRTR0_IN_326	CTRL_MMR0_ACCESS_ERR_0	327
R5FSS1_INTRTR0_IN_327	WKUP_VTM0_THERM_LVL_GT_TH1_INTR_0	328
R5FSS1_INTRTR0_IN_328	WKUP_VTM0_THERM_LVL_GT_TH2_INTR_0	329
R5FSS1_INTRTR0_IN_329	WKUP_VTM0_THERM_LVL_LT_TH0_INTR_0	330

**Table 9-52. R5FSS1\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	R5FSS1_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
R5FSS1_INTRTR0_IN_331	COMPUTE_CLUSTER0_GIC_OUTPUT_WAKER_GIC_PWR0_WAKE_REQUEST_0	332
R5FSS1_INTRTR0_IN_332	COMPUTE_CLUSTER0_GIC_OUTPUT_WAKER_GIC_PWR0_WAKE_REQUEST_1	333
R5FSS1_INTRTR0_IN_335	MCU_MCAN0_MCANSS_MCAN_LVL_INT_0	336
R5FSS1_INTRTR0_IN_336	MCU_MCAN0_MCANSS_MCAN_LVL_INT_1	337
R5FSS1_INTRTR0_IN_337	MCU_MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	338
R5FSS1_INTRTR0_IN_338	MCU_MCAN1_MCANSS_MCAN_LVL_INT_0	339
R5FSS1_INTRTR0_IN_339	MCU_MCAN1_MCANSS_MCAN_LVL_INT_1	340
R5FSS1_INTRTR0_IN_340	MCU_MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	341

- (1) Do not use that setting. As it is the default one, it is strongly recommended to set the field to another value before enabling interrupt router operation.

### 9.4.3.8 C66SS0 Interrupt Map

**Table 9-53. C66SS0 Interrupt Map**

Interrupt Input Line	Interrupt ID	Interrupt Name
C66SS0_INTR_IN_0	0	GLUELOGIC_EXT_INTN_GLUE_EXT_INT_LVL_0
C66SS0_INTR_IN_4	4	C66_0_INTROUTER0_OUTL_0
C66SS0_INTR_IN_5	5	C66_0_INTROUTER0_OUTL_1
C66SS0_INTR_IN_6	6	C66_0_INTROUTER0_OUTL_2
C66SS0_INTR_IN_7	7	C66_0_INTROUTER0_OUTL_3
C66SS0_INTR_IN_8	8	C66_0_INTROUTER0_OUTL_4
C66SS0_INTR_IN_15	15	C66_0_INTROUTER0_OUTL_5
C66SS0_INTR_IN_16	16	C66_0_INTROUTER0_OUTL_6
C66SS0_INTR_IN_17	17	C66_0_INTROUTER0_OUTL_7
C66SS0_INTR_IN_18	18	C66_0_INTROUTER0_OUTL_8
C66SS0_INTR_IN_19	19	C66_0_INTROUTER0_OUTL_9
C66SS0_INTR_IN_20	20	C66_0_INTROUTER0_OUTL_10
C66SS0_INTR_IN_21	21	C66_0_INTROUTER0_OUTL_11
C66SS0_INTR_IN_22	22	C66_0_INTROUTER0_OUTL_12
C66SS0_INTR_IN_23	23	C66_0_INTROUTER0_OUTL_13
C66SS0_INTR_IN_24	24	C66_0_INTROUTER0_OUTL_14
C66SS0_INTR_IN_25	25	C66_0_INTROUTER0_OUTL_15
C66SS0_INTR_IN_26	26	C66_0_INTROUTER0_OUTL_16
C66SS0_INTR_IN_27	27	C66_0_INTROUTER0_OUTL_17
C66SS0_INTR_IN_28	28	C66_0_INTROUTER0_OUTL_18
C66SS0_INTR_IN_29	29	C66_0_INTROUTER0_OUTL_19
C66SS0_INTR_IN_30	30	C66_0_INTROUTER0_OUTL_20
C66SS0_INTR_IN_31	31	C66_0_INTROUTER0_OUTL_21
C66SS0_INTR_IN_32	32	C66_0_INTROUTER0_OUTL_22
C66SS0_INTR_IN_33	33	C66_0_INTROUTER0_OUTL_23
C66SS0_INTR_IN_34	34	C66_0_INTROUTER0_OUTL_24
C66SS0_INTR_IN_35	35	C66_0_INTROUTER0_OUTL_25
C66SS0_INTR_IN_36	36	C66_0_INTROUTER0_OUTL_26
C66SS0_INTR_IN_37	37	C66_0_INTROUTER0_OUTL_27
C66SS0_INTR_IN_38	38	C66_0_INTROUTER0_OUTL_28
C66SS0_INTR_IN_39	39	C66_0_INTROUTER0_OUTL_29
C66SS0_INTR_IN_40	40	C66_0_INTROUTER0_OUTL_30
C66SS0_INTR_IN_41	41	C66_0_INTROUTER0_OUTL_31
C66SS0_INTR_IN_42	42	C66_0_INTROUTER0_OUTL_32
C66SS0_INTR_IN_43	43	C66_0_INTROUTER0_OUTL_33
C66SS0_INTR_IN_44	44	C66_0_INTROUTER0_OUTL_34
C66SS0_INTR_IN_45	45	C66_0_INTROUTER0_OUTL_35
C66SS0_INTR_IN_46	46	C66_0_INTROUTER0_OUTL_36
C66SS0_INTR_IN_47	47	C66_0_INTROUTER0_OUTL_37
C66SS0_INTR_IN_48	48	C66_0_INTROUTER0_OUTL_38
C66SS0_INTR_IN_49	49	C66_0_INTROUTER0_OUTL_39
C66SS0_INTR_IN_50	50	C66_0_INTROUTER0_OUTL_40
C66SS0_INTR_IN_51	51	C66_0_INTROUTER0_OUTL_41
C66SS0_INTR_IN_52	52	C66_0_INTROUTER0_OUTL_42

**Table 9-53. C66SS0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
C66SS0_INTR_IN_53	53	C66_0_INTROUTER0_OUTL_43
C66SS0_INTR_IN_54	54	C66_0_INTROUTER0_OUTL_44
C66SS0_INTR_IN_55	55	C66_0_INTROUTER0_OUTL_45
C66SS0_INTR_IN_56	56	C66_0_INTROUTER0_OUTL_46
C66SS0_INTR_IN_57	57	C66_0_INTROUTER0_OUTL_47
C66SS0_INTR_IN_58	58	C66_0_INTROUTER0_OUTL_48
C66SS0_INTR_IN_59	59	C66_0_INTROUTER0_OUTL_49
C66SS0_INTR_IN_60	60	C66_0_INTROUTER0_OUTL_50
C66SS0_INTR_IN_61	61	C66_0_INTROUTER0_OUTL_51
C66SS0_INTR_IN_62	62	C66_0_INTROUTER0_OUTL_52
C66SS0_INTR_IN_63	63	C66_0_INTROUTER0_OUTL_53
C66SS0_INTR_IN_64	64	C66_0_INTROUTER0_OUTL_54
C66SS0_INTR_IN_65	65	C66_0_INTROUTER0_OUTL_55
C66SS0_INTR_IN_66	66	C66_0_INTROUTER0_OUTL_56
C66SS0_INTR_IN_67	67	C66_0_INTROUTER0_OUTL_57
C66SS0_INTR_IN_68	68	C66_0_INTROUTER0_OUTL_58
C66SS0_INTR_IN_69	69	C66_0_INTROUTER0_OUTL_59
C66SS0_INTR_IN_70	70	C66_0_INTROUTER0_OUTL_60
C66SS0_INTR_IN_71	71	C66_0_INTROUTER0_OUTL_61
C66SS0_INTR_IN_72	72	C66_0_INTROUTER0_OUTL_62
C66SS0_INTR_IN_73	73	C66_0_INTROUTER0_OUTL_63
C66SS0_INTR_IN_74	74	C66_0_INTROUTER0_OUTL_64
C66SS0_INTR_IN_75	75	C66_0_INTROUTER0_OUTL_65
C66SS0_INTR_IN_76	76	C66_0_INTROUTER0_OUTL_66
C66SS0_INTR_IN_77	77	C66_0_INTROUTER0_OUTL_67
C66SS0_INTR_IN_78	78	C66_0_INTROUTER0_OUTL_68
C66SS0_INTR_IN_79	79	C66_0_INTROUTER0_OUTL_69
C66SS0_INTR_IN_80	80	C66_0_INTROUTER0_OUTL_70
C66SS0_INTR_IN_81	81	C66_0_INTROUTER0_OUTL_71
C66SS0_INTR_IN_82	82	C66_0_INTROUTER0_OUTL_72
C66SS0_INTR_IN_83	83	C66_0_INTROUTER0_OUTL_73
C66SS0_INTR_IN_84	84	C66_0_INTROUTER0_OUTL_74
C66SS0_INTR_IN_85	85	C66_0_INTROUTER0_OUTL_75
C66SS0_INTR_IN_86	86	C66_0_INTROUTER0_OUTL_76
C66SS0_INTR_IN_87	87	C66_0_INTROUTER0_OUTL_77
C66SS0_INTR_IN_88	88	C66_0_INTROUTER0_OUTL_78
C66SS0_INTR_IN_89	89	C66_0_INTROUTER0_OUTL_79
C66SS0_INTR_IN_90	90	C66_0_INTROUTER0_OUTL_80
C66SS0_INTR_IN_91	91	C66_0_INTROUTER0_OUTL_81
C66SS0_INTR_IN_92	92	C66_0_INTROUTER0_OUTL_82
C66SS0_INTR_IN_93	93	C66_0_INTROUTER0_OUTL_83
C66SS0_INTR_IN_94	94	C66_0_INTROUTER0_OUTL_84
C66SS0_INTR_IN_95	95	C66_0_INTROUTER0_OUTL_85
C66SS0_INTR_IN_99	99	C66_0_INTROUTER0_OUTL_86
C66SS0_INTR_IN_102	102	C66_0_INTROUTER0_OUTL_87

**Table 9-53. C66SS0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
C66SS0_INTR_IN_103	103	C66_0_INTROUTER0_OUTL_88
C66SS0_INTR_IN_104	104	C66_0_INTROUTER0_OUTL_89
C66SS0_INTR_IN_105	105	C66_0_INTROUTER0_OUTL_90
C66SS0_INTR_IN_106	106	C66_0_INTROUTER0_OUTL_91
C66SS0_INTR_IN_107	107	C66_0_INTROUTER0_OUTL_92
C66SS0_INTR_IN_108	108	C66_0_INTROUTER0_OUTL_93
C66SS0_INTR_IN_109	109	C66_0_INTROUTER0_OUTL_94
C66SS0_INTR_IN_114	114	C66_0_INTROUTER0_OUTL_95
C66SS0_INTR_IN_115	115	C66_0_INTROUTER0_OUTL_96

### 9.4.3.9 C66SS1 Interrupt Map

**Table 9-54. C66SS1 Interrupt Map**

Interrupt Input Line	Interrupt ID	Interrupt Name
C66SS1_INTR_IN_0	0	GLUELOGIC_EXT_INTN_GLUE_EXT_INT_LVL_0
C66SS1_INTR_IN_4	4	C66_1_INTROUTER0_OUTL_0
C66SS1_INTR_IN_5	5	C66_1_INTROUTER0_OUTL_1
C66SS1_INTR_IN_6	6	C66_1_INTROUTER0_OUTL_2
C66SS1_INTR_IN_7	7	C66_1_INTROUTER0_OUTL_3
C66SS1_INTR_IN_8	8	C66_1_INTROUTER0_OUTL_4
C66SS1_INTR_IN_15	15	C66_1_INTROUTER0_OUTL_5
C66SS1_INTR_IN_16	16	C66_1_INTROUTER0_OUTL_6
C66SS1_INTR_IN_17	17	C66_1_INTROUTER0_OUTL_7
C66SS1_INTR_IN_18	18	C66_1_INTROUTER0_OUTL_8
C66SS1_INTR_IN_19	19	C66_1_INTROUTER0_OUTL_9
C66SS1_INTR_IN_20	20	C66_1_INTROUTER0_OUTL_10
C66SS1_INTR_IN_21	21	C66_1_INTROUTER0_OUTL_11
C66SS1_INTR_IN_22	22	C66_1_INTROUTER0_OUTL_12
C66SS1_INTR_IN_23	23	C66_1_INTROUTER0_OUTL_13
C66SS1_INTR_IN_24	24	C66_1_INTROUTER0_OUTL_14
C66SS1_INTR_IN_25	25	C66_1_INTROUTER0_OUTL_15
C66SS1_INTR_IN_26	26	C66_1_INTROUTER0_OUTL_16
C66SS1_INTR_IN_27	27	C66_1_INTROUTER0_OUTL_17
C66SS1_INTR_IN_28	28	C66_1_INTROUTER0_OUTL_18
C66SS1_INTR_IN_29	29	C66_1_INTROUTER0_OUTL_19
C66SS1_INTR_IN_30	30	C66_1_INTROUTER0_OUTL_20
C66SS1_INTR_IN_31	31	C66_1_INTROUTER0_OUTL_21
C66SS1_INTR_IN_32	32	C66_1_INTROUTER0_OUTL_22
C66SS1_INTR_IN_33	33	C66_1_INTROUTER0_OUTL_23
C66SS1_INTR_IN_34	34	C66_1_INTROUTER0_OUTL_24
C66SS1_INTR_IN_35	35	C66_1_INTROUTER0_OUTL_25
C66SS1_INTR_IN_36	36	C66_1_INTROUTER0_OUTL_26
C66SS1_INTR_IN_37	37	C66_1_INTROUTER0_OUTL_27
C66SS1_INTR_IN_38	38	C66_1_INTROUTER0_OUTL_28
C66SS1_INTR_IN_39	39	C66_1_INTROUTER0_OUTL_29
C66SS1_INTR_IN_40	40	C66_1_INTROUTER0_OUTL_30
C66SS1_INTR_IN_41	41	C66_1_INTROUTER0_OUTL_31
C66SS1_INTR_IN_42	42	C66_1_INTROUTER0_OUTL_32
C66SS1_INTR_IN_43	43	C66_1_INTROUTER0_OUTL_33
C66SS1_INTR_IN_44	44	C66_1_INTROUTER0_OUTL_34
C66SS1_INTR_IN_45	45	C66_1_INTROUTER0_OUTL_35
C66SS1_INTR_IN_46	46	C66_1_INTROUTER0_OUTL_36
C66SS1_INTR_IN_47	47	C66_1_INTROUTER0_OUTL_37
C66SS1_INTR_IN_48	48	C66_1_INTROUTER0_OUTL_38
C66SS1_INTR_IN_49	49	C66_1_INTROUTER0_OUTL_39
C66SS1_INTR_IN_50	50	C66_1_INTROUTER0_OUTL_40
C66SS1_INTR_IN_51	51	C66_1_INTROUTER0_OUTL_41

**Table 9-54. C66SS1 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
C66SS1_INTR_IN_52	52	C66_1_INTROUTER0_OUTL_42
C66SS1_INTR_IN_53	53	C66_1_INTROUTER0_OUTL_43
C66SS1_INTR_IN_54	54	C66_1_INTROUTER0_OUTL_44
C66SS1_INTR_IN_55	55	C66_1_INTROUTER0_OUTL_45
C66SS1_INTR_IN_56	56	C66_1_INTROUTER0_OUTL_46
C66SS1_INTR_IN_57	57	C66_1_INTROUTER0_OUTL_47
C66SS1_INTR_IN_58	58	C66_1_INTROUTER0_OUTL_48
C66SS1_INTR_IN_59	59	C66_1_INTROUTER0_OUTL_49
C66SS1_INTR_IN_60	60	C66_1_INTROUTER0_OUTL_50
C66SS1_INTR_IN_61	61	C66_1_INTROUTER0_OUTL_51
C66SS1_INTR_IN_62	62	C66_1_INTROUTER0_OUTL_52
C66SS1_INTR_IN_63	63	C66_1_INTROUTER0_OUTL_53
C66SS1_INTR_IN_64	64	C66_1_INTROUTER0_OUTL_54
C66SS1_INTR_IN_65	65	C66_1_INTROUTER0_OUTL_55
C66SS1_INTR_IN_66	66	C66_1_INTROUTER0_OUTL_56
C66SS1_INTR_IN_67	67	C66_1_INTROUTER0_OUTL_57
C66SS1_INTR_IN_68	68	C66_1_INTROUTER0_OUTL_58
C66SS1_INTR_IN_69	69	C66_1_INTROUTER0_OUTL_59
C66SS1_INTR_IN_70	70	C66_1_INTROUTER0_OUTL_60
C66SS1_INTR_IN_71	71	C66_1_INTROUTER0_OUTL_61
C66SS1_INTR_IN_72	72	C66_1_INTROUTER0_OUTL_62
C66SS1_INTR_IN_73	73	C66_1_INTROUTER0_OUTL_63
C66SS1_INTR_IN_74	74	C66_1_INTROUTER0_OUTL_64
C66SS1_INTR_IN_75	75	C66_1_INTROUTER0_OUTL_65
C66SS1_INTR_IN_76	76	C66_1_INTROUTER0_OUTL_66
C66SS1_INTR_IN_77	77	C66_1_INTROUTER0_OUTL_67
C66SS1_INTR_IN_78	78	C66_1_INTROUTER0_OUTL_68
C66SS1_INTR_IN_79	79	C66_1_INTROUTER0_OUTL_69
C66SS1_INTR_IN_80	80	C66_1_INTROUTER0_OUTL_70
C66SS1_INTR_IN_81	81	C66_1_INTROUTER0_OUTL_71
C66SS1_INTR_IN_82	82	C66_1_INTROUTER0_OUTL_72
C66SS1_INTR_IN_83	83	C66_1_INTROUTER0_OUTL_73
C66SS1_INTR_IN_84	84	C66_1_INTROUTER0_OUTL_74
C66SS1_INTR_IN_85	85	C66_1_INTROUTER0_OUTL_75
C66SS1_INTR_IN_86	86	C66_1_INTROUTER0_OUTL_76
C66SS1_INTR_IN_87	87	C66_1_INTROUTER0_OUTL_77
C66SS1_INTR_IN_88	88	C66_1_INTROUTER0_OUTL_78
C66SS1_INTR_IN_89	89	C66_1_INTROUTER0_OUTL_79
C66SS1_INTR_IN_90	90	C66_1_INTROUTER0_OUTL_80
C66SS1_INTR_IN_91	91	C66_1_INTROUTER0_OUTL_81
C66SS1_INTR_IN_92	92	C66_1_INTROUTER0_OUTL_82
C66SS1_INTR_IN_93	93	C66_1_INTROUTER0_OUTL_83
C66SS1_INTR_IN_94	94	C66_1_INTROUTER0_OUTL_84
C66SS1_INTR_IN_95	95	C66_1_INTROUTER0_OUTL_85

**Table 9-54. C66SS1 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
C66SS1_INTR_IN_99	99	C66_1_INTROUTER0_OUTL_86
C66SS1_INTR_IN_102	102	C66_1_INTROUTER0_OUTL_87
C66SS1_INTR_IN_103	103	C66_1_INTROUTER0_OUTL_88
C66SS1_INTR_IN_104	104	C66_1_INTROUTER0_OUTL_89
C66SS1_INTR_IN_105	105	C66_1_INTROUTER0_OUTL_90
C66SS1_INTR_IN_106	106	C66_1_INTROUTER0_OUTL_91
C66SS1_INTR_IN_107	107	C66_1_INTROUTER0_OUTL_92
C66SS1_INTR_IN_108	108	C66_1_INTROUTER0_OUTL_93
C66SS1_INTR_IN_109	109	C66_1_INTROUTER0_OUTL_94
C66SS1_INTR_IN_114	114	C66_1_INTROUTER0_OUTL_95
C66SS1_INTR_IN_115	115	C66_1_INTROUTER0_OUTL_96



#### 9.4.3.10 C66SS0\_INTRTR0 Interrupt Map

Table 9-55 lists all the interrupts that are mapped to the C66SS0\_INTRTR0 inputs, along with the corresponding register programming values used for mux control. Any C66SS0\_INTRTR0 interrupt input that is not listed in this table should be considered as unused.

**Table 9-55. C66SS0\_INTRTR0 Interrupt Map**

Interrupt Input Line	Peripheral Interrupt	C66SS0_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
RESERVED	RESERVED	0 <sup>(1)</sup>
C66SS0_INTRTR0_IN_0	TIMER0_INTR_PEND_0	1
C66SS0_INTRTR0_IN_1	TIMER1_INTR_PEND_0	2
C66SS0_INTRTR0_IN_2	TIMER2_INTR_PEND_0	3
C66SS0_INTRTR0_IN_3	TIMER3_INTR_PEND_0	4
C66SS0_INTRTR0_IN_4	TIMER4_INTR_PEND_0	5
C66SS0_INTRTR0_IN_5	TIMER5_INTR_PEND_0	6
C66SS0_INTRTR0_IN_6	TIMER6_INTR_PEND_0	7
C66SS0_INTRTR0_IN_7	TIMER7_INTR_PEND_0	8
C66SS0_INTRTR0_IN_8	TIMER8_INTR_PEND_0	9
C66SS0_INTRTR0_IN_9	TIMER9_INTR_PEND_0	10
C66SS0_INTRTR0_IN_10	TIMER10_INTR_PEND_0	11
C66SS0_INTRTR0_IN_11	TIMER11_INTR_PEND_0	12
C66SS0_INTRTR0_IN_12	TIMER16_INTR_PEND_0	13
C66SS0_INTRTR0_IN_13	TIMER17_INTR_PEND_0	14
C66SS0_INTRTR0_IN_14	TIMER18_INTR_PEND_0	15
C66SS0_INTRTR0_IN_15	TIMER19_INTR_PEND_0	16
C66SS0_INTRTR0_IN_16	MCSP13_INTR_SPI_0	17
C66SS0_INTRTR0_IN_17	MCSP14_INTR_SPI_0	18
C66SS0_INTRTR0_IN_18	MCSP15_INTR_SPI_0	19
C66SS0_INTRTR0_IN_19	MCSP16_INTR_SPI_0	20
C66SS0_INTRTR0_IN_20	MCAN2_MCANSS_MCAN_LVL_INT_0	21
C66SS0_INTRTR0_IN_21	MCAN2_MCANSS_MCAN_LVL_INT_1	22
C66SS0_INTRTR0_IN_22	MCAN2_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	23
C66SS0_INTRTR0_IN_23	MCAN3_MCANSS_MCAN_LVL_INT_0	24
C66SS0_INTRTR0_IN_24	MCAN3_MCANSS_MCAN_LVL_INT_1	25
C66SS0_INTRTR0_IN_25	MCAN3_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	26
C66SS0_INTRTR0_IN_26	I2C3_POINTRPEND_0	27
C66SS0_INTRTR0_IN_27	I2C4_POINTRPEND_0	28
C66SS0_INTRTR0_IN_28	I2C5_POINTRPEND_0	29
C66SS0_INTRTR0_IN_29	I2C6_POINTRPEND_0	30
C66SS0_INTRTR0_IN_30	GLUELOGIC_SOCA_INT_GLUE_SOCA_INT_0	31
C66SS0_INTRTR0_IN_31	GLUELOGIC_SOCB_INT_GLUE_SOCB_INT_0	32
C66SS0_INTRTR0_IN_32	EHRPWM0_EPWM_ETINT_0	33
C66SS0_INTRTR0_IN_33	EHRPWM1_EPWM_ETINT_0	34
C66SS0_INTRTR0_IN_34	EHRPWM2_EPWM_ETINT_0	35
C66SS0_INTRTR0_IN_35	EHRPWM3_EPWM_ETINT_0	36
C66SS0_INTRTR0_IN_36	EHRPWM4_EPWM_ETINT_0	37
C66SS0_INTRTR0_IN_37	EHRPWM5_EPWM_ETINT_0	38
C66SS0_INTRTR0_IN_38	EHRPWM0_EPWM_TRIPZINT_0	39

**Table 9-55. C66SS0\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	C66SS0_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
C66SS0_INTRTR0_IN_39	EHRPWM1_EPWM_TRIPZINT_0	40
C66SS0_INTRTR0_IN_40	EHRPWM2_EPWM_TRIPZINT_0	41
C66SS0_INTRTR0_IN_41	EHRPWM3_EPWM_TRIPZINT_0	42
C66SS0_INTRTR0_IN_42	EHRPWM4_EPWM_TRIPZINT_0	43
C66SS0_INTRTR0_IN_43	EHRPWM5_EPWM_TRIPZINT_0	44
C66SS0_INTRTR0_IN_44	ECAP0_ECAP_INT_0	45
C66SS0_INTRTR0_IN_45	ECAP1_ECAP_INT_0	46
C66SS0_INTRTR0_IN_46	ECAP2_ECAP_INT_0	47
C66SS0_INTRTR0_IN_47	EQEP0_EQEP_INT_0	48
C66SS0_INTRTR0_IN_48	EQEP1_EQEP_INT_0	49
C66SS0_INTRTR0_IN_49	EQEP2_EQEP_INT_0	50
C66SS0_INTRTR0_IN_50	UART3_USART_IRQ_0	51
C66SS0_INTRTR0_IN_51	UART4_USART_IRQ_0	52
C66SS0_INTRTR0_IN_52	UART5_USART_IRQ_0	53
C66SS0_INTRTR0_IN_53	UART6_USART_IRQ_0	54
C66SS0_INTRTR0_IN_54	UART7_USART_IRQ_0	55
C66SS0_INTRTR0_IN_55	UART8_USART_IRQ_0	56
C66SS0_INTRTR0_IN_56	UART9_USART_IRQ_0	57
C66SS0_INTRTR0_IN_57	MCAN4_MCANSS_MCAN_LVL_INT_0	58
C66SS0_INTRTR0_IN_58	MCAN4_MCANSS_MCAN_LVL_INT_1	59
C66SS0_INTRTR0_IN_59	MCAN4_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	60
C66SS0_INTRTR0_IN_60	MCAN5_MCANSS_MCAN_LVL_INT_0	61
C66SS0_INTRTR0_IN_61	MCAN5_MCANSS_MCAN_LVL_INT_1	62
C66SS0_INTRTR0_IN_62	MCAN5_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	63
C66SS0_INTRTR0_IN_63	MCAN6_MCANSS_MCAN_LVL_INT_0	64
C66SS0_INTRTR0_IN_64	MCAN6_MCANSS_MCAN_LVL_INT_1	65
C66SS0_INTRTR0_IN_65	MCAN6_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	66
C66SS0_INTRTR0_IN_66	MCAN7_MCANSS_MCAN_LVL_INT_0	67
C66SS0_INTRTR0_IN_67	MCAN7_MCANSS_MCAN_LVL_INT_1	68
C66SS0_INTRTR0_IN_68	MCAN7_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	69
C66SS0_INTRTR0_IN_69	MCAN8_MCANSS_MCAN_LVL_INT_0	70
C66SS0_INTRTR0_IN_70	MCAN8_MCANSS_MCAN_LVL_INT_1	71
C66SS0_INTRTR0_IN_71	MCAN8_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	72
C66SS0_INTRTR0_IN_72	MCAN9_MCANSS_MCAN_LVL_INT_0	73
C66SS0_INTRTR0_IN_73	MCAN9_MCANSS_MCAN_LVL_INT_1	74
C66SS0_INTRTR0_IN_74	MCAN9_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	75
C66SS0_INTRTR0_IN_75	MCAN10_MCANSS_MCAN_LVL_INT_0	76
C66SS0_INTRTR0_IN_76	MCAN10_MCANSS_MCAN_LVL_INT_1	77
C66SS0_INTRTR0_IN_77	MCAN10_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	78
C66SS0_INTRTR0_IN_78	MCAN11_MCANSS_MCAN_LVL_INT_0	79
C66SS0_INTRTR0_IN_79	MCAN11_MCANSS_MCAN_LVL_INT_1	80
C66SS0_INTRTR0_IN_80	MCAN11_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	81
C66SS0_INTRTR0_IN_81	MCAN12_MCANSS_MCAN_LVL_INT_0	82
C66SS0_INTRTR0_IN_82	MCAN12_MCANSS_MCAN_LVL_INT_1	83

**Table 9-55. C66SS0\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	C66SS0_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
C66SS0_INTRTR0_IN_83	MCAN12_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	84
C66SS0_INTRTR0_IN_84	MCAN13_MCANSS_MCAN_LVL_INT_0	85
C66SS0_INTRTR0_IN_85	MCAN13_MCANSS_MCAN_LVL_INT_1	86
C66SS0_INTRTR0_IN_86	MCAN13_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	87
C66SS0_INTRTR0_IN_88	GPU0_MISC_0_IRQ_0	89
C66SS0_INTRTR0_IN_96	NAVSS0_INTR_ROUTER_0_OUTL_INTR_344	97
C66SS0_INTRTR0_IN_97	NAVSS0_INTR_ROUTER_0_OUTL_INTR_345	98
C66SS0_INTRTR0_IN_98	NAVSS0_INTR_ROUTER_0_OUTL_INTR_346	99
C66SS0_INTRTR0_IN_99	NAVSS0_INTR_ROUTER_0_OUTL_INTR_347	100
C66SS0_INTRTR0_IN_100	NAVSS0_INTR_ROUTER_0_OUTL_INTR_348	101
C66SS0_INTRTR0_IN_101	NAVSS0_INTR_ROUTER_0_OUTL_INTR_349	102
C66SS0_INTRTR0_IN_102	NAVSS0_INTR_ROUTER_0_OUTL_INTR_350	103
C66SS0_INTRTR0_IN_103	NAVSS0_INTR_ROUTER_0_OUTL_INTR_351	104
C66SS0_INTRTR0_IN_104	DSS0_DSS_INST0_DISPC_FUNC_IRQ_PROC0_0	105
C66SS0_INTRTR0_IN_105	DSS0_DSS_INST0_DISPC_FUNC_IRQ_PROC1_0	106
C66SS0_INTRTR0_IN_106	DSS0_DSS_INST0_DISPC_SECURE_IRQ_PROC0_0	107
C66SS0_INTRTR0_IN_107	DSS0_DSS_INST0_DISPC_SECURE_IRQ_PROC1_0	108
C66SS0_INTRTR0_IN_108	DSS0_DSS_INST0_DISPC_SAFETY_ERROR_IRQ_PROC0_0	109
C66SS0_INTRTR0_IN_109	DSS0_DSS_INST0_DISPC_SAFETY_ERROR_IRQ_PROC1_0	110
C66SS0_INTRTR0_IN_110	DMPAC0_INTD_0_SYSTEM_INTR_LEVEL_0	111
C66SS0_INTRTR0_IN_111	DMPAC0_INTD_0_SYSTEM_INTR_LEVEL_1	112
C66SS0_INTRTR0_IN_114	GLUELOGIC_GPU_GPIO_REQACK_GLUE_GPU_GPIO_REQINT_LVL_0	115
C66SS0_INTRTR0_IN_115	GLUELOGIC_GPU_GPIO_REQACK_GLUE_GPU_GPIO_ACKINT_LVL_0	116
C66SS0_INTRTR0_IN_116	CPSW0_STAT_PEND_0	117
C66SS0_INTRTR0_IN_117	CPSW0_MDIO_PEND_0	118
C66SS0_INTRTR0_IN_118	CPSW0_EVNT_PEND_0	119
C66SS0_INTRTR0_IN_120	SA2_UL0_SA_UL_PKA_0	121
C66SS0_INTRTR0_IN_121	SA2_UL0_SA_UL_TRNG_0	122
C66SS0_INTRTR0_IN_122	ESM0_ESM_INT_LOW_LVL_0	123
C66SS0_INTRTR0_IN_123	ESM0_ESM_INT_HI_LVL_0	124
C66SS0_INTRTR0_IN_124	ESM0_ESM_INT_CFG_LVL_0	125
C66SS0_INTRTR0_IN_127	GPIOMUX_INTRTR0_OUTP_40	128
C66SS0_INTRTR0_IN_128	GPIOMUX_INTRTR0_OUTP_41	129
C66SS0_INTRTR0_IN_129	GPIOMUX_INTRTR0_OUTP_42	130
C66SS0_INTRTR0_IN_130	GPIOMUX_INTRTR0_OUTP_43	131
C66SS0_INTRTR0_IN_131	GPIOMUX_INTRTR0_OUTP_44	132
C66SS0_INTRTR0_IN_132	GPIOMUX_INTRTR0_OUTP_45	133
C66SS0_INTRTR0_IN_133	GPIOMUX_INTRTR0_OUTP_46	134
C66SS0_INTRTR0_IN_134	GPIOMUX_INTRTR0_OUTP_47	135
C66SS0_INTRTR0_IN_135	GPIOMUX_INTRTR0_OUTP_48	136
C66SS0_INTRTR0_IN_136	GPIOMUX_INTRTR0_OUTP_49	137
C66SS0_INTRTR0_IN_137	GPIOMUX_INTRTR0_OUTP_50	138
C66SS0_INTRTR0_IN_138	GPIOMUX_INTRTR0_OUTP_51	139

**Table 9-55. C66SS0\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	C66SS0_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
C66SS0_INTRTR0_IN_139	GPIOMUX_INTRTR0_OUTP_52	140
C66SS0_INTRTR0_IN_140	GPIOMUX_INTRTR0_OUTP_53	141
C66SS0_INTRTR0_IN_141	GPIOMUX_INTRTR0_OUTP_54	142
C66SS0_INTRTR0_IN_142	GPIOMUX_INTRTR0_OUTP_55	143
C66SS0_INTRTR0_IN_143	GPIOMUX_INTRTR0_OUTP_56	144
C66SS0_INTRTR0_IN_144	GPIOMUX_INTRTR0_OUTP_57	145
C66SS0_INTRTR0_IN_145	GPIOMUX_INTRTR0_OUTP_58	146
C66SS0_INTRTR0_IN_146	GPIOMUX_INTRTR0_OUTP_59	147
C66SS0_INTRTR0_IN_147	GPIOMUX_INTRTR0_OUTP_60	148
C66SS0_INTRTR0_IN_148	GPIOMUX_INTRTR0_OUTP_61	149
C66SS0_INTRTR0_IN_149	GPIOMUX_INTRTR0_OUTP_62	150
C66SS0_INTRTR0_IN_150	GPIOMUX_INTRTR0_OUTP_63	151
C66SS0_INTRTR0_IN_151	PRU_ICSSG0_PR1_HOST_INTR_PEND_0	152
C66SS0_INTRTR0_IN_152	PRU_ICSSG0_PR1_HOST_INTR_PEND_1	153
C66SS0_INTRTR0_IN_153	PRU_ICSSG0_PR1_HOST_INTR_PEND_2	154
C66SS0_INTRTR0_IN_154	PRU_ICSSG0_PR1_HOST_INTR_PEND_3	155
C66SS0_INTRTR0_IN_155	PRU_ICSSG0_PR1_HOST_INTR_PEND_4	156
C66SS0_INTRTR0_IN_156	PRU_ICSSG0_PR1_HOST_INTR_PEND_5	157
C66SS0_INTRTR0_IN_157	PRU_ICSSG0_PR1_HOST_INTR_PEND_6	158
C66SS0_INTRTR0_IN_158	PRU_ICSSG0_PR1_HOST_INTR_PEND_7	159
C66SS0_INTRTR0_IN_159	PRU_ICSSG1_PR1_HOST_INTR_PEND_0	160
C66SS0_INTRTR0_IN_160	PRU_ICSSG1_PR1_HOST_INTR_PEND_1	161
C66SS0_INTRTR0_IN_161	PRU_ICSSG1_PR1_HOST_INTR_PEND_2	162
C66SS0_INTRTR0_IN_162	PRU_ICSSG1_PR1_HOST_INTR_PEND_3	163
C66SS0_INTRTR0_IN_163	PRU_ICSSG1_PR1_HOST_INTR_PEND_4	164
C66SS0_INTRTR0_IN_164	PRU_ICSSG1_PR1_HOST_INTR_PEND_5	165
C66SS0_INTRTR0_IN_165	PRU_ICSSG1_PR1_HOST_INTR_PEND_6	166
C66SS0_INTRTR0_IN_166	PRU_ICSSG1_PR1_HOST_INTR_PEND_7	167
C66SS0_INTRTR0_IN_167	PRU_ICSSG0_PR1_TX_SOF_INTR_REQ_0	168
C66SS0_INTRTR0_IN_168	PRU_ICSSG0_PR1_TX_SOF_INTR_REQ_1	169
C66SS0_INTRTR0_IN_169	PRU_ICSSG0_PR1_RX_SOF_INTR_REQ_0	170
C66SS0_INTRTR0_IN_170	PRU_ICSSG0_PR1_RX_SOF_INTR_REQ_1	171
C66SS0_INTRTR0_IN_171	PRU_ICSSG1_PR1_TX_SOF_INTR_REQ_0	172
C66SS0_INTRTR0_IN_172	PRU_ICSSG1_PR1_TX_SOF_INTR_REQ_1	173
C66SS0_INTRTR0_IN_173	PRU_ICSSG1_PR1_RX_SOF_INTR_REQ_0	174
C66SS0_INTRTR0_IN_174	PRU_ICSSG1_PR1_RX_SOF_INTR_REQ_1	175
C66SS0_INTRTR0_IN_175	USB0_IRQ_0	176
C66SS0_INTRTR0_IN_176	USB0_IRQ_1	177
C66SS0_INTRTR0_IN_177	USB0_IRQ_2	178
C66SS0_INTRTR0_IN_178	USB0_IRQ_3	179
C66SS0_INTRTR0_IN_179	USB0_IRQ_4	180
C66SS0_INTRTR0_IN_180	USB0_IRQ_5	181
C66SS0_INTRTR0_IN_181	USB0_IRQ_6	182
C66SS0_INTRTR0_IN_182	USB0_IRQ_7	183

**Table 9-55. C66SS0\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	C66SS0_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
C66SS0_INTRTR0_IN_183	USB1_IRQ_0	184
C66SS0_INTRTR0_IN_184	USB1_IRQ_1	185
C66SS0_INTRTR0_IN_185	USB1_IRQ_2	186
C66SS0_INTRTR0_IN_186	USB1_IRQ_3	187
C66SS0_INTRTR0_IN_187	USB1_IRQ_4	188
C66SS0_INTRTR0_IN_188	USB1_IRQ_5	189
C66SS0_INTRTR0_IN_189	USB1_IRQ_6	190
C66SS0_INTRTR0_IN_190	USB1_IRQ_7	191
C66SS0_INTRTR0_IN_199	USB0_OTGIRQ_0	200
C66SS0_INTRTR0_IN_200	USB1_OTGIRQ_0	201
C66SS0_INTRTR0_IN_202	PCIE0_PCIE_LEGACY_PULSE_0	203
C66SS0_INTRTR0_IN_203	PCIE0_PCIE_DOWNSTREAM_PULSE_0	204
C66SS0_INTRTR0_IN_204	PCIE0_PCIE_FLR_PULSE_0	205
C66SS0_INTRTR0_IN_205	PCIE0_PCIE_PHY_LEVEL_0	206
C66SS0_INTRTR0_IN_206	PCIE0_PCIE_LOCAL_LEVEL_0	207
C66SS0_INTRTR0_IN_207	PCIE0_PCIE_ERROR_PULSE_0	208
C66SS0_INTRTR0_IN_208	PCIE0_PCIE_LINK_STATE_PULSE_0	209
C66SS0_INTRTR0_IN_209	PCIE0_PCIE_PWR_STATE_PULSE_0	210
C66SS0_INTRTR0_IN_210	PCIE0_PCIE_PTM_VALID_PULSE_0	211
C66SS0_INTRTR0_IN_211	PCIE0_PCIE_HOT_RESET_PULSE_0	212
C66SS0_INTRTR0_IN_212	PCIE0_PCIE_CPTS_PEND_0	213
C66SS0_INTRTR0_IN_214	PCIE1_PCIE_LEGACY_PULSE_0	215
C66SS0_INTRTR0_IN_215	PCIE1_PCIE_DOWNSTREAM_PULSE_0	216
C66SS0_INTRTR0_IN_216	PCIE1_PCIE_FLR_PULSE_0	217
C66SS0_INTRTR0_IN_217	PCIE1_PCIE_PHY_LEVEL_0	218
C66SS0_INTRTR0_IN_218	PCIE1_PCIE_LOCAL_LEVEL_0	219
C66SS0_INTRTR0_IN_219	PCIE1_PCIE_ERROR_PULSE_0	220
C66SS0_INTRTR0_IN_220	PCIE1_PCIE_LINK_STATE_PULSE_0	221
C66SS0_INTRTR0_IN_221	PCIE1_PCIE_PWR_STATE_PULSE_0	222
C66SS0_INTRTR0_IN_222	PCIE1_PCIE_PTM_VALID_PULSE_0	223
C66SS0_INTRTR0_IN_223	PCIE1_PCIE_HOT_RESET_PULSE_0	224
C66SS0_INTRTR0_IN_224	PCIE1_PCIE_CPTS_PEND_0	225
C66SS0_INTRTR0_IN_226	PCIE2_PCIE_LEGACY_PULSE_0	227
C66SS0_INTRTR0_IN_227	PCIE2_PCIE_DOWNSTREAM_PULSE_0	228
C66SS0_INTRTR0_IN_228	PCIE2_PCIE_FLR_PULSE_0	229
C66SS0_INTRTR0_IN_229	PCIE2_PCIE_PHY_LEVEL_0	230
C66SS0_INTRTR0_IN_230	PCIE2_PCIE_LOCAL_LEVEL_0	231
C66SS0_INTRTR0_IN_231	PCIE2_PCIE_ERROR_PULSE_0	232
C66SS0_INTRTR0_IN_232	PCIE2_PCIE_LINK_STATE_PULSE_0	233
C66SS0_INTRTR0_IN_233	PCIE2_PCIE_PWR_STATE_PULSE_0	234
C66SS0_INTRTR0_IN_234	PCIE2_PCIE_PTM_VALID_PULSE_0	235
C66SS0_INTRTR0_IN_235	PCIE2_PCIE_HOT_RESET_PULSE_0	236
C66SS0_INTRTR0_IN_236	PCIE2_PCIE_CPTS_PEND_0	237
C66SS0_INTRTR0_IN_238	PCIE3_PCIE_LEGACY_PULSE_0	239

**Table 9-55. C66SS0\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	C66SS0_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
C66SS0_INTRTR0_IN_239	PCIE3_PCIE_DOWNSTREAM_PULSE_0	240
C66SS0_INTRTR0_IN_240	PCIE3_PCIE_FLR_PULSE_0	241
C66SS0_INTRTR0_IN_241	PCIE3_PCIE_PHY_LEVEL_0	242
C66SS0_INTRTR0_IN_242	PCIE3_PCIE_LOCAL_LEVEL_0	243
C66SS0_INTRTR0_IN_243	PCIE3_PCIE_ERROR_PULSE_0	244
C66SS0_INTRTR0_IN_244	PCIE3_PCIE_LINK_STATE_PULSE_0	245
C66SS0_INTRTR0_IN_245	PCIE3_PCIE_PWR_STATE_PULSE_0	246
C66SS0_INTRTR0_IN_246	PCIE3_PCIE_PTM_VALID_PULSE_0	247
C66SS0_INTRTR0_IN_247	PCIE3_PCIE_HOT_RESET_PULSE_0	248
C66SS0_INTRTR0_IN_248	PCIE3_PCIE_CPTS_PEND_0	249
C66SS0_INTRTR0_IN_252	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_16	253
C66SS0_INTRTR0_IN_253	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_17	254
C66SS0_INTRTR0_IN_254	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_18	255
C66SS0_INTRTR0_IN_255	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_19	256
C66SS0_INTRTR0_IN_256	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_20	257
C66SS0_INTRTR0_IN_257	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_21	258
C66SS0_INTRTR0_IN_258	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_22	259
C66SS0_INTRTR0_IN_259	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_23	260
C66SS0_INTRTR0_IN_260	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_24	261
C66SS0_INTRTR0_IN_261	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_25	262
C66SS0_INTRTR0_IN_262	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_26	263
C66SS0_INTRTR0_IN_263	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_27	264
C66SS0_INTRTR0_IN_264	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_28	265
C66SS0_INTRTR0_IN_265	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_29	266
C66SS0_INTRTR0_IN_266	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_30	267
C66SS0_INTRTR0_IN_267	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_31	268
C66SS0_INTRTR0_IN_268	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_0	269
C66SS0_INTRTR0_IN_269	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_1	270
C66SS0_INTRTR0_IN_270	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_2	271
C66SS0_INTRTR0_IN_271	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_3	272
C66SS0_INTRTR0_IN_272	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_4	273
C66SS0_INTRTR0_IN_273	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_5	274
C66SS0_INTRTR0_IN_276	USB0_HOST_SYSTEM_ERROR_0	277
C66SS0_INTRTR0_IN_277	USB1_HOST_SYSTEM_ERROR_0	278
C66SS0_INTRTR0_IN_279	MCU_CPSW0_STAT_PEND_0	280
C66SS0_INTRTR0_IN_280	MCU_CPSW0_MDIO_PEND_0	281
C66SS0_INTRTR0_IN_281	MCU_CPSW0_EVNT_PEND_0	282
C66SS0_INTRTR0_IN_283	MCU_TIMER0_INTR_PEND_0	284
C66SS0_INTRTR0_IN_284	MCU_TIMER1_INTR_PEND_0	285
C66SS0_INTRTR0_IN_285	MCU_TIMER2_INTR_PEND_0	286
C66SS0_INTRTR0_IN_286	MCU_TIMER3_INTR_PEND_0	287
C66SS0_INTRTR0_IN_287	MCU_TIMER4_INTR_PEND_0	288
C66SS0_INTRTR0_IN_288	MCU_TIMER5_INTR_PEND_0	289
C66SS0_INTRTR0_IN_289	MCU_TIMER6_INTR_PEND_0	290

**Table 9-55. C66SS0\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	C66SS0_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
C66SS0_INTRTR0_IN_290	MCU_TIMER7_INTR_PEND_0	291
C66SS0_INTRTR0_IN_291	MCU_TIMER8_INTR_PEND_0	292
C66SS0_INTRTR0_IN_292	MCU_TIMER9_INTR_PEND_0	293
C66SS0_INTRTR0_IN_295	MCU_ESM0_ESM_INT_LOW_LVL_0	296
C66SS0_INTRTR0_IN_296	MCU_ESM0_ESM_INT_HI_LVL_0	297
C66SS0_INTRTR0_IN_297	MCU_ESM0_ESM_INT_CFG_LVL_0	298
C66SS0_INTRTR0_IN_298	WKUP_ESM0_ESM_INT_LOW_LVL_0	299
C66SS0_INTRTR0_IN_299	WKUP_ESM0_ESM_INT_HI_LVL_0	300
C66SS0_INTRTR0_IN_300	WKUP_ESM0_ESM_INT_CFG_LVL_0	301
C66SS0_INTRTR0_IN_303	TIMER12_INTR_PEND_0	304
C66SS0_INTRTR0_IN_304	TIMER13_INTR_PEND_0	305
C66SS0_INTRTR0_IN_305	TIMER14_INTR_PEND_0	306
C66SS0_INTRTR0_IN_306	TIMER15_INTR_PEND_0	307
C66SS0_INTRTR0_IN_307	RTI24_INTR_WWD_0	308
C66SS0_INTRTR0_IN_308	CTRL_MMR0_IPC_SET6_IPC_SET_IPCFG_0	309
C66SS0_INTRTR0_IN_309	MCASP0_REC_INTR_PEND_0	310
C66SS0_INTRTR0_IN_310	MCASP1_REC_INTR_PEND_0	311
C66SS0_INTRTR0_IN_311	MCASP2_REC_INTR_PEND_0	312
C66SS0_INTRTR0_IN_312	MCASP3_REC_INTR_PEND_0	313
C66SS0_INTRTR0_IN_313	MCASP4_REC_INTR_PEND_0	314
C66SS0_INTRTR0_IN_314	MCASP5_REC_INTR_PEND_0	315
C66SS0_INTRTR0_IN_315	MCASP6_REC_INTR_PEND_0	316
C66SS0_INTRTR0_IN_316	MCASP7_REC_INTR_PEND_0	317
C66SS0_INTRTR0_IN_317	MCASP8_REC_INTR_PEND_0	318
C66SS0_INTRTR0_IN_318	MCASP9_REC_INTR_PEND_0	319
C66SS0_INTRTR0_IN_319	MCASP10_REC_INTR_PEND_0	320
C66SS0_INTRTR0_IN_320	MCASP11_REC_INTR_PEND_0	321
C66SS0_INTRTR0_IN_321	MCASP0_XMIT_INTR_PEND_0	322
C66SS0_INTRTR0_IN_322	MCASP1_XMIT_INTR_PEND_0	323
C66SS0_INTRTR0_IN_323	MCASP2_XMIT_INTR_PEND_0	324
C66SS0_INTRTR0_IN_324	MCASP3_XMIT_INTR_PEND_0	325
C66SS0_INTRTR0_IN_325	MCASP4_XMIT_INTR_PEND_0	326
C66SS0_INTRTR0_IN_326	MCASP5_XMIT_INTR_PEND_0	327
C66SS0_INTRTR0_IN_327	MCASP6_XMIT_INTR_PEND_0	328
C66SS0_INTRTR0_IN_328	MCASP7_XMIT_INTR_PEND_0	329
C66SS0_INTRTR0_IN_329	MCASP8_XMIT_INTR_PEND_0	330
C66SS0_INTRTR0_IN_330	MCASP9_XMIT_INTR_PEND_0	331
C66SS0_INTRTR0_IN_331	MCASP10_XMIT_INTR_PEND_0	332
C66SS0_INTRTR0_IN_332	MCASP11_XMIT_INTR_PEND_0	333
C66SS0_INTRTR0_IN_333	AASRC0_INFIFO_LEVEL_0	334
C66SS0_INTRTR0_IN_334	AASRC0_INGROUP_LEVEL_0	335
C66SS0_INTRTR0_IN_335	AASRC0_OUTFIFO_LEVEL_0	336
C66SS0_INTRTR0_IN_336	AASRC0_OUTGROUP_LEVEL_0	337
C66SS0_INTRTR0_IN_337	AASRC0_ERR_LEVEL_0	338



**Table 9-55. C66SS0\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	C66SS0_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
C66SS0_INTRTR0_IN_338	I3C0_I3C_INT_0	339
C66SS0_INTRTR0_IN_339	MCSPi0_INTR_SPI_0	340
C66SS0_INTRTR0_IN_340	MCSPi1_INTR_SPI_0	341
C66SS0_INTRTR0_IN_341	MCSPi2_INTR_SPI_0	342
C66SS0_INTRTR0_IN_342	MCSPi7_INTR_SPI_0	343
C66SS0_INTRTR0_IN_343	CMPEVENT_INTRTR0_OUTP_12	344
C66SS0_INTRTR0_IN_344	CMPEVENT_INTRTR0_OUTP_13	345
C66SS0_INTRTR0_IN_345	CMPEVENT_INTRTR0_OUTP_14	346
C66SS0_INTRTR0_IN_346	CMPEVENT_INTRTR0_OUTP_15	347
C66SS0_INTRTR0_IN_347	I2C0_POINTRPEND_0	348
C66SS0_INTRTR0_IN_348	I2C1_POINTRPEND_0	349
C66SS0_INTRTR0_IN_349	I2C2_POINTRPEND_0	350
C66SS0_INTRTR0_IN_350	UART0_USART_IRQ_0	351
C66SS0_INTRTR0_IN_351	UART1_USART_IRQ_0	352
C66SS0_INTRTR0_IN_352	UART2_USART_IRQ_0	353
C66SS0_INTRTR0_IN_353	MCU_I3C0_I3C_INT_0	354
C66SS0_INTRTR0_IN_354	MCU_I3C1_I3C_INT_0	355
C66SS0_INTRTR0_IN_355	C66DEBUGSS0_AQCMPINTR_LEVEL_0	356
C66SS0_INTRTR0_IN_356	C66SS0_KSBUS_RAT_0_C66_RAT_INTR_0	357
C66SS0_INTRTR0_IN_357	NAVSS0_INTR_ROUTER_0_OUTL_INTR_320	358
C66SS0_INTRTR0_IN_358	NAVSS0_INTR_ROUTER_0_OUTL_INTR_321	359
C66SS0_INTRTR0_IN_359	NAVSS0_INTR_ROUTER_0_OUTL_INTR_322	360
C66SS0_INTRTR0_IN_360	NAVSS0_INTR_ROUTER_0_OUTL_INTR_323	361
C66SS0_INTRTR0_IN_361	NAVSS0_INTR_ROUTER_0_OUTL_INTR_324	362
C66SS0_INTRTR0_IN_362	NAVSS0_INTR_ROUTER_0_OUTL_INTR_325	363
C66SS0_INTRTR0_IN_363	NAVSS0_INTR_ROUTER_0_OUTL_INTR_326	364
C66SS0_INTRTR0_IN_364	NAVSS0_INTR_ROUTER_0_OUTL_INTR_327	365
C66SS0_INTRTR0_IN_365	NAVSS0_INTR_ROUTER_0_OUTL_INTR_328	366
C66SS0_INTRTR0_IN_366	NAVSS0_INTR_ROUTER_0_OUTL_INTR_329	367
C66SS0_INTRTR0_IN_367	NAVSS0_INTR_ROUTER_0_OUTL_INTR_330	368
C66SS0_INTRTR0_IN_368	NAVSS0_INTR_ROUTER_0_OUTL_INTR_331	369
C66SS0_INTRTR0_IN_369	NAVSS0_INTR_ROUTER_0_OUTL_INTR_332	370
C66SS0_INTRTR0_IN_370	NAVSS0_INTR_ROUTER_0_OUTL_INTR_333	371
C66SS0_INTRTR0_IN_371	NAVSS0_INTR_ROUTER_0_OUTL_INTR_334	372
C66SS0_INTRTR0_IN_372	NAVSS0_INTR_ROUTER_0_OUTL_INTR_335	373
C66SS0_INTRTR0_IN_373	NAVSS0_INTR_ROUTER_0_OUTL_INTR_336	374
C66SS0_INTRTR0_IN_374	NAVSS0_INTR_ROUTER_0_OUTL_INTR_337	375
C66SS0_INTRTR0_IN_375	NAVSS0_INTR_ROUTER_0_OUTL_INTR_338	376
C66SS0_INTRTR0_IN_376	NAVSS0_INTR_ROUTER_0_OUTL_INTR_339	377
C66SS0_INTRTR0_IN_377	NAVSS0_INTR_ROUTER_0_OUTL_INTR_340	378
C66SS0_INTRTR0_IN_378	NAVSS0_INTR_ROUTER_0_OUTL_INTR_341	379
C66SS0_INTRTR0_IN_379	NAVSS0_INTR_ROUTER_0_OUTL_INTR_342	380
C66SS0_INTRTR0_IN_380	NAVSS0_INTR_ROUTER_0_OUTL_INTR_343	381
C66SS0_INTRTR0_IN_381	MCAN0_MCANSS_MCAN_LVL_INT_0	382



**Table 9-55. C66SS0\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	C66SS0_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
C66SS0_INTRTR0_IN_382	MCAN0_MCANSS_MCAN_LVL_INT_1	383
C66SS0_INTRTR0_IN_383	MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	384
C66SS0_INTRTR0_IN_384	MCAN1_MCANSS_MCAN_LVL_INT_0	385
C66SS0_INTRTR0_IN_385	MCAN1_MCANSS_MCAN_LVL_INT_1	386
C66SS0_INTRTR0_IN_386	MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	387
C66SS0_INTRTR0_IN_387	MLB0_MLBSS_MLB_AHB_INT_0	388
C66SS0_INTRTR0_IN_388	MLB0_MLBSS_MLB_AHB_INT_1	389
C66SS0_INTRTR0_IN_389	MLB0_MLBSS_MLB_INT_0	390
C66SS0_INTRTR0_IN_390	GPIOMUX_INTRTR0_OUTP_32	391
C66SS0_INTRTR0_IN_391	GPIOMUX_INTRTR0_OUTP_33	392
C66SS0_INTRTR0_IN_392	GPIOMUX_INTRTR0_OUTP_34	393
C66SS0_INTRTR0_IN_393	GPIOMUX_INTRTR0_OUTP_35	394
C66SS0_INTRTR0_IN_394	GPIOMUX_INTRTR0_OUTP_36	395
C66SS0_INTRTR0_IN_395	GPIOMUX_INTRTR0_OUTP_37	396
C66SS0_INTRTR0_IN_396	GPIOMUX_INTRTR0_OUTP_38	397
C66SS0_INTRTR0_IN_397	GPIOMUX_INTRTR0_OUTP_39	398
C66SS0_INTRTR0_IN_398	MCU_ADC0_GEN_LEVEL_0	399
C66SS0_INTRTR0_IN_399	MCU_ADC1_GEN_LEVEL_0	400

- (1) Do not use that setting. As it is the default one, it is strongly recommended to set the field to another value before enabling interrupt router operation.

### 9.4.3.11 C66SS1\_INTRTR0 Interrupt Map

Table 9-56 lists all the interrupts that are mapped to the C66SS1\_INTRTR0 inputs, along with the corresponding register programming values used for mux control. Any C66SS1\_INTRTR0 interrupt input that is not listed in this table should be considered as unused.

**Table 9-56. C66SS1\_INTRTR0 Interrupt Map**

Interrupt Input Line	Peripheral Interrupt	C66SS1_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
RESERVED	RESERVED	0 <sup>(1)</sup>
C66SS1_INTRTR0_IN_0	TIMER0_INTR_PEND_0	1
C66SS1_INTRTR0_IN_1	TIMER1_INTR_PEND_0	2
C66SS1_INTRTR0_IN_2	TIMER2_INTR_PEND_0	3
C66SS1_INTRTR0_IN_3	TIMER3_INTR_PEND_0	4
C66SS1_INTRTR0_IN_4	TIMER4_INTR_PEND_0	5
C66SS1_INTRTR0_IN_5	TIMER5_INTR_PEND_0	6
C66SS1_INTRTR0_IN_6	TIMER6_INTR_PEND_0	7
C66SS1_INTRTR0_IN_7	TIMER7_INTR_PEND_0	8
C66SS1_INTRTR0_IN_8	TIMER8_INTR_PEND_0	9
C66SS1_INTRTR0_IN_9	TIMER9_INTR_PEND_0	10
C66SS1_INTRTR0_IN_10	TIMER10_INTR_PEND_0	11
C66SS1_INTRTR0_IN_11	TIMER11_INTR_PEND_0	12
C66SS1_INTRTR0_IN_12	TIMER16_INTR_PEND_0	13
C66SS1_INTRTR0_IN_13	TIMER17_INTR_PEND_0	14
C66SS1_INTRTR0_IN_14	TIMER18_INTR_PEND_0	15
C66SS1_INTRTR0_IN_15	TIMER19_INTR_PEND_0	16
C66SS1_INTRTR0_IN_16	MCSP13_INTR_SPI_0	17
C66SS1_INTRTR0_IN_17	MCSP14_INTR_SPI_0	18
C66SS1_INTRTR0_IN_18	MCSP15_INTR_SPI_0	19
C66SS1_INTRTR0_IN_19	MCSP16_INTR_SPI_0	20
C66SS1_INTRTR0_IN_20	MCAN2_MCANSS_MCAN_LVL_INT_0	21
C66SS1_INTRTR0_IN_21	MCAN2_MCANSS_MCAN_LVL_INT_1	22
C66SS1_INTRTR0_IN_22	MCAN2_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	23
C66SS1_INTRTR0_IN_23	MCAN3_MCANSS_MCAN_LVL_INT_0	24
C66SS1_INTRTR0_IN_24	MCAN3_MCANSS_MCAN_LVL_INT_1	25
C66SS1_INTRTR0_IN_25	MCAN3_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	26
C66SS1_INTRTR0_IN_26	I2C3_POINTRPEND_0	27
C66SS1_INTRTR0_IN_27	I2C4_POINTRPEND_0	28
C66SS1_INTRTR0_IN_28	I2C5_POINTRPEND_0	29
C66SS1_INTRTR0_IN_29	I2C6_POINTRPEND_0	30
C66SS1_INTRTR0_IN_30	GLUELOGIC_SOCA_INT_GLUE_SOCA_INT_0	31
C66SS1_INTRTR0_IN_31	GLUELOGIC_SOCB_INT_GLUE_SOCB_INT_0	32
C66SS1_INTRTR0_IN_32	EHRPWM0_EPWM_ETINT_0	33
C66SS1_INTRTR0_IN_33	EHRPWM1_EPWM_ETINT_0	34
C66SS1_INTRTR0_IN_34	EHRPWM2_EPWM_ETINT_0	35
C66SS1_INTRTR0_IN_35	EHRPWM3_EPWM_ETINT_0	36
C66SS1_INTRTR0_IN_36	EHRPWM4_EPWM_ETINT_0	37
C66SS1_INTRTR0_IN_37	EHRPWM5_EPWM_ETINT_0	38
C66SS1_INTRTR0_IN_38	EHRPWM0_EPWM_TRIPZINT_0	39

**Table 9-56. C66SS1\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	C66SS1_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
C66SS1_INTRTR0_IN_39	EHRPWM1_EPWM_TRIPZINT_0	40
C66SS1_INTRTR0_IN_40	EHRPWM2_EPWM_TRIPZINT_0	41
C66SS1_INTRTR0_IN_41	EHRPWM3_EPWM_TRIPZINT_0	42
C66SS1_INTRTR0_IN_42	EHRPWM4_EPWM_TRIPZINT_0	43
C66SS1_INTRTR0_IN_43	EHRPWM5_EPWM_TRIPZINT_0	44
C66SS1_INTRTR0_IN_44	ECAP0_ECAP_INT_0	45
C66SS1_INTRTR0_IN_45	ECAP1_ECAP_INT_0	46
C66SS1_INTRTR0_IN_46	ECAP2_ECAP_INT_0	47
C66SS1_INTRTR0_IN_47	EQEP0_EQEP_INT_0	48
C66SS1_INTRTR0_IN_48	EQEP1_EQEP_INT_0	49
C66SS1_INTRTR0_IN_49	EQEP2_EQEP_INT_0	50
C66SS1_INTRTR0_IN_50	UART3_USART_IRQ_0	51
C66SS1_INTRTR0_IN_51	UART4_USART_IRQ_0	52
C66SS1_INTRTR0_IN_52	UART5_USART_IRQ_0	53
C66SS1_INTRTR0_IN_53	UART6_USART_IRQ_0	54
C66SS1_INTRTR0_IN_54	UART7_USART_IRQ_0	55
C66SS1_INTRTR0_IN_55	UART8_USART_IRQ_0	56
C66SS1_INTRTR0_IN_56	UART9_USART_IRQ_0	57
C66SS1_INTRTR0_IN_57	MCAN4_MCANSS_MCAN_LVL_INT_0	58
C66SS1_INTRTR0_IN_58	MCAN4_MCANSS_MCAN_LVL_INT_1	59
C66SS1_INTRTR0_IN_59	MCAN4_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	60
C66SS1_INTRTR0_IN_60	MCAN5_MCANSS_MCAN_LVL_INT_0	61
C66SS1_INTRTR0_IN_61	MCAN5_MCANSS_MCAN_LVL_INT_1	62
C66SS1_INTRTR0_IN_62	MCAN5_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	63
C66SS1_INTRTR0_IN_63	MCAN6_MCANSS_MCAN_LVL_INT_0	64
C66SS1_INTRTR0_IN_64	MCAN6_MCANSS_MCAN_LVL_INT_1	65
C66SS1_INTRTR0_IN_65	MCAN6_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	66
C66SS1_INTRTR0_IN_66	MCAN7_MCANSS_MCAN_LVL_INT_0	67
C66SS1_INTRTR0_IN_67	MCAN7_MCANSS_MCAN_LVL_INT_1	68
C66SS1_INTRTR0_IN_68	MCAN7_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	69
C66SS1_INTRTR0_IN_69	MCAN8_MCANSS_MCAN_LVL_INT_0	70
C66SS1_INTRTR0_IN_70	MCAN8_MCANSS_MCAN_LVL_INT_1	71
C66SS1_INTRTR0_IN_71	MCAN8_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	72
C66SS1_INTRTR0_IN_72	MCAN9_MCANSS_MCAN_LVL_INT_0	73
C66SS1_INTRTR0_IN_73	MCAN9_MCANSS_MCAN_LVL_INT_1	74
C66SS1_INTRTR0_IN_74	MCAN9_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	75
C66SS1_INTRTR0_IN_75	MCAN10_MCANSS_MCAN_LVL_INT_0	76
C66SS1_INTRTR0_IN_76	MCAN10_MCANSS_MCAN_LVL_INT_1	77
C66SS1_INTRTR0_IN_77	MCAN10_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	78
C66SS1_INTRTR0_IN_78	MCAN11_MCANSS_MCAN_LVL_INT_0	79
C66SS1_INTRTR0_IN_79	MCAN11_MCANSS_MCAN_LVL_INT_1	80
C66SS1_INTRTR0_IN_80	MCAN11_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	81
C66SS1_INTRTR0_IN_81	MCAN12_MCANSS_MCAN_LVL_INT_0	82
C66SS1_INTRTR0_IN_82	MCAN12_MCANSS_MCAN_LVL_INT_1	83

**Table 9-56. C66SS1\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	C66SS1_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
C66SS1_INTRTR0_IN_83	MCAN12_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	84
C66SS1_INTRTR0_IN_84	MCAN13_MCANSS_MCAN_LVL_INT_0	85
C66SS1_INTRTR0_IN_85	MCAN13_MCANSS_MCAN_LVL_INT_1	86
C66SS1_INTRTR0_IN_86	MCAN13_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	87
C66SS1_INTRTR0_IN_88	GPU0_MISC_0_IRQ_0	89
C66SS1_INTRTR0_IN_96	NAVSS0_INTR_ROUTER_0_OUTL_INTR_376	97
C66SS1_INTRTR0_IN_97	NAVSS0_INTR_ROUTER_0_OUTL_INTR_377	98
C66SS1_INTRTR0_IN_98	NAVSS0_INTR_ROUTER_0_OUTL_INTR_378	99
C66SS1_INTRTR0_IN_99	NAVSS0_INTR_ROUTER_0_OUTL_INTR_379	100
C66SS1_INTRTR0_IN_100	NAVSS0_INTR_ROUTER_0_OUTL_INTR_380	101
C66SS1_INTRTR0_IN_101	NAVSS0_INTR_ROUTER_0_OUTL_INTR_381	102
C66SS1_INTRTR0_IN_102	NAVSS0_INTR_ROUTER_0_OUTL_INTR_382	103
C66SS1_INTRTR0_IN_103	NAVSS0_INTR_ROUTER_0_OUTL_INTR_383	104
C66SS1_INTRTR0_IN_104	DSS0_DSS_INST0_DISPC_FUNC_IRQ_PROC0_0	105
C66SS1_INTRTR0_IN_105	DSS0_DSS_INST0_DISPC_FUNC_IRQ_PROC1_0	106
C66SS1_INTRTR0_IN_106	DSS0_DSS_INST0_DISPC_SECURE_IRQ_PROC0_0	107
C66SS1_INTRTR0_IN_107	DSS0_DSS_INST0_DISPC_SECURE_IRQ_PROC1_0	108
C66SS1_INTRTR0_IN_108	DSS0_DSS_INST0_DISPC_SAFETY_ERROR_IRQ_PROC0_0	109
C66SS1_INTRTR0_IN_109	DSS0_DSS_INST0_DISPC_SAFETY_ERROR_IRQ_PROC1_0	110
C66SS1_INTRTR0_IN_110	DMPAC0_INTD_0_SYSTEM_INTR_LEVEL_0	111
C66SS1_INTRTR0_IN_111	DMPAC0_INTD_0_SYSTEM_INTR_LEVEL_1	112
C66SS1_INTRTR0_IN_114	GLUELOGIC_GPU_GPIO_REQACK_GLUE_GPU_GPIO_REQINT_LVL_0	115
C66SS1_INTRTR0_IN_115	GLUELOGIC_GPU_GPIO_REQACK_GLUE_GPU_GPIO_ACKINT_LVL_0	116
C66SS1_INTRTR0_IN_116	CPSW0_STAT_PEND_0	117
C66SS1_INTRTR0_IN_117	CPSW0_MDIO_PEND_0	118
C66SS1_INTRTR0_IN_118	CPSW0_EVNT_PEND_0	119
C66SS1_INTRTR0_IN_119	SA2_UL0_SA_UL_PKA_0	120
C66SS1_INTRTR0_IN_120	SA2_UL0_SA_UL_TRNG_0	121
C66SS1_INTRTR0_IN_121	ESM0_ESM_INT_LOW_LVL_0	122
C66SS1_INTRTR0_IN_122	ESM0_ESM_INT_HI_LVL_0	123
C66SS1_INTRTR0_IN_123	ESM0_ESM_INT_CFG_LVL_0	124
C66SS1_INTRTR0_IN_127	GPIOMUX_INTRTR0_OUTP_40	128
C66SS1_INTRTR0_IN_128	GPIOMUX_INTRTR0_OUTP_41	129
C66SS1_INTRTR0_IN_129	GPIOMUX_INTRTR0_OUTP_42	130
C66SS1_INTRTR0_IN_130	GPIOMUX_INTRTR0_OUTP_43	131
C66SS1_INTRTR0_IN_131	GPIOMUX_INTRTR0_OUTP_44	132
C66SS1_INTRTR0_IN_132	GPIOMUX_INTRTR0_OUTP_45	133
C66SS1_INTRTR0_IN_133	GPIOMUX_INTRTR0_OUTP_46	134
C66SS1_INTRTR0_IN_134	GPIOMUX_INTRTR0_OUTP_47	135
C66SS1_INTRTR0_IN_135	GPIOMUX_INTRTR0_OUTP_48	136
C66SS1_INTRTR0_IN_136	GPIOMUX_INTRTR0_OUTP_49	137
C66SS1_INTRTR0_IN_137	GPIOMUX_INTRTR0_OUTP_50	138
C66SS1_INTRTR0_IN_138	GPIOMUX_INTRTR0_OUTP_51	139

**Table 9-56. C66SS1\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	C66SS1_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
C66SS1_INTRTR0_IN_139	GPIOMUX_INTRTR0_OUTP_52	140
C66SS1_INTRTR0_IN_140	GPIOMUX_INTRTR0_OUTP_53	141
C66SS1_INTRTR0_IN_141	GPIOMUX_INTRTR0_OUTP_54	142
C66SS1_INTRTR0_IN_142	GPIOMUX_INTRTR0_OUTP_55	143
C66SS1_INTRTR0_IN_143	GPIOMUX_INTRTR0_OUTP_56	144
C66SS1_INTRTR0_IN_144	GPIOMUX_INTRTR0_OUTP_57	145
C66SS1_INTRTR0_IN_145	GPIOMUX_INTRTR0_OUTP_58	146
C66SS1_INTRTR0_IN_146	GPIOMUX_INTRTR0_OUTP_59	147
C66SS1_INTRTR0_IN_147	GPIOMUX_INTRTR0_OUTP_60	148
C66SS1_INTRTR0_IN_148	GPIOMUX_INTRTR0_OUTP_61	149
C66SS1_INTRTR0_IN_149	GPIOMUX_INTRTR0_OUTP_62	150
C66SS1_INTRTR0_IN_150	GPIOMUX_INTRTR0_OUTP_63	151
C66SS1_INTRTR0_IN_151	PRU_ICSSG0_PR1_HOST_INTR_PEND_0	152
C66SS1_INTRTR0_IN_152	PRU_ICSSG0_PR1_HOST_INTR_PEND_1	153
C66SS1_INTRTR0_IN_153	PRU_ICSSG0_PR1_HOST_INTR_PEND_2	154
C66SS1_INTRTR0_IN_154	PRU_ICSSG0_PR1_HOST_INTR_PEND_3	155
C66SS1_INTRTR0_IN_155	PRU_ICSSG0_PR1_HOST_INTR_PEND_4	156
C66SS1_INTRTR0_IN_156	PRU_ICSSG0_PR1_HOST_INTR_PEND_5	157
C66SS1_INTRTR0_IN_157	PRU_ICSSG0_PR1_HOST_INTR_PEND_6	158
C66SS1_INTRTR0_IN_158	PRU_ICSSG0_PR1_HOST_INTR_PEND_7	159
C66SS1_INTRTR0_IN_159	PRU_ICSSG1_PR1_HOST_INTR_PEND_0	160
C66SS1_INTRTR0_IN_160	PRU_ICSSG1_PR1_HOST_INTR_PEND_1	161
C66SS1_INTRTR0_IN_161	PRU_ICSSG1_PR1_HOST_INTR_PEND_2	162
C66SS1_INTRTR0_IN_162	PRU_ICSSG1_PR1_HOST_INTR_PEND_3	163
C66SS1_INTRTR0_IN_163	PRU_ICSSG1_PR1_HOST_INTR_PEND_4	164
C66SS1_INTRTR0_IN_164	PRU_ICSSG1_PR1_HOST_INTR_PEND_5	165
C66SS1_INTRTR0_IN_165	PRU_ICSSG1_PR1_HOST_INTR_PEND_6	166
C66SS1_INTRTR0_IN_166	PRU_ICSSG1_PR1_HOST_INTR_PEND_7	167
C66SS1_INTRTR0_IN_167	PRU_ICSSG0_PR1_TX_SOF_INTR_REQ_0	168
C66SS1_INTRTR0_IN_168	PRU_ICSSG0_PR1_TX_SOF_INTR_REQ_1	169
C66SS1_INTRTR0_IN_169	PRU_ICSSG0_PR1_RX_SOF_INTR_REQ_0	170
C66SS1_INTRTR0_IN_170	PRU_ICSSG0_PR1_RX_SOF_INTR_REQ_1	171
C66SS1_INTRTR0_IN_171	PRU_ICSSG1_PR1_TX_SOF_INTR_REQ_0	172
C66SS1_INTRTR0_IN_172	PRU_ICSSG1_PR1_TX_SOF_INTR_REQ_1	173
C66SS1_INTRTR0_IN_173	PRU_ICSSG1_PR1_RX_SOF_INTR_REQ_0	174
C66SS1_INTRTR0_IN_174	PRU_ICSSG1_PR1_RX_SOF_INTR_REQ_1	175
C66SS1_INTRTR0_IN_175	USB0_IRQ_0	176
C66SS1_INTRTR0_IN_176	USB0_IRQ_1	177
C66SS1_INTRTR0_IN_177	USB0_IRQ_2	178
C66SS1_INTRTR0_IN_178	USB0_IRQ_3	179
C66SS1_INTRTR0_IN_179	USB0_IRQ_4	180
C66SS1_INTRTR0_IN_180	USB0_IRQ_5	181
C66SS1_INTRTR0_IN_181	USB0_IRQ_6	182
C66SS1_INTRTR0_IN_182	USB0_IRQ_7	183

**Table 9-56. C66SS1\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	C66SS1_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
C66SS1_INTRTR0_IN_183	USB1_IRQ_0	184
C66SS1_INTRTR0_IN_184	USB1_IRQ_1	185
C66SS1_INTRTR0_IN_185	USB1_IRQ_2	186
C66SS1_INTRTR0_IN_186	USB1_IRQ_3	187
C66SS1_INTRTR0_IN_187	USB1_IRQ_4	188
C66SS1_INTRTR0_IN_188	USB1_IRQ_5	189
C66SS1_INTRTR0_IN_189	USB1_IRQ_6	190
C66SS1_INTRTR0_IN_190	USB1_IRQ_7	191
C66SS1_INTRTR0_IN_199	USB0_OTGIRQ_0	200
C66SS1_INTRTR0_IN_200	USB1_OTGIRQ_0	201
C66SS1_INTRTR0_IN_202	PCIE0_PCIE_LEGACY_PULSE_0	203
C66SS1_INTRTR0_IN_203	PCIE0_PCIE_DOWNSTREAM_PULSE_0	204
C66SS1_INTRTR0_IN_204	PCIE0_PCIE_FLR_PULSE_0	205
C66SS1_INTRTR0_IN_205	PCIE0_PCIE_PHY_LEVEL_0	206
C66SS1_INTRTR0_IN_206	PCIE0_PCIE_LOCAL_LEVEL_0	207
C66SS1_INTRTR0_IN_207	PCIE0_PCIE_ERROR_PULSE_0	208
C66SS1_INTRTR0_IN_208	PCIE0_PCIE_LINK_STATE_PULSE_0	209
C66SS1_INTRTR0_IN_209	PCIE0_PCIE_PWR_STATE_PULSE_0	210
C66SS1_INTRTR0_IN_210	PCIE0_PCIE_PTM_VALID_PULSE_0	211
C66SS1_INTRTR0_IN_211	PCIE0_PCIE_HOT_RESET_PULSE_0	212
C66SS1_INTRTR0_IN_212	PCIE0_PCIE_CPTS_PEND_0	213
C66SS1_INTRTR0_IN_214	PCIE1_PCIE_LEGACY_PULSE_0	215
C66SS1_INTRTR0_IN_215	PCIE1_PCIE_DOWNSTREAM_PULSE_0	216
C66SS1_INTRTR0_IN_216	PCIE1_PCIE_FLR_PULSE_0	217
C66SS1_INTRTR0_IN_217	PCIE1_PCIE_PHY_LEVEL_0	218
C66SS1_INTRTR0_IN_218	PCIE1_PCIE_LOCAL_LEVEL_0	219
C66SS1_INTRTR0_IN_219	PCIE1_PCIE_ERROR_PULSE_0	220
C66SS1_INTRTR0_IN_220	PCIE1_PCIE_LINK_STATE_PULSE_0	221
C66SS1_INTRTR0_IN_221	PCIE1_PCIE_PWR_STATE_PULSE_0	222
C66SS1_INTRTR0_IN_222	PCIE1_PCIE_PTM_VALID_PULSE_0	223
C66SS1_INTRTR0_IN_223	PCIE1_PCIE_HOT_RESET_PULSE_0	224
C66SS1_INTRTR0_IN_224	PCIE1_PCIE_CPTS_PEND_0	225
C66SS1_INTRTR0_IN_226	PCIE2_PCIE_LEGACY_PULSE_0	227
C66SS1_INTRTR0_IN_227	PCIE2_PCIE_DOWNSTREAM_PULSE_0	228
C66SS1_INTRTR0_IN_228	PCIE2_PCIE_FLR_PULSE_0	229
C66SS1_INTRTR0_IN_229	PCIE2_PCIE_PHY_LEVEL_0	230
C66SS1_INTRTR0_IN_230	PCIE2_PCIE_LOCAL_LEVEL_0	231
C66SS1_INTRTR0_IN_231	PCIE2_PCIE_ERROR_PULSE_0	232
C66SS1_INTRTR0_IN_232	PCIE2_PCIE_LINK_STATE_PULSE_0	233
C66SS1_INTRTR0_IN_233	PCIE2_PCIE_PWR_STATE_PULSE_0	234
C66SS1_INTRTR0_IN_234	PCIE2_PCIE_PTM_VALID_PULSE_0	235
C66SS1_INTRTR0_IN_235	PCIE2_PCIE_HOT_RESET_PULSE_0	236
C66SS1_INTRTR0_IN_236	PCIE2_PCIE_CPTS_PEND_0	237
C66SS1_INTRTR0_IN_238	PCIE3_PCIE_LEGACY_PULSE_0	239

**Table 9-56. C66SS1\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	C66SS1_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
C66SS1_INTRTR0_IN_239	PCIE3_PCIE_DOWNSTREAM_PULSE_0	240
C66SS1_INTRTR0_IN_240	PCIE3_PCIE_FLR_PULSE_0	241
C66SS1_INTRTR0_IN_241	PCIE3_PCIE_PHY_LEVEL_0	242
C66SS1_INTRTR0_IN_242	PCIE3_PCIE_LOCAL_LEVEL_0	243
C66SS1_INTRTR0_IN_243	PCIE3_PCIE_ERROR_PULSE_0	244
C66SS1_INTRTR0_IN_244	PCIE3_PCIE_LINK_STATE_PULSE_0	245
C66SS1_INTRTR0_IN_245	PCIE3_PCIE_PWR_STATE_PULSE_0	246
C66SS1_INTRTR0_IN_246	PCIE3_PCIE_PTM_VALID_PULSE_0	247
C66SS1_INTRTR0_IN_247	PCIE3_PCIE_HOT_RESET_PULSE_0	248
C66SS1_INTRTR0_IN_248	PCIE3_PCIE_CPTS_PEND_0	249
C66SS1_INTRTR0_IN_252	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_16	253
C66SS1_INTRTR0_IN_253	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_17	254
C66SS1_INTRTR0_IN_254	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_18	255
C66SS1_INTRTR0_IN_255	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_19	256
C66SS1_INTRTR0_IN_256	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_20	257
C66SS1_INTRTR0_IN_257	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_21	258
C66SS1_INTRTR0_IN_258	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_22	259
C66SS1_INTRTR0_IN_259	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_23	260
C66SS1_INTRTR0_IN_260	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_24	261
C66SS1_INTRTR0_IN_261	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_25	262
C66SS1_INTRTR0_IN_262	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_26	263
C66SS1_INTRTR0_IN_263	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_27	264
C66SS1_INTRTR0_IN_264	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_28	265
C66SS1_INTRTR0_IN_265	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_29	266
C66SS1_INTRTR0_IN_266	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_30	267
C66SS1_INTRTR0_IN_267	COMPUTE_CLUSTER0_CLEC_SOC_EVENTS_OUT_LEVEL_31	268
C66SS1_INTRTR0_IN_268	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_0	269
C66SS1_INTRTR0_IN_269	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_1	270
C66SS1_INTRTR0_IN_270	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_2	271
C66SS1_INTRTR0_IN_271	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_3	272
C66SS1_INTRTR0_IN_272	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_4	273
C66SS1_INTRTR0_IN_273	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_5	274
C66SS1_INTRTR0_IN_276	USB0_HOST_SYSTEM_ERROR_0	277
C66SS1_INTRTR0_IN_277	USB1_HOST_SYSTEM_ERROR_0	278
C66SS1_INTRTR0_IN_279	MCU_CPSW0_STAT_PEND_0	280
C66SS1_INTRTR0_IN_280	MCU_CPSW0_MDIO_PEND_0	281
C66SS1_INTRTR0_IN_281	MCU_CPSW0_EVNT_PEND_0	282
C66SS1_INTRTR0_IN_283	MCU_TIMER0_INTR_PEND_0	284
C66SS1_INTRTR0_IN_284	MCU_TIMER1_INTR_PEND_0	285
C66SS1_INTRTR0_IN_285	MCU_TIMER2_INTR_PEND_0	286
C66SS1_INTRTR0_IN_286	MCU_TIMER3_INTR_PEND_0	287
C66SS1_INTRTR0_IN_287	MCU_TIMER4_INTR_PEND_0	288
C66SS1_INTRTR0_IN_288	MCU_TIMER5_INTR_PEND_0	289
C66SS1_INTRTR0_IN_289	MCU_TIMER6_INTR_PEND_0	290



**Table 9-56. C66SS1\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	C66SS1_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
C66SS1_INTRTR0_IN_290	MCU_TIMER7_INTR_PEND_0	291
C66SS1_INTRTR0_IN_291	MCU_TIMER8_INTR_PEND_0	292
C66SS1_INTRTR0_IN_292	MCU_TIMER9_INTR_PEND_0	293
C66SS1_INTRTR0_IN_295	MCU_ESM0_ESM_INT_LOW_LVL_0	296
C66SS1_INTRTR0_IN_296	MCU_ESM0_ESM_INT_HI_LVL_0	297
C66SS1_INTRTR0_IN_297	MCU_ESM0_ESM_INT_CFG_LVL_0	298
C66SS1_INTRTR0_IN_298	WKUP_ESM0_ESM_INT_LOW_LVL_0	299
C66SS1_INTRTR0_IN_299	WKUP_ESM0_ESM_INT_HI_LVL_0	300
C66SS1_INTRTR0_IN_300	WKUP_ESM0_ESM_INT_CFG_LVL_0	301
C66SS1_INTRTR0_IN_303	TIMER12_INTR_PEND_0	304
C66SS1_INTRTR0_IN_304	TIMER13_INTR_PEND_0	305
C66SS1_INTRTR0_IN_305	TIMER14_INTR_PEND_0	306
C66SS1_INTRTR0_IN_306	TIMER15_INTR_PEND_0	307
C66SS1_INTRTR0_IN_307	RTI25_INTR_WWD_0	308
C66SS1_INTRTR0_IN_308	CTRL_MMR0_IPC_SET7_IPC_SET_IPCFG_0	309
C66SS1_INTRTR0_IN_309	MCASP0_REC_INTR_PEND_0	310
C66SS1_INTRTR0_IN_310	MCASP1_REC_INTR_PEND_0	311
C66SS1_INTRTR0_IN_311	MCASP2_REC_INTR_PEND_0	312
C66SS1_INTRTR0_IN_312	MCASP3_REC_INTR_PEND_0	313
C66SS1_INTRTR0_IN_313	MCASP4_REC_INTR_PEND_0	314
C66SS1_INTRTR0_IN_314	MCASP5_REC_INTR_PEND_0	315
C66SS1_INTRTR0_IN_315	MCASP6_REC_INTR_PEND_0	316
C66SS1_INTRTR0_IN_316	MCASP7_REC_INTR_PEND_0	317
C66SS1_INTRTR0_IN_317	MCASP8_REC_INTR_PEND_0	318
C66SS1_INTRTR0_IN_318	MCASP9_REC_INTR_PEND_0	319
C66SS1_INTRTR0_IN_319	MCASP10_REC_INTR_PEND_0	320
C66SS1_INTRTR0_IN_320	MCASP11_REC_INTR_PEND_0	321
C66SS1_INTRTR0_IN_321	MCASP0_XMIT_INTR_PEND_0	322
C66SS1_INTRTR0_IN_322	MCASP1_XMIT_INTR_PEND_0	323
C66SS1_INTRTR0_IN_323	MCASP2_XMIT_INTR_PEND_0	324
C66SS1_INTRTR0_IN_324	MCASP3_XMIT_INTR_PEND_0	325
C66SS1_INTRTR0_IN_325	MCASP4_XMIT_INTR_PEND_0	326
C66SS1_INTRTR0_IN_326	MCASP5_XMIT_INTR_PEND_0	327
C66SS1_INTRTR0_IN_327	MCASP6_XMIT_INTR_PEND_0	328
C66SS1_INTRTR0_IN_328	MCASP7_XMIT_INTR_PEND_0	329
C66SS1_INTRTR0_IN_329	MCASP8_XMIT_INTR_PEND_0	330
C66SS1_INTRTR0_IN_330	MCASP9_XMIT_INTR_PEND_0	331
C66SS1_INTRTR0_IN_331	MCASP10_XMIT_INTR_PEND_0	332
C66SS1_INTRTR0_IN_332	MCASP11_XMIT_INTR_PEND_0	333
C66SS1_INTRTR0_IN_333	AASRC0_INFIFO_LEVEL_0	334
C66SS1_INTRTR0_IN_334	AASRC0_INGROUP_LEVEL_0	335
C66SS1_INTRTR0_IN_335	AASRC0_OUTFIFO_LEVEL_0	336
C66SS1_INTRTR0_IN_336	AASRC0_OUTGROUP_LEVEL_0	337
C66SS1_INTRTR0_IN_337	AASRC0_ERR_LEVEL_0	338



**Table 9-56. C66SS1\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	C66SS1_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
C66SS1_INTRTR0_IN_338	I3C0_I3C_INT_0	339
C66SS1_INTRTR0_IN_339	MCSPi0_INTR_SPI_0	340
C66SS1_INTRTR0_IN_340	MCSPi1_INTR_SPI_0	341
C66SS1_INTRTR0_IN_341	MCSPi2_INTR_SPI_0	342
C66SS1_INTRTR0_IN_342	MCSPi7_INTR_SPI_0	343
C66SS1_INTRTR0_IN_343	CMPEVENT_INTRTR0_OUTP_12	344
C66SS1_INTRTR0_IN_344	CMPEVENT_INTRTR0_OUTP_13	345
C66SS1_INTRTR0_IN_345	CMPEVENT_INTRTR0_OUTP_14	346
C66SS1_INTRTR0_IN_346	CMPEVENT_INTRTR0_OUTP_15	347
C66SS1_INTRTR0_IN_347	I2C0_POINTRPEND_0	348
C66SS1_INTRTR0_IN_348	I2C1_POINTRPEND_0	349
C66SS1_INTRTR0_IN_349	I2C2_POINTRPEND_0	350
C66SS1_INTRTR0_IN_350	UART0_USART_IRQ_0	351
C66SS1_INTRTR0_IN_351	UART1_USART_IRQ_0	352
C66SS1_INTRTR0_IN_352	UART2_USART_IRQ_0	353
C66SS1_INTRTR0_IN_353	MCU_I3C0_I3C_INT_0	354
C66SS1_INTRTR0_IN_354	MCU_I3C1_I3C_INT_0	355
C66SS1_INTRTR0_IN_355	C66DEBUGSS1_AQCMPINTR_LEVEL_0	356
C66SS1_INTRTR0_IN_356	C66SS1_KSBUS_RAT_0_C66_RAT_INTR_0	357
C66SS1_INTRTR0_IN_357	NAVSS0_INTR_ROUTER_0_OUTL_INTR_352	358
C66SS1_INTRTR0_IN_358	NAVSS0_INTR_ROUTER_0_OUTL_INTR_353	359
C66SS1_INTRTR0_IN_359	NAVSS0_INTR_ROUTER_0_OUTL_INTR_354	360
C66SS1_INTRTR0_IN_360	NAVSS0_INTR_ROUTER_0_OUTL_INTR_355	361
C66SS1_INTRTR0_IN_361	NAVSS0_INTR_ROUTER_0_OUTL_INTR_356	362
C66SS1_INTRTR0_IN_362	NAVSS0_INTR_ROUTER_0_OUTL_INTR_357	363
C66SS1_INTRTR0_IN_363	NAVSS0_INTR_ROUTER_0_OUTL_INTR_358	364
C66SS1_INTRTR0_IN_364	NAVSS0_INTR_ROUTER_0_OUTL_INTR_359	365
C66SS1_INTRTR0_IN_365	NAVSS0_INTR_ROUTER_0_OUTL_INTR_360	366
C66SS1_INTRTR0_IN_366	NAVSS0_INTR_ROUTER_0_OUTL_INTR_361	367
C66SS1_INTRTR0_IN_367	NAVSS0_INTR_ROUTER_0_OUTL_INTR_362	368
C66SS1_INTRTR0_IN_368	NAVSS0_INTR_ROUTER_0_OUTL_INTR_363	369
C66SS1_INTRTR0_IN_369	NAVSS0_INTR_ROUTER_0_OUTL_INTR_364	370
C66SS1_INTRTR0_IN_370	NAVSS0_INTR_ROUTER_0_OUTL_INTR_365	371
C66SS1_INTRTR0_IN_371	NAVSS0_INTR_ROUTER_0_OUTL_INTR_366	372
C66SS1_INTRTR0_IN_372	NAVSS0_INTR_ROUTER_0_OUTL_INTR_367	373
C66SS1_INTRTR0_IN_373	NAVSS0_INTR_ROUTER_0_OUTL_INTR_368	374
C66SS1_INTRTR0_IN_374	NAVSS0_INTR_ROUTER_0_OUTL_INTR_369	375
C66SS1_INTRTR0_IN_375	NAVSS0_INTR_ROUTER_0_OUTL_INTR_370	376
C66SS1_INTRTR0_IN_376	NAVSS0_INTR_ROUTER_0_OUTL_INTR_371	377
C66SS1_INTRTR0_IN_377	NAVSS0_INTR_ROUTER_0_OUTL_INTR_372	378
C66SS1_INTRTR0_IN_378	NAVSS0_INTR_ROUTER_0_OUTL_INTR_373	379
C66SS1_INTRTR0_IN_379	NAVSS0_INTR_ROUTER_0_OUTL_INTR_374	380
C66SS1_INTRTR0_IN_380	NAVSS0_INTR_ROUTER_0_OUTL_INTR_375	381
C66SS1_INTRTR0_IN_381	MCAN0_MCANSS_MCAN_LVL_INT_0	382

**Table 9-56. C66SS1\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	C66SS1_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
C66SS1_INTRTR0_IN_382	MCAN0_MCANSS_MCAN_LVL_INT_1	383
C66SS1_INTRTR0_IN_383	MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	384
C66SS1_INTRTR0_IN_384	MCAN1_MCANSS_MCAN_LVL_INT_0	385
C66SS1_INTRTR0_IN_385	MCAN1_MCANSS_MCAN_LVL_INT_1	386
C66SS1_INTRTR0_IN_386	MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	387
C66SS1_INTRTR0_IN_387	MLB0_MLBSS_MLB_AHB_INT_0	388
C66SS1_INTRTR0_IN_388	MLB0_MLBSS_MLB_AHB_INT_1	389
C66SS1_INTRTR0_IN_389	MLB0_MLBSS_MLB_INT_0	390
C66SS1_INTRTR0_IN_390	GPIOMUX_INTRTR0_OUTP_32	391
C66SS1_INTRTR0_IN_391	GPIOMUX_INTRTR0_OUTP_33	392
C66SS1_INTRTR0_IN_392	GPIOMUX_INTRTR0_OUTP_34	393
C66SS1_INTRTR0_IN_393	GPIOMUX_INTRTR0_OUTP_35	394
C66SS1_INTRTR0_IN_394	GPIOMUX_INTRTR0_OUTP_36	395
C66SS1_INTRTR0_IN_395	GPIOMUX_INTRTR0_OUTP_37	396
C66SS1_INTRTR0_IN_396	GPIOMUX_INTRTR0_OUTP_38	397
C66SS1_INTRTR0_IN_397	GPIOMUX_INTRTR0_OUTP_39	398
C66SS1_INTRTR0_IN_398	MCU_ADC0_GEN_LEVEL_0	399
C66SS1_INTRTR0_IN_399	MCU_ADC1_GEN_LEVEL_0	400

- (1) Do not use that setting. As it is the default one, it is strongly recommended to set the field to another value before enabling interrupt router operation.

### 9.4.3.12 PRU-ICSSG0 Interrupt Map

**Table 9-57. PRU-ICSSG0 Interrupt Map**

Interrupt Input Line	Interrupt ID	Interrupt Name
PRU_ICSSG0_PR1_SLV_INTR_IN_0	0	CTRL_MMR0_IPC_SET20_IPC_SET_IPCFG_0
PRU_ICSSG0_PR1_SLV_INTR_IN_1	1	CTRL_MMR0_IPC_SET21_IPC_SET_IPCFG_0
PRU_ICSSG0_PR1_SLV_INTR_IN_2	2	MCU_ADC0_GEN_LEVEL_0
PRU_ICSSG0_PR1_SLV_INTR_IN_3	3	MCU_ADC1_GEN_LEVEL_0
PRU_ICSSG0_PR1_SLV_INTR_IN_4	4	PRU_ICSSG1_PR1_HOST_INTR_PEND_5
PRU_ICSSG0_PR1_SLV_INTR_IN_5	5	PRU_ICSSG1_PR1_HOST_INTR_PEND_6
PRU_ICSSG0_PR1_SLV_INTR_IN_6	6	PRU_ICSSG1_PR1_HOST_INTR_PEND_7
PRU_ICSSG0_PR1_SLV_INTR_IN_7	7	GPMC0_GPMC_SINTERRUPT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_8	8	ECAP1_ECAP_INT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_9	9	ECAP2_ECAP_INT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_10	10	GLUELOGIC_SOCA_INT_GLUE_SOCA_INT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_11	11	GLUELOGIC_SOCB_INT_GLUE_SOCB_INT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_12	12	EHRPWM0_EPWM_ETINT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_13	13	EHRPWM1_EPWM_ETINT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_14	14	EHRPWM2_EPWM_ETINT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_15	15	EHRPWM3_EPWM_ETINT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_16	16	EHRPWM4_EPWM_ETINT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_17	17	EHRPWM5_EPWM_ETINT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_18	18	EHRPWM0_EPWM_TRIPZINT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_19	19	EHRPWM1_EPWM_TRIPZINT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_20	20	EHRPWM2_EPWM_TRIPZINT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_21	21	EHRPWM3_EPWM_TRIPZINT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_22	22	EHRPWM4_EPWM_TRIPZINT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_23	23	EHRPWM5_EPWM_TRIPZINT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_24	24	EQEP0_EQEP_INT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_25	25	EQEP1_EQEP_INT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_26	26	EQEP2_EQEP_INT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_27	27	ECAP0_ECAP_INT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_28	28	I2C0_POINTRPEND_0
PRU_ICSSG0_PR1_SLV_INTR_IN_29	29	I2C1_POINTRPEND_0
PRU_ICSSG0_PR1_SLV_INTR_IN_30	30	I2C2_POINTRPEND_0
PRU_ICSSG0_PR1_SLV_INTR_IN_31	31	I2C3_POINTRPEND_0
PRU_ICSSG0_PR1_SLV_INTR_IN_32	32	MCSPi0_INTR_SPI_0
PRU_ICSSG0_PR1_SLV_INTR_IN_33	33	MCSPi1_INTR_SPI_0
PRU_ICSSG0_PR1_SLV_INTR_IN_34	34	MCSPi2_INTR_SPI_0
PRU_ICSSG0_PR1_SLV_INTR_IN_35	35	MCSPi3_INTR_SPI_0
PRU_ICSSG0_PR1_SLV_INTR_IN_46	46	NAVSS0_INTR_ROUTER_0_OUTL_INTR_384
PRU_ICSSG0_PR1_SLV_INTR_IN_47	47	NAVSS0_INTR_ROUTER_0_OUTL_INTR_385
PRU_ICSSG0_PR1_SLV_INTR_IN_48	48	NAVSS0_INTR_ROUTER_0_OUTL_INTR_386
PRU_ICSSG0_PR1_SLV_INTR_IN_49	49	NAVSS0_INTR_ROUTER_0_OUTL_INTR_387
PRU_ICSSG0_PR1_SLV_INTR_IN_50	50	NAVSS0_INTR_ROUTER_0_OUTL_INTR_388
PRU_ICSSG0_PR1_SLV_INTR_IN_51	51	NAVSS0_INTR_ROUTER_0_OUTL_INTR_389
PRU_ICSSG0_PR1_SLV_INTR_IN_52	52	NAVSS0_INTR_ROUTER_0_OUTL_INTR_390
PRU_ICSSG0_PR1_SLV_INTR_IN_53	53	NAVSS0_INTR_ROUTER_0_OUTL_INTR_391

**Table 9-57. PRU-ICSSG0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
PRU_ICSSG0_PR1_SLV_INTR_IN_54	54	MCASP0_XMIT_INTR_PEND_0
PRU_ICSSG0_PR1_SLV_INTR_IN_55	55	MCASP0_REC_INTR_PEND_0
PRU_ICSSG0_PR1_SLV_INTR_IN_56	56	MCASP1_XMIT_INTR_PEND_0
PRU_ICSSG0_PR1_SLV_INTR_IN_57	57	MCASP1_REC_INTR_PEND_0
PRU_ICSSG0_PR1_SLV_INTR_IN_58	58	MCASP2_XMIT_INTR_PEND_0
PRU_ICSSG0_PR1_SLV_INTR_IN_59	59	MCASP2_REC_INTR_PEND_0
PRU_ICSSG0_PR1_SLV_INTR_IN_60	60	UART0_USART_IRQ_0
PRU_ICSSG0_PR1_SLV_INTR_IN_61	61	UART1_USART_IRQ_0
PRU_ICSSG0_PR1_SLV_INTR_IN_62	62	UART2_USART_IRQ_0
PRU_ICSSG0_PR1_SLV_INTR_IN_63	63	UART3_USART_IRQ_0
PRU_ICSSG0_PR1_SLV_INTR_IN_64	64	UART4_USART_IRQ_0
PRU_ICSSG0_PR1_SLV_INTR_IN_65	65	UART5_USART_IRQ_0
PRU_ICSSG0_PR1_SLV_INTR_IN_66	66	UART6_USART_IRQ_0
PRU_ICSSG0_PR1_SLV_INTR_IN_67	67	UART7_USART_IRQ_0
PRU_ICSSG0_PR1_SLV_INTR_IN_68	68	UART8_USART_IRQ_0
PRU_ICSSG0_PR1_SLV_INTR_IN_69	69	UART9_USART_IRQ_0
PRU_ICSSG0_PR1_SLV_INTR_IN_72	72	MCAN0_MCANSS_MCAN_LVL_INT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_73	73	MCAN0_MCANSS_MCAN_LVL_INT_1
PRU_ICSSG0_PR1_SLV_INTR_IN_74	74	MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_75	75	MCAN1_MCANSS_MCAN_LVL_INT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_76	76	MCAN1_MCANSS_MCAN_LVL_INT_1
PRU_ICSSG0_PR1_SLV_INTR_IN_77	77	MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_78	78	MCAN2_MCANSS_MCAN_LVL_INT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_79	79	MCAN2_MCANSS_MCAN_LVL_INT_1
PRU_ICSSG0_PR1_SLV_INTR_IN_80	80	MCAN2_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_81	81	MCAN3_MCANSS_MCAN_LVL_INT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_82	82	MCAN3_MCANSS_MCAN_LVL_INT_1
PRU_ICSSG0_PR1_SLV_INTR_IN_83	83	MCAN3_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_84	84	MCAN4_MCANSS_MCAN_LVL_INT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_85	85	MCAN4_MCANSS_MCAN_LVL_INT_1
PRU_ICSSG0_PR1_SLV_INTR_IN_86	86	MCAN4_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_87	87	MCAN5_MCANSS_MCAN_LVL_INT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_88	88	MCAN5_MCANSS_MCAN_LVL_INT_1
PRU_ICSSG0_PR1_SLV_INTR_IN_89	89	MCAN5_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_90	90	MCAN6_MCANSS_MCAN_LVL_INT_0
PRU_ICSSG0_PR1_SLV_INTR_IN_91	91	MCAN6_MCANSS_MCAN_LVL_INT_1
PRU_ICSSG0_PR1_SLV_INTR_IN_92	92	MCAN6_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
PRU_ICSSG0_PR1_IEP0_CAP_INTR_REQ0	0	GPIOMUX_INTRTR0_OUTP_52
PRU_ICSSG0_PR1_IEP0_CAP_INTR_REQ1	1	GPIOMUX_INTRTR0_OUTP_53
PRU_ICSSG0_PR1_IEP0_CAP_INTR_REQ2	2	GPIOMUX_INTRTR0_OUTP_54
PRU_ICSSG0_PR1_IEP0_CAP_INTR_REQ3	3	GPIOMUX_INTRTR0_OUTP_55
PRU_ICSSG0_PR1_IEP0_CAP_INTR_REQ4	4	GPIOMUX_INTRTR0_OUTP_56
PRU_ICSSG0_PR1_IEP0_CAP_INTR_REQ5	5	GPIOMUX_INTRTR0_OUTP_57
PRU_ICSSG0_PR1_IEP1_CAP_INTR_REQ0	0	GPIOMUX_INTRTR0_OUTP_58

**Table 9-57. PRU-ICSSG0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
PRU_ICSSG0_PR1_IEP1_CAP_INTR_REQ1	1	GPIOMUX_INTRTR0_OUTP_59
PRU_ICSSG0_PR1_IEP1_CAP_INTR_REQ2	2	GPIOMUX_INTRTR0_OUTP_60
PRU_ICSSG0_PR1_IEP1_CAP_INTR_REQ3	3	GPIOMUX_INTRTR0_OUTP_61
PRU_ICSSG0_PR1_IEP1_CAP_INTR_REQ4	4	GPIOMUX_INTRTR0_OUTP_62
PRU_ICSSG0_PR1_IEP1_CAP_INTR_REQ5	5	GPIOMUX_INTRTR0_OUTP_63

### 9.4.3.13 PRU-ICSSG1 Interrupt Map

**Table 9-58. PRU-ICSSG1 Interrupt Map**

Interrupt Input Line	Interrupt ID	Interrupt Name
PRU_ICSSG1_PR1_SLV_INTR_IN_0	0	CTRL_MMR0_IPC_SET22_IPC_SET_IPCFG_0
PRU_ICSSG1_PR1_SLV_INTR_IN_1	1	CTRL_MMR0_IPC_SET23_IPC_SET_IPCFG_0
PRU_ICSSG1_PR1_SLV_INTR_IN_2	2	MCU_ADC0_GEN_LEVEL_0
PRU_ICSSG1_PR1_SLV_INTR_IN_3	3	MCU_ADC1_GEN_LEVEL_0
PRU_ICSSG1_PR1_SLV_INTR_IN_4	4	PRU_ICSSG0_PR1_HOST_INTR_PEND_5
PRU_ICSSG1_PR1_SLV_INTR_IN_5	5	PRU_ICSSG0_PR1_HOST_INTR_PEND_6
PRU_ICSSG1_PR1_SLV_INTR_IN_6	6	PRU_ICSSG0_PR1_HOST_INTR_PEND_7
PRU_ICSSG1_PR1_SLV_INTR_IN_7	7	GPMC0_GPMC_SINTERRUPT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_8	8	ECAP1_ECAP_INT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_9	9	ECAP2_ECAP_INT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_10	10	GLUELOGIC_SOCA_INT_GLUE_SOCA_INT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_11	11	GLUELOGIC_SOCB_INT_GLUE_SOCB_INT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_12	12	EHRPWM0_EPWM_ETINT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_13	13	EHRPWM1_EPWM_ETINT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_14	14	EHRPWM2_EPWM_ETINT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_15	15	EHRPWM3_EPWM_ETINT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_16	16	EHRPWM4_EPWM_ETINT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_17	17	EHRPWM5_EPWM_ETINT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_18	18	EHRPWM0_EPWM_TRIPZINT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_19	19	EHRPWM1_EPWM_TRIPZINT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_20	20	EHRPWM2_EPWM_TRIPZINT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_21	21	EHRPWM3_EPWM_TRIPZINT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_22	22	EHRPWM4_EPWM_TRIPZINT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_23	23	EHRPWM5_EPWM_TRIPZINT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_24	24	EQEP0_EQEP_INT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_25	25	EQEP1_EQEP_INT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_26	26	EQEP2_EQEP_INT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_27	27	ECAP0_ECAP_INT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_28	28	I2C0_POINTRPEND_0
PRU_ICSSG1_PR1_SLV_INTR_IN_29	29	I2C1_POINTRPEND_0
PRU_ICSSG1_PR1_SLV_INTR_IN_30	30	I2C2_POINTRPEND_0
PRU_ICSSG1_PR1_SLV_INTR_IN_31	31	I2C3_POINTRPEND_0
PRU_ICSSG1_PR1_SLV_INTR_IN_32	32	MCSPi0_INTR_SPI_0
PRU_ICSSG1_PR1_SLV_INTR_IN_33	33	MCSPi1_INTR_SPI_0
PRU_ICSSG1_PR1_SLV_INTR_IN_34	34	MCSPi2_INTR_SPI_0
PRU_ICSSG1_PR1_SLV_INTR_IN_35	35	MCSPi3_INTR_SPI_0
PRU_ICSSG1_PR1_SLV_INTR_IN_46	46	NAVSS0_INTR_ROUTER_0_OUTL_INTR_392
PRU_ICSSG1_PR1_SLV_INTR_IN_47	47	NAVSS0_INTR_ROUTER_0_OUTL_INTR_393
PRU_ICSSG1_PR1_SLV_INTR_IN_48	48	NAVSS0_INTR_ROUTER_0_OUTL_INTR_394
PRU_ICSSG1_PR1_SLV_INTR_IN_49	49	NAVSS0_INTR_ROUTER_0_OUTL_INTR_395
PRU_ICSSG1_PR1_SLV_INTR_IN_50	50	NAVSS0_INTR_ROUTER_0_OUTL_INTR_396
PRU_ICSSG1_PR1_SLV_INTR_IN_51	51	NAVSS0_INTR_ROUTER_0_OUTL_INTR_397
PRU_ICSSG1_PR1_SLV_INTR_IN_52	52	NAVSS0_INTR_ROUTER_0_OUTL_INTR_398
PRU_ICSSG1_PR1_SLV_INTR_IN_53	53	NAVSS0_INTR_ROUTER_0_OUTL_INTR_399

**Table 9-58. PRU-ICSSG1 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
PRU_ICSSG1_PR1_SLV_INTR_IN_54	54	MCASP0_XMIT_INTR_PEND_0
PRU_ICSSG1_PR1_SLV_INTR_IN_55	55	MCASP0_REC_INTR_PEND_0
PRU_ICSSG1_PR1_SLV_INTR_IN_56	56	MCASP1_XMIT_INTR_PEND_0
PRU_ICSSG1_PR1_SLV_INTR_IN_57	57	MCASP1_REC_INTR_PEND_0
PRU_ICSSG1_PR1_SLV_INTR_IN_58	58	MCASP2_XMIT_INTR_PEND_0
PRU_ICSSG1_PR1_SLV_INTR_IN_59	59	MCASP2_REC_INTR_PEND_0
PRU_ICSSG1_PR1_SLV_INTR_IN_60	60	UART0_USART_IRQ_0
PRU_ICSSG1_PR1_SLV_INTR_IN_61	61	UART1_USART_IRQ_0
PRU_ICSSG1_PR1_SLV_INTR_IN_62	62	UART2_USART_IRQ_0
PRU_ICSSG1_PR1_SLV_INTR_IN_63	63	UART3_USART_IRQ_0
PRU_ICSSG1_PR1_SLV_INTR_IN_64	64	UART4_USART_IRQ_0
PRU_ICSSG1_PR1_SLV_INTR_IN_65	65	UART5_USART_IRQ_0
PRU_ICSSG1_PR1_SLV_INTR_IN_66	66	UART6_USART_IRQ_0
PRU_ICSSG1_PR1_SLV_INTR_IN_67	67	UART7_USART_IRQ_0
PRU_ICSSG1_PR1_SLV_INTR_IN_68	68	UART8_USART_IRQ_0
PRU_ICSSG1_PR1_SLV_INTR_IN_69	69	UART9_USART_IRQ_0
PRU_ICSSG1_PR1_SLV_INTR_IN_72	72	MCAN7_MCANSS_MCAN_LVL_INT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_73	73	MCAN7_MCANSS_MCAN_LVL_INT_1
PRU_ICSSG1_PR1_SLV_INTR_IN_74	74	MCAN7_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_75	75	MCAN8_MCANSS_MCAN_LVL_INT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_76	76	MCAN8_MCANSS_MCAN_LVL_INT_1
PRU_ICSSG1_PR1_SLV_INTR_IN_77	77	MCAN8_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_78	78	MCAN9_MCANSS_MCAN_LVL_INT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_79	79	MCAN9_MCANSS_MCAN_LVL_INT_1
PRU_ICSSG1_PR1_SLV_INTR_IN_80	80	MCAN9_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_81	81	MCAN10_MCANSS_MCAN_LVL_INT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_82	82	MCAN10_MCANSS_MCAN_LVL_INT_1
PRU_ICSSG1_PR1_SLV_INTR_IN_83	83	MCAN10_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_84	84	MCAN11_MCANSS_MCAN_LVL_INT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_85	85	MCAN11_MCANSS_MCAN_LVL_INT_1
PRU_ICSSG1_PR1_SLV_INTR_IN_86	86	MCAN11_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_87	87	MCAN12_MCANSS_MCAN_LVL_INT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_88	88	MCAN12_MCANSS_MCAN_LVL_INT_1
PRU_ICSSG1_PR1_SLV_INTR_IN_89	89	MCAN12_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_90	90	MCAN13_MCANSS_MCAN_LVL_INT_0
PRU_ICSSG1_PR1_SLV_INTR_IN_91	91	MCAN13_MCANSS_MCAN_LVL_INT_1
PRU_ICSSG1_PR1_SLV_INTR_IN_92	92	MCAN13_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0
PRU_ICSSG1_PR1_IEP0_CAP_INTR_REQ0	0	GPIOMUX_INTRTR0_OUTP_52
PRU_ICSSG1_PR1_IEP0_CAP_INTR_REQ1	1	GPIOMUX_INTRTR0_OUTP_53
PRU_ICSSG1_PR1_IEP0_CAP_INTR_REQ2	2	GPIOMUX_INTRTR0_OUTP_54
PRU_ICSSG1_PR1_IEP0_CAP_INTR_REQ3	3	GPIOMUX_INTRTR0_OUTP_55
PRU_ICSSG1_PR1_IEP0_CAP_INTR_REQ4	4	GPIOMUX_INTRTR0_OUTP_56
PRU_ICSSG1_PR1_IEP0_CAP_INTR_REQ5	5	GPIOMUX_INTRTR0_OUTP_57
PRU_ICSSG1_PR1_IEP1_CAP_INTR_REQ0	0	GPIOMUX_INTRTR0_OUTP_58

**Table 9-58. PRU-ICSSG1 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
PRU_ICSSG1_PR1_IEP1_CAP_INTR_REQ1	1	GPIOMUX_INTRTR0_OUTP_59
PRU_ICSSG1_PR1_IEP1_CAP_INTR_REQ2	2	GPIOMUX_INTRTR0_OUTP_60
PRU_ICSSG1_PR1_IEP1_CAP_INTR_REQ3	3	GPIOMUX_INTRTR0_OUTP_61
PRU_ICSSG1_PR1_IEP1_CAP_INTR_REQ4	4	GPIOMUX_INTRTR0_OUTP_62
PRU_ICSSG1_PR1_IEP1_CAP_INTR_REQ5	5	GPIOMUX_INTRTR0_OUTP_63



#### 9.4.3.14 MAIN2MCU\_LVL\_INTRTR0 Interrupt Map

Table 9-59 lists all the interrupts that are mapped to the MAIN2MCU\_LVL\_INTRTR0 inputs, along with the corresponding register programming values used for mux control. Any MAIN2MCU\_LVL\_INTRTR0 interrupt input that is not listed in this table should be considered as unused.

**Table 9-59. MAIN2MCU\_LVL\_INTRTR0 Interrupt Map**

Interrupt Input Line	Peripheral Interrupt	MAIN2MCU_LVL_INTRTR0_M UXCNTL_n[8-0] ENABLE Field Value (DEC)
MAIN2MCU_LVL_INTRTR0_IN_0	RESERVED	0
MAIN2MCU_LVL_INTRTR0_IN_1	GLUELOGIC_EXT_INTN_GLUE_EXT_INT_LVL_0	1
MAIN2MCU_LVL_INTRTR0_IN_4	SA2_UL0_SA_UL_TRNG_0	4
MAIN2MCU_LVL_INTRTR0_IN_5	SA2_UL0_SA_UL_PKA_0	5
MAIN2MCU_LVL_INTRTR0_IN_6	CTRL_MMR0_ACCESS_ERR_0	6
MAIN2MCU_LVL_INTRTR0_IN_7	ELM0_ELM_POROCPSINTERRUPT_LVL_0	7
MAIN2MCU_LVL_INTRTR0_IN_8	GPMC0_GPMC_SINTERRUPT_0	8
MAIN2MCU_LVL_INTRTR0_IN_9	DDR0_DDRSS_PLL_FREQ_CHANGE_REQ_0	9
MAIN2MCU_LVL_INTRTR0_IN_10	DDR0_DDRSS_CONTROLLER_0	10
MAIN2MCU_LVL_INTRTR0_IN_11	DDR0_DDRSS_V2A_OTHER_ERR_LVL_0	11
MAIN2MCU_LVL_INTRTR0_IN_12	DDR0_DDRSS_HS_PHY_GLOBAL_ERROR_0	12
MAIN2MCU_LVL_INTRTR0_IN_13	CCDEBUGSS0_AQCMPINTR_LEVEL_0	13
MAIN2MCU_LVL_INTRTR0_IN_14	DEBUGSS0_AQCMPINTR_LEVEL_0	14
MAIN2MCU_LVL_INTRTR0_IN_15	DEBUGSS0_CTM_LEVEL_0	15
MAIN2MCU_LVL_INTRTR0_IN_16	MCAN0_MCANSS_MCAN_LVL_INT_0	16
MAIN2MCU_LVL_INTRTR0_IN_17	MCAN0_MCANSS_MCAN_LVL_INT_1	17
MAIN2MCU_LVL_INTRTR0_IN_18	MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	18
MAIN2MCU_LVL_INTRTR0_IN_19	MCAN1_MCANSS_MCAN_LVL_INT_0	19
MAIN2MCU_LVL_INTRTR0_IN_20	MCAN1_MCANSS_MCAN_LVL_INT_1	20
MAIN2MCU_LVL_INTRTR0_IN_21	MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	21
MAIN2MCU_LVL_INTRTR0_IN_22	MCAN2_MCANSS_MCAN_LVL_INT_0	22
MAIN2MCU_LVL_INTRTR0_IN_23	MCAN2_MCANSS_MCAN_LVL_INT_1	23
MAIN2MCU_LVL_INTRTR0_IN_24	MCAN2_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	24
MAIN2MCU_LVL_INTRTR0_IN_25	MCAN3_MCANSS_MCAN_LVL_INT_0	25
MAIN2MCU_LVL_INTRTR0_IN_26	MCAN3_MCANSS_MCAN_LVL_INT_1	26
MAIN2MCU_LVL_INTRTR0_IN_27	MCAN3_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	27
MAIN2MCU_LVL_INTRTR0_IN_28	MMCS00_EMMCSS_INTR_0	28
MAIN2MCU_LVL_INTRTR0_IN_29	MMCS01_EMMCSDSS_INTR_0	29
MAIN2MCU_LVL_INTRTR0_IN_30	MMCS02_EMMCSDSS_INTR_0	30
MAIN2MCU_LVL_INTRTR0_IN_32	PRU_ICSSG0_PR1_HOST_INTR_PEND_0	32
MAIN2MCU_LVL_INTRTR0_IN_33	PRU_ICSSG0_PR1_HOST_INTR_PEND_1	33
MAIN2MCU_LVL_INTRTR0_IN_34	PRU_ICSSG0_PR1_HOST_INTR_PEND_2	34
MAIN2MCU_LVL_INTRTR0_IN_35	PRU_ICSSG0_PR1_HOST_INTR_PEND_3	35
MAIN2MCU_LVL_INTRTR0_IN_36	PRU_ICSSG0_PR1_HOST_INTR_PEND_4	36
MAIN2MCU_LVL_INTRTR0_IN_37	PRU_ICSSG0_PR1_HOST_INTR_PEND_5	37
MAIN2MCU_LVL_INTRTR0_IN_38	PRU_ICSSG0_PR1_HOST_INTR_PEND_6	38
MAIN2MCU_LVL_INTRTR0_IN_39	PRU_ICSSG0_PR1_HOST_INTR_PEND_7	39
MAIN2MCU_LVL_INTRTR0_IN_40	PRU_ICSSG1_PR1_HOST_INTR_PEND_0	40
MAIN2MCU_LVL_INTRTR0_IN_41	PRU_ICSSG1_PR1_HOST_INTR_PEND_1	41

**Table 9-59. MAIN2MCU\_LVL\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	MAIN2MCU_LVL_INTRTR0_M UXCNTL_n[8-0] ENABLE Field Value (DEC)
MAIN2MCU_LVL_INTRTR0_IN_42	PRU_ICSSG1_PR1_HOST_INTR_PEND_2	42
MAIN2MCU_LVL_INTRTR0_IN_43	PRU_ICSSG1_PR1_HOST_INTR_PEND_3	43
MAIN2MCU_LVL_INTRTR0_IN_44	PRU_ICSSG1_PR1_HOST_INTR_PEND_4	44
MAIN2MCU_LVL_INTRTR0_IN_45	PRU_ICSSG1_PR1_HOST_INTR_PEND_5	45
MAIN2MCU_LVL_INTRTR0_IN_46	PRU_ICSSG1_PR1_HOST_INTR_PEND_6	46
MAIN2MCU_LVL_INTRTR0_IN_47	PRU_ICSSG1_PR1_HOST_INTR_PEND_7	47
MAIN2MCU_LVL_INTRTR0_IN_48	MCSPi0_INTR_SPI_0	48
MAIN2MCU_LVL_INTRTR0_IN_49	MCSPi1_INTR_SPI_0	49
MAIN2MCU_LVL_INTRTR0_IN_50	MCSPi2_INTR_SPI_0	50
MAIN2MCU_LVL_INTRTR0_IN_51	MCSPi3_INTR_SPI_0	51
MAIN2MCU_LVL_INTRTR0_IN_52	MCSPi4_INTR_SPI_0	52
MAIN2MCU_LVL_INTRTR0_IN_53	MCSPi5_INTR_SPI_0	53
MAIN2MCU_LVL_INTRTR0_IN_54	MCSPi6_INTR_SPI_0	54
MAIN2MCU_LVL_INTRTR0_IN_55	MCSPi7_INTR_SPI_0	55
MAIN2MCU_LVL_INTRTR0_IN_56	I2C0_POINTRPEND_0	56
MAIN2MCU_LVL_INTRTR0_IN_57	I2C1_POINTRPEND_0	57
MAIN2MCU_LVL_INTRTR0_IN_58	I2C2_POINTRPEND_0	58
MAIN2MCU_LVL_INTRTR0_IN_59	I2C3_POINTRPEND_0	59
MAIN2MCU_LVL_INTRTR0_IN_60	I2C4_POINTRPEND_0	60
MAIN2MCU_LVL_INTRTR0_IN_61	I2C5_POINTRPEND_0	61
MAIN2MCU_LVL_INTRTR0_IN_62	I2C6_POINTRPEND_0	62
MAIN2MCU_LVL_INTRTR0_IN_67	PCIE0_PCIE_PHY_LEVEL_0	67
MAIN2MCU_LVL_INTRTR0_IN_68	PCIE0_PCIE_LOCAL_LEVEL_0	68
MAIN2MCU_LVL_INTRTR0_IN_69	PCIE0_PCIE_CPTS_PEND_0	69
MAIN2MCU_LVL_INTRTR0_IN_73	PCIE1_PCIE_PHY_LEVEL_0	73
MAIN2MCU_LVL_INTRTR0_IN_74	PCIE1_PCIE_LOCAL_LEVEL_0	74
MAIN2MCU_LVL_INTRTR0_IN_75	PCIE1_PCIE_CPTS_PEND_0	75
MAIN2MCU_LVL_INTRTR0_IN_79	PCIE2_PCIE_PHY_LEVEL_0	79
MAIN2MCU_LVL_INTRTR0_IN_80	PCIE2_PCIE_LOCAL_LEVEL_0	80
MAIN2MCU_LVL_INTRTR0_IN_81	PCIE2_PCIE_CPTS_PEND_0	81
MAIN2MCU_LVL_INTRTR0_IN_85	PCIE3_PCIE_PHY_LEVEL_0	85
MAIN2MCU_LVL_INTRTR0_IN_86	PCIE3_PCIE_LOCAL_LEVEL_0	86
MAIN2MCU_LVL_INTRTR0_IN_87	PCIE3_PCIE_CPTS_PEND_0	87
MAIN2MCU_LVL_INTRTR0_IN_88	DCC0_INTR_DONE_LEVEL_0	88
MAIN2MCU_LVL_INTRTR0_IN_89	DCC1_INTR_DONE_LEVEL_0	89
MAIN2MCU_LVL_INTRTR0_IN_90	DCC2_INTR_DONE_LEVEL_0	90
MAIN2MCU_LVL_INTRTR0_IN_91	DCC3_INTR_DONE_LEVEL_0	91
MAIN2MCU_LVL_INTRTR0_IN_92	DCC4_INTR_DONE_LEVEL_0	92
MAIN2MCU_LVL_INTRTR0_IN_93	DCC5_INTR_DONE_LEVEL_0	93
MAIN2MCU_LVL_INTRTR0_IN_94	DCC6_INTR_DONE_LEVEL_0	94
MAIN2MCU_LVL_INTRTR0_IN_95	DCC7_INTR_DONE_LEVEL_0	95
MAIN2MCU_LVL_INTRTR0_IN_96	UART0_USART_IRQ_0	96
MAIN2MCU_LVL_INTRTR0_IN_97	UART1_USART_IRQ_0	97
MAIN2MCU_LVL_INTRTR0_IN_98	UART2_USART_IRQ_0	98

**Table 9-59. MAIN2MCU\_LVL\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	MAIN2MCU_LVL_INTRTR0_M UXCNTL_n[8-0] ENABLE Field Value (DEC)
MAIN2MCU_LVL_INTRTR0_IN_99	UART3_USART_IRQ_0	99
MAIN2MCU_LVL_INTRTR0_IN_100	UART4_USART_IRQ_0	100
MAIN2MCU_LVL_INTRTR0_IN_101	UART5_USART_IRQ_0	101
MAIN2MCU_LVL_INTRTR0_IN_102	UART6_USART_IRQ_0	102
MAIN2MCU_LVL_INTRTR0_IN_103	UART7_USART_IRQ_0	103
MAIN2MCU_LVL_INTRTR0_IN_104	UART8_USART_IRQ_0	104
MAIN2MCU_LVL_INTRTR0_IN_105	UART9_USART_IRQ_0	105
MAIN2MCU_LVL_INTRTR0_IN_108	TIMER0_INTR_PEND_0	108
MAIN2MCU_LVL_INTRTR0_IN_109	TIMER1_INTR_PEND_0	109
MAIN2MCU_LVL_INTRTR0_IN_110	TIMER2_INTR_PEND_0	110
MAIN2MCU_LVL_INTRTR0_IN_111	TIMER3_INTR_PEND_0	111
MAIN2MCU_LVL_INTRTR0_IN_112	TIMER4_INTR_PEND_0	112
MAIN2MCU_LVL_INTRTR0_IN_113	TIMER5_INTR_PEND_0	113
MAIN2MCU_LVL_INTRTR0_IN_114	TIMER6_INTR_PEND_0	114
MAIN2MCU_LVL_INTRTR0_IN_115	TIMER7_INTR_PEND_0	115
MAIN2MCU_LVL_INTRTR0_IN_116	TIMER8_INTR_PEND_0	116
MAIN2MCU_LVL_INTRTR0_IN_117	TIMER9_INTR_PEND_0	117
MAIN2MCU_LVL_INTRTR0_IN_118	TIMER10_INTR_PEND_0	118
MAIN2MCU_LVL_INTRTR0_IN_119	TIMER11_INTR_PEND_0	119
MAIN2MCU_LVL_INTRTR0_IN_120	TIMER12_INTR_PEND_0	120
MAIN2MCU_LVL_INTRTR0_IN_121	TIMER13_INTR_PEND_0	121
MAIN2MCU_LVL_INTRTR0_IN_122	TIMER14_INTR_PEND_0	122
MAIN2MCU_LVL_INTRTR0_IN_123	TIMER15_INTR_PEND_0	123
MAIN2MCU_LVL_INTRTR0_IN_124	TIMER16_INTR_PEND_0	124
MAIN2MCU_LVL_INTRTR0_IN_125	TIMER17_INTR_PEND_0	125
MAIN2MCU_LVL_INTRTR0_IN_126	TIMER18_INTR_PEND_0	126
MAIN2MCU_LVL_INTRTR0_IN_127	TIMER19_INTR_PEND_0	127
MAIN2MCU_LVL_INTRTR0_IN_128	USB0_IRQ_0	128
MAIN2MCU_LVL_INTRTR0_IN_129	USB0_IRQ_1	129
MAIN2MCU_LVL_INTRTR0_IN_130	USB0_IRQ_2	130
MAIN2MCU_LVL_INTRTR0_IN_131	USB0_IRQ_3	131
MAIN2MCU_LVL_INTRTR0_IN_132	USB0_IRQ_4	132
MAIN2MCU_LVL_INTRTR0_IN_133	USB0_IRQ_5	133
MAIN2MCU_LVL_INTRTR0_IN_134	USB0_IRQ_6	134
MAIN2MCU_LVL_INTRTR0_IN_135	USB0_IRQ_7	135
MAIN2MCU_LVL_INTRTR0_IN_136	USB1_IRQ_0	136
MAIN2MCU_LVL_INTRTR0_IN_137	USB1_IRQ_1	137
MAIN2MCU_LVL_INTRTR0_IN_138	USB1_IRQ_2	138
MAIN2MCU_LVL_INTRTR0_IN_139	USB1_IRQ_3	139
MAIN2MCU_LVL_INTRTR0_IN_140	USB1_IRQ_4	140
MAIN2MCU_LVL_INTRTR0_IN_141	USB1_IRQ_5	141
MAIN2MCU_LVL_INTRTR0_IN_142	USB1_IRQ_6	142
MAIN2MCU_LVL_INTRTR0_IN_143	USB1_IRQ_7	143
MAIN2MCU_LVL_INTRTR0_IN_152	USB0_OTGIRQ_0	152

**Table 9-59. MAIN2MCU\_LVL\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	MAIN2MCU_LVL_INTRTR0_M UXCNTL_n[8-0] ENABLE Field Value (DEC)
MAIN2MCU_LVL_INTRTR0_IN_153	USB1_OTGIRQ_0	153
MAIN2MCU_LVL_INTRTR0_IN_156	GLUELOGIC_MAIN_CBASS_INTR_OR_GLUE_MAIN_CBASS_AG G_ERR_INTR_0	156
MAIN2MCU_LVL_INTRTR0_IN_157	USB0_HOST_SYSTEM_ERROR_0	157
MAIN2MCU_LVL_INTRTR0_IN_158	USB1_HOST_SYSTEM_ERROR_0	158
MAIN2MCU_LVL_INTRTR0_IN_167	CBASS_INFRA0_DEFAULT_ERR_INTR_0	167
MAIN2MCU_LVL_INTRTR0_IN_176	MCASP0_XMIT_INTR_PEND_0	176
MAIN2MCU_LVL_INTRTR0_IN_177	MCASP0_REC_INTR_PEND_0	177
MAIN2MCU_LVL_INTRTR0_IN_178	MCASP1_XMIT_INTR_PEND_0	178
MAIN2MCU_LVL_INTRTR0_IN_179	MCASP1_REC_INTR_PEND_0	179
MAIN2MCU_LVL_INTRTR0_IN_180	MCASP2_XMIT_INTR_PEND_0	180
MAIN2MCU_LVL_INTRTR0_IN_181	MCASP2_REC_INTR_PEND_0	181
MAIN2MCU_LVL_INTRTR0_IN_182	MCASP3_XMIT_INTR_PEND_0	182
MAIN2MCU_LVL_INTRTR0_IN_183	MCASP3_REC_INTR_PEND_0	183
MAIN2MCU_LVL_INTRTR0_IN_184	MCASP4_XMIT_INTR_PEND_0	184
MAIN2MCU_LVL_INTRTR0_IN_185	MCASP4_REC_INTR_PEND_0	185
MAIN2MCU_LVL_INTRTR0_IN_186	MCASP5_XMIT_INTR_PEND_0	186
MAIN2MCU_LVL_INTRTR0_IN_187	MCASP5_REC_INTR_PEND_0	187
MAIN2MCU_LVL_INTRTR0_IN_188	MCASP6_XMIT_INTR_PEND_0	188
MAIN2MCU_LVL_INTRTR0_IN_189	MCASP6_REC_INTR_PEND_0	189
MAIN2MCU_LVL_INTRTR0_IN_190	MCASP7_XMIT_INTR_PEND_0	190
MAIN2MCU_LVL_INTRTR0_IN_191	MCASP7_REC_INTR_PEND_0	191
MAIN2MCU_LVL_INTRTR0_IN_192	MCASP8_XMIT_INTR_PEND_0	192
MAIN2MCU_LVL_INTRTR0_IN_193	MCASP8_REC_INTR_PEND_0	193
MAIN2MCU_LVL_INTRTR0_IN_194	MCASP9_XMIT_INTR_PEND_0	194
MAIN2MCU_LVL_INTRTR0_IN_195	MCASP9_REC_INTR_PEND_0	195
MAIN2MCU_LVL_INTRTR0_IN_196	MCASP10_XMIT_INTR_PEND_0	196
MAIN2MCU_LVL_INTRTR0_IN_197	MCASP10_REC_INTR_PEND_0	197
MAIN2MCU_LVL_INTRTR0_IN_198	MCASP11_XMIT_INTR_PEND_0	198
MAIN2MCU_LVL_INTRTR0_IN_199	MCASP11_REC_INTR_PEND_0	199
MAIN2MCU_LVL_INTRTR0_IN_200	AASRC0_INFIFO_LEVEL_0	200
MAIN2MCU_LVL_INTRTR0_IN_201	AASRC0_INGROUP_LEVEL_0	201
MAIN2MCU_LVL_INTRTR0_IN_202	AASRC0_OUTFIFO_LEVEL_0	202
MAIN2MCU_LVL_INTRTR0_IN_203	AASRC0_OUTGROUP_LEVEL_0	203
MAIN2MCU_LVL_INTRTR0_IN_204	AASRC0_ERR_LEVEL_0	204
MAIN2MCU_LVL_INTRTR0_IN_205	MLB0_MLBSS_MLB_INT_0	205
MAIN2MCU_LVL_INTRTR0_IN_206	MLB0_MLBSS_MLB_AHB_INT_0	206
MAIN2MCU_LVL_INTRTR0_IN_207	MLB0_MLBSS_MLB_AHB_INT_1	207
MAIN2MCU_LVL_INTRTR0_IN_208	DCC8_INTR_DONE_LEVEL_0	208
MAIN2MCU_LVL_INTRTR0_IN_209	DCC9_INTR_DONE_LEVEL_0	209
MAIN2MCU_LVL_INTRTR0_IN_210	DCC10_INTR_DONE_LEVEL_0	210
MAIN2MCU_LVL_INTRTR0_IN_211	DCC11_INTR_DONE_LEVEL_0	211
MAIN2MCU_LVL_INTRTR0_IN_212	DCC12_INTR_DONE_LEVEL_0	212
MAIN2MCU_LVL_INTRTR0_IN_215	C66DEBUGSS0_AQCOMPINTR_LEVEL_0	215

**Table 9-59. MAIN2MCU\_LVL\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	MAIN2MCU_LVL_INTRTR0_M UXCNTL_n[8-0] ENABLE Field Value (DEC)
MAIN2MCU_LVL_INTRTR0_IN_216	C66DEBUGSS1_AQCMPINTR_LEVEL_0	216
MAIN2MCU_LVL_INTRTR0_IN_217	DEBUGSS1_AQCMPINTR_LEVEL_0	217
MAIN2MCU_LVL_INTRTR0_IN_218	UFS0_UFS_INTR_0	218
MAIN2MCU_LVL_INTRTR0_IN_220	I3C0_I3C_INT_0	220
MAIN2MCU_LVL_INTRTR0_IN_221	CPSW0_STAT_PEND_0	221
MAIN2MCU_LVL_INTRTR0_IN_222	CPSW0_MDIO_PEND_0	222
MAIN2MCU_LVL_INTRTR0_IN_223	CPSW0_EVNT_PEND_0	223
MAIN2MCU_LVL_INTRTR0_IN_224	DSS_DSI0_DSI_0_FUNC_INTR_0	224
MAIN2MCU_LVL_INTRTR0_IN_226	DSS0_DSS_INST0_DISPC_FUNC_IRQ_PROC0_0	226
MAIN2MCU_LVL_INTRTR0_IN_227	DSS0_DSS_INST0_DISPC_FUNC_IRQ_PROC1_0	227
MAIN2MCU_LVL_INTRTR0_IN_228	DSS0_DSS_INST0_DISPC_SECURE_IRQ_PROC0_0	228
MAIN2MCU_LVL_INTRTR0_IN_229	DSS0_DSS_INST0_DISPC_SECURE_IRQ_PROC1_0	229
MAIN2MCU_LVL_INTRTR0_IN_230	DSS0_DSS_INST0_DISPC_SAFETY_ERROR_IRQ_PROC0_0	230
MAIN2MCU_LVL_INTRTR0_IN_231	DSS0_DSS_INST0_DISPC_SAFETY_ERROR_IRQ_PROC1_0	231
MAIN2MCU_LVL_INTRTR0_IN_238	DSS_EDP0_INTR_0	238
MAIN2MCU_LVL_INTRTR0_IN_239	DSS_EDP0_INTR_1	239
MAIN2MCU_LVL_INTRTR0_IN_240	DSS_EDP0_INTR_2	240
MAIN2MCU_LVL_INTRTR0_IN_241	DSS_EDP0_INTR_3	241
MAIN2MCU_LVL_INTRTR0_IN_250	CSI_TX_IF0_CSI_INTERRUPT_0	250
MAIN2MCU_LVL_INTRTR0_IN_251	CSI_TX_IF0_CSI_LEVEL_0	251
MAIN2MCU_LVL_INTRTR0_IN_252	DECODER0_IRQ_0	252
MAIN2MCU_LVL_INTRTR0_IN_253	ENCODER0_IRQ_0	253
MAIN2MCU_LVL_INTRTR0_IN_254	CSI_RX_IF0_CSI_IRQ_0	254
MAIN2MCU_LVL_INTRTR0_IN_255	CSI_RX_IF0_CSI_ERR_IRQ_0	255
MAIN2MCU_LVL_INTRTR0_IN_256	CSI_RX_IF0_CSI_LEVEL_0	256
MAIN2MCU_LVL_INTRTR0_IN_257	CSI_RX_IF1_CSI_IRQ_0	257
MAIN2MCU_LVL_INTRTR0_IN_258	CSI_RX_IF1_CSI_ERR_IRQ_0	258
MAIN2MCU_LVL_INTRTR0_IN_259	CSI_RX_IF1_CSI_LEVEL_0	259
MAIN2MCU_LVL_INTRTR0_IN_268	DMPAC0_INTD_0_SYSTEM_INTR_LEVEL_0	268
MAIN2MCU_LVL_INTRTR0_IN_269	DMPAC0_INTD_0_SYSTEM_INTR_LEVEL_1	269
MAIN2MCU_LVL_INTRTR0_IN_270	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_0	270
MAIN2MCU_LVL_INTRTR0_IN_271	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_1	271
MAIN2MCU_LVL_INTRTR0_IN_272	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_2	272
MAIN2MCU_LVL_INTRTR0_IN_273	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_3	273
MAIN2MCU_LVL_INTRTR0_IN_274	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_4	274
MAIN2MCU_LVL_INTRTR0_IN_275	VPAC0_INTD_0_SYSTEM_INTR_LEVEL_5	275
MAIN2MCU_LVL_INTRTR0_IN_276	VPFE0_CCDC_INTR_PEND_0	276
MAIN2MCU_LVL_INTRTR0_IN_277	VPFE0_RAT_EXP_INTR_0	277
MAIN2MCU_LVL_INTRTR0_IN_278	MCAN4_MCANSS_MCAN_LVL_INT_0	278
MAIN2MCU_LVL_INTRTR0_IN_279	MCAN4_MCANSS_MCAN_LVL_INT_1	279
MAIN2MCU_LVL_INTRTR0_IN_280	MCAN4_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	280
MAIN2MCU_LVL_INTRTR0_IN_281	MCAN5_MCANSS_MCAN_LVL_INT_0	281
MAIN2MCU_LVL_INTRTR0_IN_282	MCAN5_MCANSS_MCAN_LVL_INT_1	282
MAIN2MCU_LVL_INTRTR0_IN_283	MCAN5_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	283

**Table 9-59. MAIN2MCU\_LVL\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	MAIN2MCU_LVL_INTRTR0_M UXCNTL_n[8-0] ENABLE Field Value (DEC)
MAIN2MCU_LVL_INTRTR0_IN_284	MCAN6_MCANSS_MCAN_LVL_INT_0	284
MAIN2MCU_LVL_INTRTR0_IN_285	MCAN6_MCANSS_MCAN_LVL_INT_1	285
MAIN2MCU_LVL_INTRTR0_IN_286	MCAN6_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	286
MAIN2MCU_LVL_INTRTR0_IN_287	MCAN7_MCANSS_MCAN_LVL_INT_0	287
MAIN2MCU_LVL_INTRTR0_IN_288	MCAN7_MCANSS_MCAN_LVL_INT_1	288
MAIN2MCU_LVL_INTRTR0_IN_289	MCAN7_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	289
MAIN2MCU_LVL_INTRTR0_IN_290	MCAN8_MCANSS_MCAN_LVL_INT_0	290
MAIN2MCU_LVL_INTRTR0_IN_291	MCAN8_MCANSS_MCAN_LVL_INT_1	291
MAIN2MCU_LVL_INTRTR0_IN_292	MCAN8_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	292
MAIN2MCU_LVL_INTRTR0_IN_293	MCAN9_MCANSS_MCAN_LVL_INT_0	293
MAIN2MCU_LVL_INTRTR0_IN_294	MCAN9_MCANSS_MCAN_LVL_INT_1	294
MAIN2MCU_LVL_INTRTR0_IN_295	MCAN9_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	295
MAIN2MCU_LVL_INTRTR0_IN_296	MCAN10_MCANSS_MCAN_LVL_INT_0	296
MAIN2MCU_LVL_INTRTR0_IN_297	MCAN10_MCANSS_MCAN_LVL_INT_1	297
MAIN2MCU_LVL_INTRTR0_IN_298	MCAN10_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	298
MAIN2MCU_LVL_INTRTR0_IN_299	MCAN11_MCANSS_MCAN_LVL_INT_0	299
MAIN2MCU_LVL_INTRTR0_IN_300	MCAN11_MCANSS_MCAN_LVL_INT_1	300
MAIN2MCU_LVL_INTRTR0_IN_301	MCAN11_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	301
MAIN2MCU_LVL_INTRTR0_IN_302	MCAN12_MCANSS_MCAN_LVL_INT_0	302
MAIN2MCU_LVL_INTRTR0_IN_303	MCAN12_MCANSS_MCAN_LVL_INT_1	303
MAIN2MCU_LVL_INTRTR0_IN_304	MCAN12_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	304
MAIN2MCU_LVL_INTRTR0_IN_305	MCAN13_MCANSS_MCAN_LVL_INT_0	305
MAIN2MCU_LVL_INTRTR0_IN_306	MCAN13_MCANSS_MCAN_LVL_INT_1	306
MAIN2MCU_LVL_INTRTR0_IN_307	MCAN13_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	307
MAIN2MCU_LVL_INTRTR0_IN_310	GLUELOGIC_GPU_GPIO_REQACK_GLUE_GPU_GPIO_REQINT_LVL_0	310
MAIN2MCU_LVL_INTRTR0_IN_311	GLUELOGIC_GPU_GPIO_REQACK_GLUE_GPU_GPIO_ACKINT_LVL_0	311
MAIN2MCU_LVL_INTRTR0_IN_312	GPU0_MISC_0_IRQ_0	312

#### 9.4.3.15 MAIN2MCU\_PLS\_INTRTR0 Interrupt Map

Table 9-60 lists all the interrupts that are mapped to the MAIN2MCU\_PLS\_INTRTR0 inputs, along with the corresponding register programming values used for mux control. Any MAIN2MCU\_PLS\_INTRTR0 interrupt input that is not listed in this table should be considered as unused.

**Table 9-60. MAIN2MCU\_PLS\_INTRTR0 Interrupt Map**

Interrupt Input Line	Peripheral Interrupt	MAIN2MCU_PLS_INTRTR0_MUXCNTL_n [6-0] ENABLE Field Value (DEC)
RESERVED	RESERVED	0 <sup>(1)</sup>
MAIN2MCU_PLS_INTRTR0_IN_1	EHRPWM0_EPWM_ETINT_0	2
MAIN2MCU_PLS_INTRTR0_IN_2	EHRPWM1_EPWM_ETINT_0	3
MAIN2MCU_PLS_INTRTR0_IN_3	EHRPWM2_EPWM_ETINT_0	4
MAIN2MCU_PLS_INTRTR0_IN_4	EHRPWM3_EPWM_ETINT_0	5
MAIN2MCU_PLS_INTRTR0_IN_5	EHRPWM4_EPWM_ETINT_0	6
MAIN2MCU_PLS_INTRTR0_IN_6	EHRPWM5_EPWM_ETINT_0	7
MAIN2MCU_PLS_INTRTR0_IN_7	EHRPWM0_EPWM_TRIPZINT_0	8
MAIN2MCU_PLS_INTRTR0_IN_8	EHRPWM1_EPWM_TRIPZINT_0	9
MAIN2MCU_PLS_INTRTR0_IN_9	EHRPWM2_EPWM_TRIPZINT_0	10
MAIN2MCU_PLS_INTRTR0_IN_10	EHRPWM3_EPWM_TRIPZINT_0	11
MAIN2MCU_PLS_INTRTR0_IN_11	EHRPWM4_EPWM_TRIPZINT_0	12
MAIN2MCU_PLS_INTRTR0_IN_12	EHRPWM5_EPWM_TRIPZINT_0	13
MAIN2MCU_PLS_INTRTR0_IN_13	EQEP0_EQEP_INT_0	14
MAIN2MCU_PLS_INTRTR0_IN_14	EQEP1_EQEP_INT_0	15
MAIN2MCU_PLS_INTRTR0_IN_15	EQEP2_EQEP_INT_0	16
MAIN2MCU_PLS_INTRTR0_IN_16	ECAP0_ECAP_INT_0	17
MAIN2MCU_PLS_INTRTR0_IN_17	ECAP1_ECAP_INT_0	18
MAIN2MCU_PLS_INTRTR0_IN_18	ECAP2_ECAP_INT_0	19
MAIN2MCU_PLS_INTRTR0_IN_19	PRU_ICSSG0_PR1_TX_SOF_INTR_REQ_0	20
MAIN2MCU_PLS_INTRTR0_IN_20	PRU_ICSSG0_PR1_TX_SOF_INTR_REQ_1	21
MAIN2MCU_PLS_INTRTR0_IN_21	PRU_ICSSG0_PR1_RX_SOF_INTR_REQ_0	22
MAIN2MCU_PLS_INTRTR0_IN_22	PRU_ICSSG0_PR1_RX_SOF_INTR_REQ_1	23
MAIN2MCU_PLS_INTRTR0_IN_23	PRU_ICSSG1_PR1_TX_SOF_INTR_REQ_0	24
MAIN2MCU_PLS_INTRTR0_IN_24	PRU_ICSSG1_PR1_TX_SOF_INTR_REQ_1	25
MAIN2MCU_PLS_INTRTR0_IN_25	PRU_ICSSG1_PR1_RX_SOF_INTR_REQ_0	26
MAIN2MCU_PLS_INTRTR0_IN_26	PRU_ICSSG1_PR1_RX_SOF_INTR_REQ_1	27
MAIN2MCU_PLS_INTRTR0_IN_27	GLUELOGIC_SOCA_INT_GLUE_SOCA_INT_0	28
MAIN2MCU_PLS_INTRTR0_IN_28	GLUELOGIC_SOCB_INT_GLUE_SOCB_INT_0	29
MAIN2MCU_PLS_INTRTR0_IN_31	PCIE0_PCIE_LEGACY_PULSE_0	32
MAIN2MCU_PLS_INTRTR0_IN_32	PCIE0_PCIE_DOWNSTREAM_PULSE_0	33
MAIN2MCU_PLS_INTRTR0_IN_33	PCIE0_PCIE_FLR_PULSE_0	34
MAIN2MCU_PLS_INTRTR0_IN_34	PCIE0_PCIE_ERROR_PULSE_0	35
MAIN2MCU_PLS_INTRTR0_IN_35	PCIE0_PCIE_LINK_STATE_PULSE_0	36
MAIN2MCU_PLS_INTRTR0_IN_36	PCIE0_PCIE_PWR_STATE_PULSE_0	37
MAIN2MCU_PLS_INTRTR0_IN_37	PCIE0_PCIE_PTM_VALID_PULSE_0	38
MAIN2MCU_PLS_INTRTR0_IN_38	PCIE0_PCIE_HOT_RESET_PULSE_0	39
MAIN2MCU_PLS_INTRTR0_IN_39	PCIE1_PCIE_LEGACY_PULSE_0	40
MAIN2MCU_PLS_INTRTR0_IN_40	PCIE1_PCIE_DOWNSTREAM_PULSE_0	41
MAIN2MCU_PLS_INTRTR0_IN_41	PCIE1_PCIE_FLR_PULSE_0	42



**Table 9-60. MAIN2MCU\_PLS\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	MAIN2MCU_PLS_INTRTR0_MUXCNTL_n [6-0] ENABLE Field Value (DEC)
MAIN2MCU_PLS_INTRTR0_IN_42	PCIE1_PCIE_ERROR_PULSE_0	43
MAIN2MCU_PLS_INTRTR0_IN_43	PCIE1_PCIE_LINK_STATE_PULSE_0	44
MAIN2MCU_PLS_INTRTR0_IN_44	PCIE1_PCIE_PWR_STATE_PULSE_0	45
MAIN2MCU_PLS_INTRTR0_IN_45	PCIE1_PCIE_PTM_VALID_PULSE_0	46
MAIN2MCU_PLS_INTRTR0_IN_46	PCIE1_PCIE_HOT_RESET_PULSE_0	47
MAIN2MCU_PLS_INTRTR0_IN_47	PCIE2_PCIE_LEGACY_PULSE_0	48
MAIN2MCU_PLS_INTRTR0_IN_48	PCIE2_PCIE_DOWNSTREAM_PULSE_0	49
MAIN2MCU_PLS_INTRTR0_IN_49	PCIE2_PCIE_FLR_PULSE_0	50
MAIN2MCU_PLS_INTRTR0_IN_50	PCIE2_PCIE_ERROR_PULSE_0	51
MAIN2MCU_PLS_INTRTR0_IN_51	PCIE2_PCIE_LINK_STATE_PULSE_0	52
MAIN2MCU_PLS_INTRTR0_IN_52	PCIE2_PCIE_PWR_STATE_PULSE_0	53
MAIN2MCU_PLS_INTRTR0_IN_53	PCIE2_PCIE_PTM_VALID_PULSE_0	54
MAIN2MCU_PLS_INTRTR0_IN_54	PCIE2_PCIE_HOT_RESET_PULSE_0	55
MAIN2MCU_PLS_INTRTR0_IN_55	PCIE3_PCIE_LEGACY_PULSE_0	56
MAIN2MCU_PLS_INTRTR0_IN_56	PCIE3_PCIE_DOWNSTREAM_PULSE_0	57
MAIN2MCU_PLS_INTRTR0_IN_57	PCIE3_PCIE_FLR_PULSE_0	58
MAIN2MCU_PLS_INTRTR0_IN_58	PCIE3_PCIE_ERROR_PULSE_0	59
MAIN2MCU_PLS_INTRTR0_IN_59	PCIE3_PCIE_LINK_STATE_PULSE_0	60
MAIN2MCU_PLS_INTRTR0_IN_60	PCIE3_PCIE_PWR_STATE_PULSE_0	61
MAIN2MCU_PLS_INTRTR0_IN_61	PCIE3_PCIE_PTM_VALID_PULSE_0	62
MAIN2MCU_PLS_INTRTR0_IN_62	PCIE3_PCIE_HOT_RESET_PULSE_0	63
MAIN2MCU_PLS_INTRTR0_IN_63	GPIOMUX_INTRTR0_OUTP_0	64
MAIN2MCU_PLS_INTRTR0_IN_64	GPIOMUX_INTRTR0_OUTP_1	65
MAIN2MCU_PLS_INTRTR0_IN_65	GPIOMUX_INTRTR0_OUTP_2	66
MAIN2MCU_PLS_INTRTR0_IN_66	GPIOMUX_INTRTR0_OUTP_3	67
MAIN2MCU_PLS_INTRTR0_IN_67	GPIOMUX_INTRTR0_OUTP_4	68
MAIN2MCU_PLS_INTRTR0_IN_68	GPIOMUX_INTRTR0_OUTP_5	69
MAIN2MCU_PLS_INTRTR0_IN_69	GPIOMUX_INTRTR0_OUTP_6	70
MAIN2MCU_PLS_INTRTR0_IN_70	GPIOMUX_INTRTR0_OUTP_7	71
MAIN2MCU_PLS_INTRTR0_IN_71	GPIOMUX_INTRTR0_OUTP_8	72
MAIN2MCU_PLS_INTRTR0_IN_72	GPIOMUX_INTRTR0_OUTP_9	73
MAIN2MCU_PLS_INTRTR0_IN_73	GPIOMUX_INTRTR0_OUTP_10	74
MAIN2MCU_PLS_INTRTR0_IN_74	GPIOMUX_INTRTR0_OUTP_11	75
MAIN2MCU_PLS_INTRTR0_IN_75	GPIOMUX_INTRTR0_OUTP_12	76
MAIN2MCU_PLS_INTRTR0_IN_76	GPIOMUX_INTRTR0_OUTP_13	77
MAIN2MCU_PLS_INTRTR0_IN_77	GPIOMUX_INTRTR0_OUTP_14	78
MAIN2MCU_PLS_INTRTR0_IN_78	GPIOMUX_INTRTR0_OUTP_15	79
MAIN2MCU_PLS_INTRTR0_IN_79	GPIOMUX_INTRTR0_OUTP_16	80
MAIN2MCU_PLS_INTRTR0_IN_80	GPIOMUX_INTRTR0_OUTP_17	81
MAIN2MCU_PLS_INTRTR0_IN_81	GPIOMUX_INTRTR0_OUTP_18	82
MAIN2MCU_PLS_INTRTR0_IN_82	GPIOMUX_INTRTR0_OUTP_19	83
MAIN2MCU_PLS_INTRTR0_IN_83	GPIOMUX_INTRTR0_OUTP_20	84
MAIN2MCU_PLS_INTRTR0_IN_84	GPIOMUX_INTRTR0_OUTP_21	85
MAIN2MCU_PLS_INTRTR0_IN_85	GPIOMUX_INTRTR0_OUTP_22	86



**Table 9-60. MAIN2MCU\_PLS\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	MAIN2MCU_PLS_INTRTR0_MUXCNTL_n [6-0] ENABLE Field Value (DEC)
MAIN2MCU_PLS_INTRTR0_IN_86	GPIOMUX_INTRTR0_OUTP_23	87
MAIN2MCU_PLS_INTRTR0_IN_87	GPIOMUX_INTRTR0_OUTP_24	88
MAIN2MCU_PLS_INTRTR0_IN_88	GPIOMUX_INTRTR0_OUTP_25	89
MAIN2MCU_PLS_INTRTR0_IN_89	GPIOMUX_INTRTR0_OUTP_26	90
MAIN2MCU_PLS_INTRTR0_IN_90	GPIOMUX_INTRTR0_OUTP_27	91
MAIN2MCU_PLS_INTRTR0_IN_91	GPIOMUX_INTRTR0_OUTP_28	92
MAIN2MCU_PLS_INTRTR0_IN_92	GPIOMUX_INTRTR0_OUTP_29	93
MAIN2MCU_PLS_INTRTR0_IN_93	GPIOMUX_INTRTR0_OUTP_30	94
MAIN2MCU_PLS_INTRTR0_IN_94	GPIOMUX_INTRTR0_OUTP_31	95
MAIN2MCU_PLS_INTRTR0_IN_95	CMPEVENT_INTRTR0_OUTP_16	96
MAIN2MCU_PLS_INTRTR0_IN_96	CMPEVENT_INTRTR0_OUTP_17	97
MAIN2MCU_PLS_INTRTR0_IN_97	CMPEVENT_INTRTR0_OUTP_18	98
MAIN2MCU_PLS_INTRTR0_IN_98	CMPEVENT_INTRTR0_OUTP_19	99
MAIN2MCU_PLS_INTRTR0_IN_99	CMPEVENT_INTRTR0_OUTP_20	100
MAIN2MCU_PLS_INTRTR0_IN_100	CMPEVENT_INTRTR0_OUTP_21	101
MAIN2MCU_PLS_INTRTR0_IN_101	CMPEVENT_INTRTR0_OUTP_22	102
MAIN2MCU_PLS_INTRTR0_IN_102	CMPEVENT_INTRTR0_OUTP_23	103

- (1) Do not use that setting. As it is the default one, it is strongly recommended to set the field to another value before enabling interrupt router operation.

### 9.4.3.16 GPIOMUX\_INTRTR0 Interrupt Map

Table 9-61 lists all the interrupts that are mapped to the GPIOMUX\_INTRTR0 inputs, along with the corresponding register programming values used for mux control. Any GPIOMUX\_INTRTR0 interrupt input that is not listed in this table should be considered as unused.

**Table 9-61. GPIOMUX\_INTRTR0 Interrupt Map**

Interrupt Input Line	Peripheral Interrupt	GPIOMUX_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
GPIOMUX_INTRTR0_IN_0	MAIN_GPIO0_VIRT_OUT0_0	0
GPIOMUX_INTRTR0_IN_1	MAIN_GPIO0_VIRT_OUT0_1_0	1
GPIOMUX_INTRTR0_IN_2	MAIN_GPIO0_VIRT_OUT0_2_0	2
GPIOMUX_INTRTR0_IN_3	MAIN_GPIO0_VIRT_OUT0_3_0	3
GPIOMUX_INTRTR0_IN_4	MAIN_GPIO0_VIRT_OUT0_4_0	4
GPIOMUX_INTRTR0_IN_5	MAIN_GPIO0_VIRT_OUT0_5_0	5
GPIOMUX_INTRTR0_IN_6	MAIN_GPIO0_VIRT_OUT0_6_0	6
GPIOMUX_INTRTR0_IN_7	MAIN_GPIO0_VIRT_OUT0_7_0	7
GPIOMUX_INTRTR0_IN_8	MAIN_GPIO0_VIRT_OUT0_8_0	8
GPIOMUX_INTRTR0_IN_9	MAIN_GPIO0_VIRT_OUT0_9_0	9
GPIOMUX_INTRTR0_IN_10	MAIN_GPIO0_VIRT_OUT0_10_0	10
GPIOMUX_INTRTR0_IN_11	MAIN_GPIO0_VIRT_OUT0_11_0	11
GPIOMUX_INTRTR0_IN_12	MAIN_GPIO0_VIRT_OUT0_12_0	12
GPIOMUX_INTRTR0_IN_13	MAIN_GPIO0_VIRT_OUT0_13_0	13
GPIOMUX_INTRTR0_IN_14	MAIN_GPIO0_VIRT_OUT0_14_0	14
GPIOMUX_INTRTR0_IN_15	MAIN_GPIO0_VIRT_OUT0_15_0	15
GPIOMUX_INTRTR0_IN_16	MAIN_GPIO0_VIRT_OUT0_16_0	16
GPIOMUX_INTRTR0_IN_17	MAIN_GPIO0_VIRT_OUT0_17_0	17
GPIOMUX_INTRTR0_IN_18	MAIN_GPIO0_VIRT_OUT0_18_0	18
GPIOMUX_INTRTR0_IN_19	MAIN_GPIO0_VIRT_OUT0_19_0	19
GPIOMUX_INTRTR0_IN_20	MAIN_GPIO0_VIRT_OUT0_20_0	20
GPIOMUX_INTRTR0_IN_21	MAIN_GPIO0_VIRT_OUT0_21_0	21
GPIOMUX_INTRTR0_IN_22	MAIN_GPIO0_VIRT_OUT0_22_0	22
GPIOMUX_INTRTR0_IN_23	MAIN_GPIO0_VIRT_OUT0_23_0	23
GPIOMUX_INTRTR0_IN_24	MAIN_GPIO0_VIRT_OUT0_24_0	24
GPIOMUX_INTRTR0_IN_25	MAIN_GPIO0_VIRT_OUT0_25_0	25
GPIOMUX_INTRTR0_IN_26	MAIN_GPIO0_VIRT_OUT0_26_0	26
GPIOMUX_INTRTR0_IN_27	MAIN_GPIO0_VIRT_OUT0_27_0	27
GPIOMUX_INTRTR0_IN_28	MAIN_GPIO0_VIRT_OUT0_28_0	28
GPIOMUX_INTRTR0_IN_29	MAIN_GPIO0_VIRT_OUT0_29_0	29
GPIOMUX_INTRTR0_IN_30	MAIN_GPIO0_VIRT_OUT0_30_0	30
GPIOMUX_INTRTR0_IN_31	MAIN_GPIO0_VIRT_OUT0_31_0	31
GPIOMUX_INTRTR0_IN_32	MAIN_GPIO0_VIRT_OUT0_32_0	32
GPIOMUX_INTRTR0_IN_33	MAIN_GPIO0_VIRT_OUT0_33_0	33
GPIOMUX_INTRTR0_IN_34	MAIN_GPIO0_VIRT_OUT0_34_0	34
GPIOMUX_INTRTR0_IN_35	MAIN_GPIO0_VIRT_OUT0_35_0	35
GPIOMUX_INTRTR0_IN_36	MAIN_GPIO0_VIRT_OUT0_36_0	36
GPIOMUX_INTRTR0_IN_37	MAIN_GPIO0_VIRT_OUT0_37_0	37
GPIOMUX_INTRTR0_IN_38	MAIN_GPIO0_VIRT_OUT0_38_0	38
GPIOMUX_INTRTR0_IN_39	MAIN_GPIO0_VIRT_OUT0_39_0	39

**Table 9-61. GPIOMUX\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	GPIOMUX_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
GPIOMUX_INTRTR0_IN_40	MAIN_GPIO0_VIRT_OUT0_40_0	40
GPIOMUX_INTRTR0_IN_41	MAIN_GPIO0_VIRT_OUT0_41_0	41
GPIOMUX_INTRTR0_IN_42	MAIN_GPIO0_VIRT_OUT0_42_0	42
GPIOMUX_INTRTR0_IN_43	MAIN_GPIO0_VIRT_OUT0_43_0	43
GPIOMUX_INTRTR0_IN_44	MAIN_GPIO0_VIRT_OUT0_44_0	44
GPIOMUX_INTRTR0_IN_45	MAIN_GPIO0_VIRT_OUT0_45_0	45
GPIOMUX_INTRTR0_IN_46	MAIN_GPIO0_VIRT_OUT0_46_0	46
GPIOMUX_INTRTR0_IN_47	MAIN_GPIO0_VIRT_OUT0_47_0	47
GPIOMUX_INTRTR0_IN_48	MAIN_GPIO0_VIRT_OUT0_48_0	48
GPIOMUX_INTRTR0_IN_49	MAIN_GPIO0_VIRT_OUT0_49_0	49
GPIOMUX_INTRTR0_IN_50	MAIN_GPIO0_VIRT_OUT0_50_0	50
GPIOMUX_INTRTR0_IN_51	MAIN_GPIO0_VIRT_OUT0_51_0	51
GPIOMUX_INTRTR0_IN_52	MAIN_GPIO0_VIRT_OUT0_52_0	52
GPIOMUX_INTRTR0_IN_53	MAIN_GPIO0_VIRT_OUT0_53_0	53
GPIOMUX_INTRTR0_IN_54	MAIN_GPIO0_VIRT_OUT0_54_0	54
GPIOMUX_INTRTR0_IN_55	MAIN_GPIO0_VIRT_OUT0_55_0	55
GPIOMUX_INTRTR0_IN_56	MAIN_GPIO0_VIRT_OUT0_56_0	56
GPIOMUX_INTRTR0_IN_57	MAIN_GPIO0_VIRT_OUT0_57_0	57
GPIOMUX_INTRTR0_IN_58	MAIN_GPIO0_VIRT_OUT0_58_0	58
GPIOMUX_INTRTR0_IN_59	MAIN_GPIO0_VIRT_OUT0_59_0	59
GPIOMUX_INTRTR0_IN_60	MAIN_GPIO0_VIRT_OUT0_60_0	60
GPIOMUX_INTRTR0_IN_61	MAIN_GPIO0_VIRT_OUT0_61_0	61
GPIOMUX_INTRTR0_IN_62	MAIN_GPIO0_VIRT_OUT0_62_0	62
GPIOMUX_INTRTR0_IN_63	MAIN_GPIO0_VIRT_OUT0_63_0	63
GPIOMUX_INTRTR0_IN_64	MAIN_GPIO0_VIRT_OUT0_64_0	64
GPIOMUX_INTRTR0_IN_65	MAIN_GPIO0_VIRT_OUT0_65_0	65
GPIOMUX_INTRTR0_IN_66	MAIN_GPIO0_VIRT_OUT0_66_0	66
GPIOMUX_INTRTR0_IN_67	MAIN_GPIO0_VIRT_OUT0_67_0	67
GPIOMUX_INTRTR0_IN_68	MAIN_GPIO0_VIRT_OUT0_68_0	68
GPIOMUX_INTRTR0_IN_69	MAIN_GPIO0_VIRT_OUT0_69_0	69
GPIOMUX_INTRTR0_IN_70	MAIN_GPIO0_VIRT_OUT0_70_0	70
GPIOMUX_INTRTR0_IN_71	MAIN_GPIO0_VIRT_OUT0_71_0	71
GPIOMUX_INTRTR0_IN_72	MAIN_GPIO0_VIRT_OUT0_72_0	72
GPIOMUX_INTRTR0_IN_73	MAIN_GPIO0_VIRT_OUT0_73_0	73
GPIOMUX_INTRTR0_IN_74	MAIN_GPIO0_VIRT_OUT0_74_0	74
GPIOMUX_INTRTR0_IN_75	MAIN_GPIO0_VIRT_OUT0_75_0	75
GPIOMUX_INTRTR0_IN_76	MAIN_GPIO0_VIRT_OUT0_76_0	76
GPIOMUX_INTRTR0_IN_77	MAIN_GPIO0_VIRT_OUT0_77_0	77
GPIOMUX_INTRTR0_IN_78	MAIN_GPIO0_VIRT_OUT0_78_0	78
GPIOMUX_INTRTR0_IN_79	MAIN_GPIO0_VIRT_OUT0_79_0	79
GPIOMUX_INTRTR0_IN_80	MAIN_GPIO0_VIRT_OUT0_80_0	80
GPIOMUX_INTRTR0_IN_81	MAIN_GPIO0_VIRT_OUT0_81_0	81
GPIOMUX_INTRTR0_IN_82	MAIN_GPIO0_VIRT_OUT0_82_0	82
GPIOMUX_INTRTR0_IN_83	MAIN_GPIO0_VIRT_OUT0_83_0	83

**Table 9-61. GPIOMUX\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	GPIOMUX_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
GPIOMUX_INTRTR0_IN_84	MAIN_GPIO0_VIRT_OUT0_84_0	84
GPIOMUX_INTRTR0_IN_85	MAIN_GPIO0_VIRT_OUT0_85_0	85
GPIOMUX_INTRTR0_IN_86	MAIN_GPIO0_VIRT_OUT0_86_0	86
GPIOMUX_INTRTR0_IN_87	MAIN_GPIO0_VIRT_OUT0_87_0	87
GPIOMUX_INTRTR0_IN_88	MAIN_GPIO0_VIRT_OUT0_88_0	88
GPIOMUX_INTRTR0_IN_89	MAIN_GPIO0_VIRT_OUT0_89_0	89
GPIOMUX_INTRTR0_IN_90	MAIN_GPIO0_VIRT_OUT0_90_0	90
GPIOMUX_INTRTR0_IN_91	MAIN_GPIO0_VIRT_OUT0_91_0	91
GPIOMUX_INTRTR0_IN_92	MAIN_GPIO0_VIRT_OUT0_92_0	92
GPIOMUX_INTRTR0_IN_93	MAIN_GPIO0_VIRT_OUT0_93_0	93
GPIOMUX_INTRTR0_IN_94	MAIN_GPIO0_VIRT_OUT0_94_0	94
GPIOMUX_INTRTR0_IN_95	MAIN_GPIO0_VIRT_OUT0_95_0	95
GPIOMUX_INTRTR0_IN_96	MAIN_GPIO0_VIRT_OUT0_96_0	96
GPIOMUX_INTRTR0_IN_97	MAIN_GPIO0_VIRT_OUT0_97_0	97
GPIOMUX_INTRTR0_IN_98	MAIN_GPIO0_VIRT_OUT0_98_0	98
GPIOMUX_INTRTR0_IN_99	MAIN_GPIO0_VIRT_OUT0_99_0	99
GPIOMUX_INTRTR0_IN_100	MAIN_GPIO0_VIRT_OUT0_100_0	100
GPIOMUX_INTRTR0_IN_101	MAIN_GPIO0_VIRT_OUT0_101_0	101
GPIOMUX_INTRTR0_IN_102	MAIN_GPIO0_VIRT_OUT0_102_0	102
GPIOMUX_INTRTR0_IN_103	MAIN_GPIO0_VIRT_OUT0_103_0	103
GPIOMUX_INTRTR0_IN_104	MAIN_GPIO0_VIRT_OUT0_104_0	104
GPIOMUX_INTRTR0_IN_105	MAIN_GPIO0_VIRT_OUT0_105_0	105
GPIOMUX_INTRTR0_IN_106	MAIN_GPIO0_VIRT_OUT0_106_0	106
GPIOMUX_INTRTR0_IN_107	MAIN_GPIO0_VIRT_OUT0_107_0	107
GPIOMUX_INTRTR0_IN_108	MAIN_GPIO0_VIRT_OUT0_108_0	108
GPIOMUX_INTRTR0_IN_109	MAIN_GPIO0_VIRT_OUT0_109_0	109
GPIOMUX_INTRTR0_IN_110	MAIN_GPIO0_VIRT_OUT0_110_0	110
GPIOMUX_INTRTR0_IN_111	MAIN_GPIO0_VIRT_OUT0_111_0	111
GPIOMUX_INTRTR0_IN_112	MAIN_GPIO0_VIRT_OUT0_112_0	112
GPIOMUX_INTRTR0_IN_113	MAIN_GPIO0_VIRT_OUT0_113_0	113
GPIOMUX_INTRTR0_IN_114	MAIN_GPIO0_VIRT_OUT0_114_0	114
GPIOMUX_INTRTR0_IN_115	MAIN_GPIO0_VIRT_OUT0_115_0	115
GPIOMUX_INTRTR0_IN_116	MAIN_GPIO0_VIRT_OUT0_116_0	116
GPIOMUX_INTRTR0_IN_117	MAIN_GPIO0_VIRT_OUT0_117_0	117
GPIOMUX_INTRTR0_IN_118	MAIN_GPIO0_VIRT_OUT0_118_0	118
GPIOMUX_INTRTR0_IN_119	MAIN_GPIO0_VIRT_OUT0_119_0	119
GPIOMUX_INTRTR0_IN_120	MAIN_GPIO0_VIRT_OUT0_120_0	120
GPIOMUX_INTRTR0_IN_121	MAIN_GPIO0_VIRT_OUT0_121_0	121
GPIOMUX_INTRTR0_IN_122	MAIN_GPIO0_VIRT_OUT0_122_0	122
GPIOMUX_INTRTR0_IN_123	MAIN_GPIO0_VIRT_OUT0_123_0	123
GPIOMUX_INTRTR0_IN_124	MAIN_GPIO0_VIRT_OUT0_124_0	124
GPIOMUX_INTRTR0_IN_125	MAIN_GPIO0_VIRT_OUT0_125_0	125
GPIOMUX_INTRTR0_IN_126	MAIN_GPIO0_VIRT_OUT0_126_0	126
GPIOMUX_INTRTR0_IN_127	MAIN_GPIO0_VIRT_OUT0_127_0	127

**Table 9-61. GPIOMUX\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	GPIOMUX_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
GPIOMUX_INTRTR0_IN_128	MAIN_GPIO1_VIRT_OUT0_0	128
GPIOMUX_INTRTR0_IN_129	MAIN_GPIO1_VIRT_OUT0_1_0	129
GPIOMUX_INTRTR0_IN_130	MAIN_GPIO1_VIRT_OUT0_2_0	130
GPIOMUX_INTRTR0_IN_131	MAIN_GPIO1_VIRT_OUT0_3_0	131
GPIOMUX_INTRTR0_IN_132	MAIN_GPIO1_VIRT_OUT0_4_0	132
GPIOMUX_INTRTR0_IN_133	MAIN_GPIO1_VIRT_OUT0_5_0	133
GPIOMUX_INTRTR0_IN_134	MAIN_GPIO1_VIRT_OUT0_6_0	134
GPIOMUX_INTRTR0_IN_135	MAIN_GPIO1_VIRT_OUT0_7_0	135
GPIOMUX_INTRTR0_IN_136	MAIN_GPIO1_VIRT_OUT0_8_0	136
GPIOMUX_INTRTR0_IN_137	MAIN_GPIO1_VIRT_OUT0_9_0	137
GPIOMUX_INTRTR0_IN_138	MAIN_GPIO1_VIRT_OUT0_10_0	138
GPIOMUX_INTRTR0_IN_139	MAIN_GPIO1_VIRT_OUT0_11_0	139
GPIOMUX_INTRTR0_IN_140	MAIN_GPIO1_VIRT_OUT0_12_0	140
GPIOMUX_INTRTR0_IN_141	MAIN_GPIO1_VIRT_OUT0_13_0	141
GPIOMUX_INTRTR0_IN_142	MAIN_GPIO1_VIRT_OUT0_14_0	142
GPIOMUX_INTRTR0_IN_143	MAIN_GPIO1_VIRT_OUT0_15_0	143
GPIOMUX_INTRTR0_IN_144	MAIN_GPIO1_VIRT_OUT0_16_0	144
GPIOMUX_INTRTR0_IN_145	MAIN_GPIO1_VIRT_OUT0_17_0	145
GPIOMUX_INTRTR0_IN_146	MAIN_GPIO1_VIRT_OUT0_18_0	146
GPIOMUX_INTRTR0_IN_147	MAIN_GPIO1_VIRT_OUT0_19_0	147
GPIOMUX_INTRTR0_IN_148	MAIN_GPIO1_VIRT_OUT0_20_0	148
GPIOMUX_INTRTR0_IN_149	MAIN_GPIO1_VIRT_OUT0_21_0	149
GPIOMUX_INTRTR0_IN_150	MAIN_GPIO1_VIRT_OUT0_22_0	150
GPIOMUX_INTRTR0_IN_151	MAIN_GPIO1_VIRT_OUT0_23_0	151
GPIOMUX_INTRTR0_IN_152	MAIN_GPIO1_VIRT_OUT0_24_0	152
GPIOMUX_INTRTR0_IN_153	MAIN_GPIO1_VIRT_OUT0_25_0	153
GPIOMUX_INTRTR0_IN_154	MAIN_GPIO1_VIRT_OUT0_26_0	154
GPIOMUX_INTRTR0_IN_155	MAIN_GPIO1_VIRT_OUT0_27_0	155
GPIOMUX_INTRTR0_IN_156	MAIN_GPIO1_VIRT_OUT0_28_0	156
GPIOMUX_INTRTR0_IN_157	MAIN_GPIO1_VIRT_OUT0_29_0	157
GPIOMUX_INTRTR0_IN_158	MAIN_GPIO1_VIRT_OUT0_30_0	158
GPIOMUX_INTRTR0_IN_159	MAIN_GPIO1_VIRT_OUT0_31_0	159
GPIOMUX_INTRTR0_IN_160	MAIN_GPIO1_VIRT_OUT0_32_0	160
GPIOMUX_INTRTR0_IN_161	MAIN_GPIO1_VIRT_OUT0_33_0	161
GPIOMUX_INTRTR0_IN_162	MAIN_GPIO1_VIRT_OUT0_34_0	162
GPIOMUX_INTRTR0_IN_163	MAIN_GPIO1_VIRT_OUT0_35_0	163
GPIOMUX_INTRTR0_IN_256	GPIO0_GPIO_BANK_0	256
GPIOMUX_INTRTR0_IN_257	GPIO0_GPIO_BANK_1	257
GPIOMUX_INTRTR0_IN_258	GPIO0_GPIO_BANK_2	258
GPIOMUX_INTRTR0_IN_259	GPIO0_GPIO_BANK_3	259
GPIOMUX_INTRTR0_IN_260	GPIO0_GPIO_BANK_4	260
GPIOMUX_INTRTR0_IN_261	GPIO0_GPIO_BANK_5	261
GPIOMUX_INTRTR0_IN_262	GPIO0_GPIO_BANK_6	262
GPIOMUX_INTRTR0_IN_263	GPIO0_GPIO_BANK_7	263

**Table 9-61. GPIOMUX\_INTRTR0 Interrupt Map (continued)**

Interrupt Input Line	Peripheral Interrupt	GPIOMUX_INTRTR0_MUXCNTL_n [8-0] ENABLE Field Value (DEC)
GPIOMUX_INTRTR0_IN_264	GPIO2_GPIO_BANK_0	264
GPIOMUX_INTRTR0_IN_265	GPIO2_GPIO_BANK_1	265
GPIOMUX_INTRTR0_IN_266	GPIO2_GPIO_BANK_2	266
GPIOMUX_INTRTR0_IN_267	GPIO2_GPIO_BANK_3	267
GPIOMUX_INTRTR0_IN_268	GPIO2_GPIO_BANK_4	268
GPIOMUX_INTRTR0_IN_269	GPIO2_GPIO_BANK_5	269
GPIOMUX_INTRTR0_IN_270	GPIO2_GPIO_BANK_6	270
GPIOMUX_INTRTR0_IN_271	GPIO2_GPIO_BANK_7	271
GPIOMUX_INTRTR0_IN_272	GPIO4_GPIO_BANK_0	272
GPIOMUX_INTRTR0_IN_273	GPIO4_GPIO_BANK_1	273
GPIOMUX_INTRTR0_IN_274	GPIO4_GPIO_BANK_2	274
GPIOMUX_INTRTR0_IN_275	GPIO4_GPIO_BANK_3	275
GPIOMUX_INTRTR0_IN_276	GPIO4_GPIO_BANK_4	276
GPIOMUX_INTRTR0_IN_277	GPIO4_GPIO_BANK_5	277
GPIOMUX_INTRTR0_IN_278	GPIO4_GPIO_BANK_6	278
GPIOMUX_INTRTR0_IN_279	GPIO4_GPIO_BANK_7	279
GPIOMUX_INTRTR0_IN_280	GPIO6_GPIO_BANK_0	280
GPIOMUX_INTRTR0_IN_281	GPIO6_GPIO_BANK_1	281
GPIOMUX_INTRTR0_IN_282	GPIO6_GPIO_BANK_2	282
GPIOMUX_INTRTR0_IN_283	GPIO6_GPIO_BANK_3	283
GPIOMUX_INTRTR0_IN_284	GPIO6_GPIO_BANK_4	284
GPIOMUX_INTRTR0_IN_285	GPIO6_GPIO_BANK_5	285
GPIOMUX_INTRTR0_IN_286	GPIO6_GPIO_BANK_6	286
GPIOMUX_INTRTR0_IN_287	GPIO6_GPIO_BANK_7	287
GPIOMUX_INTRTR0_IN_288	GPIO1_GPIO_BANK_0	288
GPIOMUX_INTRTR0_IN_289	GPIO1_GPIO_BANK_1	289
GPIOMUX_INTRTR0_IN_290	GPIO1_GPIO_BANK_2	290
GPIOMUX_INTRTR0_IN_292	GPIO3_GPIO_BANK_0	292
GPIOMUX_INTRTR0_IN_293	GPIO3_GPIO_BANK_1	293
GPIOMUX_INTRTR0_IN_294	GPIO3_GPIO_BANK_2	294
GPIOMUX_INTRTR0_IN_296	GPIO5_GPIO_BANK_0	296
GPIOMUX_INTRTR0_IN_297	GPIO5_GPIO_BANK_1	297
GPIOMUX_INTRTR0_IN_298	GPIO5_GPIO_BANK_2	298
GPIOMUX_INTRTR0_IN_300	GPIO7_GPIO_BANK_0	300
GPIOMUX_INTRTR0_IN_301	GPIO7_GPIO_BANK_1	301
GPIOMUX_INTRTR0_IN_302	GPIO7_GPIO_BANK_2	302

### 9.4.3.17 ESM0 Interrupt Map

**Table 9-62. ESM0 Interrupt Map**

Interrupt Input Line	Interrupt ID	Interrupt Name
ESM0_LVL_IN_0	0	PLLFRAC2_SSMOD0_LOCKLOSS_IPCFG_0
ESM0_LVL_IN_1	1	PLLFRAC2_SSMOD1_LOCKLOSS_IPCFG_0
ESM0_LVL_IN_2	2	PLLFRAC2_SSMOD2_LOCKLOSS_IPCFG_0
ESM0_LVL_IN_3	3	PLLFRAC2_SSMOD3_LOCKLOSS_IPCFG_0
ESM0_LVL_IN_4	4	PLLFRAC2_SSMOD4_LOCKLOSS_IPCFG_0
ESM0_LVL_IN_5	5	PLLFRAC2_SSMOD5_LOCKLOSS_IPCFG_0
ESM0_LVL_IN_6	6	PLLFRAC2_SSMOD6_LOCKLOSS_IPCFG_0
ESM0_LVL_IN_7	7	PLLFRAC2_SSMOD7_LOCKLOSS_IPCFG_0
ESM0_LVL_IN_8	8	PLLFRAC2_SSMOD8_LOCKLOSS_IPCFG_0
ESM0_LVL_IN_12	12	PLLFRAC2_SSMOD12_LOCKLOSS_IPCFG_0
ESM0_LVL_IN_13	13	PLLFRAC2_SSMOD13_LOCKLOSS_IPCFG_0
ESM0_LVL_IN_14	14	PLLFRAC2_SSMOD14_LOCKLOSS_IPCFG_0
ESM0_LVL_IN_15	15	PLLFRAC2_SSMOD15_LOCKLOSS_IPCFG_0
ESM0_LVL_IN_16	16	PLLFRAC2_SSMOD16_LOCKLOSS_IPCFG_0
ESM0_LVL_IN_17	17	PLLFRAC2_SSMOD17_LOCKLOSS_IPCFG_0
ESM0_LVL_IN_18	18	PLLFRAC2_SSMOD18_LOCKLOSS_IPCFG_0
ESM0_LVL_IN_19	19	PLLFRAC2_SSMOD19_LOCKLOSS_IPCFG_0
ESM0_LVL_IN_23	23	PLLFRAC2_SSMOD23_LOCKLOSS_IPCFG_0
ESM0_LVL_IN_24	24	PLLDESKEW24_LOCKLOSS_IPCFG_0
ESM0_LVL_IN_25	25	PLLFRAC2_SSMOD25_LOCKLOSS_IPCFG_0
ESM0_LVL_IN_32	32	DDR0_DDRSS_CONTROLLER_GLOBAL_ERROR_NONFATAL_0
ESM0_LVL_IN_33	33	DDR0_DDRSS_CONTROLLER_GLOBAL_ERROR_FATAL_0
ESM0_LVL_IN_34	34	DDR0_DDRSS_HS_PHY_GLOBAL_ERROR_0
ESM0_LVL_IN_40	40	A72SS0_CORE0_INTERRIRQ_0
ESM0_LVL_IN_41	41	A72SS0_CORE0_EXTERRIRQ_0
ESM0_LVL_IN_48	48	COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_0
ESM0_LVL_IN_49	49	COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_1
ESM0_LVL_IN_50	50	COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_2
ESM0_LVL_IN_51	51	COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_3
ESM0_LVL_IN_52	52	COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_4
ESM0_LVL_IN_53	53	COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_5
ESM0_LVL_IN_54	54	COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_6
ESM0_LVL_IN_55	55	COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_7
ESM0_LVL_IN_56	56	COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_8
ESM0_LVL_IN_57	57	COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_9
ESM0_LVL_IN_58	58	COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_10
ESM0_LVL_IN_59	59	COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_11
ESM0_LVL_IN_60	60	COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_12
ESM0_LVL_IN_61	61	COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_13
ESM0_LVL_IN_62	62	COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_14
ESM0_LVL_IN_63	63	COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_15
ESM0_LVL_IN_64	64	COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_16
ESM0_LVL_IN_65	65	COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_17
ESM0_LVL_IN_66	66	COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_18

**Table 9-62. ESM0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
ESM0_LVL_IN_67	67	COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_19
ESM0_LVL_IN_68	68	COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_20
ESM0_LVL_IN_69	69	COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_39
ESM0_LVL_IN_70	70	COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_40
ESM0_LVL_IN_71	71	COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_41
ESM0_LVL_IN_72	72	COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_46
ESM0_LVL_IN_73	73	COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_47
ESM0_LVL_IN_74	74	COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_48
ESM0_LVL_IN_75	75	COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_62
ESM0_LVL_IN_76	76	COMPUTE_CLUSTER0_CLEC_ESM_EVENTS_OUT_LEVEL_63
ESM0_LVL_IN_94	94	UFS0_UFS_INTR_NONFATAL_0
ESM0_LVL_IN_95	95	UFS0_UFS_INTR_FATAL_0
ESM0_LVL_IN_96	96	UFS0_HCLK_ECC_CORR_LVL_0
ESM0_LVL_IN_97	97	UFS0_HCLK_ECC_UNCORR_LVL_0
ESM0_LVL_IN_104	104	DCC0_INTR_ERR_LEVEL_0
ESM0_LVL_IN_105	105	DCC1_INTR_ERR_LEVEL_0
ESM0_LVL_IN_106	106	DCC2_INTR_ERR_LEVEL_0
ESM0_LVL_IN_107	107	DCC3_INTR_ERR_LEVEL_0
ESM0_LVL_IN_108	108	DCC4_INTR_ERR_LEVEL_0
ESM0_LVL_IN_109	109	DCC5_INTR_ERR_LEVEL_0
ESM0_LVL_IN_110	110	DCC6_INTR_ERR_LEVEL_0
ESM0_LVL_IN_111	111	DCC7_INTR_ERR_LEVEL_0
ESM0_LVL_IN_112	112	DCC8_INTR_ERR_LEVEL_0
ESM0_LVL_IN_113	113	DCC9_INTR_ERR_LEVEL_0
ESM0_LVL_IN_114	114	DCC10_INTR_ERR_LEVEL_0
ESM0_LVL_IN_115	115	DCC11_INTR_ERR_LEVEL_0
ESM0_LVL_IN_116	116	DCC12_INTR_ERR_LEVEL_0
ESM0_LVL_IN_120	120	PDMA5_ECC_SEC_PEND_0
ESM0_LVL_IN_121	121	PDMA5_ECC_DED_PEND_0
ESM0_LVL_IN_124	124	USB0_ASF_INT_NONFATAL_0
ESM0_LVL_IN_125	125	USB0_ASF_INT_FATAL_0
ESM0_LVL_IN_126	126	USB0_A_ECC_AGGR_CORRECTED_ERR_LEVEL_0
ESM0_LVL_IN_127	127	USB0_A_ECC_AGGR_UNCORRECTED_ERR_LEVEL_0
ESM0_LVL_IN_130	130	USB1_ASF_INT_NONFATAL_0
ESM0_LVL_IN_131	131	USB1_ASF_INT_FATAL_0
ESM0_LVL_IN_132	132	USB1_A_ECC_AGGR_CORRECTED_ERR_LEVEL_0
ESM0_LVL_IN_133	133	USB1_A_ECC_AGGR_UNCORRECTED_ERR_LEVEL_0
ESM0_LVL_IN_144	144	PRU_ICSSG0_PR1_EDIO0_WD_TRIIG_0
ESM0_LVL_IN_145	145	PRU_ICSSG0_PR1_EDIO1_WD_TRIIG_0
ESM0_LVL_IN_146	146	PRU_ICSSG1_PR1_EDIO0_WD_TRIIG_0
ESM0_LVL_IN_147	147	PRU_ICSSG1_PR1_EDIO1_WD_TRIIG_0
ESM0_LVL_IN_148	148	PRU_ICSSG0_PR1_ECC_SEC_ERR_PEND_0
ESM0_LVL_IN_149	149	PRU_ICSSG0_PR1_ECC_DED_ERR_PEND_0
ESM0_LVL_IN_150	150	PRU_ICSSG1_PR1_ECC_SEC_ERR_PEND_0
ESM0_LVL_IN_151	151	PRU_ICSSG1_PR1_ECC_DED_ERR_PEND_0



**Table 9-62. ESM0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
ESM0_LVL_IN_152	152	I3C0_PCLK_ECC_UNCORR_LVL_0
ESM0_LVL_IN_153	153	I3C0_SCLK_ECC_UNCORR_LVL_0
ESM0_LVL_IN_154	154	I3C0_I3C_NONFATAL__INT_0
ESM0_LVL_IN_155	155	I3C0_I3C_FATAL__INT_0
ESM0_LVL_IN_160	160	NAVSS0_MODSS_ECC_AGGR_0_ECC_CORRECTED_ERR_LEVEL_0
ESM0_LVL_IN_161	161	NAVSS0_MODSS_ECC_AGGR_0_ECC_UNCORRECTED_ERR_LEVEL_0
ESM0_LVL_IN_162	162	NAVSS0_UDMASS_ECC_AGGR_0_ECC_CORRECTED_ERR_LEVEL_0
ESM0_LVL_IN_163	163	NAVSS0_UDMASS_ECC_AGGR_0_ECC_UNCORRECTED_ERR_LEVEL_0
ESM0_LVL_IN_164	164	NAVSS0_NBSS_ECC_AGGR_0_ECC_CORRECTED_ERR_LEVEL_0
ESM0_LVL_IN_165	165	NAVSS0_NBSS_ECC_AGGR_0_ECC_UNCORRECTED_ERR_LEVEL_0
ESM0_LVL_IN_166	166	NAVSS0_VIRTSS_ECC_AGGR_0_ECC_CORRECTED_ERR_LEVEL_0
ESM0_LVL_IN_167	167	NAVSS0_VIRTSS_ECC_AGGR_0_ECC_UNCORRECTED_ERR_LEVEL_0
ESM0_LVL_IN_168	168	MLB0_MLBSS_ECC_CORR_LVL_0
ESM0_LVL_IN_169	169	MLB0_MLBSS_ECC_UNCORR_LVL_0
ESM0_LVL_IN_170	170	MSRAM16KX256E0_ECC_CORR_LEVEL_0
ESM0_LVL_IN_171	171	MSRAM16KX256E0_ECC_UNCORR_LEVEL_0
ESM0_LVL_IN_172	172	PSRAMECC0_ECC_CORR_LEVEL_0
ESM0_LVL_IN_173	173	PSRAMECC0_ECC_UNCORR_LEVEL_0
ESM0_LVL_IN_174	174	PSRAM2KECC0_ECC_CORR_LEVEL_0
ESM0_LVL_IN_175	175	PSRAM2KECC0_ECC_UNCORR_LEVEL_0
ESM0_LVL_IN_176	176	MMCS00_EMMCSS_RXMEM_CORR_ERR_LVL_0
ESM0_LVL_IN_177	177	MMCS00_EMMCSS_RXMEM_UNCORR_ERR_LVL_0
ESM0_LVL_IN_178	178	MMCS00_EMMCSS_TXMEM_CORR_ERR_LVL_0
ESM0_LVL_IN_179	179	MMCS00_EMMCSS_TXMEM_UNCORR_ERR_LVL_0
ESM0_LVL_IN_180	180	MMCS01_EMMCSDSS_RXMEM_CORR_ERR_LVL_0
ESM0_LVL_IN_181	181	MMCS01_EMMCSDSS_RXMEM_UNCORR_ERR_LVL_0
ESM0_LVL_IN_182	182	MMCS01_EMMCSDSS_TXMEM_CORR_ERR_LVL_0
ESM0_LVL_IN_183	183	MMCS01_EMMCSDSS_TXMEM_UNCORR_ERR_LVL_0
ESM0_LVL_IN_184	184	MMCS02_EMMCSDSS_RXMEM_CORR_ERR_LVL_0
ESM0_LVL_IN_185	185	MMCS02_EMMCSDSS_RXMEM_UNCORR_ERR_LVL_0
ESM0_LVL_IN_186	186	MMCS02_EMMCSDSS_TXMEM_CORR_ERR_LVL_0
ESM0_LVL_IN_187	187	MMCS02_EMMCSDSS_TXMEM_UNCORR_ERR_LVL_0
ESM0_LVL_IN_192	192	DSS0_DSS_INST0_DISPC_SAFETY_ERROR_IRQ_PROC0_0
ESM0_LVL_IN_193	193	DSS0_DSS_INST0_DISPC_SAFETY_ERROR_IRQ_PROC1_0
ESM0_LVL_IN_194	194	DSS0_DSS_INST0_DISPC_SAFETY_ERROR_IRQ_PROC2_0
ESM0_LVL_IN_195	195	DSS0_DSS_INST0_DISPC_SAFETY_ERROR_IRQ_PROC3_0
ESM0_LVL_IN_196	196	CSI_RX_IF0_CSI_ERR_IRQ_0
ESM0_LVL_IN_197	197	CSI_RX_IF1_CSI_ERR_IRQ_0
ESM0_LVL_IN_200	200	CSI_RX_IF0_CSI_FATAL_0
ESM0_LVL_IN_201	201	CSI_RX_IF0_CSI_NONFATAL_0
ESM0_LVL_IN_202	202	CSI_RX_IF1_CSI_FATAL_0
ESM0_LVL_IN_203	203	CSI_RX_IF1_CSI_NONFATAL_0
ESM0_LVL_IN_206	206	CSI_RX_IF0_CORR_LEVEL_0
ESM0_LVL_IN_207	207	CSI_RX_IF0_UNCORR_LEVEL_0
ESM0_LVL_IN_208	208	CSI_RX_IF1_CORR_LEVEL_0

**Table 9-62. ESM0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
ESM0_LVL_IN_209	209	CSI_RX_IF1_UNCORR_LEVEL_0
ESM0_LVL_IN_212	212	CSI_TX_IF0_CSI_FATAL_0
ESM0_LVL_IN_213	213	CSI_TX_IF0_CSI_NONFATAL_0
ESM0_LVL_IN_214	214	CSI_TX_IF0_CDNS_RAM_CORR_LEVEL_0
ESM0_LVL_IN_215	215	CSI_TX_IF0_CDNS_RAM_UNCORR_LEVEL_0
ESM0_LVL_IN_216	216	CSI_TX_IF0_CORR_LEVEL_0
ESM0_LVL_IN_217	217	CSI_TX_IF0_UNCORR_LEVEL_0
ESM0_LVL_IN_218	218	DMPAC0_ECC_AGGR_0_ECC_CORRECTED_ERR_LEVEL_0
ESM0_LVL_IN_219	219	DMPAC0_ECC_AGGR_0_ECC_UNCORRECTED_ERR_LEVEL_0
ESM0_LVL_IN_220	220	VPAC0_ECC_AGGR_0_ECC_CORRECTED_ERR_LEVEL_0
ESM0_LVL_IN_221	221	VPAC0_ECC_AGGR_0_ECC_UNCORRECTED_ERR_LEVEL_0
ESM0_LVL_IN_222	222	VPAC0_VISS_0_CORR_LEVEL_INTR_0
ESM0_LVL_IN_223	223	VPAC0_VISS_0_UNCORR_LEVEL_INTR_0
ESM0_LVL_IN_226	226	VPAC0_LDC_0_CORR_LEVEL_INTR_0
ESM0_LVL_IN_227	227	VPAC0_LDC_0_UNCORR_LEVEL_INTR_0
ESM0_LVL_IN_230	230	DSS_EDP0_INTR_ASF_0
ESM0_LVL_IN_231	231	DSS_EDP0_INTR_ASF_1
ESM0_LVL_IN_232	232	DSS_EDP0_INTR_ASF_2
ESM0_LVL_IN_233	233	DSS_EDP0_INTR_ASF_3
ESM0_LVL_IN_234	234	DSS_EDP0_INTR_ASF_4
ESM0_LVL_IN_235	235	DSS_EDP0_INTR_ASF_5
ESM0_LVL_IN_236	236	DSS_EDP0_INTR_ASF_6
ESM0_LVL_IN_244	244	DSS_DSI0_DSI_0_SAFETY_ERROR_NONFATAL_INTR_0
ESM0_LVL_IN_245	245	DSS_DSI0_DSI_0_SAFETY_ERROR_FATAL_INTR_0
ESM0_LVL_IN_246	246	DSS_DSI0_ECC_INTR_UNCORR_LEVEL_SYS_0
ESM0_LVL_IN_253	253	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_96
ESM0_LVL_IN_254	254	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_97
ESM0_LVL_IN_255	255	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_98
ESM0_LVL_IN_256	256	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_110
ESM0_LVL_IN_257	257	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_111
ESM0_LVL_IN_258	258	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_112
ESM0_LVL_IN_259	259	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_113
ESM0_LVL_IN_260	260	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_116
ESM0_LVL_IN_261	261	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_117
ESM0_LVL_IN_262	262	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_118
ESM0_LVL_IN_263	263	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_119
ESM0_LVL_IN_264	264	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_120
ESM0_LVL_IN_265	265	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_121
ESM0_LVL_IN_266	266	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_122
ESM0_LVL_IN_267	267	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_123
ESM0_LVL_IN_268	268	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_124
ESM0_LVL_IN_269	269	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_125
ESM0_LVL_IN_270	270	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_126
ESM0_LVL_IN_271	271	C66SS0_CORE0_GEM_EVENT_OUT_SYNC_127
ESM0_LVL_IN_272	272	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_96

**Table 9-62. ESM0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
ESM0_LVL_IN_273	273	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_97
ESM0_LVL_IN_274	274	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_98
ESM0_LVL_IN_275	275	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_110
ESM0_LVL_IN_276	276	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_111
ESM0_LVL_IN_277	277	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_112
ESM0_LVL_IN_278	278	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_113
ESM0_LVL_IN_279	279	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_116
ESM0_LVL_IN_280	280	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_117
ESM0_LVL_IN_281	281	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_118
ESM0_LVL_IN_282	282	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_119
ESM0_LVL_IN_283	283	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_120
ESM0_LVL_IN_284	284	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_121
ESM0_LVL_IN_285	285	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_122
ESM0_LVL_IN_286	286	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_123
ESM0_LVL_IN_287	287	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_124
ESM0_LVL_IN_288	288	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_125
ESM0_LVL_IN_289	289	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_126
ESM0_LVL_IN_290	290	C66SS1_CORE0_GEM_EVENT_OUT_SYNC_127
ESM0_LVL_IN_292	292	DDR0_DDRSS_CFG_ECC_AGGR_CORR_ERR_LVL_0
ESM0_LVL_IN_293	293	DDR0_DDRSS_CFG_ECC_AGGR_UNCORR_ERR_LVL_0
ESM0_LVL_IN_294	294	DDR0_DDRSS_CTL_ECC_AGGR_CORR_ERR_LVL_0
ESM0_LVL_IN_295	295	DDR0_DDRSS_CTL_ECC_AGGR_UNCORR_ERR_LVL_0
ESM0_LVL_IN_296	296	DDR0_DDRSS_VBUS_ECC_AGGR_CORR_ERR_LVL_0
ESM0_LVL_IN_297	297	DDR0_DDRSS_VBUS_ECC_AGGR_UNCORR_ERR_LVL_0
ESM0_LVL_IN_298	298	DDR0_DDRSS_DRAM_ECC_CORR_ERR_LVL_0
ESM0_LVL_IN_299	299	DDR0_DDRSS_DRAM_ECC_UNCORR_ERR_LVL_0
ESM0_LVL_IN_304	304	SA2_UL0_SA_UL_ECC_CORR_LEVEL_0
ESM0_LVL_IN_305	305	SA2_UL0_SA_UL_ECC_UNCORR_LEVEL_0
ESM0_LVL_IN_306	306	CPSW0_ECC_SEC_PEND_0
ESM0_LVL_IN_307	307	CPSW0_ECC_DED_PEND_0
ESM0_LVL_IN_312	312	MCAN0_MCANSS_ECC_CORR_LVL_INT_0
ESM0_LVL_IN_313	313	MCAN0_MCANSS_ECC_UNCORR_LVL_INT_0
ESM0_LVL_IN_314	314	MCAN1_MCANSS_ECC_CORR_LVL_INT_0
ESM0_LVL_IN_315	315	MCAN1_MCANSS_ECC_UNCORR_LVL_INT_0
ESM0_LVL_IN_316	316	MCAN2_MCANSS_ECC_CORR_LVL_INT_0
ESM0_LVL_IN_317	317	MCAN2_MCANSS_ECC_UNCORR_LVL_INT_0
ESM0_LVL_IN_318	318	MCAN3_MCANSS_ECC_CORR_LVL_INT_0
ESM0_LVL_IN_319	319	MCAN3_MCANSS_ECC_UNCORR_LVL_INT_0
ESM0_LVL_IN_320	320	MCAN4_MCANSS_ECC_CORR_LVL_INT_0
ESM0_LVL_IN_321	321	MCAN4_MCANSS_ECC_UNCORR_LVL_INT_0
ESM0_LVL_IN_322	322	MCAN5_MCANSS_ECC_CORR_LVL_INT_0
ESM0_LVL_IN_323	323	MCAN5_MCANSS_ECC_UNCORR_LVL_INT_0
ESM0_LVL_IN_324	324	MCAN6_MCANSS_ECC_CORR_LVL_INT_0
ESM0_LVL_IN_325	325	MCAN6_MCANSS_ECC_UNCORR_LVL_INT_0
ESM0_LVL_IN_326	326	MCAN7_MCANSS_ECC_CORR_LVL_INT_0

**Table 9-62. ESM0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
ESM0_LVL_IN_327	327	MCAN7_MCANSS_ECC_UNCORR_LVL_INT_0
ESM0_LVL_IN_328	328	MCAN8_MCANSS_ECC_CORR_LVL_INT_0
ESM0_LVL_IN_329	329	MCAN8_MCANSS_ECC_UNCORR_LVL_INT_0
ESM0_LVL_IN_330	330	MCAN9_MCANSS_ECC_CORR_LVL_INT_0
ESM0_LVL_IN_331	331	MCAN9_MCANSS_ECC_UNCORR_LVL_INT_0
ESM0_LVL_IN_332	332	MCAN10_MCANSS_ECC_CORR_LVL_INT_0
ESM0_LVL_IN_333	333	MCAN10_MCANSS_ECC_UNCORR_LVL_INT_0
ESM0_LVL_IN_334	334	MCAN11_MCANSS_ECC_CORR_LVL_INT_0
ESM0_LVL_IN_335	335	MCAN11_MCANSS_ECC_UNCORR_LVL_INT_0
ESM0_LVL_IN_336	336	MCAN12_MCANSS_ECC_CORR_LVL_INT_0
ESM0_LVL_IN_337	337	MCAN12_MCANSS_ECC_UNCORR_LVL_INT_0
ESM0_LVL_IN_338	338	MCAN13_MCANSS_ECC_CORR_LVL_INT_0
ESM0_LVL_IN_339	339	MCAN13_MCANSS_ECC_UNCORR_LVL_INT_0
ESM0_LVL_IN_344	344	RTI0_INTR_WWD_0
ESM0_LVL_IN_345	345	RTI1_INTR_WWD_0
ESM0_LVL_IN_352	352	RTI15_INTR_WWD_0
ESM0_LVL_IN_353	353	RTI16_INTR_WWD_0
ESM0_LVL_IN_357	357	RTI24_INTR_WWD_0
ESM0_LVL_IN_358	358	RTI25_INTR_WWD_0
ESM0_LVL_IN_359	359	RTI28_INTR_WWD_0
ESM0_LVL_IN_360	360	RTI29_INTR_WWD_0
ESM0_LVL_IN_361	361	RTI30_INTR_WWD_0
ESM0_LVL_IN_362	362	RTI31_INTR_WWD_0
ESM0_LVL_IN_369	369	PCIE0_PCIE_ECC0_CORR_LEVEL_0
ESM0_LVL_IN_370	370	PCIE0_PCIE_ECC0_UNCORR_LEVEL_0
ESM0_LVL_IN_371	371	PCIE0_PCIE_ECC1_UNCORR_LEVEL_0
ESM0_LVL_IN_373	373	PCIE1_PCIE_ECC0_CORR_LEVEL_0
ESM0_LVL_IN_374	374	PCIE1_PCIE_ECC0_UNCORR_LEVEL_0
ESM0_LVL_IN_375	375	PCIE1_PCIE_ECC1_UNCORR_LEVEL_0
ESM0_LVL_IN_377	377	PCIE2_PCIE_ECC0_CORR_LEVEL_0
ESM0_LVL_IN_378	378	PCIE2_PCIE_ECC0_UNCORR_LEVEL_0
ESM0_LVL_IN_379	379	PCIE2_PCIE_ECC1_UNCORR_LEVEL_0
ESM0_LVL_IN_381	381	PCIE3_PCIE_ECC0_CORR_LEVEL_0
ESM0_LVL_IN_382	382	PCIE3_PCIE_ECC0_UNCORR_LEVEL_0
ESM0_LVL_IN_383	383	PCIE3_PCIE_ECC1_UNCORR_LEVEL_0
ESM0_LVL_IN_392	392	R5FSS0_CORE0_EXP_INTR_0
ESM0_LVL_IN_393	393	R5FSS0_CORE1_EXP_INTR_0
ESM0_LVL_IN_394	394	R5FSS1_CORE0_EXP_INTR_0
ESM0_LVL_IN_395	395	R5FSS1_CORE1_EXP_INTR_0
ESM0_LVL_IN_396	396	C66SS0_RAT0_C66_RAT_INTR_0
ESM0_LVL_IN_397	397	C66SS1_RAT0_C66_RAT_INTR_0
ESM0_LVL_IN_416	416	ECC_AGGR0_CORR_LEVEL_0
ESM0_LVL_IN_417	417	ECC_AGGR0_UNCORR_LEVEL_0
ESM0_LVL_IN_424	424	ECC_AGGR16_CORR_LEVEL_0
ESM0_LVL_IN_425	425	ECC_AGGR16_UNCORR_LEVEL_0

**Table 9-62. ESM0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
ESM0_LVL_IN_426	426	ECC_AGGR17_CORR_LEVEL_0
ESM0_LVL_IN_427	427	ECC_AGGR17_UNCORR_LEVEL_0
ESM0_LVL_IN_428	428	ECC_AGGR18_CORR_LEVEL_0
ESM0_LVL_IN_429	429	ECC_AGGR18_UNCORR_LEVEL_0
ESM0_LVL_IN_430	430	ECC_AGGR19_CORR_LEVEL_0
ESM0_LVL_IN_431	431	ECC_AGGR19_UNCORR_LEVEL_0
ESM0_LVL_IN_432	432	ECC_AGGR4_CORR_LEVEL_0
ESM0_LVL_IN_433	433	ECC_AGGR4_UNCORR_LEVEL_0
ESM0_LVL_IN_434	434	ECC_AGGR5_CORR_LEVEL_0
ESM0_LVL_IN_435	435	ECC_AGGR5_UNCORR_LEVEL_0
ESM0_LVL_IN_436	436	ECC_AGGR6_CORR_LEVEL_0
ESM0_LVL_IN_437	437	ECC_AGGR6_UNCORR_LEVEL_0
ESM0_LVL_IN_444	444	ECC_AGGR10_CORR_LEVEL_0
ESM0_LVL_IN_445	445	ECC_AGGR10_UNCORR_LEVEL_0
ESM0_LVL_IN_446	446	ECC_AGGR11_CORR_LEVEL_0
ESM0_LVL_IN_447	447	ECC_AGGR11_UNCORR_LEVEL_0
ESM0_LVL_IN_456	456	DFTSS0_DFT_SAFETY_123_0
ESM0_LVL_IN_457	457	DFTSS0_DFT_SAFETY_MULTI_0
ESM0_LVL_IN_458	458	DFTSS0_DFT_SAFETY_ONE_0
ESM0_LVL_IN_459	459	PBIST6_DFT_PBIST_SAFETY_ERROR_0
ESM0_LVL_IN_461	461	PBIST0_DFT_PBIST_SAFETY_ERROR_0
ESM0_LVL_IN_462	462	PBIST1_DFT_PBIST_SAFETY_ERROR_0
ESM0_LVL_IN_463	463	PBIST4_DFT_PBIST_SAFETY_ERROR_0
ESM0_LVL_IN_464	464	PBIST2_DFT_PBIST_SAFETY_ERROR_0
ESM0_LVL_IN_465	465	PBIST3_DFT_PBIST_SAFETY_ERROR_0
ESM0_LVL_IN_466	466	PBIST7_DFT_PBIST_SAFETY_ERROR_0
ESM0_LVL_IN_467	467	PBIST9_DFT_PBIST_SAFETY_ERROR_0
ESM0_LVL_IN_468	468	PBIST10_DFT_PBIST_SAFETY_ERROR_0
ESM0_LVL_IN_470	470	PBIST5_DFT_PBIST_SAFETY_ERROR_0
ESM0_LVL_IN_471	471	GPU0_DFT_PBIST_0_DFT_PBIST_SAFETY_ERROR_0
ESM0_LVL_IN_472	472	C66SS0_PBIST0_DFT_PBIST_SAFETY_ERROR_0
ESM0_LVL_IN_473	473	C66SS1_PBIST0_DFT_PBIST_SAFETY_ERROR_0
ESM0_LVL_IN_480	480	PSC0_PSC_MOD_MNLP_MAIN_ALWAYS_ON_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_481	481	PSC0_PSC_MOD_MNLP_MAIN_TEST_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_482	482	PSC0_PSC_MOD_MNLP_MAIN_PBIST_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_483	483	PSC0_PSC_MOD_MNLP_PER_AUDIO_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_484	484	PSC0_PSC_MOD_MNLP_PER_ATL_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_485	485	PSC0_PSC_MOD_MNLP_PER_MLB_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_486	486	PSC0_PSC_MOD_MNLP_PER_MOTOR_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_487	487	PSC0_PSC_MOD_MNLP_PER_MISCIO_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_488	488	PSC0_PSC_MOD_MNLP_PER_GPMC_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_489	489	PSC0_PSC_MOD_MNLP_PER_VPFE_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_490	490	PSC0_PSC_MOD_MNLP_PER_VPE_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_491	491	PSC0_PSC_MOD_MNLP_PER_SPARE0_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_492	492	PSC0_PSC_MOD_MNLP_PER_SPARE1_CS1_CLKSTOP_REQ_0

**Table 9-62. ESM0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
ESM0_LVL_IN_493	493	PSC0_PSC_MOD_MNLP_MAIN_DEBUG_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_494	494	PSC0_PSC_MOD_MNLP_EMIF_DATA_0_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_495	495	PSC0_PSC_MOD_MNLP_EMIF_CFG_0_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_496	496	PSC0_PSC_MOD_MNLP_EMIF_DATA_1_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_497	497	PSC0_PSC_MOD_MNLP_EMIF_CFG_1_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_498	498	PSC0_PSC_MOD_MNLP_PER_SPARE2_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_499	499	PSC0_PSC_MOD_MNLP_CC_TOP_PBIST_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_500	500	PSC0_PSC_MOD_MNLP_USB_0_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_501	501	PSC0_PSC_MOD_MNLP_USB_1_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_502	502	PSC0_PSC_MOD_MNLP_USB_2_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_503	503	PSC0_PSC_MOD_MNLP_MMC4B_0_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_504	504	PSC0_PSC_MOD_MNLP_MMC4B_1_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_505	505	PSC0_PSC_MOD_MNLP_MMC8B_0_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_506	506	PSC0_PSC_MOD_MNLP_UFS_0_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_507	507	PSC0_PSC_MOD_MNLP_UFS_1_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_508	508	PSC0_PSC_MOD_MNLP_PCIE_0_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_509	509	PSC0_PSC_MOD_MNLP_PCIE_1_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_510	510	PSC0_PSC_MOD_MNLP_PCIE_2_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_511	511	PSC0_PSC_MOD_MNLP_PCIE_3_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_512	512	PSC0_PSC_MOD_MNLP_SAUL_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_513	513	PSC0_PSC_MOD_MNLP_PER_I3C_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_514	514	PSC0_PSC_MOD_MNLP_MAIN_MCANSS_0_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_515	515	PSC0_PSC_MOD_MNLP_MAIN_MCANSS_1_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_516	516	PSC0_PSC_MOD_MNLP_MAIN_MCANSS_2_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_517	517	PSC0_PSC_MOD_MNLP_MAIN_MCANSS_3_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_518	518	PSC0_PSC_MOD_MNLP_MAIN_MCANSS_4_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_519	519	PSC0_PSC_MOD_MNLP_MAIN_MCANSS_5_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_520	520	PSC0_PSC_MOD_MNLP_MAIN_MCANSS_6_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_521	521	PSC0_PSC_MOD_MNLP_MAIN_MCANSS_7_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_522	522	PSC0_PSC_MOD_MNLP_MAIN_MCANSS_8_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_523	523	PSC0_PSC_MOD_MNLP_MAIN_MCANSS_9_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_524	524	PSC0_PSC_MOD_MNLP_MAIN_MCANSS_10_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_525	525	PSC0_PSC_MOD_MNLP_MAIN_MCANSS_11_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_526	526	PSC0_PSC_MOD_MNLP_MAIN_MCANSS_12_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_527	527	PSC0_PSC_MOD_MNLP_MAIN_MCANSS_13_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_528	528	PSC0_PSC_MOD_MNLP_DSS_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_529	529	PSC0_PSC_MOD_MNLP_DSS_PBIST_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_530	530	PSC0_PSC_MOD_MNLP_DSI_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_531	531	PSC0_PSC_MOD_MNLP_EDP_0_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_532	532	PSC0_PSC_MOD_MNLP_EDP_1_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_533	533	PSC0_PSC_MOD_MNLP_CSIRX_0_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_534	534	PSC0_PSC_MOD_MNLP_CSIRX_1_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_535	535	PSC0_PSC_MOD_MNLP_CSIRX_2_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_536	536	PSC0_PSC_MOD_MNLP_CSITX_0_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_537	537	PSC0_PSC_MOD_MNLP_TX_DPHY_0_CS1_CLKSTOP_REQ_0

**Table 9-62. ESM0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
ESM0_LVL_IN_538	538	PSC0_PSC_MOD_MNLP_CSIRX_PHY_0_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_539	539	PSC0_PSC_MOD_MNLP_CSIRX_PHY_1_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_540	540	PSC0_PSC_MOD_MNLP_CSIRX_PHY_2_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_541	541	PSC0_PSC_MOD_MNLP_ICSSG_0_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_542	542	PSC0_PSC_MOD_MNLP_ICSSG_1_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_543	543	PSC0_PSC_MOD_MNLP_9GSS_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_544	544	PSC0_PSC_MOD_MNLP_SERDES_0_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_545	545	PSC0_PSC_MOD_MNLP_SERDES_1_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_546	546	PSC0_PSC_MOD_MNLP_SERDES_2_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_547	547	PSC0_PSC_MOD_MNLP_SERDES_3_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_548	548	PSC0_PSC_MOD_MNLP_SERDES_4_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_549	549	PSC0_PSC_MOD_MNLP_SERDES_5_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_550	550	PSC0_PSC_MOD_MNLP_DMTIMER_0_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_551	551	PSC0_PSC_MOD_MNLP_DMTIMER_1_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_552	552	PSC0_PSC_MOD_MNLP_DMTIMER_2_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_553	553	PSC0_PSC_MOD_MNLP_DMTIMER_3_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_554	554	PSC0_PSC_MOD_MNLP_C71X_0_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_555	555	PSC0_PSC_MOD_MNLP_C71X_0_PBIIST_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_556	556	PSC0_PSC_MOD_MNLP_C71X_1_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_557	557	PSC0_PSC_MOD_MNLP_C71X_1_PBIIST_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_558	558	PSC0_PSC_MOD_MNLP_A72_CLUSTER_0_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_559	559	PSC0_PSC_MOD_MNLP_A72_CLUSTER_0_PBIIST_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_560	560	PSC0_PSC_MOD_MNLP_A72_0_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_561	561	PSC0_PSC_MOD_MNLP_A72_1_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_562	562	PSC0_PSC_MOD_MNLP_A72_CLUSTER_1_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_563	563	PSC0_PSC_MOD_MNLP_A72_CLUSTER_1_PBIIST_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_564	564	PSC0_PSC_MOD_MNLP_A72_2_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_565	565	PSC0_PSC_MOD_MNLP_A72_3_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_566	566	PSC0_PSC_MOD_MNLP_GPUCOM_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_567	567	PSC0_PSC_MOD_MNLP_GPUPBIIST_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_568	568	PSC0_PSC_MOD_MNLP_GPUCORE_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_569	569	PSC0_PSC_MOD_MNLP_C66X_0_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_570	570	PSC0_PSC_MOD_MNLP_C66X_PBIIST_0_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_571	571	PSC0_PSC_MOD_MNLP_C66X_1_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_572	572	PSC0_PSC_MOD_MNLP_C66X_PBIIST_1_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_573	573	PSC0_PSC_MOD_MNLP_PULSAR_0_R5_0_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_574	574	PSC0_PSC_MOD_MNLP_PULSAR_0_R5_1_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_575	575	PSC0_PSC_MOD_MNLP_PULSAR_PBIIST_0_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_576	576	PSC0_PSC_MOD_MNLP_PULSAR_1_R5_0_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_577	577	PSC0_PSC_MOD_MNLP_PULSAR_1_R5_1_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_578	578	PSC0_PSC_MOD_MNLP_PULSAR_PBIIST_1_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_579	579	PSC0_PSC_MOD_MNLP_DECODE_0_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_580	580	PSC0_PSC_MOD_MNLP_DECODE_PBIIST_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_581	581	PSC0_PSC_MOD_MNLP_ENCODE_0_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_582	582	PSC0_PSC_MOD_MNLP_ENCODE_PBIIST_CS1_CLKSTOP_REQ_0



**Table 9-62. ESM0 Interrupt Map (continued)**

Interrupt Input Line	Interrupt ID	Interrupt Name
ESM0_LVL_IN_583	583	PSC0_PSC_MOD_MNLP_DMPAC_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_584	584	PSC0_PSC_MOD_MNLP_SDE_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_585	585	PSC0_PSC_MOD_MNLP_DMPAC_PBIIST_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_586	586	PSC0_PSC_MOD_MNLP_VPAC_CS1_CLKSTOP_REQ_0
ESM0_LVL_IN_587	587	PSC0_PSC_MOD_MNLP_VPAC_PBIIST_CS1_CLKSTOP_REQ_0
ESM0_PLS_IN_608	608	R5FSS0_CORE0_ECC_CORRECTED_PULSE_0
ESM0_PLS_IN_609	609	R5FSS0_CORE0_ECC_UNCORRECTED_PULSE_0
ESM0_PLS_IN_610	610	R5FSS0_CORE1_ECC_CORRECTED_PULSE_0
ESM0_PLS_IN_611	611	R5FSS0_CORE1_ECC_UNCORRECTED_PULSE_0
ESM0_PLS_IN_612	612	R5FSS0_SELFTEST_ERR_PULSE_0
ESM0_PLS_IN_613	613	R5FSS0_COMPARE_ERR_PULSE_0
ESM0_PLS_IN_614	614	R5FSS0_BUS_MONITOR_ERR_PULSE_0
ESM0_PLS_IN_615	615	R5FSS0_VIM_COMPARE_ERR_PULSE_0
ESM0_PLS_IN_616	616	R5FSS0_CCM_COMPARE_STAT_PULSE_INTR_0
ESM0_PLS_IN_618	618	R5FSS1_CORE0_ECC_CORRECTED_PULSE_0
ESM0_PLS_IN_619	619	R5FSS1_CORE0_ECC_UNCORRECTED_PULSE_0
ESM0_PLS_IN_620	620	R5FSS1_CORE1_ECC_CORRECTED_PULSE_0
ESM0_PLS_IN_621	621	R5FSS1_CORE1_ECC_UNCORRECTED_PULSE_0
ESM0_PLS_IN_622	622	R5FSS1_SELFTEST_ERR_PULSE_0
ESM0_PLS_IN_623	623	R5FSS1_COMPARE_ERR_PULSE_0
ESM0_PLS_IN_624	624	R5FSS1_BUS_MONITOR_ERR_PULSE_0
ESM0_PLS_IN_625	625	R5FSS1_VIM_COMPARE_ERR_PULSE_0
ESM0_PLS_IN_626	626	R5FSS1_CCM_COMPARE_STAT_PULSE_INTR_0
ESM0_PLS_IN_628	628	COMPUTE_CLUSTER0_GIC500SS_AXIM_ERR_0
ESM0_PLS_IN_629	629	COMPUTE_CLUSTER0_GIC500SS_ECC_FATAL_0
ESM0_PLS_IN_632	632	GPIOMUX_INTRTR0_OUTP_0
ESM0_PLS_IN_633	633	GPIOMUX_INTRTR0_OUTP_1
ESM0_PLS_IN_634	634	GPIOMUX_INTRTR0_OUTP_2
ESM0_PLS_IN_635	635	GPIOMUX_INTRTR0_OUTP_3
ESM0_PLS_IN_636	636	GPIOMUX_INTRTR0_OUTP_4
ESM0_PLS_IN_637	637	GPIOMUX_INTRTR0_OUTP_5
ESM0_PLS_IN_638	638	GPIOMUX_INTRTR0_OUTP_6
ESM0_PLS_IN_639	639	GPIOMUX_INTRTR0_OUTP_7
ESM0_PLS_IN_648	648	PCIE0_PCIE_ASF_PULSE_0
ESM0_PLS_IN_649	649	PCIE1_PCIE_ASF_PULSE_0
ESM0_PLS_IN_650	650	PCIE2_PCIE_ASF_PULSE_0
ESM0_PLS_IN_651	651	PCIE3_PCIE_ASF_PULSE_0



Chapter 10

**Data Movement Architecture (DMA)**

---



10.1 DMA Architecture.....	1120
10.2 Navigator Subsystem (NAVSS).....	1186
10.3 Peripheral DMA (PDMA).....	1276
10.4 Data Routing Unit (DRU).....	1307

## 10.1 DMA Architecture

This section describes the DMA (data movement) architecture in the device.

### 10.1.1 Overview

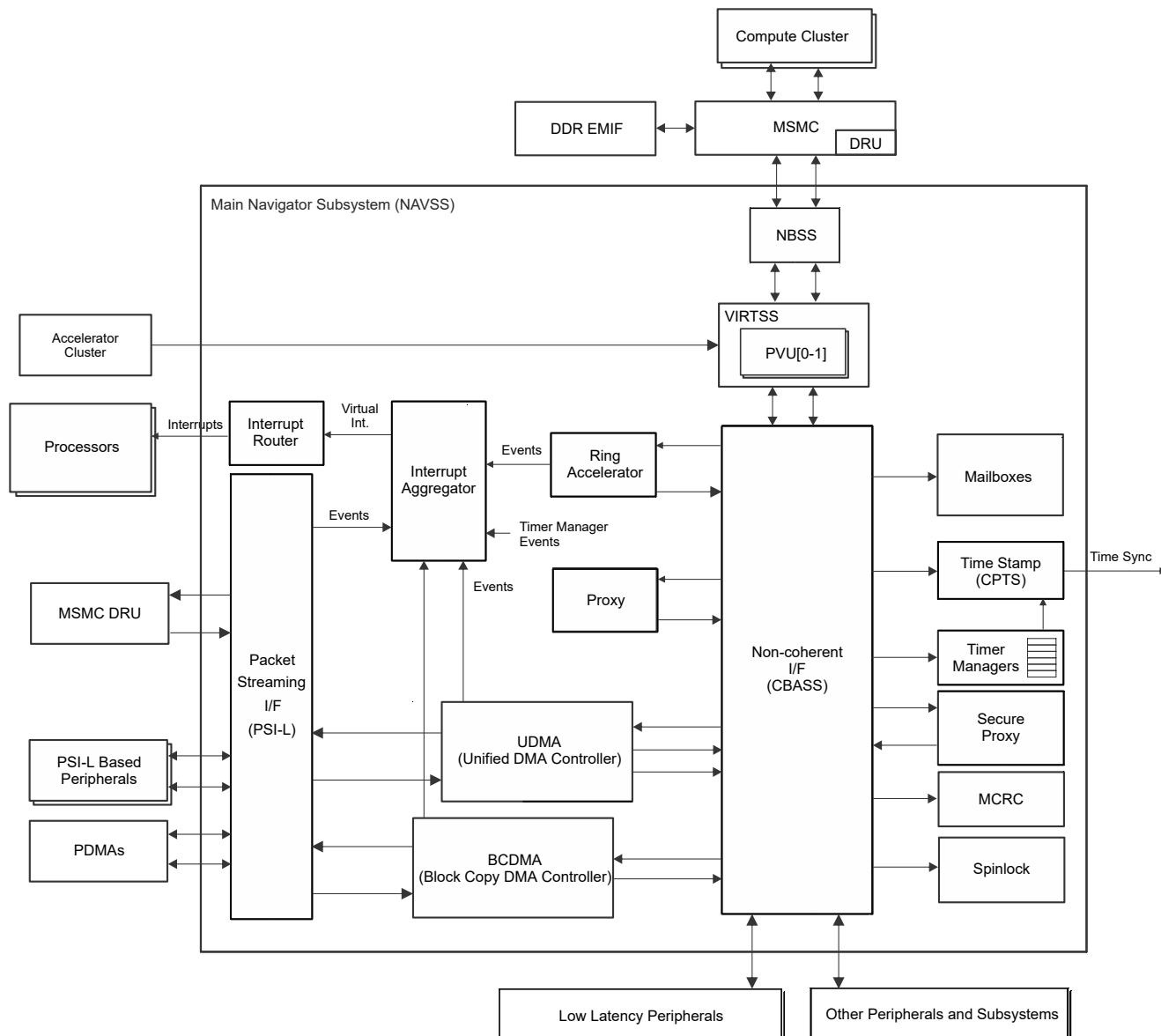
The DMA architecture specifies the data structures used by Texas Instruments standard communications modules to facilitate direct memory access (DMA) and to provide a consistent application programming interface (API) to the host software in multi-core devices. The data structures and the API used to manipulate them will be jointly referred to as NAVSS.

Frequent tasks are commonly offloaded from the host processor to peripheral hardware to increase system performance. Significant performance gains may result from careful design of the host software and communication module interface. In networking systems, packet transmission and reception are critical tasks. Texas Instruments has developed the NAVSS standard, which is aimed at maximizing the efficiency of interaction between the host software and communications modules.

The design goals for NAVSS are as follows:

- Minimize host interaction
- Maximize memory use efficiency
- Maximize bus burst efficiency
- Maximize symmetry between transmit/receive operations
- Maximize scalability for number of connections, buffer sizes, queue sizes, and protocols supported
- Minimize protocol specific features
- Minimize complexity

NAVSS architecture is shown on [Figure 10-1](#).



**Figure 10-1. NAVSS Overview**

#### 10.1.1.1 Navigator Subsystem

The Navigator Subsystem (NAVSS hardware) is a container which groups together as much as possible the components which provide the Hardware to Software boundary for data movement control in the system. All of the components which control work handoff (queuing, interrupts, monitoring and debugging) are included in the NAVSS which in turn is located as close as possible to the various host processors to which services are provided. Other DMA components such as DRUs, UTC, and PDMAs exist outside the NAVSS but all are controlled by the root complexes provided in NAVSS.

The SoC has two NAVSSes: one in main and one in MCU domain.

#### 10.1.1.2 Ring Accelerator (RA)

The ring accelerator (RINGACC or RA) is a hardware module that is responsible for accelerating management of various types of queues in the system. The RA accelerates passing of packets between a producer and

consumer in normal system memory (including cached memory). The RA is capable of accelerating the read/write pointer maintenance and occupancy tracking operations.

The ring accelerator operates like a bus infrastructure bridge in that it passes transactions through it between a source and destination interface. As transactions are passed, the RA modifies the address, byte count, and transaction identifiers and internally updates the occupancies/pointers.

Each queue that the RA provides can operate in either exposed-ring-mode or private-queue-mode.

- The exposed ring mode allows software to directly access the underlying ring structure to add or remove items from the tail or head of the ring respectively. Whenever items are added or removed from a ring, the host software is required to write to a corresponding doorbell register to increment or decrement the ring occupancy. When using the exposed ring mode, no proxy is required between the software and the RA but all queue add operations require a memory fence to be performed to ensure that the data has landed in a snooperable cache before the doorbell register is written.
- The private queue mode provides an abstract view of the ring so that the host software does not need to know the actual physical address location or current read or write indexes of the ring. The private queue mode provides a memory window for each ring which when written redirects the write to the address pointed to by the write pointer and when read redirects the read to the address pointed to by the read pointer. The same address range is always used to push or pop elements from a given queue.

Rings are an implementation of a logical queue with the following limitations:

- Each ring has a finite size. When rings are used in exposed ring mode the following conditions apply:
  - A separate operation is required to write/read the contents of a ring and to update the occupancy of the ring
  - It is not straightforward to allow multiple producers to write to a ring or multiple consumers to read from a ring. Additional synchronization and pointer passing is required since rings require software to manage one of the pointers for the queue.
  - An exposed ring cannot be used when software or hardware needs to both read and write the same queue (such as for a DMA RX free queue where errors can have the DMA write the element back onto the same RX free queue). Since software owns one side of the pointers and hardware owns the other, they cannot be kept coherent if either side needs to update either at any time.

The RA allows each ring to be configured to a different primary element size. Element sized chunks of data are placed onto and retrieved from rings when queuing or de-queuing occurs. Element sizes can be as small as 4 bytes or as large as 256 bytes.

#### 10.1.1.3 Proxy

The proxy provides a mechanism for software to access the RA coherently when the processor does not support large bursts. The RA requires a single burst for each operation so that it maintains atomicity and coherence by relying on the atomicity of the bursts on the bus interconnect fabric, since it only delivers a single burst to the RA before the next. Since processors are usually limited in the size of a burst they can deliver natively, such as 32 to 128 bits, they cannot be used directly with the data access region of the RA for larger element sizes. The proxy solves this gap by providing a temporary space for the software to form a larger burst of data before it is sent to the RA. The proxy allows smaller processor accesses to the data and only forwards to the RA when the entire data burst is complete, as directed by the software. Each proxy hardware can support multiple threads of software (whether on the same or different processors) operating on bursts at the same time, and they do not interfere with each other, as long as each thread of software only operates on its thread in the proxy. Each proxy thread also supports access to any RA queue via different offsets similar to how the RA provides access to the queues via different offsets.

For writing data to the RA, the software will write the data in native sized writes to the proxy thread it has been assigned. The proxy simply accepts the writes into a buffer reserved for that thread. Only when the software writes to the completion byte offset (last byte of the burst) then will the proxy take the entire burst of data written (including the final write data) and send it to the RA as a single write burst. After the completion write, the software can begin building a new data burst. For reading data from the RA, the software will read the data in

native sized reads to the proxy thread. For the first read, the proxy will read the entire burst from the RA, so that it atomically gets the next element off the queue. This entire data is stored in the proxy thread buffer, and the requested portion is returned to software. Then software can read any location within the burst and the proxy will read it from the buffer. When software is done reading the data, it must read the completion byte offset so that the proxy 'knows' that the read burst is completed, and that for the next read it must read a burst from the RA queue again. A single proxy thread can only be used for a single data burst at a time, so once starting a write it should complete before starting another write or a read, and similarly once starting a read, it should complete before starting another read or a write. If software needs to read and write at the same time, it should use two proxy threads. Similarly if there are multiple threads of software that need to access RA at the same time, they should use separate proxy threads.

#### 10.1.1.4 Secure Proxy

A special version of a proxy is the secure proxy that provides secured access to RA queues for communication to the security master of the system. The secure proxy provides basically the same function as a normal proxy. The data burst size is fixed for message sizes to the security master, so there is no support for all the element sizes of the RA. The security master locks down the RA queue to access and direction for each proxy thread so that the software cannot access anything they shouldn't, so the software only sees a single queue (unlike the normal proxy), and any attempts to access another queue or access it in violation of how the security master defined and an error will be flagged. This allows for a much more controlled setup for passing messages to the security master.

#### 10.1.1.5 Interrupt Aggregator (INTA)

The interrupt aggregator (INTR\_AGGR or INTA) is a hardware module which provides a persistent view of the real time status of various DMA components within the overall SoC DMA system. The main function of the module is to convert between 'global events' (assertion and de-assertion events transmitted on the event-transport-lane bus) and 'local events' (level sensitive signals on output, and level or edge sensitive signals on input).

The INTA block provides the following functions:

- Conversion of local event signal lines into global events
- Performing an 'Y-split' operation of global events
- Counting global events
- Steering and aggregation of global events into specified bit positions in one of N interrupt cause registers
- SoC interrupt-architecture-compliant set/clear and mask set/clear functionality to cover interrupt cause registers
- Output of virtual interrupt sources to the Interrupt Router module

#### 10.1.1.6 Interrupt Router (IR)

The interrupt router (INTR\_ROUTER or IR) is a hardware module which allows for virtual interrupt sources which were output from the INTA block to be crossbar switched onto physical interrupts which are connected to the system level Interrupt Controller (ARM GIC or other interrupt controller).

#### 10.1.1.7 Unified DMA – Third Party Channel Controller (UDMA-C)

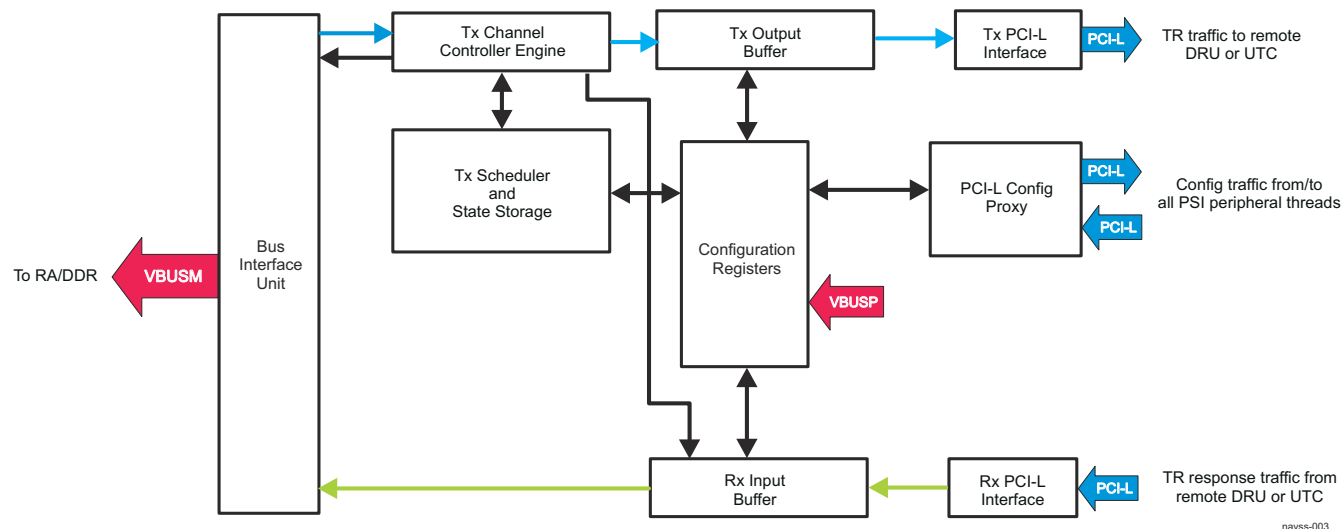
##### Note

In this SoC, UDMA-P (see [Section 10.1.1.10](#)) serves UDMA-C functions. That is, there is no discrete UDMA-C controller in this SoC device.

The Unified DMA 3<sup>rd</sup>-Party Channel Controller is intended to perform similar (but significantly upgraded) functions to the EDMA Channel Controller engine on previous SoC devices. The UDMA-C module supports the transmission of Transfer Request packets from memory mapped data structures to data channels within UTCs in the system and reception of Transfer Response packets from data channels in UTCs in the system into memory mapped data structures. Up to 512 (configuration specific) third-party DMA control channels are provided within the UDMA-C. Each control channel corresponds to a single third party DMA data channel in a separate UTC.

Transfer Request messages provide transfer parameters which are to be used by the UTC channels to move data in the system (previously implemented as PARAM entries). Transfer Response messages are sent back from the UTC in a one to one relationship with Transfer Request packets and provide an indication of both completion of a data transfer and whether or not any exception or error occurred during the data transfer.

The UDMA-C controller maintains state information for each of its channels which allows Transfer Request packet transmission and Transfer Response packet reception operations to be time division multiplexed between channels in order to share the underlying DMA hardware. An internal DMA scheduler is used to control the ordering and rate at which this multiplexing occurs for the transmission of Transfer Request packets. The ordering and rate of Transfer Response receive operations is directly controlled by the order in which those packets are received on the Rx PSI-L interface. A block diagram of a UDMA-C module is shown Figure 10-2.



**Figure 10-2. UDMA-C Block Diagram**

#### 10.1.1.8 Unified Transfer Controller (UTC)

The Unified Transfer Controller is intended to perform similar functions to the EDMA Transfer Controller engine used on previous devices. The UTC engine is generally classified as a third-party DMA. This designation comes from the fact that the engine is not actually the source or sink of the data which is being moved but is instead an intermediary 3<sup>rd</sup>-party that performs the data move on behalf of the source and sink.

The UTC engine accepts Transfer Response messages from the UDMA-P via a PSI-L interface which provide instructions to copy data between a source read interface and a destination write interface. The sequence of operations that can be instructed includes up to 4-dimensional nested loops. Multiple types of Transfer Request messages are specified and each UTC instance in the system may support all types or any specified subset. The TR formats are specified in detail in a later section. When a Transfer Request has been completed, the UTC returns a Transfer Response message back to the originating UDMA-P. The UTC has the ability to generate events at specified completion points when processing a Transfer Request. These events are sent back to the Interrupt Aggregator block.

An UTC instance may be configured to only support VBUSM to VBUSM block copies or they may also optionally support 'split' operations where a read engine is instructed by a TR to read data from a VBUSM interface and send the data to the Packet Streaming fabric via a PSI-L master interface. A similar split operation is supported for writes where data is received into a PSI-L slave interface and is then routed to a write engine which has been instructed by a TR to move the data into a set of memory locations.

#### 10.1.1.9 Data Routing Unit (DRU)

The DRU is a special version of a UTC that also includes support for advanced operations which are critical to cache maintenance and data IO to the C7x DSP cores.

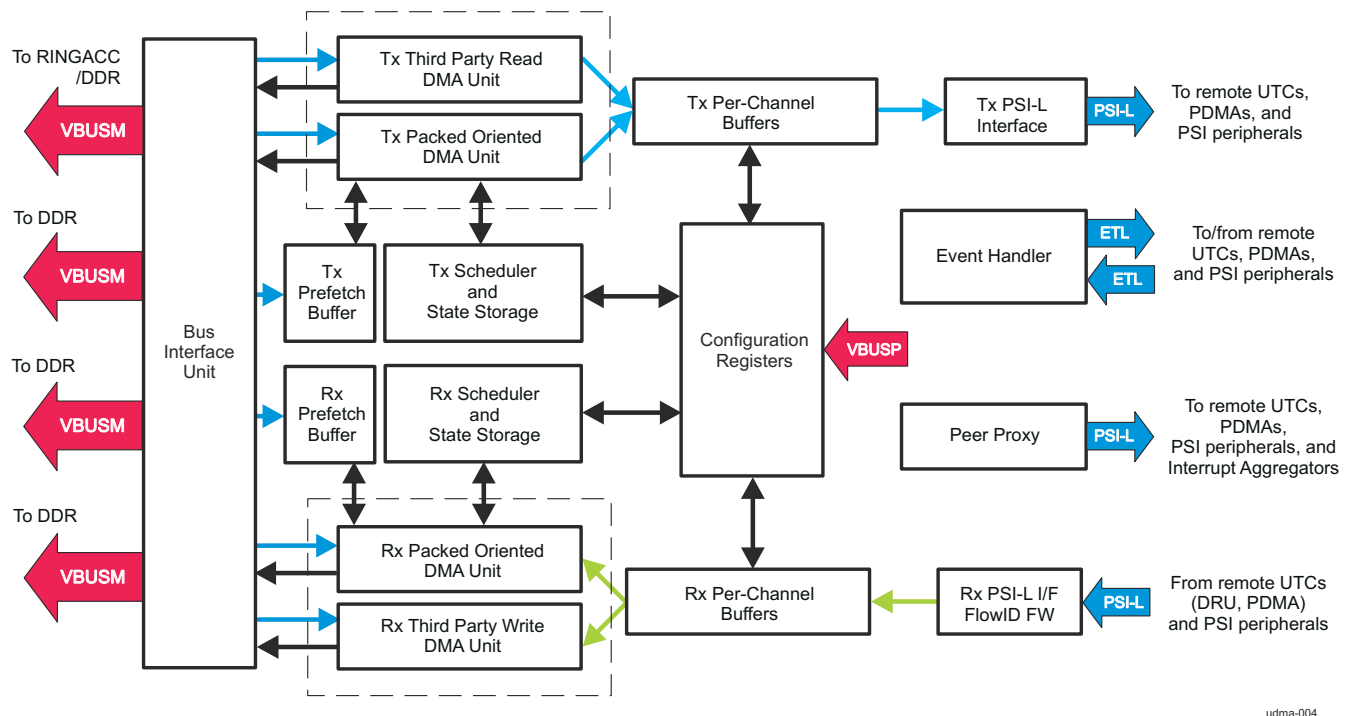
For purposes of the K3 Data Movement architecture, the DRU is essentially a UTC which supports only the block copy mode subset of the Transfer Request message formats. The DRU typically will provide the highest performance block copy data movement capability of any DMA engine within the SoC. The data routing unit (DRU) is a high bandwidth, flexible routing engine with programmable DMA transfer requests. It enables user to perform high speed data transfers between memory mapped slave endpoints, processor caches and shared caches. DRU behaves like a DMA transfer controller, moving data at CC\_ARMSS frequency.

#### 10.1.1.10 Unified DMA – Peripheral Root Complex (UDMA-P)

The UDMA-P is intended to perform similar (but significantly upgraded) functions as the packet-oriented DMA used on previous SoC devices. The UDMA-P module supports the transmission and reception of various packet types. The UDMA-P is architected to facilitate the segmentation and reassembly of SoC DMA data structure compliant packets to/from smaller data blocks that are natively compatible with the specific requirements of each connected peripheral. Multiple Tx and Rx channels are provided within the DMA which allow multiple segmentation or reassembly operations to be ongoing. The DMA controller maintains state information for each of the channels which allows packet segmentation and reassembly operations to be time division multiplexed between channels in order to share the underlying DMA hardware. An external DMA scheduler is used to control the ordering and rate at which this multiplexing occurs for Transmit operations. The ordering and rate of Receive operations is indirectly controlled by the order in which blocks are pushed into the DMA on the Rx PSI-L interface.

The UDMA-P also supports acting as both a UTC and UDMA-C for its internal channels. Channels in the UDMA-P can be configured to be either Packet-Based or Third-Party channels on a channel by channel basis.

A block diagram of the UDMA-P Controller is shown in [Figure 10-3](#).



**Figure 10-3. UDMA-P Block Diagram**

##### 10.1.1.10.1 Channel Classes

Not all channels within a DMA controller require the same performance capabilities. Raw bandwidth and latency tolerance requirements will vary greatly between data transfers at various points in the system. Several different classes of DMA channels are provided in the UDMA-P and are described in [Table 10-1](#)



**Table 10-1. Channel Classes**

Class Name	Description
Normal Capacity	Provides baseline amount of descriptor and TR prefetch and Tx/Rx control and data buffering. Suitable for most peripheral transfers which are communicating with on-chip memories and DDR
High Capacity	Provides an elevated amount of descriptor and TR prefetch and custom Tx/Rx control and data buffering. Suitable for applications which require moderate per-channel bandwidth with significantly increased latency (for example Transferring data to/from PCIe)
Ultra-high Capacity	Provides identical descriptor and TR prefetch as High Capacity but with increased Tx data buffering to allow for more read data in flight. Suitable for applications requiring high per-channel bandwidth with significantly increased latency (for example 16+ Gbps transfers to/from PCIe).

#### 10.1.1.11 Peripheral DMA (PDMA)

The Peripheral DMA is a simple DMA which has been architected to specifically meet the data transfer needs of peripherals which perform data transfers using memory mapped registers accessed via a standard non-coherent bus fabric. The PDMA module is intended to be located close to one or more peripherals which require an external DMA for data movement and is architected to reduce cost by using VBUSP interfaces and supporting only statically configured Transfer Request operations. The PDMA is only responsible for performing the data movement transactions which interact with the peripherals themselves. Data which is read from a given peripheral is packed by a PDMA source channel into a PSI-L data stream which is then sent to a remote peer UDMA-P destination channel which then performs the movement of the data into memory. Likewise, a remote UDMA-P source channel fetches data from memory and transfers it to a peer PDMA destination channel over PSI-L which then performs the writes to the peripheral.

The Peripheral DMA architecture is intentionally heterogeneous (UDMA-P + PDMA) to right size the data transfer complexity at each point in the system to match the requirements of whatever is being transferred to or from. Peripherals are typically FIFO based and do not require multi-dimensional transfers beyond their FIFO dimensioning requirements, so the PDMA transfer engines are kept simple with only a few dimensions (typically for sample size and FIFO depth), hardcoded address maps, and simple triggering capabilities.

Multiple source and destination channels are provided within the PDMA which allow multiple simultaneous transfer operations to be ongoing. The DMA controller maintains state information for each of the channels and employs round-robin scheduling between channels in order to share the underlying DMA hardware.

#### 10.1.1.12 Embedded DMA

Embedded DMA refers to a DMA-capable master module which either follows an industry standard hardware/software Application Programming Interface (API) or is a module which does not natively comply with the TI DMA architecture requirements. It is assumed that modules which include Embedded DMA will provide their own dedicated software drivers and interoperability with other DMA entities in the system will be accomplished with software bridging code. None of the remainder of this chapter has any effect on the operation or architecture of Embedded DMA functions.



### 10.1.1.13 Definition of Terms

<b>Host</b>	The host is an intelligent system resource that configures and manages each communications control module. The host is responsible for allocating memory, initializing all data structures, and responding to port interrupts.
<b>Channel</b>	A channel refers to the sub-division of information (flows) that is transported across a DMA engine. Each channel has associated state information. Channels are used to segregate information flows based on the protocol used, scheduling requirements (for example, CBR, VBR, ABR), or concurrency requirements (that is, blocking avoidance). Information flow within a channel is a stream of strongly ordered information.
<b>Data Buffer</b>	A data buffer is a single data structure that contains payload information for transmission to or reception from a channel.
<b>Buffer Descriptor</b>	A buffer descriptor is a single data structure that contains information about one or more data buffers.
<b>Packet Descriptor</b>	A packet descriptor is another name for the first buffer descriptor within a packet. Some fields within a data buffer descriptor are only valid when it is a packet descriptor including the tags, packet length, packet type, and flags. All Monolithic type descriptors are packet descriptors (and are also a Data Buffer).
<b>Queue</b>	A queue is a list of strongly ordered entries which is typically used to pass work between a producer and a consumer. Queue entries in most cases are references to a work payload which is being passed but in some cases (Transfer Request Packets for example) queue entries may actually contain data which is being transferred. Queues are used throughout UDMA whenever communication is required between entities. Queues can have multiple different implementations and UDMA uses two of the most common: linked lists and rings.
<b>Linked list</b>	A linked list is a data structure in which each entry stores not only the entry data but also a chaining pointer to the next entry in the list. The last entry in each list has its chaining pointer set to NULL (typically encoded as 0x0). The list manager maintains a pointer to the head element on the list and to the tail element on the list. Since the chaining pointer is stored with the entry data, linked lists have a length which is dynamically changeable and limited only by the ability to allocate additional entries which are to be queued/de-queued. Linked lists are present in Host Descriptors to chain multiple descriptors to form a packet
<b>Ring</b>	A ring is a data structure in which a contiguous memory block defined by M N-byte entries (total size is M × N bytes) is statically allocated and sequentially written/read in order to pass data or data references. Rings are also referred to as circular buffers because when the last element in the contiguous memory array is written, the pointers wrap back to the beginning address for the ring and start the process all over again. The Ring Accelerator component uses rings in order to implement logical queues
<b>Free Descriptor/Buffer Queue</b>	A free descriptor/buffer queue is a list of available descriptors with pre-linked empty buffers that are to be used by the receive ports for host type descriptors. Free Descriptor/Buffer Queues are implemented by the Ring Accelerator.
<b>Free Descriptor Queue</b>	A free descriptor queue is a list of available descriptors that are not yet linked with buffers that are to be used by the receive ports for monolithic type descriptors. Free Descriptor Queues are implemented by the Ring Accelerator.
<b>Packet Queue</b>	A packet queue is a list of valid (that is, populated) packet descriptors that is used for forwarding a packet from one entity to another for any number of purposes. Packet Queues are implemented by the Ring Accelerator.
<b>Memory</b>	Memory is an area of data storage managed by the host. This area is visible to the port as a 64-bit addressable area.
<b>Device Driver</b>	A device driver is application independent software that runs on the host for purposes abstracting the low level hardware so that upper level software can use the hardware without

knowing every bit field location or initialization sequence.. General device driver functions include port initialization, transmit packet queuing, and receive packet processing.

**SOP** Start of Packet. This refers to the descriptor/buffer that is the first buffer in a packet.

**MOP** Middle of Packet. This refers to the descriptors/buffers that are neither the first or last buffers in a packet.

**EOP** End of Packet. This refers to the descriptor/buffer that is the last buffer in a packet.

### 10.1.2 DMA Hardware/Software Interface

The intent of the TI Data Movement (DMA) architecture is to provide a uniform hardware/software interface which includes a rich set of mechanisms that software can make use of to transfer data with low overhead and reasonable complexity. To this goal, the number of components which software is required to interact with on an ongoing real time basis has been kept to a minimum and is as follows (in order of anticipated frequency of access):

- Interrupt Aggregator
- Ring Accelerator
- UDMA-P/C (for managed flow control and error/exception handling only)

When interrupts are used, the interrupt aggregator will provide the vast majority of interrupt sources from all DMA components in the system. All non-exception/non-debug packet and TR completion signaling originates from the Interrupt Aggregator and the INTA provides a uniform set of registers that can be queried to quickly determine the cause of a specific interrupt. Events from PSI-L/ETL components are all routed to the host via the Interrupt Aggregator.

The Ring Accelerator provides the primary means by which work is sent to or received from the UDMA-P DMA engines (and the UTC/DRU, and PDMA engines whose control operations are proxied by the UDMA-P).

The UDMA-P are primarily fronted by the RA and the INTA but for real time operations like software controlled flow control (via pause) some registers in the UDMA-P blocks may be directly manipulated.

Data structures are used to pass anytime information between components in the system. These components may be hardware or software. The following sections describe the data structures which are used within a UDMA based system for passing information. These data structures include data buffers, packet descriptors, buffer descriptors, queues (including transmit queues, transmit completion queues, and receive queues) and the configuration registers that are provided in the various components. The following sections provide a detailed description of these data structures.

#### 10.1.2.1 Data Buffers

A data buffer is a byte aligned contiguous block of memory used to store packet payload data. Each buffer is described in an entry in either a packet descriptor or in a buffer descriptor. A data buffer may hold any portion of a packet and may be linked together (via descriptors) with other buffers to form packets. Data buffers may be allocated anywhere within the 64-bit memory space.

The Buffer Length field of the packet/buffer descriptor indicates the number of valid data bytes in the buffer. There may be from 1 to 4M-1 valid data bytes in each buffer.

#### 10.1.2.2 Descriptors

Descriptors are so named because their primary function is to describe other data structures.

The UDMA architecture provides 4 basic types of descriptors each of which has specific characteristics intended to address different requirements. [Table 10-2](#) shows the different types of descriptors and their characteristics.

**Table 10-2. UDMA Descriptor Types and Attributes**

Descriptor Type	Includes Valid Packet Info	Provides Number of Slots to Link In External Data Buffers	Provides Local Packet Data Storage	Provides Local Protocol Specific Storage	Provides Slot to Link Additional Descriptors Within Same Packet	Description
Host Packet Descriptor	✓	1		✓	✓	Used to describe packet and SOP buffer in applications that require Host OS compatible data structures (i.e. applications where the descriptors and buffers cannot be managed independently but must instead be pre-linked by the Host software). These applications inherently require a separate descriptor for each buffer.

**Table 10-2. UDMA Descriptor Types and Attributes (continued)**

Descriptor Type	Includes Valid Packet Info	Provides Number of Slots to Link In External Data Buffers	Provides Local Packet Data Storage	Provides Local Protocol Specific Storage	Provides Slot to Link Additional Descriptors Within Same Packet	Description
Host Buffer Descriptor		1			✓	Used to describe non-SOP buffers in applications that require Host OS compatible data structures
Monolithic Packet Descriptor	✓	0	✓	✓		Used to provide descriptor and data information in one contiguous data structure.
Transfer Request Packet Descriptor		0				Used to feed transfer request sequences to the UDMA third party channel controller

As [Table 10-2](#) shows, two of the four different descriptor types (the Host Packet Descriptor and Monolithic Packet Descriptor) provide packet level information that is useful to both the ports and the Host in order to properly process the packet. These descriptors are referred to as Packet Descriptors and will always appear as the first descriptor (or only descriptor in the case of Monolithic types) within a packet.

Software must allocate descriptors on 16-byte address boundaries. 16-byte is the minimum granularity that UDMA supports. This is intended for better memory utilization by allowing on-chip descriptors which are not a power of 2 in length to be packed into arrays with little wasted memory space.

Even though descriptors and buffers may be allocated on any 16-byte alignment, careful consideration of the alignment effects should be made based on the storage location and any cache related affects that may exist. If data structures are placed in off-chip SDRAM the burst size and alignment restrictions of the memory devices must be considered in order to avoid performance issues related to continually fetching mis-aligned blocks. In this case, the memory efficiency can be reduced to 50% because two memory bank lines are read for every line sized data fetch. Similarly, placing more than one descriptor or buffer object within a single cache line can cause the adjacent object to become corrupted during cache line writeback operations.

Software/application specific control information may be added to the end of any of the packet descriptor types using as many extra words as necessary but the above alignment rules still apply.

#### 10.1.2.2.1 Host Packet Descriptor

Host Packet Descriptors are designed to be used when the application requires support for true, unlimited fragment count scatter/gather type operations. The Host Packet Descriptor contains the following information:

- Indicator which identifies the descriptor as a Host Packet Descriptor
- Source and Destination Tags
- Packet Type
- Packet Length
- Protocol Specific Region Size
- Protocol Specific Control/Status Bits
- Pointer to the first valid byte in the SOP data buffer
- Length of the SOP data buffer
- Pointer to the next buffer descriptor in the packet
- Software specific information

Host Packet Descriptors always contain 48 bytes of required information and may also contain optional software specific information and protocol specific information. How much optional information (and therefore the allocated size of the descriptors) is required is application dependent.

The Host Packet descriptor layout is shown below.

Packet Info (16 bytes)
Linking Info (8 bytes)
Buffer Info (12 bytes)
Original Buffer Info (12 bytes)
Extended Packet Info Block (Optional) Includes Timestamp and Software Data (16 bytes)
Protocol Specific Data (Optional) (0 to M bytes where M is a multiple of 4)
Other Software Data (Optional and User Defined)

Host Packet Descriptors may be linked with zero or more additional Host Buffer Descriptors in a singly linked list fashion to form packets. Each Host Packet consists of a single Host Packet Descriptor followed by a chain of zero or more Host Buffer Descriptors linked together using the Next Descriptor Pointer fields in the descriptors. The last descriptor in a Host packet has a zero Next Descriptor Pointer.

The 'Other Software Data' portion of the descriptor exists after all of the defined words and is reserved for use by the host software to store completely private data. This region is not used in any way by hardware components in a UDMA system and these modules will not modify any bytes within this region.

The contents of the Host Packet Descriptor words are detailed in [Table 10-3](#) through [Table 10-19](#).

**Table 10-3. Host Packet Descriptor Packet Information Word 0 (PD Word 0)**

Bits	Name	Description	Rx Overwrite
31:30	2'd1	64-bit Host Packet Descriptor Type Identifier	Yes
29	Extended Packet Info Block Present	This field indicates the presence of the Extended Packet Info Block in the descriptor. 0 = EPIB is not present 1 = 16 byte EPIB is present	Yes
28	Protocol Specific Region Location	This field indicates the location of the Protocol Specific Words: 0 = PS Words are located in the descriptor 1 = PS Words are located in the SOP Buffer immediately prior to the data.	Yes
27:22	Protocol Specific Valid Word Count	This field indicates the valid number of 32-bit words in the protocol specific region. This is encoded in increments of 4 bytes as follows: 0 = 0 bytes 1 = 4 bytes ... 16 = 64 bytes ... 32 = 128 bytes 33-63 = RESERVED	Yes
21:0	Packet Length	The length of the packet in bytes. If the Packet Length is less than the sum of the buffer lengths, then the packet data will be truncated. A Packet Length greater than the sum of the buffers is an error unless the packet length is set to 0x3FFFFFF. If the packet length is set to all 1s (0x3FFFFFF) the Tx port will disable truncation and will transmit as much data as is specified in the Buffer Length fields. On Rx, if the received data exceeds 4M-1 bytes, the packet length field will saturate to a value of 0x3FFFFFF. The valid range for an exact packet length is 0 to 4M-1 bytes. If the packet length is set to 0, the port will not actually transmit any information. Instead, the port will perform buffer/descriptor reclamation as instructed in the return information in word 2.	Yes

**Table 10-4. Host Packet Descriptor Packet Information Word 1 (PD Word 1)**

Bits	Name	Description	Rx Overwrite
31:28	Error Flags	This field contains error flags that can be assigned based on the packet type	Yes
27:24	Protocol Specific Flags	This field contains protocol specific flags/information that can be assigned based on the packet type.	Yes
23:14	Packet ID	Unique Packet ID for packet within FlowID	Yes
13:0	Flow ID	Flow ID within which this packet is being transported. The FlowID is used by downstream blocks to make decisions about packet steering and resource allocations. FlowIDs are also used to allow specific packets to be received into specific sets of buffers.	Yes

**Table 10-5. Host Packet Descriptor Packet Information Word 2 (PD Word 2)**

Bits	Name	Description	Rx Overwrite
31:27	Packet Type	This field indicates the type of this packet and is encoded as follows: 0-31 = Application specific	Yes
26:19	RESERVED	Reserved	No
18	Return Policy	This field indicates the return policy for this packet. 0 = Entire packet (still linked together) should be returned to queue specified in bits 15:0. 1 = Each buffer should be returned to queue specified in bits 15:0 of Word 2 in their respective descriptors. The Tx DMA will return each buffer in sequence.	No
17	Early Return	This field indicates that each buffer pointer should be immediately returned to the specified queue when data transfer is started from the packet instead of waiting for the entire buffer to be emptied. This flag is used to enable in-place reception of packets on a Receive Channel while the source packet is in the process of being transferred on a Transmit Channel. 0 = Buffer/Packet descriptor pointers should only be returned after all reads have been completed 1 = Buffer/Packet descriptor pointers should be returned immediately upon fetching the descriptor and beginning to transfer data.	No
16	Return Push Policy	This field indicates how a Transmit or Receive DMA should return the descriptor pointers to the free queues This field is encoded as follows: 0 = Descriptor must be returned to tail of queue 1 = Descriptor must be returned to head of queue This bit is only used when the Return Policy bit is set to 1. The Rx DMA will only use this field when an error occurs during reception and the DMA must return descriptors back to the free queue from which they came. This field must be set to 0 for descriptors which will be placed on queues managed by the Ring Accelerator.	No
15:0	Packet Return Queue/ Ring Num	This field indicates the ring number in the RA that the descriptor is to be returned to after transmission is complete. The value 0xFFFF is reserved.	No

**Table 10-6. Host Packet Descriptor Packet Information Word 3 (PD Word 3)**

Bits	Name	Description	Rx Overwrite
31:24	Source Tag – Hi	This field is application specific. During Packet reception, the DMA controller in the port will overwrite this field as specified in the rx_src_tag_hi_sel field in the flow configuration table entry.	Configurable
23:16	Source Tag – Lo	This field is application specific. During Packet reception, the DMA controller in the port will overwrite this field as specified in the rx_src_tag_lo_sel field in the flow configuration table entry.	Configurable
15:8	Dest Tag – Hi	This field is application specific. During Packet reception, the DMA controller in the port will overwrite this field as specified in the rx_dest_tag_hi_sel field in the flow configuration table entry.	Configurable

**Table 10-6. Host Packet Descriptor Packet Information Word 3 (PD Word 3) (continued)**

Bits	Name	Description	Rx Overwrite
15:0	Dest Tag – Lo	This field is application specific. During Packet reception, the DMA controller in the port will overwrite this field as specified in the rx_dest_tag_lo_sel field in the flow configuration table entry.	Configurable

**Table 10-7. Host Packet Descriptor Linking Word 0 (PD Word 4)**

Bits	Name	Description	Rx Overwrite
31:0	Next Descriptor Pointer LSB	The 32 LSBs of the 48-bit, 16-byte aligned (min), memory address of the next buffer descriptor in the packet. If the value of this pointer is zero then the current buffer is the last buffer in the packet. The host sets the Next Descriptor Pointer.	Yes

**Table 10-8. Host Packet Descriptor Linking Word 1 (PD Word 5)**

Bits	Name	Description	Rx Overwrite
31:20	RESERVED	Reserved	Yes
19:16	Next Descriptor Pointer Address Space Select	Effectively bits 51:48 of the address. The value given in this field will be output by the DMA masters on the casel pin which is used by the infrastructure as an identifier for which address space this particular memory region is located within. Address space 0 is the default unified address space for a given device. Address spaces 1-15 are used for alternate address maps which may be external to the device (PCIe/Hyperlink) or in other 'tiles' on large devices.	Yes
15:0	Next Descriptor Pointer MSB	The 16 MSBs of the 48-bit next descriptor pointer	Yes

**Table 10-9. Host Packet Descriptor Buffer 0 Info Word 0 (PD Word 6)**

Bits	Name	Description	Rx Overwrite
31:0	Buffer 0 Pointer LSB	The Buffer Pointer is the byte aligned memory address of the buffer associated with the buffer descriptor. This value will be written during reception. If the protocol specific words are placed at the beginning of the SOP buffer, this pointer will point to the PS words. The offset to the data in that case must be calculated by the consumer using the Protocol Specific Valid Count from Word 2. These are the 32 LSBs of the 48-bit buffer pointer.	Yes

**Table 10-10. Host Packet Descriptor Buffer 0 Info Word 1 (PD Word 7)**

Bits	Name	Description	Rx Overwrite
31:20	RESERVED	Reserved	Yes
19:16	Buffer 0 Pointer Address Space Select	Effectively bits 51:48 of the address. The value given in this field will be output by the DMA masters on the casel pin which is used by the infrastructure as an identifier for which address space this particular memory region is located within. Address space 0 is the default unified address space for a given device. Address spaces 1-15 are used for alternate address maps which may be external to the device (PCIe/Hyperlink) or in other 'tiles' on large devices.	Yes
15:0	Buffer 0 Pointer MSB	The 16 MSBs of the 48-bit buffer pointer	Yes

**Table 10-11. Host Packet Descriptor Buffer 0 Info Word 2 (PD Word 8)**

Bits	Name	Description	Rx Overwrite
31:22	RESERVED	Written to 0	Yes
21:0	Buffer 0 Length	The Buffer Length field indicates how many valid data bytes are in the buffer. Unused or protocol specific bytes at the beginning of the buffer are not counted in the Buffer Length field. This value will be overwritten during reception.	Yes

**Table 10-12. Host Packet Descriptor Original Buffer Info Word 0 (PD Word 9)**

Bits	Name	Description	Rx Overwrite
31:22	RESERVED		No



**Table 10-12. Host Packet Descriptor Original Buffer Info Word 0 (PD Word 9) (continued)**

Bits	Name	Description	Rx Overwrite
21:0	Original Buffer 0 Length	The Buffer Length field indicates the original size of the buffer in bytes. Data bytes are in the buffer. This value will not be overwritten during reception. This value is read by the Rx DMA to determine the actual buffer size as allocated by the host at initialization. Since the buffer length in PD Word 8 is overwritten by the Rx port during reception, this field is necessary to permanently store the buffer size information.	No

**Table 10-13. Host Packet Descriptor Original Buffer Info Word 1 (PD Word 10)**

Bits	Name	Description	Rx Overwrite
31:0	Original Buffer 0 Pointer LSB	The Buffer Pointer is the byte aligned memory address of the buffer associated with the buffer descriptor. This value will not be overwritten during reception. This value is read by the RX DMA to determine the actual buffer location as allocated by the host at initialization. Since the buffer pointer in PD Words 6/7 is overwritten by the Rx port during reception, this field is necessary to permanently store the buffer pointer information. This field contains the 32 LSBs of the 48-bit original buffer pointer	No

**Table 10-14. Host Packet Descriptor Original Buffer Info Word 2 (PD Word 11)**

Bits	Name	Description	Rx Overwrite
31:20	RESERVED	Reserved	Yes
19:16	Original Buffer 0 Pointer Address Space Select	Effectively bits 51:48 of the address. The value given in this field will be output by the DMA masters on the casel pin which is used by the infrastructure as an identifier for which address space this particular memory region is located within. Address space 0 is the default unified address space for a given device. Address spaces 1-15 are used for alternate address maps which may be external to the device (PCIe/Hyperlink) or in other 'tiles' on large devices.	Yes
15:0	Original Buffer 0 Pointer MSB	The 16 MSBs of the 48-bit original buffer pointer	No

**Table 10-15. Host Packet Descriptor Extended Packet Info Block Word 0 (Optional)**

Bits	Name	Description	Rx Overwrite
31:0	Timestamp Info	This field contains an application specific timestamp which can be used for traffic shaping in a QoS enabled system.	Configurable

**Table 10-16. Host Packet Descriptor Extended Packet Info Block Word 1 (Optional)**

Bits	Name	Description	Rx Overwrite
31:0	Software Info 0	This field stores software centric information that needs to travel with the packet through the stack. This information will be copied from the source descriptor to the destination descriptor whenever a prefetch operation is performed or when transferring through an infrastructure DMA node.	Configurable

**Table 10-17. Host Packet Descriptor Extended Packet Info Block Word 2 (Optional)**

Bits	Name	Description	Rx Overwrite
31:0	Software Info 1	This field stores software centric information that needs to travel with the packet through the stack. This information will be copied from the source descriptor to the destination descriptor whenever a prefetch operation is performed or when transferring through an infrastructure DMA node.	Configurable

**Table 10-18. Host Packet Descriptor Extended Packet Info Block Word 3 (Optional)**

Bits	Name	Description	Rx Overwrite
31:0	Software Info 2	This field stores software centric information that needs to travel with the packet through the stack. This information will be copied from the source descriptor to the destination descriptor whenever a prefetch operation is performed or when transferring through an infrastructure DMA node.	Configurable



**Table 10-19. Host Packet Descriptor Protocol Specific Word N (Optional)**

Bits	Name	Description	Rx Overwrite
31:0	Protocol Specific Data N	This field stores information which varies depending on the block and packet type.	Configurable

#### 10.1.2.2.2 Host Buffer Descriptor

The Host Buffer Descriptor is identical in size and organization to a Host Packet Descriptor but does not include valid information in the packet level fields and does not include a populated region for protocol specific information. Host Buffer Descriptors are designed to be linked onto a Host Packet Descriptor or another Host Buffer Descriptor to provide support for unlimited scatter/gather type operations. Host Buffer Descriptors provide information about a single corresponding data buffer.

Every Host buffer descriptor stores the following information:

1. Pointer to the first valid byte in the data buffer
2. Length of the data buffer
3. Pointer to the next buffer descriptor in the packet

Host Buffer Descriptors always contain 48 bytes of required information. Since it is a requirement that it is possible to convert a Host descriptor between a Buffer Descriptor and a Packet Descriptor (by filling in the appropriate fields) in practice, Host Buffer Descriptors will be allocated using the same sizes as Host Packet Descriptors.

The Host Buffer Descriptor layout is shown below.

Buffer Reclamation Info (16 bytes)
Linking Info (8 bytes)
Buffer Info (12 bytes)
Original Buffer Info (12 bytes)

A Host Packet Descriptor and zero or more Host Buffer Descriptors may be linked together using the Next Descriptor Pointer fields to form packets. The last descriptor in a packet has a zero Next Descriptor Pointer. Each Host Buffer descriptor also points to a single data buffer.

The contents of the Host Buffer Descriptor words are detailed in [Table 10-20](#) through [Table 10-31](#).

**Table 10-20. Host Buffer Descriptor Reserved Word 0 (BD Word 0)**

Bits	Name	Description	Rx Overwrite
31:0	RESERVED	Reserved	No

**Table 10-21. Host Buffer Descriptor Reserved Word 1 (BD Word 1)**

Bits	Name	Description	Rx Overwrite
31:0	RESERVED	Reserved	No

**Table 10-22. Host Buffer Descriptor Buffer Reclamation Info (BD Word 2)**

Bits	Name	Description	Rx Overwrite
31:18	RESERVED		No

**Table 10-22. Host Buffer Descriptor Buffer Reclamation Info (BD Word 2) (continued)**

Bits	Name	Description	Rx Overwrite
17	Early Return	This field indicates that each buffer pointer should be immediately returned to the specified queue when data transfer is started from the packet instead of waiting for the entire buffer to be emptied. This flag is used to enable in-place reception of packets on a Receive Channel while the source packet is in the process of being transferred on a Transmit Channel. 0 = Buffer/Packet descriptor pointers should only be returned after all reads have been completed 1 = Buffer/Packet descriptor pointers should be returned immediately upon fetching the descriptor and beginning to transfer data.	No
16	Return Push Policy	This field indicates how a Transmit or Receive DMA should return the descriptor pointers to the free queues. This field is encoded as follows: 0 = Descriptor must be returned to tail of queue 1 = Descriptor must be returned to head of queue This bit is only used when the Return Policy bit is set to 1. The Rx DMA will only use this field when an error occurs during reception and the DMA must return descriptors back to the free queue from which they came. This field must be set to 0 for descriptors which will be placed on queues managed by the Ring Accelerator.	No
15:0	Packet Return Queue Num	This field indicates the ring number within the Ring Accelerator that the descriptor is to be returned to after transmission is complete.	No

**Table 10-23. Host Buffer Descriptor Reserved Word 3 (BD Word 3)**

Bits	Name	Description	Rx Overwrite
31:0	RESERVED	Reserved	No

**Table 10-24. Host Buffer Descriptor Linking Word 0 (BD Word 4)**

Bits	Name	Description	Rx Overwrite
31:0	Next Descriptor Pointer LSB	The 32 LSBs of the 48-bit, 16-byte aligned (min), memory address of the next buffer descriptor in the packet. If the value of this pointer is zero then the current buffer is the last buffer in the packet. The host sets the Next Descriptor Pointer.	Yes

**Table 10-25. Host Buffer Descriptor Linking Word 1 (BD Word 5)**

Bits	Name	Description	Rx Overwrite
31:20	RESERVED		Yes
19:16	Next Descriptor Pointer Address Space Select	Effectively bits 51:48 of the address. The value given in this field will be output by the DMA masters on the casel pin which is used by the infrastructure as an identifier for which address space this particular memory region is located within. Address space 0 is the default unified address space for a given device. Address spaces 1-15 are used for alternate address maps which may be external to the device (PCIe/Hyperlink) or in other 'tiles' on large devices.	Yes
15:0	Next Descriptor Pointer MSB	The 16 MSBs of the 48-bit next descriptor pointer	Yes

**Table 10-26. Host Buffer Descriptor Buffer N Info Word 1 (BD Word 6)**

Bits	Name	Description	Rx Overwrite
31:0	Buffer 0 Pointer LSB	The Buffer Pointer is the byte aligned memory address of the buffer associated with the buffer descriptor. This value will be written during reception. If the protocol specific words are placed at the beginning of the SOP buffer, this pointer will point to the PS words. The offset to the data in that case must be calculated by the consumer using the Protocol Specific Valid Word Count from Word 2. These are the 32 LSBs of the 48-bit buffer pointer.	Yes

**Table 10-27. Host Buffer Descriptor Buffer N Info Word 1 (BD Word 7)**

Bits	Name	Description	Rx Overwrite
31:20	RESERVED		Yes

**Table 10-27. Host Buffer Descriptor Buffer N Info Word 1 (BD Word 7) (continued)**

Bits	Name	Description	Rx Overwrite
19:16	Buffer 0 Pointer Address Space Select	Effectively bits 51:48 of the address. The value given in this field will be output by the DMA masters on the casel pin which is used by the infrastructure as an identifier for which address space this particular memory region is located within. Address space 0 is the default unified address space for a given device. Address spaces 1-15 are used for alternate address maps which may be external to the device (PCIe/Hyperlink) or in other 'tiles' on large devices.	Yes
15:0	Buffer 0 Pointer MSB	The MSBs of the 48-bit buffer pointer	Yes

**Table 10-28. Host Buffer Descriptor Buffer N Info Word 2 (BD Word 8)**

Bits	Name	Description	Rx Overwrite
31:22	RESERVED		Yes
21:0	Buffer N Length	The Buffer Length field indicates how many valid data bytes are in the buffer. Unused or protocol specific bytes at the beginning of the buffer are not counted in the Buffer Length field. This value will be overwritten during reception.	Yes

**Table 10-29. Host Buffer Descriptor Original Buffer Info Word 0 (BD Word 9)**

Bits	Name	Description	Rx Overwrite
31:28	RESERVED		No
27:0	Original Buffer 0 Length	The Buffer Length field indicates the original size of the buffer in bytes. Data bytes are in the buffer. This value will not be overwritten during reception. This value is read by the Rx DMA to determine the actual buffer size as allocated by the host at initialization. Since the buffer length in BD Word 8 is overwritten by the Rx port during reception, this field is necessary to permanently store the buffer size information.	No

**Table 10-30. Host Buffer Descriptor Original Buffer Info Word 1 (BD Word 10)**

Bits	Name	Description	Rx Overwrite
31:0	Original Buffer 0 Pointer LSB	The Buffer Pointer is the byte aligned memory address of the buffer associated with the buffer descriptor. This value will not be overwritten during reception. This value is read by the RX DMA to determine the actual buffer location as allocated by the host at initialization. Since the buffer pointer in BD Words 6/7 is overwritten by the Rx port during reception, this field is necessary to permanently store the buffer pointer information. This field contains the 32 LSBs of the 48-bit original buffer pointer	No

**Table 10-31. Host Buffer Descriptor Original Buffer Info Word 2 (BD Word 11)**

Bits	Name	Description	Rx Overwrite
31:20	RESERVED		Yes
19:16	Original Buffer 0 Pointer Address Space Select	Effectively bits 51:48 of the address – treated special by the infrastructure as an identifier for which address space this particular memory region is located within. Address space 0 is the default unified address space for a given device. Address spaces 1-15 are used for alternate address maps which may be external to the device (PCIe/Hyperlink) or in other 'tiles' on large devices.	Yes
15:0	Original Buffer 0 Pointer MSB	The 16 MSBs of the 48-bit original buffer pointer	No

#### 10.1.2.2.3 Monolithic Packet Descriptor

The Monolithic Packet Descriptor contains the following information:

- Indicator which identifies the descriptor as a Monolithic Packet Descriptor
- Source and Destination Tags
- Packet Type
- Packet Length
- Packet Error Indicator
- Packet Return Information
- Protocol Specific Region Size
- Protocol Specific Region Offset

- Protocol Specific Control/Status Bits
- Packet Data

The maximum size of a Monolithic Packet Descriptor is 4 MB. Of this, Monolithic Packet Descriptors always contain 16 bytes of required information and may also contain 16 bytes of software specific tagging information and up to 128 bytes (indicated in 4 byte increments) of protocol specific information. How much protocol specific information (and therefore the allocated size of the descriptors) is application dependent.

The Monolithic Packet descriptor layout is shown below.

Packet Info (16 bytes)
Extended Packet Info Block (Optional) Includes Timestamp, and software Data Words (16 bytes)
Protocol Specific Data (Optional) (0 to M bytes where M is a multiple of 4)
Null Region (0 to 511 bytes)
Packet Data (0 to 4M – 1)
Other software Data (Optional and User Defined)

The 'Other software Data' portion of the descriptor exists after all of the defined words and is reserved for use by the host software to store private data. This region is not used in any way by the hardware components in a UDMA system and these modules will not modify any bytes within this region.

The contents of the Monolithic Packet Descriptor words are detailed in [Table 10-32](#) through [Table 10-41](#)

**Table 10-32. Monolithic Packet Descriptor Word 0**

Bits	Name	Description	Rx Overwrite
31:30	2'd2	Monolithic Packet Descriptor type.	Yes
29	Extended Packet Info Block Present	This field indicates the presence of the Extended Packet Info Block in the descriptor. 0 = EPIB is not present 1 = 16 byte EPIB is present	Yes
28	RESERVED		Yes
27:22	Protocol Specific Valid Word Count	This field indicates the valid number of 32-bit words in the protocol specific region. This is encoded in increments of 4 bytes as follows: 0 = 0 bytes 1 = 4 bytes ... 16 = 64 bytes ... 32 = 128 bytes 33-63 = RESERVED	Yes
21:0	Packet Length	The length of the packet in bytes. The valid range for the packet length is 0 to 4M-1 bytes. If the packet length is set to 0, the port will not actually transmit any information. Instead, the port will perform buffer/descriptor reclamation as instructed in the return information in word 2.	Yes

**Table 10-33. Monolithic Packet Descriptor Word 1**

Bits	Name	Description	Rx Overwrite
31:28	Error Flags	This field contains error flags that can be assigned based on the packet type	Yes
27:24	Protocol Specific Flags	This field contains protocol specific flags/information that can be assigned based on the packet type.	Yes
23:14	Packet ID	Unique Packet ID for packet within FlowID	Yes

**Table 10-33. Monolithic Packet Descriptor Word 1 (continued)**

Bits	Name	Description	Rx Overwrite
13:0	Flow ID	Flow ID within which this packet is being transported. The FlowID is used by downstream blocks to make decisions about packet steering and resource allocations. FlowIDs are also used to allow specific packets to be received into specific sets of buffers.	Yes

**Table 10-34. Monolithic Packet Descriptor Word 2**

Bits	Name	Description	Rx Overwrite
31:27	Packet Type	This field indicates the type of this packet and is encoded as follows: 0-31 = To Be Assigned	Yes
26:18	Data Offset	This field indicates the byte offset from byte 0 of this descriptor to the location where the valid data begins. On Rx, this value is set equal to the value for the SOP offset given in the Rx DMA channel's Monolithic control register. When a monolithic packet is processed, this value may be modified in order to add or remove bytes to from the beginning of the packet. The value for this field can range from 0-511 bytes. Note that the value of this field must always be greater than or equal to 4 times the value given in the Protocol Specific Valid Word Count field.	Yes
17	Early Return	This field indicates that each buffer pointer should be immediately returned to the specified queue when data transfer is started from the packet instead of waiting for the entire buffer to be emptied. This flag is used to enable in-place reception of packets on a Receive Channel while the source packet is in the process of being transferred on a Transmit Channel. 0 = Buffer/Package descriptor pointers should only be returned after all reads have been completed 1 = Buffer/Package descriptor pointers should be returned immediately upon fetching the descriptor and beginning to transfer data.	No
16	Return Push Policy	This field indicates how a Transmit DMA should return the descriptor pointers to the free queues. This field is encoded as follows: 0 = Descriptor must be returned to tail of queue 1 = Descriptor must be returned to head of queue This bit is only used when the Return Policy bit is set to 1. This field must be set to 0 for descriptors which will be placed on queues managed by the Ring Accelerator.	No
15:0	Packet Return Queue Num	This field indicates the queue number that the descriptor is to be returned to after transmission is complete.	No

**Table 10-35. Monolithic Packet Descriptor Word 3**

Bits	Name	Description	Rx Overwrite
31:24	Source Tag – Hi	This field is application specific. During Packet reception, the DMA controller in the port will overwrite this field as specified in the rx_src_tag_hi_sel field in the flow configuration table entry.	Configurable
23:16	Source Tag – Lo	This field is application specific. During Packet reception, the DMA controller in the port will overwrite this field as specified in the rx_src_tag_lo_sel field in the flow configuration table entry.	Configurable
15:8	Dest Tag – Hi	This field is application specific. During Packet reception, the DMA controller in the port will overwrite this field as specified in the rx_dest_tag_hi_sel field in the flow configuration table entry.	Configurable
15:0	Dest Tag – Lo	This field is application specific. During Packet reception, the DMA controller in the port will overwrite this field as specified in the rx_dest_tag_lo_sel field in the flow configuration table entry.	Configurable

**Table 10-36. Monolithic Extended Packet Info Word 0 (Optional)**

Bits	Name	Description	Rx Overwrite
31:0	Timestamp Info	This field contains an application specific timestamp which can be used for traffic shaping in a QoS enabled system.	Configurable

This word is only present if the Extended Packet Info Block present bit is set in Word 0.

**Table 10-37. Monolithic Extended Packet Info Word 1 (Optional)**

Bits	Name	Description	Rx Overwrite
31:0	Software Info 0	This field stores software centric information that needs to travel with the packet through the stack. This information will be copied from the source descriptor to the destination descriptor whenever a prefetch operation is performed or when transferring through an infrastructure DMA node.	Configurable

This word is only present if the Extended Packet Info Block present bit is set in Word 0.

**Table 10-38. Monolithic Extended Packet Info Word 2 (Optional)**

Bits	Name	Description	Rx Overwrite
31:0	Software Info 1	This field stores software centric information that needs to travel with the packet through the stack. This information will be copied from the source descriptor to the destination descriptor whenever a prefetch operation is performed or when transferring through an infrastructure DMA node.	Configurable

This word is only present if the Extended Packet Info Block present bit is set in Word 0.

**Table 10-39. Monolithic Extended Packet Info Word 3 (Optional)**

Bits	Name	Description	Rx Overwrite
31:0	Software Info 2	This field stores software centric information that needs to travel with the packet through the stack. This information will be copied from the source descriptor to the destination descriptor whenever a prefetch operation is performed or when transferring through an infrastructure DMA node.	Configurable

These words if present immediately follow the Software Data Block Information.

**Table 10-40. Monolithic Packet Descriptor Protocol Specific Word M (Optional)**

Bits	Name	Description	Rx Overwrite
31:0	Protocol Specific Data N	This field stores information which varies depending on the packet type.	Configurable

The payload data follows the protocol specific words at an offset specified in the data offset field of Word 0.

**Table 10-41. Monolithic Packet Descriptor Payload Data Words 0-N**

Bits	Name	Description	Rx Overwrite
31:0	Packet Data N	These words store the packet payload data.	Yes

This field is endian specific. In other words, this is the only field in the descriptor which changes based on the endianness of the system.

#### 10.1.2.2.4 Transfer Request Descriptor

The Transfer Request Descriptor contains the following information:

- Indicator which identifies the descriptor as a TR Descriptor
- Source and Destination Tags
- Packet Error Indicator
- Packet Return Information
- Set of one or more Transfer Request Records
- Set of one or more Transfer Response Records

Transfer Request Packet Descriptors always contain 16 bytes of required information and a variable number of Transfer Request/Transfer Response Records (request and response counts match).

The Transfer Request descriptor layout is shown below.

Packet Info (16 bytes)
Null (0-102 bytes to bring start of TR request array into natural alignment for memory fetch efficiency)
Array of Transfer Request Records (M bytes)
Array of Transfer Response Records (4 bytes per)

**Table 10-42. Transfer Request Packet Descriptor Word 0**

Bits	Name	Description	Rx Overwrite
31:30	2'd3	TR Packet Descriptor type.	Yes
29	RESERVED		Yes
28:20	Reload Count	Specifies what to do when the last entry is processed in this packet. This field specifies how many times to return to the Reload Index upon reaching the Last Entry. When an internal count is incremented to this value and the TR indicated by the Last Entry has been processed, this packet will be considered complete and the descriptor will be placed back on the return queue specified in Word 2.  A value of 0x1FF indicates that a perpetual loop is desired. In this case, the loop count is considered infinite and the internal count will not be incremented. A teardown operation on the channel will cause the loop to be broken at the nearest iteration boundary.	
19:14	Reload Index	Specifies the value to set the current processing index to when the last entry is processed and the Reload Enable is set to 1. This is basically an absolute index to jump to on the 2 <sup>nd</sup> and following passes through the TR packet.	
13:0	Last Entry	Specifies the index of the last valid entry in this packet	Yes

**Table 10-43. Transfer Request Packet Descriptor Word 1**

Bits	Name	Description	Rx Overwrite
31:28	Error Flags	This field contains error flags that can be assigned based on the packet type	Yes
27	RESERVED		Yes



**Table 10-43. Transfer Request Packet Descriptor Word 1 (continued)**

Bits	Name	Description	Rx Overwrite
26:24	Transfer Request Nominal Element Size	Specifies the stride between TR entries. The value in this field must be set large enough that any TR in the buffer will fit within the given dimension. TRs are expected to be placed on boundaries as given in this dimension. This field is also used to calculate the location where the Transfer Responses will be written back. This field is encoded as follows: 0 = 16 byte Transfer Request record size 1 = 32-byte Transfer Request record size 2 = 64-byte Transfer Request record size 3 = 128-byte Transfer Request record size 4-7 = RESERVED	Yes
23:14	Packet ID	Unique Packet ID for packet within FlowID	Yes
13:0	Flow ID	Flow ID within which this packet is being transported. The FlowID is used by downstream blocks to make decisions about packet steering and resource allocations. FlowIDs are also used to allow specific packets to be received into specific sets of buffers.	Yes

**Table 10-44. Transfer Request Packet Descriptor Word 2**

Bits	Name	Description	Rx Overwrite
31:17	RESERVED		No
16	Return Push Policy	This field indicates how a Transmit DMA should return the descriptor pointers to the free queues. This field is encoded as follows: 0 = Descriptor must be returned to tail of queue 1 = Descriptor must be returned to head of queue This bit is only used when the Return Policy bit is set to 1. This field must be set to 0 for descriptors which will be placed on queues managed by the Ring Accelerator	No
15:0	Packet Return Queue Num	This field indicates the queue number that the descriptor is to be returned to after transmission is complete.	No

**Table 10-45. Transfer Request Packet Descriptor Word 3**

Bits	Name	Description	Rx Overwrite
31:24	Source Tag – Hi	This field is application specific. During Packet reception, the DMA controller in the port will overwrite this field as specified in the rx_src_tag_hi_sel field in the flow configuration table entry.	Configurable
23:16	Source Tag – Lo	This field is application specific. During Packet reception, the DMA controller in the port will overwrite this field as specified in the rx_src_tag_lo_sel field in the flow configuration table entry.	Configurable
15:8	Dest Tag – Hi	This field is application specific. During Packet reception, the DMA controller in the port will overwrite this field as specified in the rx_dest_tag_hi_sel field in the flow configuration table entry.	Configurable



**Table 10-45. Transfer Request Packet Descriptor Word 3 (continued)**

Bits	Name	Description	Rx Overwrite
15:0	Dest Tag – Lo	This field is application specific. During Packet reception, the DMA controller in the port will overwrite this field as specified in the rx_dest_tag_lo_sel field in the flow configuration table entry.	Configurable

**Table 10-46. Transfer Request Packet Descriptor Payload Words 0-N**

Bits	Name	Description	Rx Overwrite
31:0	Transfer Control Record Array	Transfer Control Records. This is an array of records which encapsulate the Transfer Request and Transfer Response messages which are used by the UTC	Yes (partial)

### 10.1.2.3 Transfer Request Record

#### 10.1.2.3.1 Overview

This section describes the standard transfer request (TR) format to initiate a DMA transfer. The TRs can support both half and full duplex with multiple dimensions as well as cache wars. The UTC (Universal Transfer Controller) does not have to support all types of TRs. Each TR will also generate a TR response. A TR can be sent to the UTC either through a PSI-L interface or an optionally supported Direct TR submission (previously QDMA) which will have Memory Mapped TR submission registers that can be written. The Memory Mapped TR submission registers further be classified by submission type either a single burst (atomic write) or multiple burst (non-atomic). The non-atomic TR will be sent when the submission register with word 0 is written.

**Table 10-47. Terms and Definitions**

Term	Definition
TR	A transfer request to move data.
CR	A cache operation request. A request to send messages to a specified cache controller to prepare the cache for a future operation.
UTC	Universal Transfer Controller the module in the UDMA architecture that handled the actual movement of data requested in the TR.

#### 10.1.2.3.2 Addressing Algorithm

For a basic TR request the same algorithm is used. The innermost loop always consumes physically contiguous elements from memory. Its implicit dimension is 1 byte, this can be changed if the TR contains the formatting flags field and it is supported by the UTC. The pointer itself moves from fetch to fetch on the maximum byte alignments specified for the UTC, in increasing order. In each level outside the inner loop, the loop moves the pointer to a new location based on the size of that loop level's dimension.

##### 10.1.2.3.2.1 Linear Addressing (Forward)

The following code illustrates the basic algorithm for a 4-level loop nest block move. This model assumes that it transfers data 1 byte at a time and the input and output block are the same size in all dimensions.

```
// sptr is a byte pointer.
// dptr is a byte pointer.
// Source address is indirect
if (TR_ISA) {
    sptr = *sptr;
}
// Destination address is indirect
if (TR_IDA) {
    dptr = *dptr;
}
// TR_TRIGX can be selected to come from Global events, Local event, or none.
// Check for trigger of TYPE0
if (TR_TRIG0_TYPE == TYPE0) while(!TR_TRIG0);
if (TR_TRIG1_TYPE == TYPE0) while(!TR_TRIG1);
```

```

for (i3 = 0; i3 < ICNT3; i3++)
{
    sptr3 = sptr; // save current position before entering next level
    dptr3 = dptr; // save current position before entering next level
    // Check for trigger of TYPE1
    if (TR_TRIG0_TYPE == TYPE1) while(!TR_TRIG0);
    if (TR_TRIG1_TYPE == TYPE1) while(!TR_TRIG1);
    for (i2 = 0; i2 < ICNT2; i2++) {
        sptr2 = sptr; // save current position before entering next level
        dptr2 = dptr; // save current position before entering next level
        // Check for trigger of TYPE2
        if (TR_TRIG0_TYPE == TYPE2) while(!TR_TRIG0);
        if (TR_TRIG1_TYPE == TYPE2) while(!TR_TRIG1);
        for (i1 = 0; i1 < ICNT1; i1++) {
            sptr1 = sptr; // save current position before entering next level
            dptr1 = dptr; // save current position before entering next level
            // Check for trigger of TYPE3
            if (TR_TRIG0_TYPE == TYPE3) while(!TR_TRIG0);
            if (TR_TRIG1_TYPE == TYPE3) while(!TR_TRIG1);
            for (i0 = 0; i0 < ICNT0; i0++) {
                // UTC can combine these in optimized burst aligned accesses.
                *dptr = *sptr;
                sptr = sptr++;
                dptr = dptr++;
            }
            // Update based on saved pointer for this level
            sptr = sptr1 + SDIM1;
            dptr = dptr1 + DDIM1;
        }
        // Update based on saved pointer for this level
        sptr = sptr2 + SDIM2;
        dptr = dptr1 + DDIM2;
    }
    // Update based on saved pointer for this level
    sptr = sptr3 + SDIM3;
    dptr = dptr1 + DDIM3;
}

```

This form of addressing allows programs to specify regular paths through memory in a small number of parameters. Additionally, it allows for various stall points through the loop to allow for hardware or software induced pausing of the transfer. The following table defines these parameters more explicitly.

**Table 10-48. Addressing Parameters for a Basic Stream**

Parameter	Definition
ICNT0	Number of iterations for the innermost loop dimension, loop level 0. That is, DIM0 = BYTES.
ICNT1	Number of iterations for the first level above the innermost loop, loop level 1.
SDIM1	Number of bytes between the starting points for consecutive iterations of loop level 1 for the source.
DDIM1	Number of bytes between the starting points for consecutive iterations of loop level 1 for the destination.
ICNT2	Number of iterations for loop level 2.
SDIM2	Number of bytes between the starting points for consecutive iterations of loop level 2 for the source.
DDIM2	Number of bytes between the starting points for consecutive iterations of loop level 2 for the destination.
ICNT3	Number of iterations for loop level 3.
SDIM3	Number of bytes between starting points for consecutive iterations of loop level 3 for the source.
DDIM3	Number of bytes between the starting points for consecutive iterations of loop level 3 for the destination.

### 10.1.2.3.3 Transfer Request Formats

The Transfer Request format builds on the same structure adding fields to the TR to give more configuration options. The structure in [Table 10-49](#) shows the 512-bit data structure that makes up the maximum sized UTC Transfer request.

**Table 10-49. Transfer Request Template**

word 15		word 14		word 13		word 12	
DICNT3	DICNT2	DICNT1	DICNT0	DDIM3		DDIM2	
word 11		word 10		word 9		word 8	
DADDR				DDIM1		FMTFLAGS	
word 7		word 6		word 5		word 4	
DIM3		DIM2		ICNT3	ICNT2	DIM1	
word3		word 2		word 1		word0	
ADDR				ICNT1	ICNT0	FLAGS	

The UTC defines a four-level loop nest for addressing elements within the TR, using the algorithms described in the Addressing algorithm. Most of the fields in the TR template map directly to the parameters in those algorithms.

**Table 10-50. Transfer Request Fields**

Field Name	Description	Size (bits)
FLAGS	TR flags that specify type of TR and how the TR should be handled. It also supports the TR triggering and output events.	32
ICNT0	Total loop iteration count for level 0 (innermost)	16
ICNT1	Total loop iteration count for level 1	16
ADDR	Starting address for the source data or destination data if it is a half-duplex write	64
DIM1	Signed dimension for loop level 1 for the source data	32
ICNT2	Total loop iteration count for level 2	16
ICNT3	Total loop iteration count for level 3 (outermost)	16
DIM2	Signed dimension for loop level 2	32
DIM3	Signed dimension for loop level 3	32
FMTFLAGS	Flags that tell how the data is formatted either between the input and the output or if the data should use different addressing schemes or sizes. These flags are OPTIONAL and only specific features may be supported by a given DMA implementation. The UDMA/UTC capabilities register specifies which features are supported.	32
DDIM1	Signed dimension for loop level 1 for the destination data	32
DADDR	Starting address for the destination of the data	64
DDIM2	Signed dimension for loop level 2 for the destination data	32
DDIM3	Signed dimension for loop level 3 for the destination data	32
DICNT0	Total loop iteration count for level 0 (innermost) used for destination	16
DICNT1	Total loop iteration count for level 1 used for destination	16
DICNT2	Total loop iteration count for level 2 used for destination	16
DICNT3	Total loop iteration count for level 3 used for destination	16

The TR assumes all iteration counts as unsigned integers, and all dimensions as signed integers representing byte offsets.

The template above fully specifies the type of elements, length, and dimensions of the transfer.

Since all transfers will not require all the fields the TRs have the type field which allow the number of words required to be sent for the TR to be reduced.

#### 10.1.2.3.4 Flags Field Definition

[Table 10-51](#) expands the 32-bit FLAGS field. The numbers above each field here indicate bit numbers within the field.

**Table 10-51. Transfer Request Template FLAGS Field**

31								24								23								16																																							
CONFIGURATION SPECIFIC FLAGS																CMD ID																																															
15				14				13				12				11				10				9				8				7				6				5				4				3				0											
TRIGGER1_TY PE								TRIGGER1								TRIGGER0_TY PE								TRIGGER0								EVENT_SIZE								WAIT								STATI C								TYPE							

The flags field fills in the remaining details about the stream, as follows:

**Table 10-52. FLAGS Field Descriptions**

Bit	Field	Description
0-3	TYPE	The TYPE of TR that is being sent 0: One dimensional data move 1: Two dimensional data move 2: Three dimensional data move 3: Four dimensional data move 4: Four dimensional data move with data formatting 5: Four dimensional Cache Warm 6-7: Reserved 8: Four Dimensional Block Move 9: Four Dimensional Block Move with Repacking 10: Two Dimensional Block Move 11: Two Dimensional Block Move with Repacking 12-14: Reserved 15: Four Dimensional Block Move with Repacking and Indirection
4	STATIC	This field is only valid on Type 8 and type 9 TRs The TR is static and will continually reload into the channel upon completion of the TR until a channel tear down.
5	WAIT	This field is only valid on Type 15 TRs. This field indicates whether or not this TR should ensure it completes before allowing the next TR to start on this channel. The encoding is as follows: 0 = Allow next TR to start immediately 1 = Wait for this TR to complete before allowing next TR to start When set, the next TR will not be considered triggered (regardless of any other trigger conditions) until all write status responses for the current TR have landed.

**Table 10-52. FLAGS Field Descriptions (continued)**

Bit	Field	Description
6-7	EVENT_SIZE	This is how often the TR will generate an output event. 0: Event is only generated with the TR is complete 1: Event is generated when the second inner most loop (ICNT1) is decremented by 1. 2: Event is generated when the third inner most loop (ICNT2) is decremented by 1. 3: Event is generated when the outer most loop (ICNT3) is decremented by 1.
8-9	TRIGGER0	This is one of two selectable triggers. The receipt of this trigger will enable the TR to be active for enough data transfer as specified by the TRIGGER0_TYPE Field. 0: No Trigger 1: Global Trigger 0 for the channel 2: Global Trigger 1 for the channel 3: Local Event for the channel
10-11	TRIGGER0_TYPE	This is the type of data transfer that will be enabled by receiving a trigger. 0: The second inner most loop (ICNT1) will be decremented by 1. 1: The third inner most loop (ICNT2) will be decremented by 1. 2: The outer most loop (ICNT3) will be decremented by 1. 3: The entire TR will be allowed to complete.
12-13	TRIGGER1	This is one of two selectable triggers. The receipt of this trigger will enable the TR to be active for enough data transfer as specified by the TRIGGER1_TYPE Field. 0: No Trigger 1: Global Trigger 0 for the channel 2: Global Trigger 1 for the channel 3: Local Event for the channel This field is reserved for a Direct TR.
14-15	TRIGGER1_TYPE	This is the type of data transfer that will be enabled by receiving a trigger. 0: The second inner most loop (ICNT1) will be decremented by 1. 1: The third inner most loop (ICNT2) will be decremented by 1. 2: The outer most loop (ICNT3) will be decremented by 1. 3: The entire TR will be allowed to complete.
16-23	CMD ID	The Command ID for the TR. This will be sent back with the response for the Channel controller to identify the specific TR. For a Direct TR this is the event number that will be generated based on the event trigger.
24-31	Configuration Specific Flags	These are flag bits that can be specified by the specific UTC configuration. These bits will remain undefined in the global TR format to allow for customization requirements that might be required in the specific UTC implementation.

The following sections expand on each of these fields.

#### 10.1.2.3.4.1 Type: TR Type Field

The TR Type field gives the size of the TR and which fields are expected in the TR. Based on the type the number of words required to be sent to complete the TR can be decreased. The tables below show the minimum size for each TR as well if any fields will be treated as reserved.

**Table 10-53. Transfer Request Minimum Size Type 0 One Dimensional Transfer**

w 0	w 1	w 2	w 3	w 4	w 5	w 6	w 7	w 8	w 9	w 10	w 11	w 12	w 13	w 14	w 15
FLAGS	ICNT0	ADDR		RESERVED/NOT REQUIRED											

**Table 10-54. Transfer Request Minimum Size Type 1 Two Dimensional Transfer**

w 0	w 1	w 2	w 3	w 4	w 5	w 6	w 7	w 8	w 9	w 10	w 11	w 12	w 13	w 14	w 15
FLAGS	ICNT0/ 1	ADDR		DIM1	RESERVED/NOT REQUIRED										

**Table 10-55. Transfer Request Minimum Size Type 2 Three Dimensional Transfer**

w 0	w 1	w 2	w 3	w 4	w 5	w 6	w 7	w 8	w 9	w 10	w 11	w 12	w 13	w 14	w 15
FLAGS	ICNT0/ 1	ADDR		DIM1	ICNT2	DIM2	RESERVED/NOT REQUIRED								

**Table 10-56. Transfer Request Minimum Size Type 3 Four Dimensional Transfer**

w 0	w 1	w 2	w 3	w 4	w 5	w 6	w 7	w 8	w 9	w 10	w 11	w 12	w 13	w 14	w 15
FLAGS	ICNT0/ 1	ADDR		DIM1	ICNT2/ 3	DIM2	DIM3	RESERVED/NOT REQUIRED							

**Table 10-57. Transfer Request Minimum Size Type 4 Four Dimensional Transfer with Formatting**

w 0	w 1	w 2	w 3	w 4	w 5	w 6	w 7	w 8	w 9	w 10	w 11	w 12	w 13	w 14	w 15
FLAGS	ICNT0/ 1	ADDR		DIM1	ICNT2/ 3	DIM2	DIM3	FMT FLAGS	RESERVED/NOT REQUIRED						

**Table 10-58. Transfer Request Minimum Size Type 5 Cache Warm**

w 0	w 1	w 2	w 3	w 4	w 5	w 6	w 7	w 8	w 9	w 10	w 11	w 12	w 13	w 14	w 15
FLAGS	ICNT0/ 1	ADDR		DIM1	ICNT2/ 3	DIM2	DIM3	CACH E FLAGS	RESERVED/NOT REQUIRED						

**Table 10-59. Transfer Request Minimum Size Type 8 Four Dimensional Block Copy**

w 0	w 1	w 2	w 3	w 4	w 5	w 6	w 7	w 8	w 9	w 10	w 11	w 12	w 13	w 14	w 15
FLAGS	ICNT0/ 1	ADDR		DIM1	ICNT2/ 3	DIM2	DIM3	FMT FLAGS	DDIM1	DADDR		DDIM2	DDIM3	RESERVED/NOT REQUIRED	

**Table 10-60. Transfer Request Minimum Size Type 9 Four Dimensional Block Copy with Repacking**

w 0	w 1	w 2	w 3	w 4	w 5	w 6	w 7	w 8	w 9	w 10	w 11	w 12	w 13	w 14	w 15
FLAGS	ICNT0/ 1	ADDR		DIM1	ICNT2/ 3	DIM2	DIM3	FMT FLAGS	DDIM1	DADDR		DDIM2	DDIM3	DICNT 0/1	DICNT 2/3

**Table 10-61. Transfer Request Minimum Size Type 10 Two Dimensional Block Copy**

w 0	w 1	w 2	w 3	w 4	w 5	w 6	w 7	w 8	w 9	w 10	w 11	w 12	w 13	w 14	w 15
FLAGS	ICNT0/ 1	ADDR		DIM1	RESERVED			FMT FLAGS	DDIM1	DADDR		RESERVED/NOT REQUIRED			

**Table 10-62. Transfer Request Minimum Size Type 11 Two Dimensional Block Copy with Repacking**

w 0	w 1	w 2	w 3	w 4	w 5	w 6	w 7	w 8	w 9	w 10	w 11	w 12	w 13	w 14	w 15
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------

**Table 10-62. Transfer Request Minimum Size Type 11 Two Dimensional Block Copy with Repacking**  
(continued)

FLAGS	ICNT0/ 1	ADDR	DIM1	RESERVED	FMT FLAGS	DDIM1	DADDR	RESERVED	DICNT 0/1	RESE RVED/ NOT REQUI RED
-------	-------------	------	------	----------	--------------	-------	-------	----------	--------------	--------------------------------------

**Table 10-63. Transfer Request Minimum Size Type 15 Four Dimensional Block Copy with Repacking and Indirection Support**

w 0	w 1	w 2	w 3	w 4	w 5	w 6	w 7	w 8	w 9	w 10	w 11	w 12	w 13	w 14	w 15
FLAGS	ICNT0/ 1	ADDR	DIM1	ICNT2/ 3	DIM2	DIM3	RESE RVED	DDIM1	DADD R	DDIM2	DDIM3	DICNT 0/1	DICNT2/3		

#### 10.1.2.3.4.2 STATIC: Static Field Definition

The static field is used to allow for a single TR to be reused repeatedly. If the static bit is set then the UTC upon completing the specified TR will restart the TR with the same values as were in the initial TR. This TR will stay active until the channel is torn down. The initiating of a tear down of the channel will clear the STATIC field in the active TR and it will be removed from the channel FIFO after the TR completion.

#### Note

If a TR is submitted to a channel FIFO with the STATIC field set then any TRs placed in the FIFO after that will NOT be run until the teardown has been initiated.

#### 10.1.2.3.4.3 EVENT\_SIZE: Event Generation Definition

The UTC allows for an event to be generated at specified intervals for each TR. As the TR passes through the various loops in the addressing algorithm it will generate an event that then can be routed to other event receivers, such as other channels or interrupts to processors.

**Table 10-64. Event Size Encoding**

Value	Event Type	When it Fires
0	Completion Event	When the TR is complete and all status for the TR has been received.
1	ICNT1 Decrement	Type 0: When the last data transaction is sent for the TR. Type 1-11: When ICNT1 is decremented
2	ICNT2 Decrement	Type 0 – 1,10-11: When the last transaction is sent for the TR. All Other Types: When ICNT2 is decremented
3	ICNT3 Decrement	Type 0 – 2,10-11: When the last transaction is sent for the TR. All Other Types: When ICNT3 is decremented

#### 10.1.2.3.4.4 TRIGGER INFO: TR Triggers

The TR allows for two different triggers to be set to allow for the data transfer to be prevented or halted through the process of transferring the data. The triggers are specified by selecting a type of trigger and the size of the transfer that can be allowed for the receipt of a given trigger. The triggers themselves in the UTC are collected on a per channel basis and for each of three sources. Anytime the trigger event is received the internal counter is incremented. The trigger event will not be cleared until the specified block has started its transfer and it will **only** clear the triggers that are active.

### CAUTION

This does allow for one TR in the channel to use global event 0 and the next TR to use global event 1. If while the first TR is running global event 0 can increment and will decrement each time the TR reaches the specified level. At the same time global event 1 will be incremented whenever it is received but it will not decrement until the next TR runs. If the global event 1 counter reaches overflows then an error event will NOT be generated but the event will be lost.

#### 10.1.2.3.4.5 TRIGGERX\_TYPE: Trigger Type

The trigger type sets the value of TR\_TRIGX as shown in the addressing algorithm description. The TR\_TRIGX value allows for the temporary halting of the TR until the expected event has been received. After the given loop has been entered the selected trigger will be decremented.

#### 10.1.2.3.4.6 TRIGGERX: Trigger Selection

The TR is able to select up to 2 of 3 trigger sources for a given TR. The events can come from the dedicated local event on the UTC itself or from two assigned global events that come from the PSI-L interface on the UTC. The trigger counters are always enabled so that events can be received prior to the TR getting loaded. The trigger counters are only decremented when the trigger is active and the TR has scheduled the first transfer of the transaction.

#### 10.1.2.3.4.7 CMD ID: Command ID Field Definition

The command ID is used as a unique identifier for the TR. The value is not used during the data transfer but is only used by the TR Completion Response to allow for hardware and software to identify the specific TR.

#### 10.1.2.3.4.8 Configuration Specific Flags Definition

The configuration specific flags are specific to a given TR type. If a TR type does not require additional flags the field should be filled with 0's. Currently only Types 0-4 and Type 15 support the following configuration specific flags:

**Table 10-65. Configuration Specific Flags**

Bit	Field	Description
0	ISA	Indirect Source Address 0: Source address in TR is a directly usable pointer to the data 1: Source address in TR is a pointer to a 64-bit location that contains the actual pointer to the data
1	IDA	Indirect Destination Address 0: Destination address in TR is a directly usable pointer to the data 1: Destination address in TR is a pointer to a 64-bit location that contains the actual pointer to the data
2	SUPR_EVT	Suppress Event Output: 0 = Output events will be generated according to FLAGS.EVENT_SIZE field 1 = No output events will be generated for the duration of this TR execution This field is only valid on split and type 15 TRs (Type 0-3,15)
3	Reserved	Reserved for Future use.



**Table 10-65. Configuration Specific Flags (continued)**

Bit	Field	Description
6:4	EOL	<p>This field is only valid on split TRs (Type 0-3).</p> <p>On source (Read) split TRs, this field specifies whether or not the EOL delimiters should be produced on the Tx PSI-L bus. The encodings of this field for source split TRs is as follows:</p> <ul style="list-style-type: none"> <li>0 = SOL/EOL match SOP/EOP</li> <li>1 = SOL/EOL boundaries are each ICNT0 bytes</li> <li>2 = SOL/EOL boundaries are each ICNT0×ICNT1 bytes</li> <li>3 = SOL/EOL boundaries are each ICNT0×ICNT1×ICNT2 bytes</li> <li>4 = SOL/EOL boundaries are each ICNT0×ICNT1×ICNT2×ICNT3 bytes</li> </ul> <p>On destination (Write) split TRs, this field specifies how to handle EOL delimiters when they are encountered on the Rx (write) side of a TR. EOL delimiters can be produced from the Tx (read) side of a block copy operation or from a remotely paired peripheral when operating in split TR mode. The encodings of this field are as follows:</p> <ul style="list-style-type: none"> <li>0: Ignore EOL</li> <li>1: Line length is icnt0 bytes. Clear any remaining ICNT0 bytes and increment ICNT1 by 1</li> <li>2: Line length is icnt0×icnt1 bytes. Clear any remaining ICNT0/1 bytes and increment ICNT2 by 1</li> <li>3: Line length is icnt0×icnt1×icnt2 bytes. Clear any remaining ICNT0/1/2 bytes and increment ICNT3 by 1</li> <li>4: Line length is ICNT0×ICNT1×ICNT2×ICNT3 bytes. Move on to next TR</li> <li>5-7: RESERVED</li> </ul>
7	EOP	<p>This TR should generate an EOP on the streaming interface for any downstream packet oriented consumers to denote that a 'packet' of data is complete</p> <ul style="list-style-type: none"> <li>0: No EOP flag will accompany the last PSI-L data phase associated with transfers from this TR</li> <li>1: On egress the DMA will set the EOP flag coincident with transferring the last of the data for this TR. If the TR data does not complete an entire PSI-L data phase then the remaining bytes in the data phase will be skipped and the internal FIFO pointer will be updated to begin packing new data on a new data phase boundary.</li> </ul>

#### 10.1.2.3.5 TR Address and Size Attributes

The fields described below all describe the data transfer that needs to be made as described in the Linear Memory Addressing Block Move Algorithm.

##### 10.1.2.3.5.1 ICNT0

The ICNT0 is the number of elements to transfer in the inner loop of the TR. If the TR type does not contain a FMTFLAGS field then this count is assumed to be the number of bytes to transfer. If the FMTFLAGS field is included in the types then the number of bytes to be transferred will be ICNT0 times the Element size rounded up to the nearest byte.

##### 10.1.2.3.5.2 ICNT1

The ICNT1 field is the loop count for the second innermost loop count as defined in the Addressing algorithm.

##### 10.1.2.3.5.3 ADDR

The address location is the initial address that will be accessed at the start of the transfer. All of the dimensions will also be based off of this value. This address can either physical or virtual based upon settings in the DMA channel Configuration register.

While the ADDR field is 64 bits wide, the usable extent of the address on a K3 system is 48 bits of absolute offset plus a 4-bit address space selector which indicates 1 of 16 different orthogonal address spaces that the pointer is referencing within. The format of the ADDR field is given in [Table 10-66](#).

**Table 10-66. ADDR Field Format**

Bits	Subfield	Description
63:52	Reserved	Reserved
51:48	Address Space Select	Effectively bits 51:48 of the address. The value given in this field will be output by the DMA masters on the casel pin which is used by the infrastructure as an identifier for which address space this particular memory region is located within. Address space 0 is the default unified address space for a given device. Address spaces 1-15 are used for alternate address maps which may be external to the device (PCIe/Hyperlink) or in other 'tiles' on large devices.
47:0	Address	The 48-bit source or source/destination starting address for the transfer. This address will be interpreted to be either physical, virtual, or intermediate physical based on the tx_atype or rx_atype field in the DMA Tx/Rx channel's Configuration Register.

#### 10.1.2.3.5.4 DIM1

This is the offset of the address from the initial address for the first access of the second loop. This will be added to the address from the loop before. This is a signed value so that the address can be both above and below the initial address.

#### 10.1.2.3.5.5 ICNT2

This is the count for the third innermost loop in the addressing algorithm.

#### 10.1.2.3.5.6 ICNT3

This is the count for the outermost loop in the addressing.

#### 10.1.2.3.5.7 DIM2

This is the offset of the address from the initial address to the next address in the third innermost loop. This is a signed vales so that the address can be both above and below the previous address.

#### 10.1.2.3.5.8 DIM3

This is the offset of the address from the initial address to the next address in the outermost loop. This is a signed values so that the address can be both above and below the previous address.

#### 10.1.2.3.5.9 DDIM1

This is the offset of the destination address from the initial destination address for the first access of the second loop. This will be added to the destination address from the loop before. This is a signed value so that the destination address can be both above and below the initial destination address.

#### 10.1.2.3.5.10 DADDR

The destination address location is the initial destination address that will be accessed at the start of the transfer. All of the dimensions will also be based off of this value. This address can either physical or virtual based upon settings in the DMA channel Configuration register

While the DADDR field is 64 bits wide, the usable extent of the address on a K3 system is 48 bits of absolute offset plus a 4 bit address space selector which indicates 1 of 16 different orthogonal address spaces that the pointer is referencing within. The format of the DADDR field is given as follows:

**Table 10-67. DADDR Field Format**

Bits	Subfield	Description
63:52	Reserved	Reserved
51:48	Address Space Select	Effectively bits 51:48 of the address. The value given in this field will be output by the DMA masters on the casel pin which is used by the infrastructure as an identifier for which address space this particular memory region is located within. Address space 0 is the default unified address space for a given device. Address spaces 1-15 are used for alternate address maps which may be external to the device (PCIe/Hyperlink) or in other 'tiles' on large devices.

**Table 10-67. DADDR Field Format (continued)**

Bits	Subfield	Description
47:0	Address	The 48-bit starting destination address for the transfer. This address will be interpreted to be either physical, virtual, or intermediate physical based on the tx_atype field in the DMA Tx channel's Configuration Register.

#### 10.1.2.3.5.11 DDIM2

This is the offset of the destination address from the initial destination address for the first access of the third loop. This will be added to the destination address from the loop before. This is a signed value so that the destination address can be both above and below the initial destination address.

#### 10.1.2.3.5.12 DDIM3

This is the offset of the destination address from the initial destination address for the first access of the outer loop. This will be added to the destination address from the loop before. This is a signed value so that the destination address can be both above and below the initial destination address.

#### 10.1.2.3.5.13 DICNT0

This is the number of elements to use in the destination if the TR is repacking the data. This number reflects the number of elements to be transferred.

#### 10.1.2.3.5.14 DICNT1

This is the number of times to execute the second loop to use in the destination transfer.

#### 10.1.2.3.5.15 DICNT2

This is the number of times to execute the third loop to use in the destination transfer.

#### 10.1.2.3.5.16 DICNT3

This is the number of times to execute the fourth loop to use in the destination transfer. The value of DICNT0 × DICNT1 × DICNT2 × DICNT3 must equal ICNT0 × ICNT1 × ICNT2 × ICNT3. If any of the values are zero they are NOT included in the multiplication.

#### 10.1.2.3.6 FMTFLAGS

The format flags are used to specify special formatting of the transfer. [Table 10-68](#) shows the definition of the sub-fields in the FMTFLAGS field.

**Table 10-68. Transfer Request Template FMTFLAGS Field**

31										16
AMODE SPECIFIC										
15	14	13	12	11	8	7	4	3	2	0
RESERVED		SECTR		DFMT		ELTYPE		DIR		AMODE

The flags field fills in the remaining details about the stream, as follows:

**Table 10-69. FMTFLAGS Field Descriptions**

Bit	Field	Description
16-31	AMODE SPECIFIC	This 16-bit field is defined based on the addressing mode used as selected in the AMODE field.
14-15	Reserved	Reserved for Future use
12-13	SECTR	A secondary TR is required and the format of the secondary TR.
8-11	DFMT	The data reformatting function to apply in the middle of the TR
4-7	ELTYPE	The type of element and its size and potential output size for the innermost loop. If this is not specified in the TR then a byte is the assumed element.

**Table 10-69. FMTFLAGS Field Descriptions (continued)**

Bit	Field	Description
3	DIR	For a block type transfer does the addressing mode apply for source or destination. The other one will use linear. If it is a non-block type TR then this field is ignored
0-2	AMODE	The Addressing Mode of TR that is being sent 0: Linear Addressing 1: Circular Addressing 2-7: Reserved for future use

#### 10.1.2.3.6.1 AMODE: Addressing Mode Definition

The addressing mode field allows for different algorithms to be used other than the default linear addressing mode described in the Addressing algorithm.

##### 10.1.2.3.6.1.1 Linear Addressing

This is the addressing that is used if there is no FMTFLAGS field in the TR.

##### 10.1.2.3.6.1.2 Circular Addressing

Circular addressing modifies how the UTC performs address arithmetic so that addresses remain within a power-of-2 sized window. Circular addressing works by holding upper address bits constant during an address update, while allowing the lower address bits to vary. This contrasts with the default—linear addressing—which allows all of the address bits to vary.

The UTC TR provides address mode selection for each level of loop nest. Each level can select between linear addressing or circular addressing with one of two circular block sizes.

Conceptually, selectable addressing mode support replaces all address arithmetic with a function similar to the following.

```
enum addr_mode_t
{
    LINEAR, // Linear addressing mode
    CIRC0, // Circular addressing with block size 0
    CIRC1, // Circular addressing with block size 1
};
uint64_t circ_mask_0; // Circular addressing mask for block size 0
uint64_t circ_mask_1; // Circular addressing mask for block size 1
uint64_t address_add( uint64_t base, int32_t offset, addr_mode_t mode )
{
    uint64_t new_addr = base + offset;
    if ( mode == LINEAR )
        return new_addr;
    // Look up address mask based on selected circular buffer size.
    // Mask contains 1s for bits that remain fixed, and 0s elsewhere.
    uint64_t addr_mask = mode == CIRC0 ? circ_mask_0 : circ_mask_1;
    return ( base & addr_mask ) | ( new_addr & ~addr_mask );
}
```

The address\_add primitive then takes the place of plain addition for all TR address computations in this mode. The following example illustrates address\_add.

```
for (i3 = 0; i3 < ICNT3; i3++)
{
    ptr3 = ptr; // save current position before entering next level
    for (i2 = 0; i2 < ICNT2; i2++)
    {
        ptr2 = ptr; // save current position before entering next level
        for (i1 = 0; i1 < ICNT1; i1++)
        {
            ptr1 = ptr; // save current position before entering next level
            for (i0 = 0; i0 < ICNT0; i0++)
            {
```

```

fetch( ptr, ELEM_BYTES );
ptr = address_add( ptr, ELEM_BYTES, addr_mode_0 );
}
// Update based on saved pointer for this level
ptr = address_add( ptr1, DIM1, addr_mode_1 );
}
// Update based on saved pointer for this level
ptr = address_add( ptr2, DIM2, addr_mode_2 );
}
// Update based on saved pointer for this level
ptr = address_add( ptr3, DIM3, addr_mode_3 );
}

```

The values for the circular addressing can be selected by using the 16 bits of the AMODE specific Field as described in the AMODE SPECIFIC Addressing Mode Field.

#### 10.1.2.3.6.2 DIR: Addressing Mode Direction Definition

This field is used if the TR TYPE specifies the TR is for a block move if the addressing mode specified in the AMODE field applies to the source addressing or the destination addressing. A value of 0 means the source addressing will use the method selected in the AMODE field and the destination address will use the default linear addressing. A value of 1 means that the destination addressing will use the method selected in the AMODE field and the source addressing will use the default linear addressing. If the TR TYPE is not a block transfer then this field will be ignored.

#### 10.1.2.3.6.3 ELTYPE: Element Type Definition

The Element Type field allows the user to specify the basic unit that is used for the innermost loop. If the FMTFLAGS field is not present then the UTC will assume a value of 0 which is a byte size element. The element type also allows for the some expansion or contraction of the element size from the source to the destination.

**Table 10-70. ELTYPE Encoding**

Encoding	Definition
0	1 Byte per element
1	1.5 Bytes(12 bits) per element
2	2 Bytes per element
3	3 Bytes per element
4	4 Bytes per element
5	8 Bytes per element
6	16 Bytes per element
7	32 Bytes per Element
8	1 Byte per Input Element 2 Bytes per Output Element
9	1.5 Bytes per Input Element 2 Bytes per Output Element
10	2 Bytes per Input Element 1 Byte per Output Element
11	2 Bytes per Input Element 1.5 Bytes per Output Element
12-15	Reserved for Future Use

#### 10.1.2.3.6.4 DFMT: Data Formatting Algorithm Definition

The Data Formatting Algorithm is a method to specify a manipulation of the data between how it is read and how it is sent. If the TR Type is not a block operation TR then this field will be ignored. Otherwise [Table 10-71](#) specifies what the encodings meaning.

**Table 10-71. DFMT Encoding**

Encoding	Definition
0	No Change. The input and output block will remain identical
1	Constant Copy. The input block is not an address but the address is up to a 64-bit constant

**Table 10-71. DFMT Encoding (continued)**

Encoding	Definition
2	Transpose. The inner and second most inner loops are swapped so that rows become columns and columns become rows.
3	Reverse. The data in the row will be accessed in the reverse of the order that it is read. So the largest address read in a row will be the first address written.
4	Reverse Transpose. The data will be written in the reverse of the order that is read as well as transposed. So the largest address read in a row will be the address in the first line being written out.
5-15	Reserved for Future Use

#### 10.1.2.3.6.5 SECTR: Secondary Transfer Request Definition

The Secondary Transfer Request field allows the user to specify if the TR requires an additional TR located in memory. If this is used then the address given in the TR is the pointer to the Secondary TR and the secondary TR will have a pointer to the actual final TR.

**Table 10-72. Secondary Transfer Request Format**

Encoding	Size	Definition
0	0	The TR does not require a secondary TR
1	64 bytes	The TR will fetch a 64-byte Secondary TR prior to the initial read.
2	128 bytes	The TR will fetch a 128-byte Secondary TR prior to the initial read.
3	TBD	Reserved for future use

#### 10.1.2.3.6.5.1 Secondary TR Formats

The secondary TR format is a TR that is fetched by the UTC when it is executing a TR to fill in additional information that does not fit in the original TR. It is not included in the standard TR as the information required in it is not required until execution time and is kept separate so the Channel FIFO space can be minimized. The first four words of the Secondary TR are always the real start address of the transfer followed by a Secondary TR Flags field. The remaining words in the secondary TR are defined by the Secondary TR TYPE from the Secondary TR Flags field.

**Table 10-73. Secondary Transfer Request Template for 64-Byte Secondary TR**

word 15	word 14	word 13	word 12
Type Specific			
word 11	word 10	word 9	word 8
Type Specific			
word 7	word 6	word 5	word 4
Type Specific			
word 3	word 2	word 1	word 0
Type Specific	SECONDARY TR FLAGS		ADDR

#### 10.1.2.3.6.5.2 Secondary TR FLAGS

The next diagram expands the generic 32-bit Secondary TR FLAGS field. The numbers above each field here indicate bit numbers within the field.

**Table 10-74. SECONDARY TR FLAGS Field Definition**

Table 16.14. SECONDARY TR TYPE Specific Field Definition				
31				16
SEC_TR_TYPE_SPECIFIC				
15	4	3	0	
SEC TR TYPE SPECIFIC		SEC TR TYPE		

The flags field fills in the remaining details about the stream, as follows:

**Table 10-75. FLAGS Field Descriptions**

Bit	Field	Description
31-4	SEC_TR_TYPE_SPECIFIC	Reserved for the SEC_TR_TYPE to define based on its requirements
3-0	SEC_TR_TYPE	The TYPE of SEC_TR 0: Multiple Buffer Interleave 1-15: Reserved

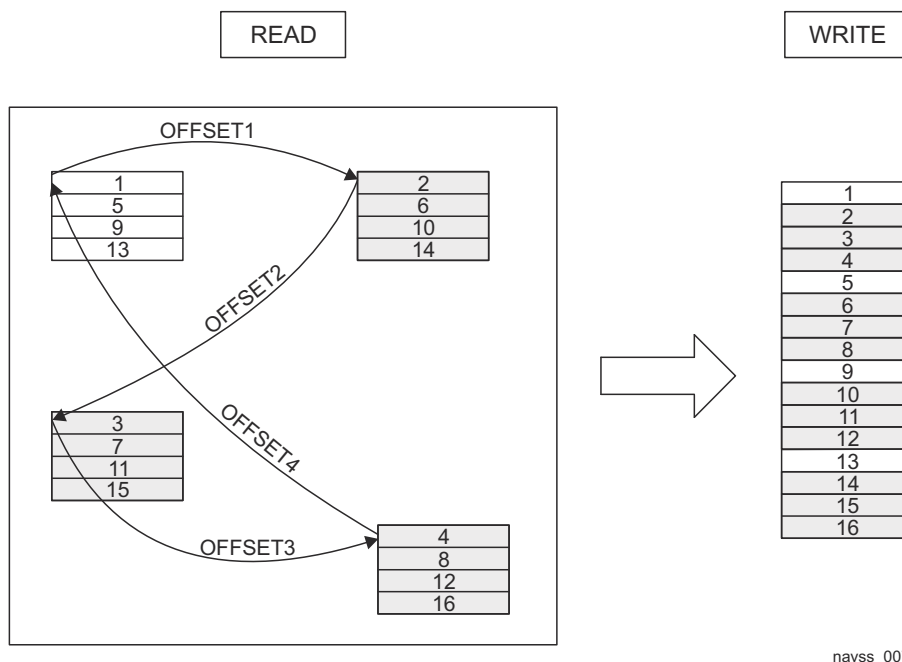
The following sections expand on each of these fields.

#### 10.1.2.3.6.5.2.1 SEC\_TR\_TYPE: Secondary TR Type Field

The Secondary TR Type field is used to state the purpose of the secondary TR. This can be defined

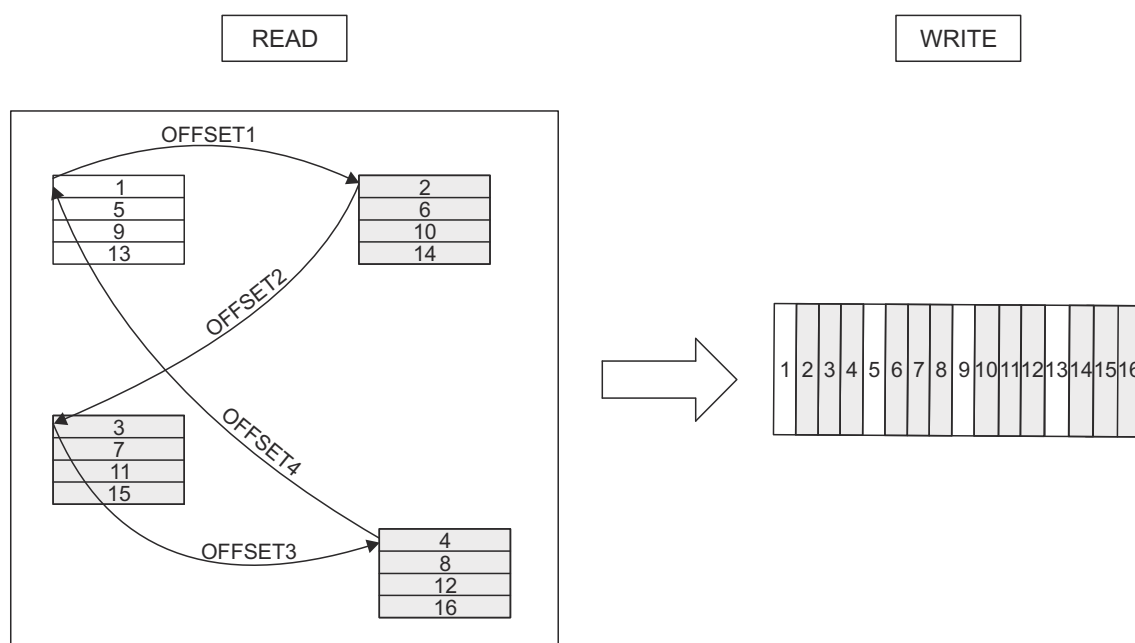
#### 10.1.2.3.6.5.2.2 Multiple Buffer Interleave

The multiple buffer interleave is used to have a single TR plus Secondary TR that can define a fetch where a single line is selected from multiple buffers and then sent as a single buffer. This can be used along with transpose of the data to present data in slices optimal for processing the same algorithm on different data sets at the same time. An example of this can be seen in [Figure 10-4](#).



**Figure 10-4. Example of Multiple Buffer Interleaving Read with 4 Offsets**

The case above provides the ability to minimize the number of TRs submitted to make the transfer but the interleaved Read main advantage comes when used with the transpose mode enabled. [Figure 10-5](#) shows the advantage of the interleaved mode along with the transpose mode to allow for maximum size transfer on both read and write transactions.



navss\_008

**Figure 10-5. Example of Interleaved Read with 4 Offsets and Transpose Enabled**

As the figure shows the data that is written out now contains one element from each block of memory and can be read in directly to processing units.

**Table 10-76. SECONDARY TR FLAGS Field Definition**

Table 16-16: SECONDARY TRF/ECF Field Definition															
31															16
SEC_TR_TYPE_SPECIFIC															
15						6		5		4		3		0	
SEC_TR_TYPE_SPECIFIC								NUMOFFSET				0 (Multiple Interleave Type)			

The flags field fills in the remaining details about the stream, as follows:

**Table 10-77. Multiple Buffer Interleave FLAGS Field Descriptions**

Bit	Field	Description
6-31	Reserved	Reserved for Future Use
4-5	NUMOFFSET	The encoded value for the number of offsets. 0: 4, 1: 8, 2: 16
0-3	SEC_TR_TYPE	0: Multiple Buffer Interleave

Secondary TR TYPE from the Secondary TR Flags field.

**Table 10-78. Multiple Buffer Interleave Secondary Transfer Request Template 128 Byte Secondary TR**

word 31	word 30	word 29	word 28
RESERVED			
word 27	word 26	word 25	word 24
RESERVED			
word 23	word 22	word 21	word 20
RESERVED			
word 19	word 18	word 17	word 16
OFFSET15	OFFSET14	OFFSET13	OFFSET12
word 15	word 14	word 13	word 12



**Table 10-78. Multiple Buffer Interleave Secondary Transfer Request Template 128 Byte Secondary TR (continued)**

OFFSET11	OFFSET10	OFFSET9	OFFSET8
word 11	word 10	word 9	word 8
OFFSET7	OFFSET6	OFFSET5	OFFSET4
word 7	word 6	word 5	word 4
OFFSET3	OFFSET2	OFFSET1	OFFSET0
word3	word 2	word 1	word0
RESERVED	SECONDARY TR FLAGS	ADDR	

Multiple Buffer Interleave will work by setting the SECTR field in the TR presented to the channel to be either 64 or 128 bytes depending on the number of offsets required. An offset number of 4 or 8 will require 64 bytes while 16 offsets will require 128 bytes. When the channel becomes active it will see that the SECTR field is nonzero and fetch the SECTR of the size specified. The first access from the channel will be a read of the secondary TR as specified in the address field of the TR. Upon receiving the Secondary TR the channel will parse the type field and seeing that it is a Multiple Buffer interleave will load the offset index fields for the channel. Additionally it will load the ADDRESS field from the secondary TR into the Address of the actual TR to allow it to process as normal.

The first address fetch will then be to this SEC\_TR\_ADDRESS. After finishing the inner loop the next address will be SEC\_TR\_ADDRESS + OFFSET0.

#### Note

ICNT1 should be equal to the height of the transfer blocks times the number of offsets.

#### 10.1.2.3.6.6 AMODE SPECIFIC: Addressing Mode Field

The AMODE Specific Fields are defined based upon the Addressing Mode being used as specified in AMODE: Addressing Mode Definition.

##### 10.1.2.3.6.6.1 Circular Address Mode Specific Flags

If the circular addressing mode is selected in AMODE the definition of the upper 16 bits of the FMTFLAGS field are used to select the type of addressing for each entry in the loop and to define 2 unique masks that can be used for the circular buffering.

The following example code shows the method used to compute the masks referred to by the address\_add function described in the 'Address Arithmetic with Selectable Linear/Circular Addressing Modes' section.

```
// Circular addressing masks.
// 1 bits in the mask indicate address bits that remain fixed.
// 0 bits in the mask indicate address bits that are allowed to vary.
uint64_t circ_mask_0; // mask for circular addressing block 0
uint64_t circ_mask_1; // mask for circular addressing block 1
void compute_circular_address_masks( int cbk0, int cbk1 )
{
    int block_size_0 = cbk0 + 9; // power-of-2 of block size in bytes
    int block_size_1 = cbk0 + cbk1 + 10;
    if ( block_size_1 > 31 ) block_size_1 = 32; // clamp to 4GB maximum size
    circ_mask_0 = (~0ULL) << block_size_0;
    circ_mask_1 = (~0ULL) << block_size_1;
}
```

#### Example 3-4 Circular Address Mask Computation

**Table 10-79. Transfer Request Template FMTFLAGS AMODE SPECIFIC Circular Addressing Field**

31	30	29	28	27	26	25	24	23	20	19	16
AM3		AM2		AM1		AM0		CBK1		CBK0	

**Table 10-80. Circular Address Mode Specific Flags Field Descriptions**

Bit	Field	Description
16-19	CBK0	Specifies the circular block 0 value as used in Circular Address Mask Computation
20-23	CBK1	Specifies the circular block 1 value as used in Circular Address Mask Computation
24-25	AM0	The address mode to use with ICNT0 loop
26-27	AM1	The address mode to use with ICNT1 loop
28-29	AM2	The address mode to use with ICNT2 loop
30-31	AM3	The address mode to use with ICNT3 loop

#### 10.1.2.3.6.6.1.1 CBK0 and CBK1: Circular Block Size Selection

The CBK0 and CBK1 fields set the circular block sizes for circular addressing. CBK0 directly determines the circular block size for block 0. CBK1 combines with CBK0 to determine the circular block size for block 1. Specifically, the streaming engine sets block 1's size according to  $CBK0 + CBK1 + 1$ . Therefore, circular block 1 is always larger than circular block 0.

The following table illustrates the resulting valid block sizes for circular block 0 and 1. For circular block 0, the use the value of CBK0 directly. Circular block 0 supports block sizes ranging from 512 bytes to 16M bytes. For circular block 1, use the value of  $CBK0 + CBK1 + 1$ . Circular block 1 supports sizes ranging from 1K bytes to 4G bytes.

**Table 10-81. Circular Block Size Decoding**

Encoded Block Size	Decoded Block Size	Encoded Block Size	Decoded Block Size	Encoded Block Size	Decoded Block Size	Encoded Block Size	Decoded Block Size
0	512	8	128K	16	32M	24	Reserved
1	1K	9	256K	17	64M	25	Reserved
2	2K	10	512K	18	128M	26	Reserved
3	4K	11	1M	19	1M	27	Reserved
4	8K	12	1M	20	1M	28	Reserved
5	16K	13	1M	21	1M	29	Reserved
6	32K	14	1M	22	1M	30	Reserved
7	64K	15	1M	23	1M	31	Reserved

#### 10.1.2.3.6.6.1.2 Amx: Addressing Mode Selection

The fields AM0 through AM3 control the addressing modes for each of the 4 dimensions. The UTC supports the following addressing modes:

**Table 10-82. AMX Address Mode Selection Encoding**

AMX	Meaning
00b	Linear addressing
01b	CKB0 (Circular Buffer 0 is used)
10b	CKB1 (Circular Buffer 1 is used)
11b	Reserved

Each dimension has its own addressing mode, independent of the other dimensions. This permits any mixture of linear and circular addressing among the 4 dimensions.

When a given dimension selects linear addressing, the UTC uses unmodified two's complement arithmetic to update addresses for that loop level.

When a given dimension selects circular addressing via CBK0 or CBK1, the UTC uses the circular address arithmetic described above in the Address Arithmetic with Selectable Linear/Circular Addressing Modes section.

### 10.1.2.3.6.7 Cache Flags

If the TR type is a cache warm the format flags field is replaced with a Cache Flags field.

Table 10-83 shows the breakdown of the 32 bit Cache Flags field.

**Table 10-83. Transfer Request Template CACHEFLAGS Field**

31	24	23	16
CACHE OPERATION		RESERVED	
15	8	7	0
RESERVED		CACHE ID	

The flags field fills in the remaining details about the stream, as follows:

**Table 10-84. FMTFLAGS Field Descriptions**

Bit	Field	Description
24-31	Cache Operation	0: Prewarm the Cache 1: Prewarm MMU 2-255: Reserved for future use
8-23	Reserved	Reserved for future use
0-7	Cache ID	The Cache number to perform the operation on. 0-254: Cores. 255: L3 Cache

### 10.1.2.4 Transfer Response Record

Upon completing a TR the UTC will send back a TR Response to the UDMA-C. The UTC TR response format is a single 32-bit word as shown in Table 10-85.

**Table 10-85. Transfer Request Response Template STATUS FLAGS Field**

31	24	23	16
CONFIGURATION SPECIFIC STATUS		CMD ID	
15	8	7	0
RESERVED		STATUS_FIELD	STATUS_TYPE

**Table 10-86. STATUS FLAGS Field Descriptions**

Bit	Field	Description
31:24	Configuration Specific Flags	These are flag bits that can be specified by the specific UTC configuration. These bits will remain undefined in the global TR format to allow for customization requirements that might be required in the specific UTC implementation. For UDMA channels which are capable of interfacing to endpoint peripherals (or via a PDMA) the format of these flags are as follows: 31:28 error_flags – the error_flags field as is provided across PSI-L 27:24 ps_flags – the ps_flags field as is provided across PSI-L
23:16	CMD ID	The Command ID for the TR. This will be sent back with the response for the Channel controller to identify the specific TR.
15:8	Reserved	Reserved for Future use
7:4	STATUS_INFO	Information that is unique based on the STATUS_TYPE returned
3:0	STATUS_TYPE	The completion status of the TR.

#### 10.1.2.4.1 STATUS Field Definition

The Status field is made up of two sub-fields STATUS\_TYPE and STATUS\_INFO. A value of 0 means it completed as requested any other value means an error has occurred. Table 10-87 states the legal STATUS\_TYPES and the use of the STATUS\_INFO in each case.

#### 10.1.2.4.1.1 STATUS\_TYPE Definition

The STATUS\_TYPE field is used to determine what type of status is being returned. Depending on the type of Status it might additionally have information in the STATUS\_FIELD to give more details about the specific reason why the status was returned.

**Table 10-87. STATUS\_TYPE Encoded Values**

Value	Error Type	Completion	STATUS FIELD Definition
0	None	Complete	None
1	Transfer Error	None to Partial	CBA Non-Complete Status Received
2	Aborted Error	None to Partial	None
3	Submission Error	None	Submission Error Type
4	Unsupported Feature	None	Feature Type
5	Transfer Exception	Partial	Exception Type
6	Teardown Flush	None	None
7-15	Reserved	Unknown	Reserved

##### 10.1.2.4.1.1.1 Transfer Error

A transfer ERROR occurs anytime that one of the CBA transactions performed by the UTC returns a status other than complete. The UTC may continue the transfer or may abort the transfer and return back to an IDLE state. Once the transfer is finished (aborted or complete) the UTC will send back the TR Response with the STATUS TYPE of Transfer Error. The STATUS FIELD will contain the 3 bit CBA Status field the upper bit specifies if it was a read status (1) or a write status (0).

##### 10.1.2.4.1.1.2 Aborted Error

An aborted error occurs if the PSI-L interface asserts the drop signal prior to the completion of the TR. The UTC will return back to an IDLE state and send back the TR Response with the STATUS TYPE. If the UTC receives a transfer error after receiving the abort the transfer error should be reported instead of the Abort Error.

##### 10.1.2.4.1.1.3 Submission Error

A submission error is returned for a TR that is received that cannot be run. The STATUS\_INFO field specifies the type of submission errors.

**Table 10-88. Submission Error STATUS\_INFO Encodings**

Value	Submission Error Type
0	ICNT0 was 0
1	Channel FIFO was full when TR received
2	Channel is not owned by the submitter. Either CC submitting to Direct or Direct submitting to a CC channel
3-15	Reserved

##### 10.1.2.4.1.1.4 Unsupported Feature

An unsupported feature error is returned for a TR that is received that cannot be run because it specifies a feature that is optional and not supported by the UTC that received the TR. The STATUS FIELD specifies the feature that was not supported. If it specifies multiple features that were not supported the UTC implementation can determine which type it wants to return

**Table 10-89. Unsupported Feature STATUS\_INFO Encodings**

Value	Submission Error Type
0	TR Type not supported
1	STATIC not supported
2	EOL not supported
3	CONFIGURATION SPECIFIC not supported

**Table 10-89. Unsupported Feature STATUS\_INFO Encodings (continued)**

Value	Submission Error Type
4	AMODE not supported
5	ELTYPE not supported
6	DFMT not supported
7	SECTR not supported
8	AMODE SPECIFIC Field not supported
9-15	Reserved

#### 10.1.2.4.1.1.5 Transfer Exception

A transfer exception is returned for a TR that completed but experienced a known exception during reception. The STATUS\_INFO field specifies the type of transfer exception that was encountered.

**Table 10-90. Transfer Exception STATUS\_INFO Encodings**

Value	Transfer Exception Type
0	EOP on incoming data stream was encountered prematurely (short packet)
1	EOP on incoming data stream was encountered late (long packet)
2-15	Reserved

#### 10.1.2.4.1.1.6 Teardown Flush

Split mode UTC Rx channels (write channels) can respond with a teardown flush in the case that they have received a teardown message, all data has been transferred, and TRs have been prefetched (in anticipation of more data being received). In this case, the UTC will simply return the prefetched TR to the completion queue with the status type set to teardown flush.

### 10.1.2.5 Queues

Queues are used to hold either values or references that need to be passed between components in the system. These components could be software or hardware. Queues are implemented by the Ring Accelerator module.

#### 10.1.2.5.1 Queue Types

##### 10.1.2.5.1.1 Transmit Queues (Pass By Reference)

Tx channels use packet queues referred to as “transmit queues” to store the packets that are waiting to be transmitted. Tx channels require one or more packet queues dedicated to each transmit channel for this purpose. Multiple queues per channel may facilitate Quality of Service (QoS) in some applications. Pass by reference Tx Queues only pass a pointer to a descriptor (and optional information like the packet length). Pass by reference Tx Queues can contain pointers to Host, Monolithic, or TR packet descriptors (depending on Tx channel mode configuration).

##### 10.1.2.5.1.2 Transmit Queues (Pass By Value)

Pass by value Tx Queues are used for direct Transfer Request submission on Tx channels which are configured for 3<sup>rd</sup> party pass by value mode. Entries on a pass by value Tx Queue include the entire Transfer Request message and optional protocol specific information. Pass by value queues are only supported by the Ring Accelerator.

##### 10.1.2.5.1.3 Transmit Completion Queues (Pass By Reference)

Tx channels also use packet queues referred to as “transmit completion queues” to return packets to the host after they are transmitted. The number of queues required by the channel or system for this purpose is driven by the requirements of the garbage collection software within the driver. Tx completion queues are only used when the packet descriptor indicates that the packet should be returned to a queue instead of directly recycling the descriptors to their queues of origin. Pass by reference Tx Completion Queues only pass a pointer to a descriptor.

#### 10.1.2.5.1.4 Transmit Completion Queues (Pass By Value)

Pass by value Tx Completion Queues are used to pass the Transfer Response messages that are returned when a channel is configured in 3<sup>rd</sup> Party pass by value mode. Entries on a pass by value Tx Completion Queue include the Transfer Response message. Pass by value queues are only supported by the Ring Accelerator.

#### 10.1.2.5.1.5 Receive Queues

Rx ports use packet queues referred to as “receive queues” to forward completed received packets to the host or another peer port entity. Rx ports may be configured in various ways to forward the received packets onto any number of receive queues. Rx ports may choose to queue their receive packets based strictly on Rx channel, protocol type, priority, direct forwarding requirements (i.e. to another ports Tx queue) or any combination of these and this is application specific. In many cases the receive queue is in fact also a transmit queue for another peer entity. It is just a matter of semantics with respect to which port is referenced in the system.

#### 10.1.2.5.1.6 Free Descriptor Queues

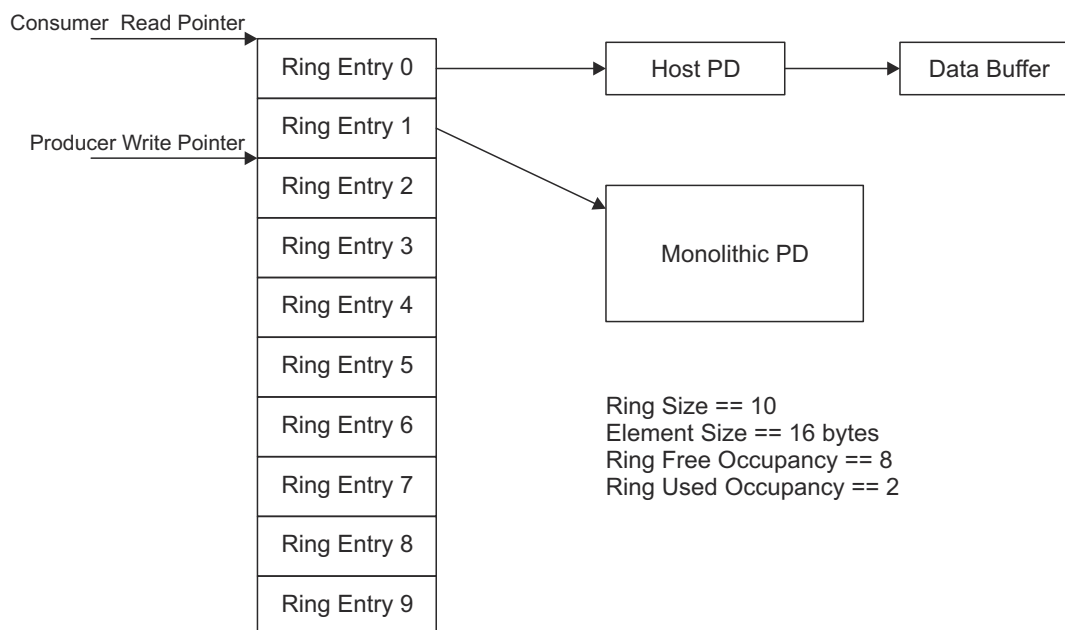
Rx ports use queues referred to as “free descriptor queues” to allocate empty monolithic descriptors. The entries on the Free Descriptor Queues do not have any buffers attached as buffers are allocated as part of the descriptor for monolithic type packets.

#### 10.1.2.5.1.7 Free Descriptor/Buffer Queues

Rx ports use queues referred to as “free descriptor/buffer queues” to allocated empty host type descriptors. The entries on the Free Descriptor/Buffer Queues have pre-attached empty buffers whose size and location are described in the “original buffer information” fields in the descriptor. The host is required to allocate both the descriptor and buffer and pre-link them prior to adding a descriptor to a free descriptor/buffer queue.

#### 10.1.2.5.2 Ring Accelerator Queues Implementation

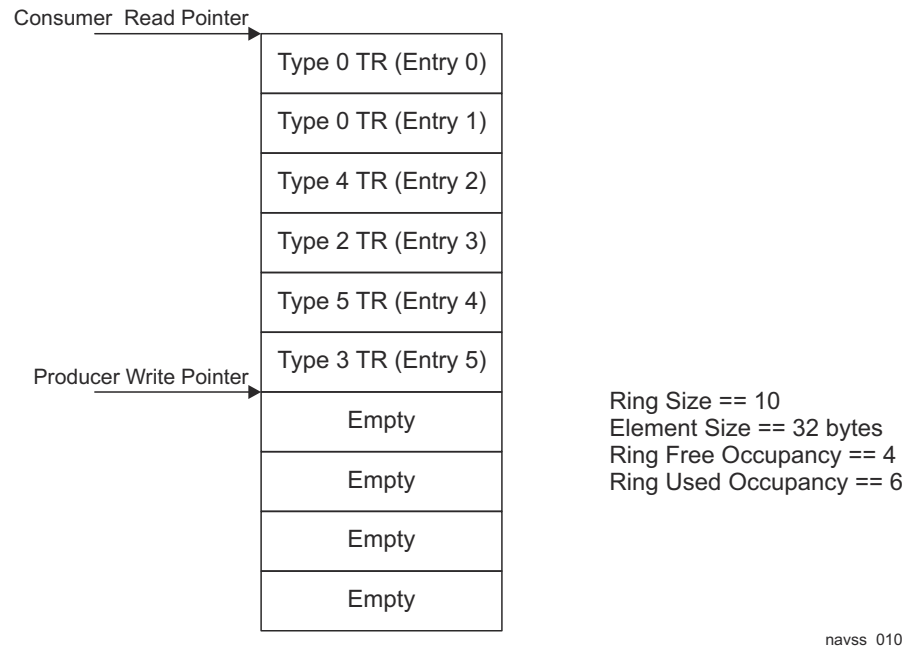
Work packets which are stored in a pass by reference queue managed by the Ring Accelerator are shown in [Figure 10-6](#).



navss\_009

**Figure 10-6. Ring Accelerator Based Queue Structure**

Work packets stored in a pass by value queue managed by the Ring Accelerator are shown in [Figure 10-7](#).



**Figure 10-7. Ring Accelerator Based Direct Transfer Request Queue Structure**

### 10.1.3 Operational Description

#### 10.1.3.1 Resource Allocation

Queues, rings, memory, interrupts, DMA channels, and flow table entries are shared resources within the data movement architecture and their ownership and use must be managed in order to provide stable, safe, and secure operation. The K3 platform provides channelized firewalls which can be used to protect arrayed resources (like queues, rings, channel control registers, etc.) from unwanted/unauthorized use as well as restricted access to the firewall configurations themselves. It is outside the scope of this document to define how the resource management system functions but it is assumed that all of these resources are isolate-able and secure-able.

#### 10.1.3.2 Ring Accelerator Operation

##### 10.1.3.2.1 Queue Initialization

The base address, size and element size for each ring must be initialized via the provided registers prior to using the ring. Ring initialization is typically done only on channel setup as once initialized the rings can be used indefinitely. Optionally, an internal initialization machine may be provided to set default values for the ring address, size, and element sizes. This capability enables the rings to be used during boot without configuration. Not all rings will necessarily support this feature and if required must be configured at design time. Whether or not this machine is present, all rings are able to be re-configured at run time through the provided registers.

##### 10.1.3.2.2 Queuing packets (Exposed Ring Mode)

Queuing of packets onto a packet queue using the Ring Accelerator is logically equivalent but requires a different set of transactions from queuing packets in queue mode. The producer (entity which is going to queue an entry onto the ring) maintains a current write pointer and a free entry occupancy count for each ring it controls. To queue an entry the producer will write the entry contents (typically a pointer to the Packet Descriptor along with optional sideband information) to the memory location pointed to by the current write pointer and will decrement its free entry occupancy. After the producer has confirmed that the data has actually landed in memory, it will then write to the doorbell register for the ring to increment the used occupancy count for the ring.

More than one entry can be queued with a single doorbell write as the entry count in the doorbell write indicates how many entries have been queued.



### 10.1.3.2.3 De-queuing packets (Exposed Ring Mode)

The consumer (entity which is going to de-queue an entry from the ring) maintains a current read pointer and a used entry occupancy count for each ring it controls. To de-queue an entry the consumer will read the entry contents from the memory location pointed to by the current read pointer and will decrement its used entry occupancy. The consumer will then write to the doorbell register for the ring to increment the free occupancy count for the ring.

### 10.1.3.2.4 Queuing packets (Queue Mode)

To queue an entry the producer will write the entry contents (typically a pointer to the Packet Descriptor along with optional sideband information) to the Proxy thread location allocated by the system, with the queue offset for the queue to access. The write within the queue offset to the completion offset (final byte of the data) will cause the entire data to be written to the RA queue.

### 10.1.3.2.5 De-queuing packets (Queue Mode)

To de-queue an entry the consumer will read the entry contents from the Proxy thread location allocated by the system, with the queue offset for the queue to access. The proxy will read the entry from the RA queue. The consumer can then read the entire contents of the entry. The completion offset (final byte of the data) will cause the proxy to flush that entry (so that it can fetch a new one from the RA upon the next read).

### 10.1.3.3 UDMA Internal Transmit Channel Setup (All Packet Types)

After a reset or a previous teardown operation but before queuing packets to a channel the host must initialize the channel's Tx Port DMA State. The host initializes the channel Tx Port DMA State by writing to the Tx Channel Configuration Register A. The Host may choose to write the enable bit in the Tx Channel Configuration Register A at the same time or after it has written all of the channel parameters but note that every write to the Tx Channel Configuration Registers will overwrite the channel state for all bytes that are enabled for the write transaction.

It should be noted that very little configuration information is required for a UDMA Tx channel compared to previous versions of UDMA. This is because UDMA has transitioned to a full modeless model for channel operation where each packet descriptor specifies the descriptor type, packet type, and the packet return parameters. Packet queues in UDMA may include combinations of host and monolithic descriptors in any order.

After a Transmit channel has been set up, packets can be added to the Queues for the channel to begin the transmit operation. The following sections describe how the transmit operations are performed for the various descriptor types.

### 10.1.3.4

### 10.1.3.5 UDMA Internal Transmit Channel Teardown (All Packet Types)

An Tx channel teardown is initiated by the host by writing the TX\_TEARDOWN bit in the Tx Channel N Realtime Control Register. When the host initiates teardown, it can choose to perform either a graceful (no data loss) or forced teardown of the channel depending on the setting of the TX\_FORCED\_TEARDOWN bit in the Tx Channel N Realtime Control Register.

If the TX\_FORCED\_TEARDOWN bit is clear, a normal teardown has been initiated. In this case, the UDMA will do the following:

1. Stops performing any additional prefetches for the channel.
2. Completes any packets normally for which a pointer/descriptor/TR prefetch has already been performed
3. Sets the tdown bit on the EOP data phase of the last packet to be sent (if any traffic was pending when teardown was initiated) or sends a zero byte packet with the tdown, sop, and eop bits asserted. This signals the destination thread that all of the data has been sent.
4. Clears the channel enable in the Tx Channel N Realtime Control Register.
5. Resets the channel state (including scoreboards, FIFOs, counters, statistics, etc.) to their after reset values.
6. Sends a single, 4-byte, teardown complete message to the queue specified in the Tx Channel Completion Queue register for the channel. The format of this message is as follows:



Bits	Field	Value	Description
31	Forced	0	Indicates that the teardown was graceful and data was not lost.
30:14	-	0	RESERVED
13:4	Channel ID	channel number	Indicates which channel the teardown completed on
3:0	Teardown Indicator	0x1	Indicates that a teardown has completed – normal pointers must be 16 byte aligned so this value indicates this is a teardown

If the TX\_FORCED\_TEARDOWN bit is set, a destructive teardown has been initiated. In this case, the UDMA will do the following:

1. Suspends any triggers to allow the channel to operate even if the trigger source is no longer functioning.
2. Completes any packets which have been prefetched with or without transferring any data or generating events (the implementation has the option of doing whichever is simplest for that implementation).
3. Clears the channel enable in the Tx Channel N Realtime Control Register.
4. Resets the channel state (including scoreboards, FIFOs, counters, statistics, etc.) to their after reset values.
5. Sends a single, 4-byte, teardown complete message to the queue specified in the Tx Channel Completion Queue register for the channel. The format of this message is as follows:

Bits	Field	Value	Description
31	Forced	1	Indicates that the teardown was not graceful. Data was potentially lost.
30:14	-	0	RESERVED
13:4	Channel ID	channel number	Indicates which channel the teardown completed on
3:0	Teardown Indicator	0x1	Indicates that a teardown has completed – normal pointers must be 16 byte aligned so this value indicates this is a teardown

The host may issue a teardown on any channel at any time, regardless of whether the channel is actively receiving a packet or not.

The host determines that a teardown is complete by periodically polling the teardown and enable bits for the channel or by waiting to receive the teardown record on the queue specified in the Tx Channel Completion Queue register for the channel.

#### 10.1.3.6 UDMA External Transmit Channel Setup

External UTC/DRU channels which are managed by the UDMA follow a modified setup sequence which is as follows:

**Table 10-91. External Transmit Channel Setup Sequence**

Step	Description	module.mregion.register.field
1	Configure the UTC-DRU starting thread number	udma.gcfcg.utc_ctrl.utc_chan_start
2	Configure the UTC-DRU to accept TR's over PSIL	UTC-DRU specific
3	Enable the UDMA-C external TX channel	udma.tchanrt.tchanrt[chan]_trt_ctl.tx_enable = 1
4	Enable the UTC-DRU thread	psil.gcfcg.enable.enable(bit 31) = 1 (via psilcfc proxy)

#### 10.1.3.7 UDMA Transmit External Channel Teardown

External UTC/DRU channels which are managed by the UDMA follow a modified teardown sequence which is as follows:

**Table 10-92. External Transmit Channel Teardown Sequence**

Step	Description	module.mregion.register.field
1	Initiate UDMA-C external TX channel teardown	udma.tchanrt.tchanrt[chan]_trt_ctl.tx_tx_teardown = 1
2	Initiate UTC-DRU thread teardown	psil.gcfcg.enable.tdown(bit 30) = 1 (via psilcfc proxy)
3	Wait for UDMA-C external TX channel enable to be 0	udma.tchanrt.tchanrt[chan]_trt_ctl.tx_enable == 0

### 10.1.3.8 UDMA-P Transmit Channel Pause

Setting the TX\_PAUSE bit in the Tx Channel N Realtime Control Register will suspend the channel from arbitration resulting in a halting of the flow of data. Clearing this bit will cause the channel to be added back into the arbitration list. Pausing a channel has no other destructive side effects (other than potential underrun/overrun conditions from a downstream data sink or upstream data source).

### 10.1.3.9

### 10.1.3.10 UDMA-P Transmit Operation (Host Packet Type)

After a channel has been set up it can begin to be used to transmit packets. Packet transmission in Host mode involves the following steps:

1. The Host is made aware of one or more chunks of data in memory that need to be transmitted as a packet. This may involve directly sourcing data from the Host or it may involve data which has been forwarded from another data source in the system.
2. The Host allocates and populates a host packet descriptor. The host will initialize the following fields within the packet descriptor:
  - a. Descriptor Type (set to Host)
  - b. Packet Length indicating the total number of bytes that are to be read from all of the buffers for this packet.
  - c. Source Tag
  - d. Destination Tag (application specific)
  - e. Packet Type
  - f. Any protocol specific flags for the given packet type
  - g. Buffer Pointer with the byte aligned address of the first chunk of buffer data
  - h. Buffer Length with the number of bytes in the first chunk of buffer data
  - i. Descriptor reclamation policy fields indicating the mode and queue number for recycling the packet after transmission is complete.
  - j. The Next Descriptor Pointer with the address of the next descriptor in this packet. If this is the last chunk of data in the packet, this field must be set to zero
  - k. Any protocol specific descriptor sections that are required for the given packet type or system configuration
3. The Host allocates and populates host buffer descriptors as necessary to point to any remaining chunks of data that belong to this packet. The host will initialize the following fields within the host buffer descriptor:
  - a. Buffer Descriptor reclamation policy fields (if the intention is to have the DMA automatically recycle the buffer descriptors as they are transmitted).
  - b. Buffer Pointer with the byte aligned address of the given chunk of buffer data
  - c. Buffer Length with the number of bytes in the given chunk of buffer data
  - d. The Next Descriptor Pointer with the address of the next descriptor in this packet. If this is the last chunk of data in the packet, this field must be set to zero
4. The Host writes queues the packet onto one of the Transmit Queues for the desired DMA channel. Channels may provide more than one Tx Queue and may provide a particular prioritization policy between the queues. This behavior is application specific and is controlled by the DMA controller/scheduler implementation.
5. The Ring Accelerator provides a level sensitive status signal for the queue which indicates if any packets are currently pending. This level sensitive status line is sent to the hardware block which is responsible for scheduling DMA operations.
6. The DMA controller is eventually brought into context for the corresponding channel and begins to process the packet.
7. The DMA controller reads the packet descriptor pointer from the Ring Accelerator
8. The DMA controller reads the packet descriptor from memory
9. Step 9:
  - If the return policy bit (PD Word 2, bit 15) is cleared, the DMA controller empties each buffer in sequence by transmitting the contents in one or more block data moves. As each buffer is emptied, the DMA will

read the next buffer descriptor to obtain the pointer and size of the data buffer as well as the pointer to the next descriptor in the chain.

- If the return policy bit is set, the DMA controller empties each buffer in sequence by transmitting the contents in one or more block data moves.

10. Step 10:

- If the return policy bit (PD Word 2, bit 15) is cleared, when all data for the packet has been transmitted as specified in the packet size field, the DMA will write the pointer to the packet descriptor to the queue specified in the return queue number fields of the packet descriptor.
- If the return policy bit is set, as each buffer is emptied the DMA will write the buffer descriptor pointer to the queue specified in the return queue number field of the buffer descriptor. The DMA will also read the next buffer descriptor to obtain the pointer and size of the data buffer, the return queue information for the descriptor, as well as the pointer to the next descriptor in the chain.

11. Step 11:

- If the return policy bit (PD Word 2, bit 15) is cleared, after the Packet Descriptor pointer has been written, the Ring Accelerator will indicate the status of the Tx Completion Queue by sending an up event .
- If the return policy bit is set, when all data for the packet has been transmitted as specified in the packet size field, if the current buffer is not marked as end of packet (via a null next descriptor pointer), the DMA will continue to walk the list returning buffer descriptors to the appropriate queues as described in step 10.

12. Step 12:

- If the return policy bit (PD Word 2, bit 15) is cleared, the Interrupt Aggregator receives the up event and sets the corresponding bit in the interrupt status register as programmed in the interrupt mapping registers. This in turn causes an interrupt to the Host to be generated.
- If the return policy bit is set, when all buffers have been returned, the DMA will write the pointer to the packet descriptor to the queue specified in the return queue number fields of the packet descriptor.

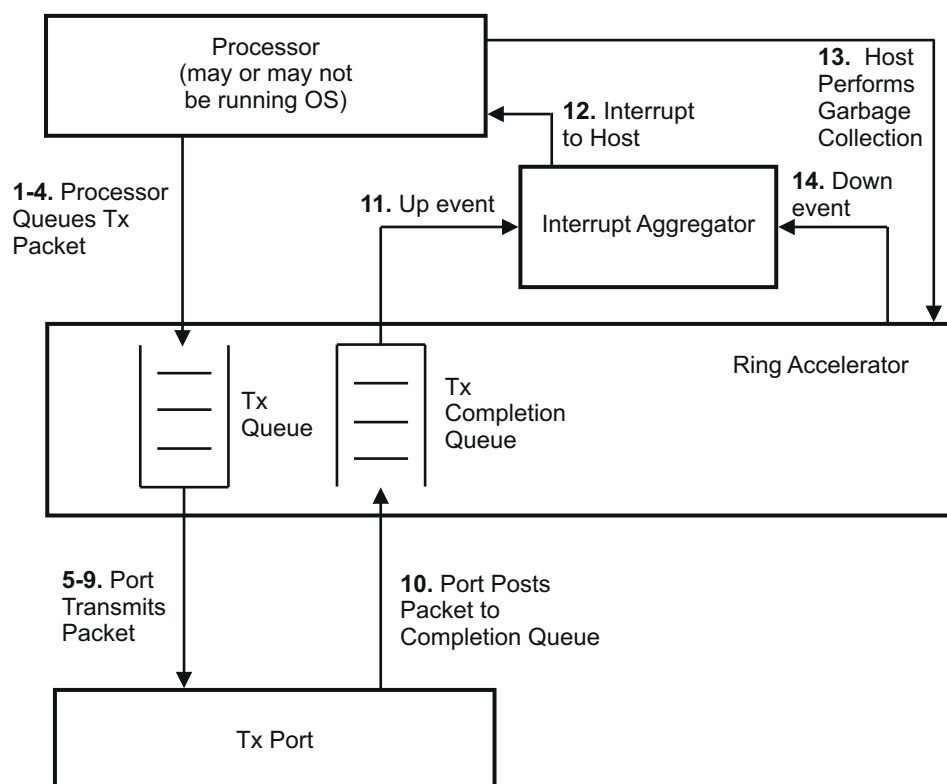
13. If the return policy bit (PD Word 2, bit 15) is cleared, host responds to status change from the Ring Accelerator (via the Interrupt Aggregator) and performs garbage collection as necessary for the packet.

14. If the return policy bit (PD Word 2, bit 15) is cleared, once the garbage collection causes the queue to become empty, the Ring Accelerator will send a down event to the Interrupt Aggregator which will clear the corresponding bit in the Interrupt Status Register and potentially de-assert the interrupt line.

If the return policy bit (PD Word 2, bit 15) is cleared, the complete process is illustrated on [Figure 10-8](#).

If the return policy bit is set, the complete process is illustrated on [Figure 10-9](#).

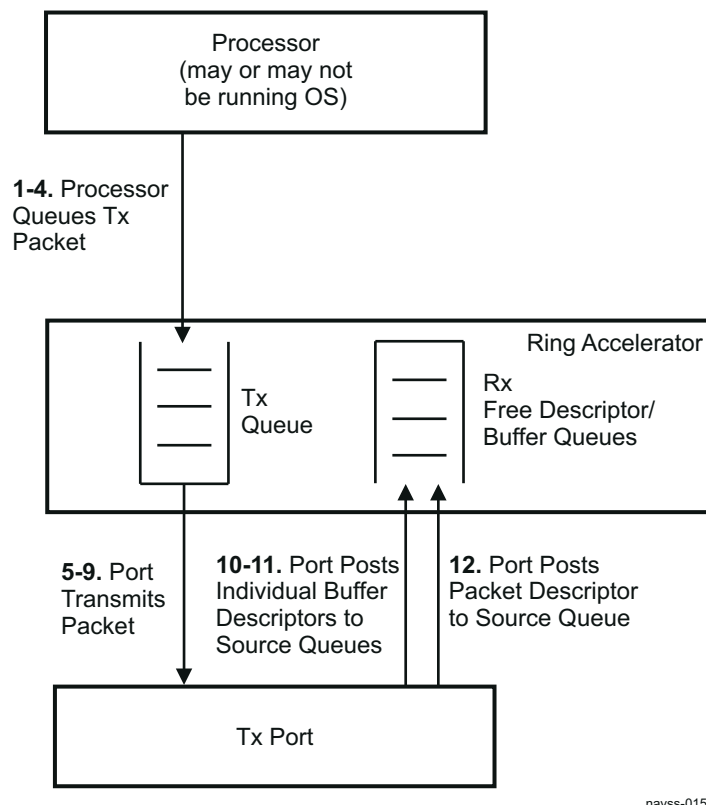
[Figure 10-8](#) shows a diagram of the host packet transmit operation – complete packet return.



navss-014

**Figure 10-8. Host Packet Tx Operation – Complete Packet Return**

Figure 10-9 shows a diagram of the host packet transmit operation – automatic buffer recycling packet return.



navss-015

**Figure 10-9. Host Packet Tx Operation – Automatic Buffer Recycling Packet Return**

#### 10.1.3.11 UDMA-P Transmit Operation (Monolithic Packet)

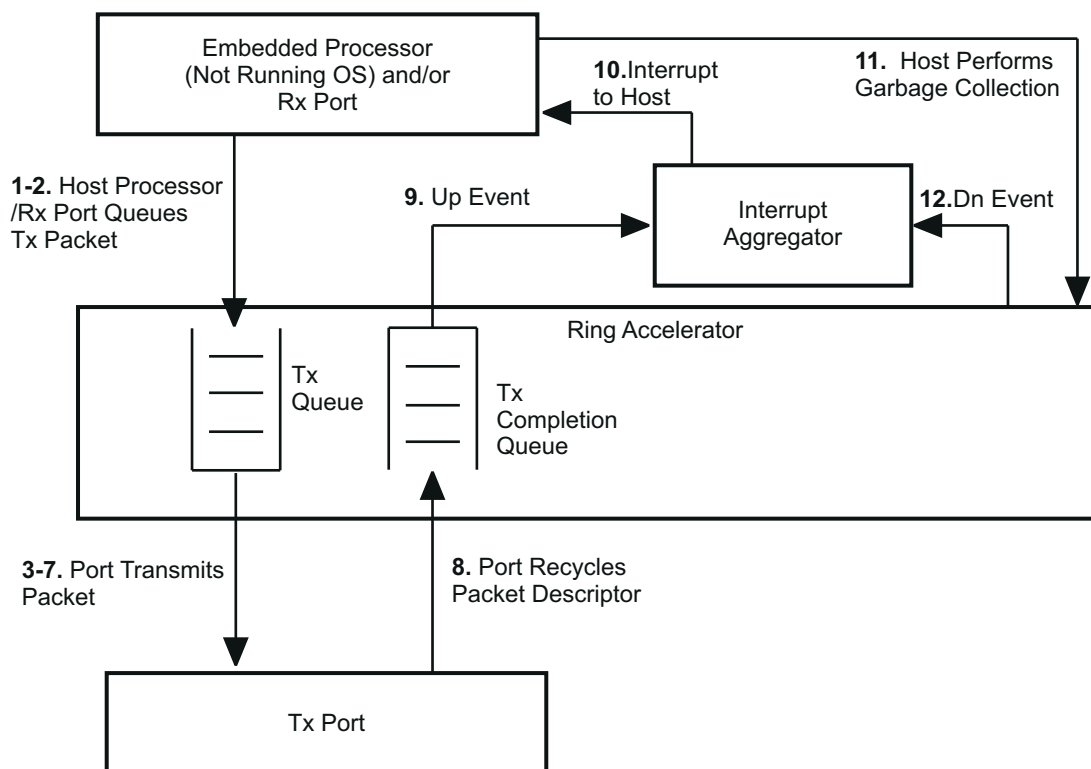
Packet transmission in Monolithic mode involves the following steps:

1. The Host allocates and populates a monolithic packet descriptor. The host will initialize the following fields within the packet descriptor:
  - a. Descriptor Type (set to Monolithic)
  - b. Packet Length indicating the total number of bytes in the packet
  - c. Source Tag
  - d. Destination Tag (application specific)
  - e. Packet Type
  - f. Any protocol specific flags for the given packet type
  - g. Offset to payload data
  - h. Descriptor reclamation policy fields indicating the mode and queue number for recycling the packet after transmission is complete.
  - i. Networking Stack Information (application specific and optional)
  - j. Any protocol specific words that are required for the given packet type
2. The Host queues the packet onto one of the Transmit Queues for the desired DMA channel. Channels may provide more than one Tx Queue and may provide a particular prioritization policy between the queues. This behavior is application specific and is controlled by the DMA controller/scheduler implementation.
3. The Ring Accelerator provides a level sensitive status signal for the queue which indicates if any packets are currently pending. This level sensitive status line is sent to the hardware block which is responsible for scheduling DMA operations.
4. The DMA controller is eventually brought into context for the corresponding channel and begins to process the packet.
5. The DMA controller reads the packet descriptor pointer from the Ring Accelerator.
6. The DMA controller reads the packet descriptor from memory

7. The DMA controller empties the data region in the descriptor by transmitting the contents in one or more block data moves. The size of these blocks is application specific. The DMA controller module specification is intended to document the block size used by a given implementation.
8. When all data for the packet has been transmitted as specified in the packet size field, the DMA will write the pointer to the packet descriptor to the queue specified in the return queue number fields of the packet descriptor.
9. After the Packet Descriptor pointer has been written, the Ring Accelerator indicates the status of the Tx Completion Queues to other ports/processors/prefetcher blocks using out-of-band level sensitive status lines. These status lines are set anytime a queue is non-empty.

Whether or not any further processing is required on the packet at this point depends on the nature of the queue that the packet was returned to. The most common case is that the monolithic descriptor will be returned to the Rx Free Descriptor queue that it was allocated from originally. If this is the case, the Tx processing is complete at this point. Alternatively, the queue may also be a Tx Completion Queue in which case, the following optional steps may also be performed:

10. While most types of peer entities and embedded processors are able to directly and efficiently use these level sensitive status lines, cached processors may require a hardware block to convert the level status into pulsed interrupts and to perform some level of aggregation of the descriptor pointers from the completion queues into lists.
11. Host responds to status change from Ring Accelerator and performs garbage collection as necessary for packet.
12. Ring Accelerator sends down event when ring is empty



navss\_011

**Figure 10-10. Monolithic Packet Tx Operation**

#### 10.1.3.12 UDMA Transmit Operation (TR Packet)

Packet transmission in TR Packet mode involves the following steps:

1. The Host allocates and populates a TR packet descriptor. The host will initialize the following fields within the packet descriptor:

- a. Descriptor Type (set to TR)
  - b. Reload Enable to 1 if looping is required, otherwise 0
  - c. Reload Index to an appropriate offset if Reload Enable set, otherwise 0
  - d. Last Entry to TR count minus 1
  - e. TR Nominal Element Size to a value that is as large as required for any given TR in the buffer
  - f. Packet ID to a value that can be used to correlate the packet when it is placed back on the Tx completion queue
  - g. Source Tag
  - h. Destination Tag (application specific)
  - i. Descriptor reclamation policy fields indicating the mode and queue number for recycling the packet after transmission is complete.
  - j. A set of one or more valid Transfer Request records whose quantity matches the last index specified previously.
2. The Host queues the packet onto one of the Transmit Queues for the desired UDMA channel. Channels may provide more than one Tx Queue and may provide a particular prioritization policy between the queues. This behavior is application specific and is controlled by the DMA controller/scheduler implementation.
  3. The Ring Accelerator provides a level sensitive status signal for the queue which indicates if any packets are currently pending. This level sensitive status line is sent to the hardware block which is responsible for scheduling DMA operations.
  4. The DMA controller is eventually brought into context for the corresponding Tx channel and begins to process the packet.
  5. The DMA controller reads the packet descriptor pointer from the RA.
  6. The DMA controller reads the packet descriptor from memory
  7. The DMA controller empties the data region in the descriptor by reading the contents in one or more nominal TR sized block data moves. These contents are then transmitted either to an internal UTC (in the case of a UDMA-P) or via PSI-L to a remote UTC (in the case of a UDMA-C).
  8. If the UDMA includes an internal UTC, all of the data transfers specified in a TR will be completed as a series of reads (for single ended transfers) or reads and corresponding writes (for block copy transfers) and a Transfer Response will be returned indicating the completion and status of the transfer.
  9. The UDMA will wait until Transfer Responses have been returned for each Transfer Request that it issued. As each Transfer Response is returned, the UDMA will write the response into the TR buffer in the Transfer Response records array. Each response is written into an array index which directly matches the index of the request record to which it corresponds.
  10. When all Transfer Requests in the packet have been processed and all Transfer Responses have been written back and confirmed to have landed in memory, the UDMA will write the pointer to the packet descriptor to the queue specified in the return queue number fields of the packet descriptor.
  11. After the Packet Descriptor pointer has been written, the Ring Accelerator indicates the status of the Tx Completion Queues to other ports/processors/prefetcher blocks using events sent to the Interrupt Aggregator. These events are then converted into standard K3 interrupts.

#### 10.1.3.13 UDMA Transmit Operation (Direct TR)

If the rings are configured in pass by value mode individual Transfer Request and Response records can be passed between software and hardware without any higher level data structures. This mode allows for the UDMA to be prompted to do a transfer by just writing memory mapped locations with a source address, destination address and byte count (in the simplest case). Direct TR is accomplished as follows:

1. The Host writes a valid Transfer Request record to the Tx Queue of the desired UDMA channel.
2. The Ring Accelerator provides a level sensitive status signal for the queue which indicates if any packets are currently pending. This level sensitive status line is sent to the hardware block which is responsible for scheduling DMA operations.
3. The DMA controller is eventually brought into context for the corresponding channel and begins to process the packet.
4. The DMA controller reads the Transfer Request record from the ring via the RA in a single nominal TR sized block data move. These contents are then transmitted either to an internal UTC (in the case of a UDMA-P) or via PSI-L to a remote UTC (in the case of a UDMA-C).



5. If the UDMA includes an internal UTC, all of the data transfers specified in a TR will be completed as a series of reads (for single ended transfers) or reads and corresponding writes (for block copy transfers) and a Transfer Response will be returned indicating the completion and status of the transfer.
6. When a Transfer Response is returned for the Transfer Request record the UDMA will write the response into the next entry in the Transmit Completion Ring. Since the Transfer Responses come back from the UTC strongly ordered each response will be written into the ring strongly ordered as well.
7. After the Transfer Response has been written, the Ring Accelerator indicates the status of the Tx Completion Queues to other ports/processors/prefetcher blocks using events sent to the Interrupt Aggregator. These events are then converted into standard K3 interrupts.

#### 10.1.3.14 UDMA Transmit Error/Exception Handling

There are 3 specific errors which may be encountered during Tx operation and which are trapped by the UDMA. The following table lists the errors and how the UDMA will process them:

**Table 10-93. UDMA Tx Error/Exception Handling**

Condition	Channel Mode	Pauses Channel	Error Flag Set in Channel	Data Transfer	TR Flush	Outputs Transfer Events
TR null icnt0	All UTC	Selectable <sup>(1)</sup>	Yes	No	Just that TR	No
Unsupported TR Type	All UTC	Selectable <sup>(1)</sup>	Yes	No	Just that TR	No
Bus Errors	All	Selectable <sup>(1)</sup>	Yes	Yes	No	Yes

(1) If PAUSE\_ON\_ERROR bit is set in channel configuration

##### 10.1.3.14.1 Null Icnt0 Error

The icnt0 parameter in a TR is never allowed to be 0 as this would cause zero byte count transactions to be issued on the bus and is unproductive wrt completion of any useful work. The Tx UTC engine in the UDMA will trap any TR for which the icnt0 is zero and will immediately return a transfer response with the TR icnt0 error conditions set without performing any data transfer and without outputting any events (including end of TR event).

##### 10.1.3.14.2 Unsupported TR Type

If a TR is provided to a channel which is of a type which is not supported by the channel, the Tx UTC engine in the UDMA-P will trap that TR and immediately return a transfer response with the Unsupported TR error conditions set without performing any data transfer and without outputting any events (including end of TR event).

##### 10.1.3.14.3 Bus Errors

If a bus error is encountered during transmit the DMA will log the error by asserting the TX\_ERROR bit for the channel (and optionally sending an error event if enabled) but the DMA will continue to attempt to complete the transfer using the returned (and potentially incorrect) data.

#### 10.1.3.15 UDMA Receive Channel Setup (All Packet Types)

After a reset or a previous teardown operation but before receiving packets on a channel the host must initialize the channel's Rx Port DMA State. The host initializes the channel Rx Port DMA State by writing to the Rx Channel Configuration Registers. The Host may choose to write the enable bit in the Rx Global Channel Configuration Register at the same time or after it has written all of the channel parameters but note that every write to the Rx Channel Configuration Registers will overwrite the channel state for all bytes that are enabled for the write transaction.

#### 10.1.3.16 UDMA Receive Channel Teardown

A Rx channel teardown is intended to be initiated at the data source (the source thread). Initiation is commanded by the host by writing the teardown bit in the PSI-L source thread register 0x408. This will cause the source of the data to gracefully terminate any packet that is in flight and send a PSI-L data phase with the tdown signal asserted. When the UDMA receives a data phase with the tdown attribute asserted it will immediately set the internal RX\_TEARDOWN bit in the Rx Channel N Realtime Control Register.



Upon seeing the RX\_TEARDOWN bit asserted the port can perform either a graceful (no data loss) or forced teardown of the channel depending on the setting of the RX\_FORCED\_TEARDOWN bit in the Rx Channel N Realtime Control Register.

If the RX\_FORCED\_TEARDOWN bit is clear, a normal teardown has been initiated. In this case, the UDMA will do the following:

1. Stops performing any additional prefetches for the channel (TR mode channels)
2. Completes any packets normally which may be pending in the ingress port buffers (what the port considers as pending packets is application specific).
3. Clears the channel enable in the Rx Channel Global Configuration Register.
4. Resets the channel state (including scoreboards, FIFOs, counters, statistics, etc.) to their after reset values.
5. Sends a single, 4-byte, teardown complete message to the queue specified in the Rx Channel Completion Queue register for the channel. The format of this message is as follows:

Bits	Field	Value	Description
31	Forced	0	Indicates that the teardown was graceful and data was not lost.
30:14	-	0	RESERVED
13:4	Channel ID	channel number	Indicates which channel the teardown completed on
3:0	Teardown Indicator	0x1	Indicates that a teardown has completed – normal pointers must be 16 byte aligned so this value indicates this is a teardown

If the RX\_FORCED\_TEARDOWN bit is set, a destructive teardown has been initiated. In this case, the UDMA will do the following:

1. Suspends any triggers to allow the channel to operate even if the trigger source is no longer functioning.
2. Completes any packets which may be pending in the ingress port buffers with or without transferring any data or generating events (the implementation has the option of doing whichever is simplest for that implementation) . Packets will be drained from the Rx FIFO either using 'null' write transfers or real transfers and the Packet Descriptors will be returned with an accurate accounting of actual bytes transferred.
3. Clears the channel enable in the Rx Channel N Realtime Control Register.
4. Resets the channel state (including scoreboards, FIFOs, counters, statistics, etc.) to their after reset values.
5. Sends a single, 4-byte, teardown complete message to the queue specified in the Rx Channel Completion Queue register for the channel. The format of this message is as follows:

Bits	Field	Value	Description
31	Forced	1	Indicates that the teardown was not graceful. Data was potentially lost.
30:14	-	0	RESERVED
13:4	Channel ID	channel number	Indicates which channel the teardown completed on
3:0	Teardown Indicator	0x1	Indicates that a teardown has completed – normal pointers must be 16 byte aligned so this value indicates this is a teardown

The host may issue a teardown on any channel at any time, regardless of whether the channel is actively receiving a packet or not. The normally intended method of issuing a teardown on a packet mode or single ended TR mode channel is to initiate the teardown at the remote PSI-L data source and allow it to flow into the UDMA. If that is not possible, the host may write the RX\_TEARDOWN bit directly to a 1.

The Host determines that a teardown is complete by periodically polling the teardown and enable bits for the channel or by waiting to receive the teardown record on the queue specified in the Tx Channel Completion Queue register for the channel.

### 10.1.3.17 UDMA-P Receive Channel Pause

Setting the RX\_PAUSE bit in the Rx Channel N Realtime Control Register will suspend the channel from arbitration resulting in a halting of the flow of data. Clearing this bit will cause the channel to be added back into the arbitration list. Pausing a channel has no other destructive side effects (other than potential underrun/overflow conditions from a downstream data sink or upstream data source).

### 10.1.3.18 UDMA-P Receive Free Descriptor/Buffer Queue Setup (Host Packets)

The Host is ultimately responsible for providing all of the free descriptors and buffers that the port uses as it receives packets. For Rx Free Descriptor/Buffer queues, descriptors are allocated in a large block by the Host and then initialized and linked with buffers before being given matched pairs where the Host initializes the descriptor to point to its corresponding buffer before being pushed onto a Receive Free Descriptor/Buffer queue. Once the host has allocated both a descriptor and a buffer, it adds them to an Rx Free Descriptor/Buffer Queue by writing the pointer to the descriptor to the Rx Free Descriptor/Buffer Queue register D for the desired queue.

Before the Host adds each Rx buffer descriptor to the queue, it must first initialize the Rx buffer descriptor values as follows:

- Write the Original Buffer Pointer with the byte aligned address of the buffer data
- Write the Original Buffer Size
- Write the Return Queue Number field in the descriptor to appropriate values (required if auto-recycling is intended)
- Write the On Chip/ Off Chip indicator in the descriptor

Depending on the software architecture, the Host may also desire to write the return policy bit in the descriptor at this point as well. The DMA will not overwrite any of these listed fields (including return policy) during reception.

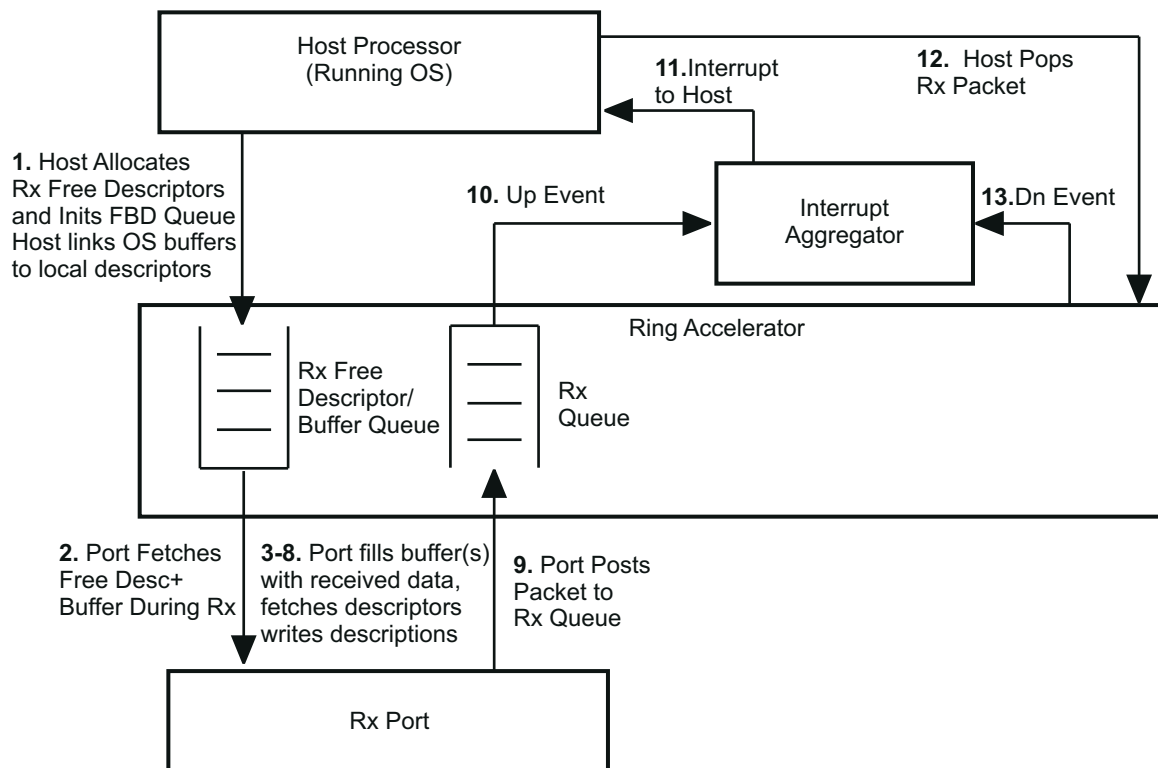
All other fields in the Descriptor do not need to be initialized as they will be overwritten by the Port on reception.

### 10.1.3.19 UDMA-P Receive FlowID Firewall Operation

Each incoming packet that enters the UDMA-P will either include an explicitly specified FlowID contained in the info words or it will have an implicitly specified default FlowID which is the same as the channel number. The Rx FlowID firewall checks the incoming FlowID on the received packet to verify that it is either the corresponding default FlowID for the channel or that the FlowID falls within the range specified in the Rx Channel Flow Range register. If the received flowID is not the channel number and it does not fall within the legal range as specified then the packet is dropped, the error is trapped in the Rx Flow ID Firewall Status register, and an optional output event is generated as specified in the Rx Flow ID Firewall Output Event Steering Register.

### 10.1.3.20 UDMA-P Receive Operation (Host Packet)

[Figure 10-11](#) shows a diagram of the overall Receive operation for Host flow mode.



navss\_012

**Figure 10-11. Host Packet Receive Operation**

When packet reception begins on a given channel, the port will begin by fetching the first descriptor + buffer from the Ring Accelerator using the Free Descriptor/Buffer Queue 0 Index that was initialized in the Rx Flow Table for the flow which is being used by the channel. If the SOP Buffer Offset in the Rx Flow Table entry is nonzero, then the port will begin writing data after the offset number of bytes in the SOP buffer. The port will then continue filling that buffer and will fetch additional descriptors + buffers as needed using the Free Descriptor/Buffer Queue 1, 2, and 3 indexes for the 2<sup>nd</sup>, 3<sup>rd</sup>, and remaining buffers in the packet.

A detailed summary is as follows:

1. Host allocates, populates, and places pointers to free descriptor/buffer structures onto Rx Free Descriptor/Buffer Queues
2. UDMA-P fetches packet descriptor pointer from RA
3. UDMA-P reads the packet descriptor to obtain the packet info, buffer pointer, and size
4. UDMA-P fills the buffer with received data
5. If packet is not complete, UDMA-P fetches next buffer descriptor pointer from RA
6. UDMA-P fills the buffer with received data

If the RX\_DESC\_TYPE field in the Rx Flow Register A is set to 0 (Normal Rx Host Mode), Steps 5-7 continue until all packet data is received at which point the Rx engine proceeds to step 8. If the RX\_DESC\_TYPE is set to 1 (Single Buffer Packet Host Mode) the Rx engine will proceed to step 8 skipping step 7.

7. UDMA-P writes previous buffer descriptor to memory. This includes the following fields:
  - a. Buffer information
  - b. Buffer Pointer with the byte aligned address of the chunk of buffer data
  - c. Buffer Length with the number of bytes in the chunk of buffer data
  - d. Pointer to next buffer descriptor in packet or 0 if this is the EOP buffer.

If the RX\_DESC\_TYPE field in the Rx Flow Register A is set to 0 (Normal Rx Host Mode), Steps 5-7 continue until all packet data is received at which point the Rx engine proceeds to step 8. If the

RX\_DESC\_TYPE is set to 1 (Single Buffer Packet Host Mode) the Rx engine proceeds to step 8 as soon as the first buffer is full. Any remaining data in the packet from the line is then received into one or more additional packets each exactly one Host buffer long.

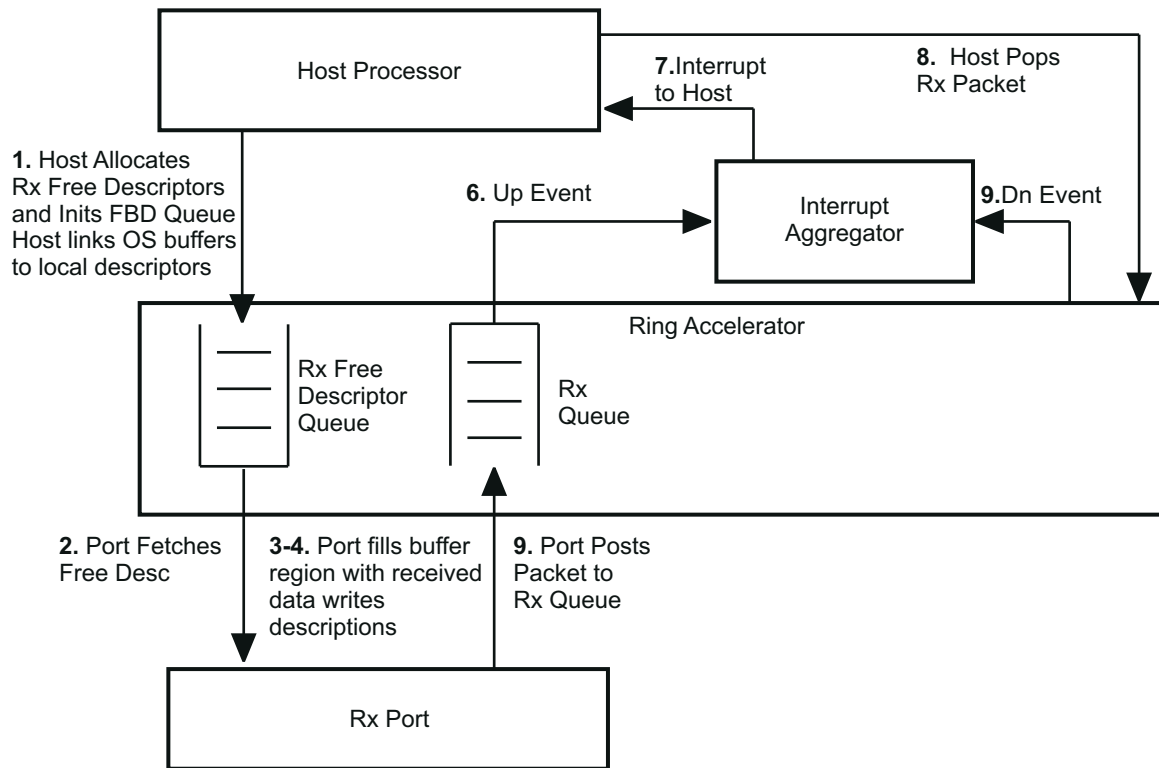
The UDMA-P performs the following operations when the entire packet has been received:

8. UDMA-P writes the packet descriptor to memory. This includes the following fields:
  - a. Descriptor Type (set to Host)
  - b. Packet Length indicating the total number of bytes that are to be read from all of the buffers for this packet.
  - c. Source Tag
  - d. Destination Tag (application specific)
  - e. Packet Type
  - f. Any protocol specific flags for the given packet type
  - g. SOP buffer information
    - i. Buffer Pointer with the byte aligned address of the chunk of buffer data
    - ii. Buffer Length with the number of bytes in the chunk of buffer data
  - h. Any protocol specific words that are required for the given packet type
  - i. Networking Stack Information (application specific and optional)
  - j. Pointer to next buffer descriptor in packet or 0 if this is the EOP buffer.
9. UDMA-P writes the packet descriptor pointer to the appropriate Rx Queue. The absolute Queue that each packet to be forwarded to on completion of reception will either be the Queue which that was specified in the rx\_dest\_qnum field in the Rx Flow Table entry or will be a queue which was provided via the PSI-L destination queue number override field.
10. Once the packet pointer is written to the Ring Accelerator it will send an up event to the Interrupt Aggregator
11. The Interrupt Aggregator upon receiving the up event will set the appropriate bit in the Interrupt Status Register indicated by the interrupt mapping table which will cause an interrupt to be asserted to the Host
12. The Host will pop the packet pointer from the Rx Queue
13. If the pop causes the queue to become empty, the Ring Accelerator will send a down event to the Interrupt Aggregator thus causing the associated bit to be cleared and also potentially clearing the interrupt to the Host.

The Ring Accelerator is responsible for indicating the status of the Receive Queues to other ports/embedded processors using out-of-band level sensitive status lines. These status lines are set anytime a queue is non-empty.

#### 10.1.3.21 UDMA-P Receive Operation (Monolithic Packet)

Figure 10-12 shows a diagram of the overall Receive operation for a monolithic descriptor type.



navss\_013

**Figure 10-12. Monolithic Packet Receive Operation**

The following sequence is a detailed summary of the Monolithic Packet Reception process:

1. The Host initializes free Descriptors and places them on an Rx Free Descriptor Queue
2. The UDMA-P fetches a single descriptor for the packet from the Ring Accelerator using the Monolithic Type Free Descriptor Queue Index that was initialized in the Rx Flow Table entry that is being used for the packet.
3. If the RX\_SOP\_OFFSET field in the Rx Flow Table entry is nonzero, then the port will begin writing data after the offset number of bytes in the payload portion of the monolithic descriptor. The port will then continue filling the payload portion of the descriptor until the entire packet has been received.  
The port performs the following operations when the entire packet has been received:
4. Writes the packet descriptor to memory. This includes the following fields:
  - a. Descriptor Type (set to Monolithic)
  - b. Packet Length indicating the total number of bytes in this packet.
  - c. Source Tag
  - d. Destination Tag (application specific)
  - e. Packet Type
  - f. Any protocol specific flags for the given packet type
  - g. Networking Stack Information (application specific and optional)
  - h. Any protocol specific words that are required for the given packet type
5. Writes the packet descriptor pointer to the appropriate Rx Queue. The absolute Queue that each packet to be forwarded to on completion of reception will either be the Queue which that was specified in the rx\_dest\_qnum fields in the Rx Flow Entry or can be overridden by the value in the PSI-L destination queue override field.
6. Once the packet pointer is written to the Ring Accelerator it will send an up event to the Interrupt Aggregator
7. The Interrupt Aggregator upon receiving the up event will set the appropriate bit in the Interrupt Status Register indicated by the interrupt mapping table which will cause an interrupt to be asserted to the Host
8. The Host will pop the packet pointer from the Rx Queue

9. If the pop causes the queue to become empty, the Ring Accelerator will send a down event to the Interrupt Aggregator thus causing the associated bit to be cleared and also potentially clearing the interrupt to the Host.

The Ring Accelerator is responsible for indicating the status of the Receive Queues to other ports/processors using out-of-band level sensitive status lines. These status lines are set anytime a queue is non-empty.

#### 10.1.3.22 UDMA Receive Operation (TR Packet)

Packet reception in TR Packet mode involves the following steps:

1. The Host allocates and populates a TR packet descriptor. The host will initialize the following fields within the packet descriptor:
  - a. Descriptor Type (set to TR)
  - b. Reload Enable to 1 if looping is required, otherwise 0
  - c. Reload Index to an appropriate offset if Reload Enable set, otherwise 0
  - d. Last Entry to TR count minus 1
  - e. TR Nominal Element Size to a value that is as large as required for any given TR in the buffer
  - f. Packet ID to a value that can be used to correlate the packet when it is placed back on the Tx completion queue
  - g. Source Tag
  - h. Destination Tag (application specific)
  - i. Descriptor reclamation policy fields indicating the mode and queue number for recycling the packet after transmission is complete.
  - j. A set of one or more valid Transfer Request records whose quantity matches the last index specified previously.
2. The Host queues the packet onto one of the Rx TR Queues for the desired UDMA channel. Channels may provide more than one Rx TR Queue and may provide a particular prioritization policy between the queues. This behavior is application specific and is controlled by the DMA controller/scheduler implementation.
3. The Ring Accelerator provides a level sensitive status signal for the queue which indicates if any packets are currently pending. This level sensitive status line is sent to the hardware block which is responsible for scheduling DMA operations.
4. The DMA controller is eventually brought into context for the corresponding channel and begins to process the packet.
5. The DMA controller reads the packet descriptor pointer from the RA.
6. The DMA controller reads the packet descriptor from memory
7. The DMA controller empties the data region in the descriptor by reading the contents in one or more nominal TR sized block data moves. These contents are then transmitted to an internal UTC.
8. If the UDMA includes an internal UTC, all of the data transfers specified in a TR will be completed as a series of writes (for single ended transfers) and a Transfer Response will be returned indicating the completion and status of the transfer.
9. The UDMA will wait until Transfer Responses have been returned for each Transfer Request that it issued. As each Transfer Response is returned, the UDMA will write the response into the TR buffer in the Transfer Response records array. Each response is written into an array index which directly matches the index of the request record to which it corresponds.
10. When all Transfer Requests in the packet have been processed and all Transfer Responses have been written back and confirmed to have landed in memory, the UDMA will write the pointer to the packet descriptor to the queue specified in the return queue number fields of the packet descriptor.
11. After the Packet Descriptor pointer has been written, the Ring Accelerator indicates the status of the Rx Completion Queues to other ports/processors/prefetcher blocks using events sent to the Interrupt Aggregator. These events are then converted into standard K3 interrupts.

#### 10.1.3.23 UDMA Receive Operation (Direct TR)

If the rings are configured in pass by value mode individual Transfer Request and Response records can be passed between software and hardware without any higher level data structures. This mode allows for

the UDMA to be prompted to do a transfer by just writing memory mapped locations with a source address, destination address and byte count (in the simplest case). Direct TR is accomplished as follows:

1. The Host writes a valid Transfer Request record to the Rx TR Queue of the desired UDMA channel.
2. The Ring Accelerator provides a level sensitive status signal for the queue which indicates if any packets are currently pending. This level sensitive status line is sent to the hardware block which is responsible for scheduling DMA operations.
3. The DMA controller is eventually brought into context for the corresponding channel and begins to process the packet.
4. The DMA controller reads the Transfer Request record from the ring via the RA in a single nominal TR sized block data move. These contents are then transmitted to an internal UTC (in the case of a UDMA-P).
5. All of the data transfers specified in a TR will be completed as a series of writes (for single ended transfers) and a Transfer Response will be returned indicating the completion and status of the transfer.
6. When a Transfer Response is returned for the Transfer Request record the UDMA will write the response into the next entry in the Transmit Completion Ring. Since the Transfer Responses come back from the UTC strongly ordered each response will be written into the ring strongly ordered as well.
7. After the Transfer Response has been written, the Ring Accelerator indicates the status of the Rx Completion Queues to other ports/processors/prefetcher blocks using events sent to the Interrupt Aggregator. These events are then converted into standard K3 interrupts.

#### 10.1.3.24 UDMA Receive Error/Exception Handling

Several types of errors/exceptions may be encountered during reception of data. The following table outlines the various errors and exceptions that can occur:

**Table 10-94. UDMA Rx Error/Exception Handling**

Condition	Channel Type	Severity	Pauses Channel	Error Flag Set	Data Transfer	Data Flush	TR Flush	Event Output
Descriptor Starvation	All Packet	Exception	No	No	Option	Option	-	No
Protocol Errors	All Packet	Info	No	No	Normal	No	-	No
Drop	Host – normal	Exception	No	No	Partial <sup>(2)</sup>	Until EOP	-	No
	Host – single buffer	Error	Selectable <sup>(1)</sup>	Yes	Partial <sup>(2)</sup>	Until EOP <sup>(3)</sup>	-	No
	Mono	Exception	No	No	Partial <sup>(2)</sup>	Until EOP	-	No
	UTC–single ended	Exception	No	No	Partial <sup>(2)</sup>	Until EOP	Until EOP <sup>(4)</sup>	Yes
	UTC- block copy	Illegal	-	-	-	-	-	-
EOL Asserted Prematurely	All Packet	Error	Selectable <sup>(1)</sup>	Yes	No	Yes	-	No
	UTC – single ended	Exception	No	No	Normal	No	No	Yes
	UTC – block copy	Illegal	-	-	-	-	-	-
EOP Asserted Prematurely (Short Packet)	UTC – single ended	Exception	No	No	Normal	No	Until EOP <sup>(4)</sup>	Yes
EOP Asserted Late (Long Packet)	UTC – single ended	Exception	No	No	Partial	Excess	No	Yes
TR null icnt0	All UTC	Error	Selectable <sup>(1)</sup>	Yes	No	No	Just that TR	No
Unsupported TR Type	All UTC	Error	Selectable <sup>(1)</sup>	Yes	No	No	Just that TR	No

(1) If PAUSE\_ON\_ERROR bit is set in channel configuration

(2) Data will not be transferred after drop is detected. All data after that point is guaranteed to be flushed and some data prior to the exact data phase on PSI-L where drop was asserted may also be flushed since data is constantly accumulated so it can be transferred in chunks.



- (3) EOP will only occur when in single buffer mode when teardown is initiated. The data is a (never ending) stream. Drop is never supposed to be asserted during this mode of operation.
- (4) EOP on TR is true for all PBV TRs, the final TR in a PBR, or if the user specified EOP in the TR flags

#### **10.1.3.24.1 Error Conditions**

The following sections describe items which fall under an error severity when encountered by the UDMA on Rx:

##### **10.1.3.24.1.1 Bus Errors**

If a bus error is encountered during receive the DMA will log the error by asserting the RX\_ERROR bit for the channel (and optionally sending an error event if enabled) but the DMA will continue to attempt to complete the transfer using the returned (and potentially incorrect) data.

##### **10.1.3.24.1.2 Null Icnt0 Error**

The icnt0 parameter in a TR is never allowed to be 0 as this would cause zero byte count transactions to be issued on the bus and it unproductive towards completion of any useful work. The Rx UTC engine in the UDMA will trap any TR for which the icnt0 is zero and will immediately return a transfer response with the TR icnt0 error conditions set without performing any data transfer and without outputting any events (including end of TR event).

##### **10.1.3.24.1.3 Unsupported TR Type**

If a TR is provided to a channel which is of a type which is not supported by the channel, the Rx UTC engine in the UDMA-P will trap that TR and immediately return a transfer response with the Unsupported TR error conditions set without performing any data transfer and without outputting any events (including end of TR event).

#### **10.1.3.24.2 Exception Conditions Exception Conditions**

There are a few cases which can occur when processing incoming packets which are not considered errors but which will require handling in order to avoid locking up the UDMA packet or TR mode write engines.

Packet mode channel exceptions include:

- Descriptor Starvation
- Protocol Errors
- Dropped Packets

Since the TR mode engine performs transfer sequencing using a count based mechanism it is susceptible to becoming out of synchronization if the source of the data and the destination for the data do not exactly match in their expectations for how much data will be transferred. The following sections outline these scenarios.

TR mode channel exceptions include:

- Reception of EOL delimiter
- Short Packets
- Long Packets

##### **10.1.3.24.2.1 Descriptor Starvation**

Descriptor starvation occurs when the Port attempts to fetch a free descriptor from an Rx Free Descriptor Queue or Rx Free Descriptor/Buffer Queue and none are available. When no descriptor is available, the Ring Accelerator will return a value of 0x0 for the descriptor pointer (Optionally, some ports may contain internal counters for the free queues which indicate how many elements are currently queued. In this case, no access to the RA is required and starvation is directly detected via a zero element count). When the port detects that descriptor starvation has occurred it will react based on the value of the RX\_ERROR\_HANDLING bit which was programmed into the Rx Flow N Configuration Register A.

- If the RX\_ERROR\_HANDLING bit is cleared:

The Port will assert a descriptor starvation indicator and will then return any descriptors that it has already used in the reception of the current packet back to the appropriate Free Descriptor or Free Descriptor/Buffer queue that they were fetched from. This is performed by writing the pointers for each descriptor to the Rx Free



Descriptor Queue N Register in sequence using the queue indexes in the Rx DMA state. The port may choose to implement a counter that counts the number of packets that were dropped due to descriptor starvation and may also desire to distinguish between whether the starvation occurred on the packet descriptor (SOP) or on a buffer starvation.

- If the `RX_ERROR_HANDLING` bit is set:

The Port will assert a descriptor starvation indicator (which may cause an interrupt to the Host) and will then behave as if it had not performed the current data block transfer which caused the starvation. The Port is required to pause its current processing saving the state of the transfer. The next time that the channel comes into context it will again try to allocate a descriptor with the intention being that if buffers/descriptors have been added that the operation will complete successfully on a subsequent retry.

#### **10.1.3.24.2.2 Protocol Errors**

Protocol errors occur when the port logic detects that the received packet did not pass protocol specific criteria that was checked during reception of the packet. Examples of protocol errors include packet length errors, CRC errors, or alignment errors.

When protocol errors are detected, the port may choose whether or not it will drop the packet. The mechanism that is used to determine which packets to drop and which to forward is application specific. If the port determines that it needs to forward the packet to the Host, it will set the Packet Error bit in the Descriptor indicating that this is a packet which experienced a protocol related error. The type of protocol related error is generally indicated in either the Protocol Specific bits in the Host or Monolithic descriptor or in the Protocol Specific bytes region. The packet length information and certain portions of the Protocol Specific region may not be correct for packets which encounter errors.

#### **10.1.3.24.2.3 Dropped Packets**

Packets can be dropped within an Rx Port for any number of reasons which with the exception of the starvation case which was covered above are application specific.

If a split TR mode channel receives instruction to drop a packet on the PSI-L interface, the UDMA/UTC will continue executing the TRs but without data transfer up until a TR is completed for which the EOP condition was set. Events will be generated as normal although data may not be actually transferred.

#### **10.1.3.24.2.4 Reception of EOL Delimiter**

If an end of line delimiter is received by the Rx (write) side of the UTC the `EOL_ADV` field in the TR application specific flags field is used to advance the appropriate indices of the TR to the next level. Each EOL that is received will cause the TR to advance to the next specified loop iteration breaking as many intermediate loops as are required.

#### **10.1.3.24.2.5 EOP Asserted Prematurely (Short Packet)**

Split TR mode channels can experience an incoming packet whose length is shorter than what was expected based on one or more TRs which form the control description for the expected packet. If an EOP is received for a TR and the TR has not been completely executed, the port is required to receive as much data as is actually provided in the incoming data placing it in accordance with the instructions in the TR and then to continue executing any remaining TR(s) such that the required events are produced to keep downstream consumers in sync. In this case, no data will be transferred to the buffers described by the TR beyond what was provided in the incoming packet.

#### **10.1.3.24.2.6 EOP Asserted Late (Long Packets)**

Split TR mode channels can experience an incoming packet whose length is longer than what was expected based on one or more TRs which form the control description for the expected packet. If a TR completes and is marked as EOP but the incoming packet is not finished, then the port must throw away any additional received data until the incoming packet reaches an EOP condition. At this point, reception will start with a new TR and a new incoming packet.

### 10.1.3.25 UTC Operation

A UTC accepts Transfer Request records from the UDMA-C and performs a sequence of transactions as specified in the TR record in accordance with the address generation algorithm that is specified in the Transfer Request format specification. The UTC will wait until an optional triggering event has occurred to load the channel state into one of its read or write engines so that the data movement operations can be completed. The UTC will complete the specified transactions until it either has transferred a pre-set number of bytes or until it has reached a point in the sequence where the TR indicates that a trigger is required to continue execution. When either of these two conditions is met, the UTC will save off the current state of the channel and will wait until the channel is re-triggered at which point the TR execution will continue. Once the entire Transfer Request specified sequence of data transfers has completed, the UTC will wait to ensure that all outstanding writes have landed at their respective destination endpoints and will then return a Transfer Response message back to the UDMA-C indicating the success or failure of the Transfer Request. The UTC processes Transfer Request operations on a given channel with strong ordering such that all Transfer Response operations will be returned in an order corresponding to the exact order that Transfer Request operations were issued.

### 10.1.3.26 UTC Receive Error/Exception Handling

Several types of errors/exceptions may be encountered during reception of data. The following table outlines the various errors and exceptions that can occur:

**Table 10-95. UTC Rx Error/Exception Handling**

Condition	Channel Type	Severity	Pauses Channel	Error Flag Set	Data Transfer	Data Flush	TR Flush	Event Output
Drop	UTC—single ended	Exception	No	No	Partial <sup>(2)</sup>	Until EOP	Until EOP <sup>(3)</sup>	Yes
	UTC- block copy	Illegal	-	-	-	-	-	-
EOL Asserted Prematurely	UTC – single ended	Exception	No	No	Normal	No	No	Yes
	UTC – block copy	Illegal	-	-	-	-	-	-
EOP Asserted Prematurely (Short Packet)	UTC – single ended	Exception	No	No	Normal	No	Until EOP <sup>(3)</sup>	Yes
EOP Asserted Late (Long Packet)	UTC – single ended	Exception	No	No	Partial	Excess	No	Yes
TR null icnt0	All UTC	Error	Selectable <sup>(1)</sup>	Yes	No	No	Just that TR	No
Unsupported TR Type	All UTC	Error	Selectable <sup>(1)</sup>	Yes	No	No	Just that TR	No

(1) If pause\_on\_error bit is set in channel configuration

(2) Data will not be transferred after drop is detected. All data after that point is guaranteed to be flushed and some data prior to the exact data phase on PSI-L where drop was asserted may also be flushed since data is constantly accumulated so it can be transferred in chunks.

(3) EOP on TR is true for all PBV TRs, the final TR in a PBR, or if the user specified EOP in the TR flags

#### 10.1.3.26.1 Error Handling

If an error occurs during the processing of a Transfer Request, the UTC will discontinue any further operations on the TR and will wait for all outstanding transactions to complete. At that point, the Transfer Response record will be returned to the UDMA-C with the appropriate codes indicating the type of failure as specified in the Transfer Response Format specification.

Depending on the value of the PAUSE\_ON\_ERROR bit in the UTC configuration register, the UTC will next do one of the following:

If the PAUSE\_ON\_ERROR bit is clear the channel will move on to the next Transfer Request record that it received from the UDMA-C.

If the PAUSE\_ON\_ERROR bit is set the channel will halt from processing any new Transfer Request records in order to allow the Host to examine the current channel state in order to determine the potential cause of the failure. The UTC is required to preserve as much of the channel state as possible as soon as a failure is determined. Note that the state will not necessarily provide exact details or a 'smoking gun' as the UTC implementation allows for large scale transaction pipelining and error status returns from the bus can occur well after the machine has moved on to other channel tasks.

#### **10.1.3.26.1.1 Null Icnt0 Error**

The icnt0 parameter in a TR is never allowed to be 0 as this would cause zero byte count transactions to be issued on the bus and it unproductive towards completion of any useful work. The Rx UTC engine in the UDMA will trap any TR for which the icnt0 is zero and will immediately return a transfer response with the TR icnt0 error conditions set without performing any data transfer and without outputting any events (including end of TR event).

#### **10.1.3.26.1.2 Unsupported TR Type**

If a TR is provided to a channel which is of a type which is not supported by the channel, the Rx UTC engine in the UDMA-P will trap that TR and immediately return a transfer response with the Unsupported TR error conditions set without performing any data transfer and without outputting any events (including end of TR event).

#### **10.1.3.26.2 Exception Conditions**

There are a few cases which can occur when processing TRs which are not considered errors but which will require handling in order to avoid locking up the UTC read or write engines. Since the UTC performs transfer sequencing using a count based mechanism it is susceptible to becoming out of synchronization if the source of the data and the destination for the data do not exactly match in their expectations for how much data will be transferred. The following outline these scenarios.

##### **10.1.3.26.2.1 Reception of EOL Delimiter**

If an end of line delimiter is received by the Rx (write) side of the UTC the EOL\_ADV field in the TR application specific flags field is used to advance the appropriate indices of the TR to the next level. Each EOL that is received will cause the TR to advance to the next specified loop iteration breaking as many intermediate loops as are required.

##### **10.1.3.26.2.2 EOP Asserted Prematurely (Short Packet)**

Split TR mode channels can experience an incoming packet whose length is shorter than what was expected based on one or more TRs which form the control description for the expected packet. If an EOP is received for a TR and the TR has not been completely executed, the port is required to receive as much data as is actually provided in the incoming data placing it in accordance with the instructions in the TR and the to continue executing any remaining TR(s) such that the required events are produced to keep downstream consumers in sync. In this case, no data will be transferred to the buffers described by the TR beyond what was provided in the incoming packet.

##### **10.1.3.26.2.3 EOP Asserted Late (Long Packets)**

Split TR mode channels can experience an incoming packet whose length is longer than what was expected based on one or more TRs which form the control description for the expected packet. If a TR completes and is marked as EOP but the incoming packet is not finished, then the port must throw away any additional received data until the incoming packet reaches an EOP condition. At this point, reception will start with a new TR and a new incoming packet.

## 10.2 Navigator Subsystem (NAVSS)

This chapter describes the features and functions of the Navigator Subsystem (NAVSS) hardware modules in the device.

The SoC device contains two NAVSSes:

- Main NAVSS (NAVSS0) – See [Section 10.2.1](#), *Main Navigator Subsystem (NAVSS)*
- MCU NAVSS (MCU\_NAVSS0) – See [Section 10.2.2](#), *MCU Navigator Subsystem (MCU NAVSS)*.

## 10.2.1 Main Navigator Subsystem (NAVSS)

This section describes the top-level integration of the main Navigator Subsystem (NAVSS) in the device.

### 10.2.1.1 NAVSS Overview

Main SoC Navigator Subsystem (NAVSS0) consists of DMA/Queue Management components – UDMA and Ring Accelerator (UDMASS), Peripherals (Module subsystem [MODSS]), Virtualization translation (VirtSS), and a North Bridge (NBSS).

**UDMASS** – UDMASS is the essential part of the *TI Data Movement Architecture* (see [Section 10.1](#)). UDMASS consists of:

- Unified DMA Controller – [Section 10.2.3](#)
- Ring Accelerator – [Section 10.2.4](#)
- Packet Streaming Interface (PSI-L) – [Section 10.2.8](#)

**MODSS** – Module subsystem is a collection of peripherals with different system-level functions, for example, interprocessor communication and time sync, among others. NAVSS0 contains the following modules:

- Mailbox – [Section 7.1](#)
- Spinlock – [Section 7.2](#)
- Two Timer Managers (Timer banks) – [Section 11.2](#)
- Time Stamp Module (CPTS) – [Section 11.1](#)
- Infrastructure components such as:
  - CBASS – *System Interconnect*
  - Proxies – [Section 10.2.5](#), [Section 10.2.6](#)
  - Interrupt aggregators – [Section 10.2.7](#)
  - Interrupt router – *NAVSS Interrupt Router Configuration*

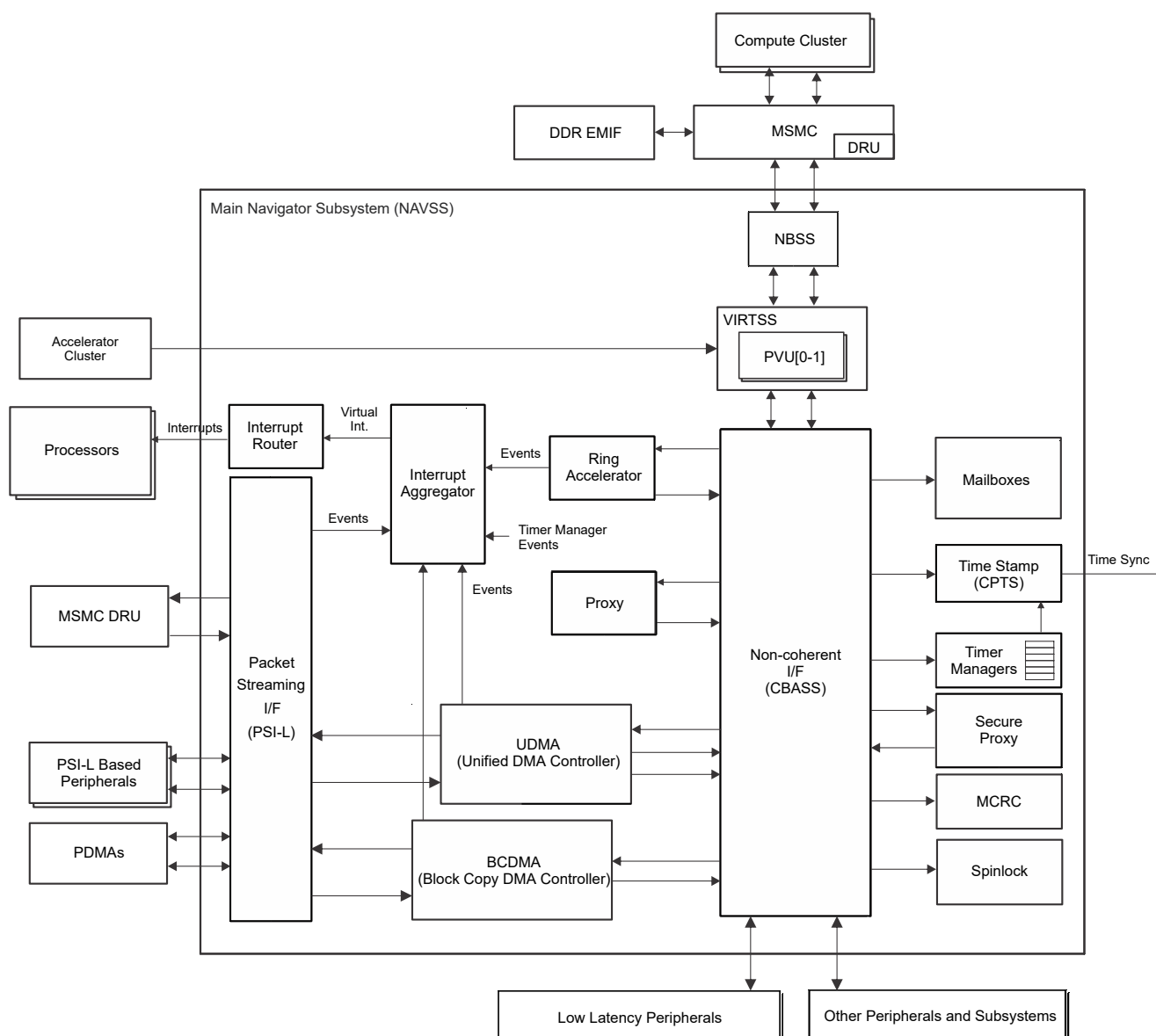
**NBSS** – North bridge – [Section 10.2.10](#)

**VirtSS** – Virtualization sub-system provides all the translation components supported in K3 for virtualization:

- Page-based Address Translator (PAT) –
- Peripheral Virtualization Units (PVU) – [Section 8.3.2](#)
- Translation look-aside Buffer Unit (TBU) –
- Translation Controller Unit (TCU) –

**ECC aggregators** – [Section 12.11.4](#)

NAVSS0 hardware components and their integration are shown on [Figure 10-13](#).



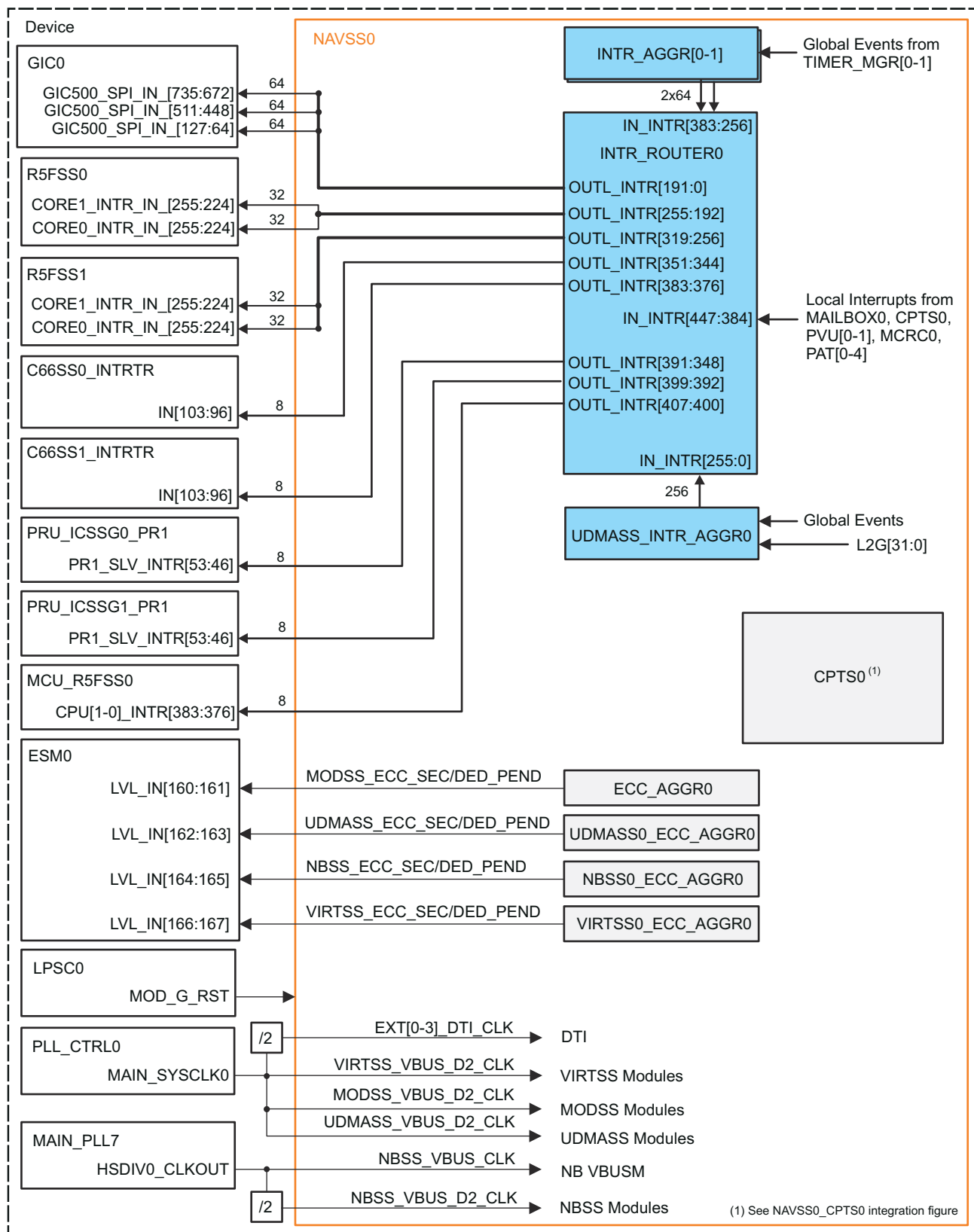
**Figure 10-13. NAVSS0 Top-Level Block Diagram**

### Note

MCU NAVSS functionality is similar to that of main NAVSS, however it is composed of fewer peripherals and has reduced bandwidth.

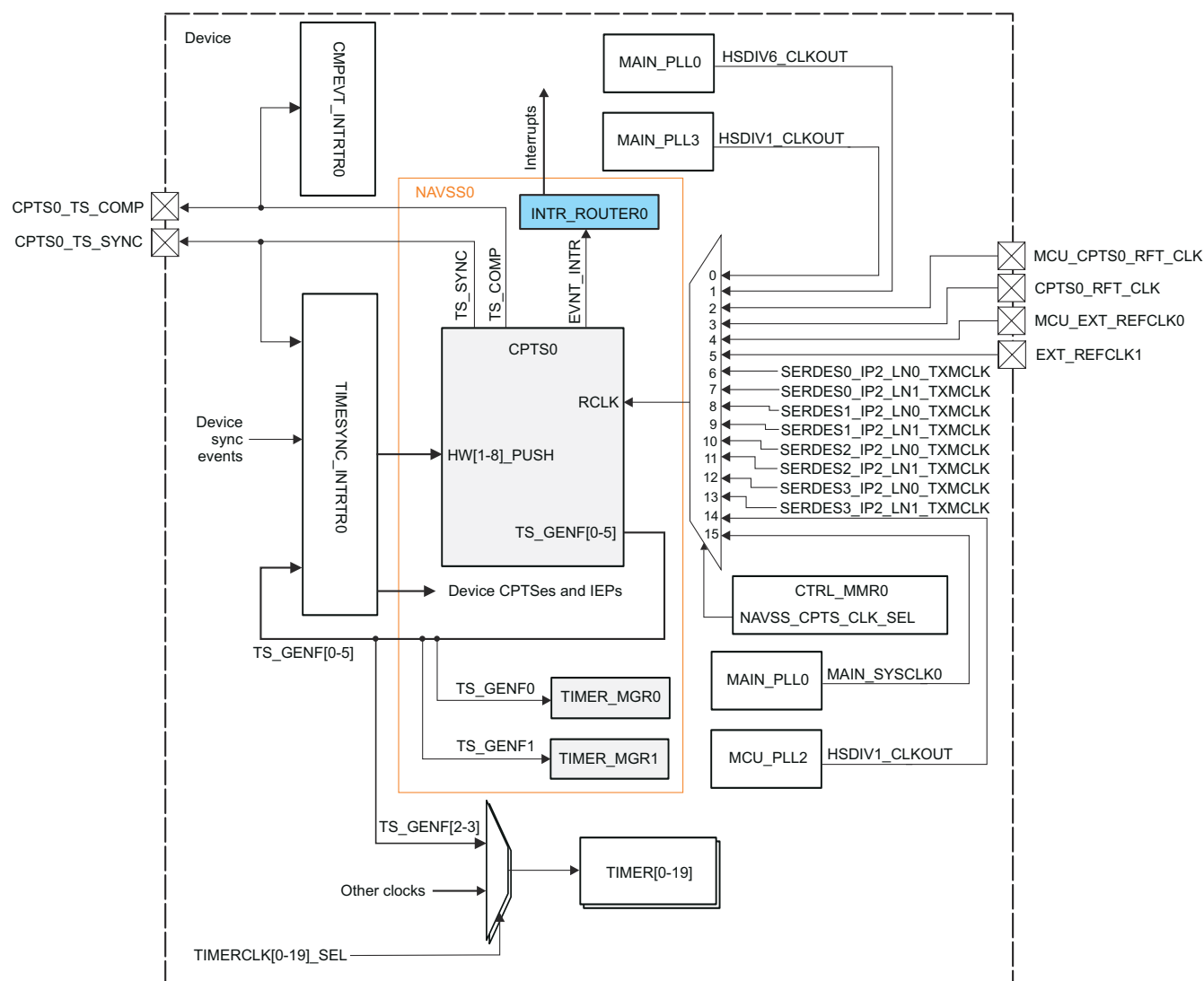
### 10.2.1.2 NAVSS Integration

Figure 10-14 to Figure 10-16 show the integration of the NAVSS0 in the device.



navss-005

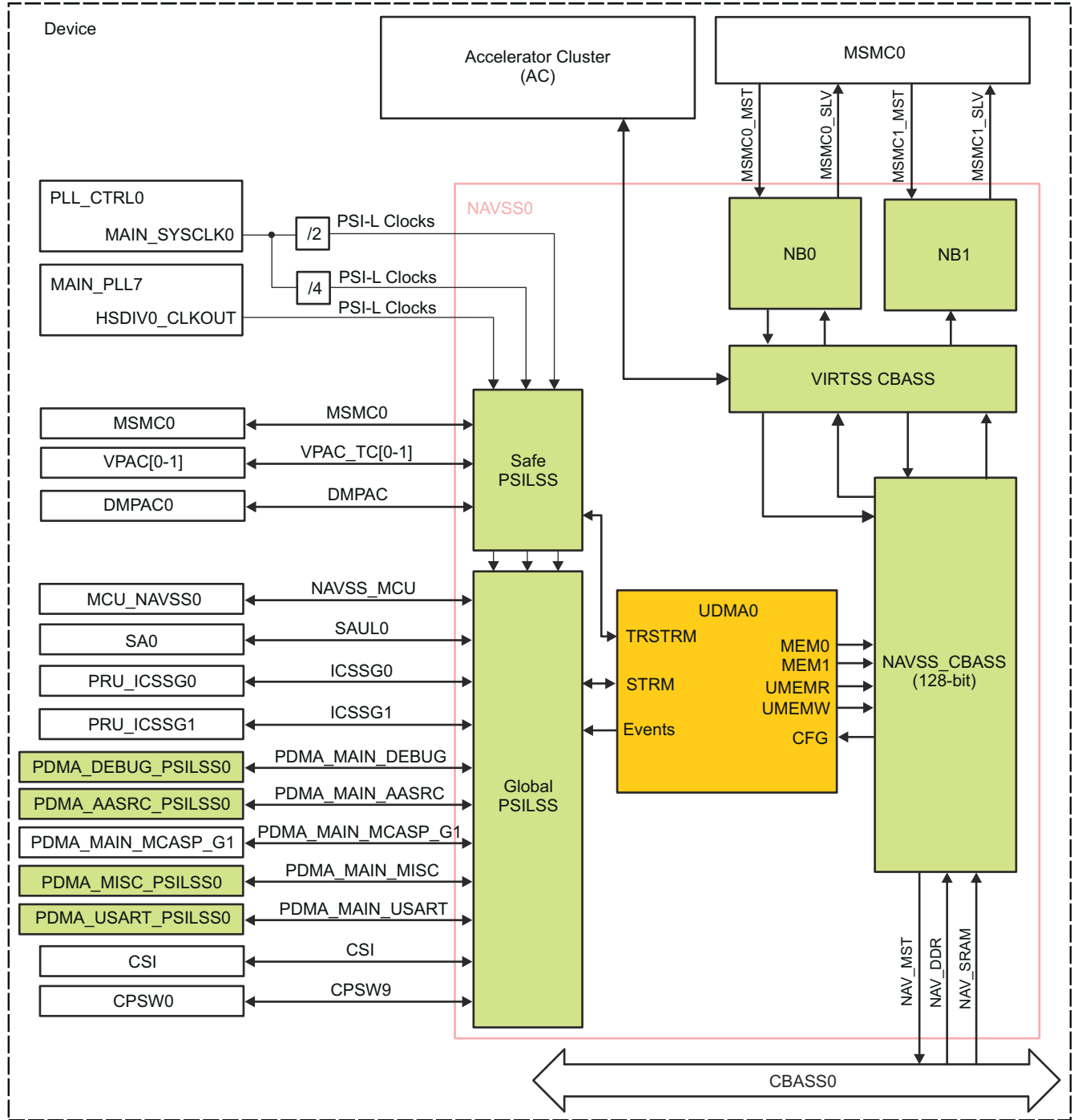
**Figure 10-14. NAVSS0 Clocks, Resets, and Interrupts**



cpts-002

**Figure 10-15. NAVSS0\_CPTS0 Integration**





navss-006

**Figure 10-16. NAVSS0 Interconnects**

Table 10-96 and Table 10-97 summarize the integration of the module in the device.

**Table 10-96. NAVSS0 Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect

**Table 10-96. NAVSS0 Integration Attributes (continued)**

NAVSS0	PSC0	GP	LPSC0	PSI-L CBASS0 VBUSP VBUSM.C CBASS0 VBUSM
--------	------	----	-------	--

**Table 10-97. NAVSS0 Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
NAVSS0_MODSS	MODSS_VBUS_D2_CLK	MAIN_SYSCLK0	PLL_CTRL0	MODSS config interface clock. This clock is used for most of the NAVSS modules (MODSS).
NAVSS0_UDMASS	UDMASS_VBUS_D2_CLK	MAIN_SYSCLK0	PLL_CTRL0	UDMASS config interface clock. This clock is used for UDMASS modules.
	MSMC0_CLK	MAIN_PLL7_HSDIV0_CLKOUT	MAIN_PLL7	MSMC0 PSI-L interface clock
	PDMA_MAIN_MISC_CLK	MAIN_SYSCLK0/4	PLL_CTRL0	PDMA_MAIN_MISC PSI-L interface clock
	PDMA_MAIN_USART_CLK	MAIN_SYSCLK0/4	PLL_CTRL0	PDMA_MAIN_USART PSI-L interface clock
	PDMA_MAIN_AASRC_CLK	MAIN_SYSCLK0/2	PLL_CTRL0	PDMA_MAIN_AASRC PSI-L interface clock
	PDMA_MAIN_DEBUG_CLK	MAIN_SYSCLK0/2	PLL_CTRL0	PDMA_MAIN_DEBUG PSI-L interface clock
	PDMA_MAIN_MCASP_G1_CLK	MAIN_SYSCLK0/2	PLL_CTRL0	PDMA_MAIN_MCASP_G1 PSI-L interface clock
	ICSS_G0_CLK	MAIN_SYSCLK0/2	PLL_CTRL0	ICSS_G0 PSI-L interface clock
	ICSS_G1_CLK	MAIN_SYSCLK0/2	PLL_CTRL0	ICSS_G1 PSI-L interface clock
NAVSS0_NBSS	NAVSS_MCU_CLK	MAIN_SYSCLK0	PLL_CTRL0	MCU_NAVSS PSI-L interface clock
	NBSS_VBUS_D2_CLK	MAIN_PLL7_HSDIV0_CLKOUT/2	MAIN_PLL7	NBSS config interface clock. This clock is used for NBSS modules (NB0-1, ECC_AGGRO).
NAVSS0_VIRTSS	NBSS_VBUS_CLK	MAIN_PLL7_HSDIV0_CLKOUT	MAIN_PLL7	NBSS VBUSM. This clock is used for NB0
	VIRTSS_VBUS_D2_CLK	MAIN_SYSCLK0	PLL_CTRL0	VIRTSS config interface clock.
	EXT[0-3]_DTI_CLK	MAIN_SYSCLK0/2	PLL_CTRL0	DTI clocks
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
NAVSS0_MODSS	MODSS_RST	MOD_G_RST	LPSC0	MODSS hardware reset
NAVSS0_UDMASS	UDMASS_RST	MOD_G_RST	LPSC0	UDMASS hardware reset
NAVSS0_NBSS	NBSS_RST	MOD_G_RST	LPSC0	NBSS hardware reset
NAVSS0_VIRTSS	VIRTSS_RST	MOD_G_RST	LPSC0	VIRTSS hardware reset

**Table 10-98. NAVSS0 Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
NAVSS0_MODSS and NAVSS0_UDMASS	INTR_ROUTER0_OUTL_IN TR[63:0]	GIC500_SPI_IN [127:64]	GIC0	INTR_PEND[191:0] interrupts to GIC SPI	Level
	INTR_ROUTER0_OUTL_IN TR[127:64]	GIC500_SPI_IN [511:448]			
	INTR_ROUTER0_OUTL_IN TR[191:128]	GIC500_SPI_IN [735:672]			

**Table 10-98. NAVSS0 Hardware Requests (continued)**

	INTR_ROUTER0_OUTL_IN TR[223:192]	CORE0_INTR_IN [255:224]	R5FSS0_CORE0	INTR_PEND[255:192] interrupts to main R5FSS0	Level
	INTR_ROUTER0_OUTL_IN TR[255:224]	CORE1_INTR_IN [255:224]	R5FSS0_CORE1		
	INTR_ROUTER0_OUTL_IN TR[287:256]	CORE0_INTR_IN [255:224]	R5FSS1_CORE0	INTR_PEND[319:256] interrupts to main R5FSS1	Level
	INTR_ROUTER0_OUTL_IN TR[319:288]	CORE1_INTR_IN [255:224]	R5FSS1_CORE1		
	INTR_ROUTER0_OUTL_IN TR[351:344]	C66SS0_INTRTR0_IN[103:9 6]	C66SS0_INTRTR 0	INTR_PEND[351:344] interrupts to C66SS0	Level
	INTR_ROUTER0_OUTL_IN TR[383:376]	C66SS0_INTRTR1_IN[103:9 6]	C66SS1_INTRTR 0	INTR_PEND[383:376] interrupts to C66SS1	Level
	INTR_ROUTER0_OUTL_IN TR[391:348]	PR1_SLV_INTR[53:46]	PRU_ICSSG0_PR 1	INTR_PEND[391:348] interrupts to PRU_ICSSG0 PR1	Level
	INTR_ROUTER0_OUTL_IN TR[399:392]	PR1_SLV_INTR[53:46]	PRU_ICSSG1_PR 1	INTR_PEND[399:392] interrupts to PRU_ICSSG1 PR1	Level
	INTR_ROUTER0_OUTL_IN TR[407:400]	CORE0_INTR_IN [383:376]  CORE1_INTR_IN [383:376]	MCU_R5FSS0	INTR_PEND[407:400] interrupts to MCU R5FSS0 CPU0 and CPU1	Level
	MODSS_ECC_SEC_PEND	ESM0_LVL_IN[160]	ESM0	SEC interrupt from MODSS ECC_AGGR0	Level
	MODSS_ECC_DED_PEND	ESM0_LVL_IN[161]	ESM0	DED interrupt from MODSS ECC_AGGR0	Level
	UDMASS_ECC_SEC_PEN D	ESM0_LVL_IN[162]	ESM0	SEC interrupt from UDMASS ECC_AGGR0	Level
	UDMASS_ECC_DED_PEN D	ESM0_LVL_IN[163]	ESM0	DED interrupt from UDMASS ECC_AGGR0	Level
NAVSS0_NBSS	NBSS_ECC_SEC_PEND	ESM0_LVL_IN[164]	ESM0	SEC interrupt from NBSS ECC_AGGR0	Level
	NBSS_ECC_DED_PEND	ESM0_LVL_IN[165]	ESM0	DED interrupt from NBSS ECC_AGGR0	Level
NAVSS0_VIRTSS	VIRTSS_ECC_SEC_PEND	ESM0_LVL_IN[166]	ESM0	SEC interrupt from VIRTSS ECC_AGGR0	Level
	VIRTSS_ECC_DED_PEND	ESM0_LVL_IN[167]	ESM0	DED interrupt from VIRTSS ECC_AGGR0	Level

**L2G Interrupt Request Inputs**

Module Instance	Module Interrupt Signal	Source Interrupt Input	Source	Description	Type
NAVSS0_UDMAS S	L2G_EVENT_PEND[7:0]	TIMESYNC_INTRTR0_OUT L [47:40]	TIMESYNC_INTR TR0	L2G interrupts from TIMESYNC_INTRTR0	Level
	L2G_EVENT_PEND[15:8]	CMPEVT_INTRTR0_OUTP [31:24]	CMPEVT_INTRTR 0	L2G interrupts from CMPEVT_INTRTR0	Level
	L2G_EVENT_PEND[31:16]	GPIOMUX_INTRTR0_OUTP [31:16]	GPIOMUX_INTRT R0	L2G interrupts from GPIOMUX_INTRTR0	Level

**DMA Events**

Module Instance	Module DMA Event	Destination DMA Event Input	Destination	Description	Type
NAVSS0	-	-	-	No PDMA channels to external DMA engines. See global event map in <a href="#">Table 10-100</a> .	-

**Time Sync Event Inputs**

Module Instance	Module Sync Input	Sync Source Signal	Source	Description	Type
-----------------	-------------------	--------------------	--------	-------------	------

**Table 10-98. NAVSS0 Hardware Requests (continued)**

NAVSS0_CPTS0	CPTS0_HW1_PUSH	TIMESYNC_INTRTR0_OUT L_0	TIMESYNC_INTR TR0	CPTS Asynchronous hardware timestamp 1 push input	Pulse
	CPTS0_HW2_PUSH	TIMESYNC_INTRTR0_OUT L_1	TIMESYNC_INTR TR0	CPTS Asynchronous hardware timestamp 2 push input	Pulse
	CPTS0_HW3_PUSH	TIMESYNC_INTRTR0_OUT L_2	TIMESYNC_INTR TR0	CPTS Asynchronous hardware timestamp 3 push input	Pulse
	CPTS0_HW4_PUSH	TIMESYNC_INTRTR0_OUT L_3	TIMESYNC_INTR TR0	CPTS Asynchronous hardware timestamp 4 push input	Pulse
	CPTS0_HW5_PUSH	TIMESYNC_INTRTR0_OUT L_4	TIMESYNC_INTR TR0	CPTS Asynchronous hardware timestamp 5 push input	Pulse
	CPTS0_HW6_PUSH	TIMESYNC_INTRTR0_OUT L_5	TIMESYNC_INTR TR0	CPTS Asynchronous hardware timestamp 6 push input	Pulse
	CPTS0_HW7_PUSH	TIMESYNC_INTRTR0_OUT L_6	TIMESYNC_INTR TR0	CPTS Asynchronous hardware timestamp 7 push input	Pulse
	CPTS0_HW8_PUSH	TIMESYNC_INTRTR0_OUT L_7	TIMESYNC_INTR TR0	CPTS Asynchronous hardware timestamp 8 push input	Pulse

**Time Sync Event Outputs**

Module Instance	Module Sync Output	Destination Sync Input	Destination	Description	Type
NAVSS0_CPTS0	CPTS0_TS_GENF0	TIMESYNC_INTRTR0_IN4	TIMESYNC_INTR TR0	CPTS Generation Function Output 0	Edge
		EON_TICK_EVT	TIMER_MGR0		
	CPTS0_TS_GENF1	TIMESYNC_INTRTR0_IN5	TIMESYNC_INTR TR0	CPTS Generation Function Output 1	Edge
		EON_TICK_EVT	TIMER_MGR1		
	CPTS0_TS_GENF2	TIMESYNC_INTRTR0_IN6	TIMESYNC_INTR TR0	CPTS Generation Function Output 2	Edge
		0xC	TIMER[0:19]_CLK MUX		
	CPTS0_TS_GENF3	TIMESYNC_INTRTR0_IN7	TIMESYNC_INTR TR0	CPTS Generation Function Output 3	Edge
		0xD	TIMER[0:19]_CLK MUX		
NAVSS0_CPTS0	CPTS0_TS_GENF4	TIMESYNC_INTRTR0_IN8	TIMESYNC_INTR TR0	CPTS Generation Function Output 4	Edge
	CPTS0_TS_GENF5	TIMESYNC_INTRTR0_IN9	TIMESYNC_INTR TR0	CPTS Generation Function Output 5	Edge
	CPTS0_TS_SYNC	TIMESYNC_INTRTR0_IN36	TIMESYNC_INTR TR0	CPTS Sync Output	Edge
		CPTS0_TS_SYNC	Pin		

**Compare Event Outputs**

Module Instance	Module Compare Output	Destination Compare Input	Destination	Description	Type
NAVSS0_CPTS0	CPTS0_TS_COMP	IN8	CMPEVT_INTRTR 0	CPTS Comparison Output	Edge
		CPTS0_TS_COMP	Pin		

#### 10.2.1.2.1 NAVSS Interrupt Router Configuration

The interrupt router is configured without an INTD (no\_intd = 1), it performs only interrupt muxing.

Table 10-99 shows the interrupt inputs listed from the higher order bits to the lower order bits.

**Table 10-99. NAVSS Interrupt Router Input Mapping**

NAVSS0_INTR_ROUTER0 Input(s)	Interrupt Mapping (MSB to LSB)
INRTR_IN[447]	PAT0_EXP_INTR
INRTR_IN[446]	PAT1_EXP_INTR
INRTR_IN[445]	PAT2_EXP_INTR
INRTR_IN[444]	PAT3_EXP_INTR
INRTR_IN[443]	PAT4_EXP_INTR
INRTR_IN[442]	IO_PVU0_EXP_INTR
INRTR_IN[441]	IO_PVU1_EXP_INTR
INRTR_IN[440]	DMA_PVU1_EXP_INTR
INRTR_IN[439:436]	MAILBOX0_CLUSTER0_PEND_INTR[3:0]
INRTR_IN[435:432]	MAILBOX0_CLUSTER1_PEND_INTR[3:0]
INRTR_IN[431:428]	MAILBOX0_CLUSTER2_PEND_INTR[3:0]
INRTR_IN[427:424]	MAILBOX0_CLUSTER3_PEND_INTR[3:0]
INRTR_IN[423:420]	MAILBOX0_CLUSTER4_PEND_INTR[3:0]
INRTR_IN[419:416]	MAILBOX0_CLUSTER5_PEND_INTR[3:0]
INRTR_IN[415:412]	MAILBOX0_CLUSTER6_PEND_INTR[3:0]
INRTR_IN[411:408]	MAILBOX0_CLUSTER7_PEND_INTR[3:0]
INRTR_IN[407:404]	MAILBOX0_CLUSTER8_PEND_INTR[3:0]
INRTR_IN[403:400]	MAILBOX0_CLUSTER9_PEND_INTR[3:0]
INRTR_IN[399:396]	MAILBOX0_CLUSTER10_PEND_INTR[3:0]
INRTR_IN[395:392]	MAILBOX0_CLUSTER11_PEND_INTR[3:0]
INRTR_IN[391]	CPTS0_EVNT_PEND_INTR
INRTR_IN[390]	Reserved
INRTR_IN[389]	Reserved
INRTR_IN[388]	MCRC0_MCRC_PEND_INTR
INRTR_IN[387:384]	MCRC0_EVENT_PEND_INTR[3:0]
INRTR_IN[383:320]	MODSS_INTA0_VINTR_PEND[63:0]
INRTR_IN[319:256]	MODSS_INTA1_VINTR_PEND[63:0]
INRTR_IN[255:0]	UDMASS_INTA0_VINTR_PEND[255:0]

#### Note

Interrupt router outputs are described in [Table 10-98, NAVSS0 Hardware Requests](#).

Interrupt router registers are described in [INTR0\\_INTR\\_ROUTER\\_CFG Registers](#).

### 10.2.1.2.2 Global Event Map

The global event map for all navigator events is shown in [Table 10-100](#).

**Table 10-100. Global Event Map**

	Destination	Offset	PSIL Routed Slots	Actual Slots
NAVSS or External	Port			
NAVSS0	UDMASS_INTR_AGGR0 SEVI <sup>(1)</sup>	0 (0k)	8192 (8k)	4608 (4.5k)
-	Reserved	8192 (8k)	-	-
MCU_NAVSS0	UDMASS_INTR_AGGR0 SEVI	16384 (16k)	2048 (2k)	1536 (1.5k)
external	DMSC_INTR_AGGR0 SEVI	18432 (18k)	2048 (2k)	-
NAVSS0	INTR_AGGR0 SEVI	20480 (20k)	2048 (2k)	1024 (1k)
NAVSS0	INTR_AGGR1 SEVI	22528 (22k)	1024 (1k)	1024 (1k)
-	Reserved	23552 (23k)	-	-
NAVSS0	UDMASS_INTR_AGGR0 MEVI <sup>(2)</sup>	32768 (32k)	2048 (2k)	512 (0.5k)
MCU_NAVSS0	UDMASS_INTR_AGGR0 MEVI	34816 (34k)	1024 (1k)	128 (0.125k)
-	Reserved	35840 (35k)	-	-
NAVSS0	UDMASS_INTR_AGGR0 GEVI <sup>(3)</sup>	36864 (36k)	2048 (2k)	512 (0.5k)
-	Reserved	38912 (38k)	-	-
MCU_NAVSS0	UDMASS_INTR_AGGR0 GEVI	39936 (39k)	1024 (1k)	256 (0.25k)
external	PDMA_MCU0 LEVI <sup>(4)</sup>	40960 (40.000k)	128	-
external	PDMA_MCU1 LEVI	41088 (40.125k)	128	-
external	PDMA_MCU2 LEVI	41216 (40.250k)	128	-
external	PDMA_ADC LEVI	41344 (40.375k)	128	-
external	PDMA_MAIN_DEBUG LEVI	41984 (41.000k)	128	-
external	PDMA_MAIN_AASRC LEVI	42112 (41.125k)	128	-
external	PDMA_MAIN_MCASP_G1 LEVI	42240 (41.250k)	128	-
external	PDMA_MAIN_MISC LEVI	42368 (41.375k)	128	-
external	PDMA_MAIN_USART LEVI	42496 (41.500k)	128	-
external	VPAC_TCO_CC	42624 (41.625k)	128	-
external	VPAC_TC1_CC	42752 (41.750k)	128	-
external	DMPAC_TCO_CC	42880 (41.875k)	128	-
external	CSI LEVI	43008 (42.000k)	128	1024 (1k)
-	Reserved	43264 (42.250k)	-	-
NAVSS0	UDMA Triggers	49152 (48k)	1024 (1k)	1024 (1k)
MCU_NAVSS0	UDMA Triggers	56320 (55k)	256 (0.25k)	256 (0.25k)
external	MSMC DRU	61440 (60k)	256 (0.25k)	64 (0.625k)
-	Reserved for MSMC DRU	61696 (60.25k)	-	-

- (1) SEVI – Mappable source input
- (2) MEVI – Multicast event input
- (3) GEVI – Global counting event input
- (4) LEVI – Local event input

### 10.2.1.2.3 PSI-L System Thread Map (All NAVSS)

Endpoints in MCU\_NAVSS0 can be paired with NAVSS0 and vice versa as shown in [Table 10-101](#) (source/destination pairs). The configuration proxy in each subsystem can configure any endpoint in the map on either subsystem without restriction.

**Table 10-101. System Thread Map for All PSI-L Threads**

Thread Number	NAVSS Instance	Endpoint
0x0000	NAVSS0	Configuration Proxy
0x0001	MCU_NAVSS0	Configuration Proxy
0x0002-0x0007	-	Reserved
0x0008	NAVSS0	UDMAP0 TRSTRM
0x0009-0x001F	NAVSS0	Reserved for future UDMA TRSTRM instances
0x0020	NAVSS0	UDMAP0 CFGSTRM
0x0021-0x003F	-	Reserved for future UDMA CFGSTRM instances
0x0040-0x0FFF	-	Reserved
0x1000-0x3FFF	NAVSS0	UDMAP0 Threads
0x4000-0x40FF	NAVSS0	SAUL0 (SA0)
0x4100-0x41FF	NAVSS0	PRU_ICSSG0
0x4200-0x42FF	NAVSS0	PRU_ICSSG1
0x4300-0x43FF	NAVSS0	MAIN_PDMA_DEBUG (to PDMA_DEBUG_PSILSS0)
0x4400-0x44FF	NAVSS0	MAIN_PDMA_AASRC (to PDMA_AASRC_PSILSS0)
0x4500-0x45FF	NAVSS0	MAIN_PDMA_MCASP_G1 (to PDMA_MAIN_MCASP_G1 directly, not through a PSILSS)
0x4600-0x46FF	NAVSS0	MAIN_PDMA_MISC (to PDMA_MISC_PSILSS0)
0x4700-0x47FF	NAVSS0	MAIN_PDMA_USART (to PDMA_USART_PSILSS0)
0x4800-0x48FF	NAVSS0	MSMC0
0x4820-0x483F	NAVSS0	VPAC TC0
0x4840-0x487F	NAVSS0	VPAC TC1
0x4880-0x48FF	NAVSS0	DMPAC
0x4900-0x49FF	NAVSS0	CSI
0x4A00-0x4AFF	NAVSS0	CPSW9 (9-port CPSW0)
0x4B00-0x5FFF	-	Reserved
0x6000-0x6FFF	MCU_NAVSS0	UDMAP0 Threads
0x7000-0x70FF	MCU_NAVSS0	CPSW0 (2-port MCU_CPSW0)
0x7100-0x71FF	MCU_NAVSS0	MCU_PDMA0
0x7200-0x72FF	MCU_NAVSS0	MCU_PDMA1
0x7300-0x73FF	MCU_NAVSS0	MCU_PDMA2
0x7400-0x74FF	MCU_NAVSS0	MCU_PDMA_ADC
0x7500-0x75FF	MCU_NAVSS0	SAUL0 (SA0)
0x7600-0x7FFF	-	Reserved

### 10.2.1.2.4 NAVSS VBUSM Route ID Table

The VBUSM Master routelds are listed in [Table 10-102](#).

**Table 10-102. VBUSM Route IDs**

VBUSM Interface	Routeld(s)	OrderId(s)
MSMC0_MST	0-127	8-15
MSMC1_MST	255	0-15
NAV_DDR0		

**Table 10-102. VBUSM Route IDs (continued)**

VBUSM Interface	RouteId(s)	OrderId(s)
NAV_DDR1		
NAV_SRAM0		
NAV_SRAM1		
PROXY0	160	0-15
SEC_PROXY0	216	0-15
RINGACC0.DST	200	0-15
UDMA0.MEM0	168	0-15
UDMA0.MEM1	169	0-15
UDMA0.UMEMW	170	0-15
UDMA0.UMEMR	171	0-15

#### 10.2.1.2.5

#### Note

For more information on the interconnects, see [Chapter 3](#), *System Interconnect*.

For more information on the power, reset, and clock management, see the corresponding sections within [Chapter 5](#), *Device Configuration*.



### 10.2.1.3 NAVSS Functional Description

See [Section 10.1](#), *DMA Architecture*, for the *TI K3 Data Movement (DMA) Architecture* specification.

See the following sections for the respective module description:

UDMASS:

- Unified DMA Controller – [Section 10.2.3](#)
- Ring Accelerator – [Section 10.2.4](#)
- Packet Streaming Interface (PSI-L) – [Section 10.2.8](#)

MODSS:

- Mailbox – [Section 7.1](#)
- Spinlock – [Section 7.2](#)
- Two Timer Managers (Timer banks) – [Section 11.2](#)
- Time Stamp Module (CPTS) – [Section 11.1](#)
- Infrastructure components:
  - CBASS – *System Interconnect*
  - Proxies – [Section 10.2.5](#)
  - Interrupt aggregators – [Section 10.2.7](#)
  - Interrupt router – *NAVSS Interrupt Router Configuration*

NBSS – North bridge – [Section 10.2.10](#)

ECC aggregators – [Section 12.11.4](#)

**VirtSS** – Virtualization sub-system provides all the translation components supported in K3 for virtualization:

- Page-based Address Translator (PAT) –
- Peripheral Virtualization Units (PVU) – [Section 8.3.2](#)
- Translation look-aside Buffer Unit (TBU) –
- Translation Controller Unit (TCU) –

### 10.2.1.4 NAVSS Interrupt Configuration

This section describes the actions needed to configure interrupts within the NAVSS. The information provided is applicable to either the NAVSS or MCU\_NAVSS subsystems.

Table 10-103 are module-specific hardcoded parameters required to properly configure the NAVSS interrupt support.

**Table 10-103. NAVSS Parameters**

Parameter	Value NAVSS0	Value MCU_NAVSS0	Description
RA_RING_CNT	1024	286	Number of total rings supported
IA_SEVI	4608	1536	UDMA Interrupt Aggregator Source Event Input (SEVI) count
IA_VINTR	256	256	UDMA Interrupt Aggregator Virtual Interrupt (VINTR) count
IR_IBASE	See NAVSS Interrupt Router Input Mapping	See Interrupt Router Input Mapping	Interrupt Router input interrupt base number for given module (hardcoded values in interrupt router)
EO	See Global Event Map	See Global Event Map	Event offset (hardcoded offsets in NAVSS)

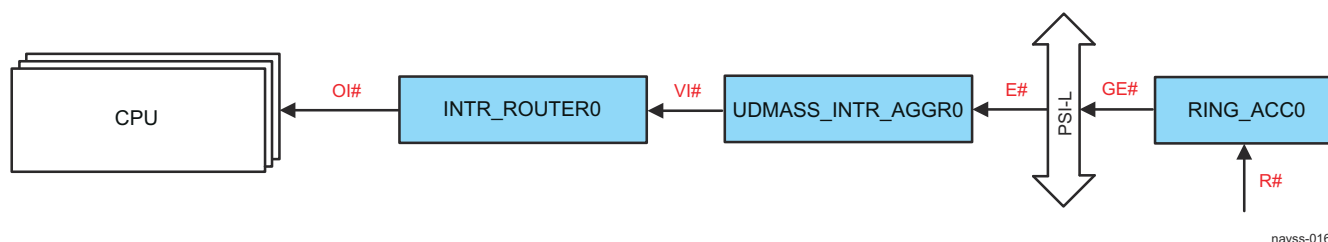
Table 10-104 lists software-configurable variables used in the examples and descriptions.

**Table 10-104. NAVSS Software Variables**

Variable	Valid Range	Description
R#	0 – RA_RING_CNT-1	Ring number
E#	0 – (destination module specific)	Event number
GE#	= EO + E#	Global event number
SB#	0 – IA_SEVI-1	Interrupt aggregator status bit number
VI#	0 – IA_VINTR-1	Virtual interrupt number
OI#	0 – 407	Interrupt router output CPU interrupt number. See NAVSS Hardware Requests

#### 10.2.1.4.1 NAVSS Event and Interrupt Flow

Figure 10-17 illustrates the event and interrupt flow within the NAVSS and output interrupts to a CPU. Information that flows between NAVSS modules is also indicated (in this case, Global Event (GE) information flows between ring accelerator and the PSILSS).



**Figure 10-17. NAVSS Interrupt Flow**

#### 10.2.1.4.1.1 NAVSS Interrupts Description

This section describes the interrupt-related information that flows within NAVSS and how this information is configured or generated.

##### 1. Ring Number (R#)

- Data is read from and written to a specific Ring Accelerator ring using direct ring-mode, proxy, or secure proxy accesses
- The Ring Accelerator has specific ring number ranges for specific purposes. The R# must match the intended purpose of the ring (see *RINGACC Ring Mapping*)

## 2. Global Event and Event Numbers (GE# and E#)

- $GE\# = EO + E\#$
- EO determines the destination module for the event. The PSILSS will route the GE# to the destination module per the NAVSS event mapping, removing EO in the process. The destination module sees only E#. See *Global event Map* for details. Examples:
  - GE# in the range 0x0000 – 0x1FFF (EO = 0x0000, E# = 0x0000 – 0x1FFF) would go to NAVSS0\_UDMASS\_INTR\_AGGR0. The INTR\_AGGR sees E#.
  - GE# in the range 0x4000 – 0x47FF (EO = 0x4000, E# = 0x0000 – 0x07FF) would go to MCU\_NAVSS0\_UDMASS\_INTR\_AGGR0. The INTR\_AGGR sees E#.
- For the INTR\_AGGR module, E# ranges from 0 to (IA\_SEVI - 1)

## 3. Virtual Interrupt (VI#)

- The INTR\_AGGR maps E# (via software configuration) to any SB#
- SB# ranges from 0 to (IA\_SEVI - 1)
- Each VI# (0 – (IA\_VINTR-1)) is driven by a group of 64 contiguous SB# numbers (VI# driven by SB#'s  $VI\# \times 64 - (VI\# \times 64) + 63$ )

## 4. Output Interrupt (OI#)

- The Interrupt Router has a hardwired set of interrupt inputs from various modules (see *NAVSS Interrupt Router Input Mapping*). Examples:
  - In the NAVSS, 256 interrupt inputs from the NAVSS0\_UDMASS\_INTR\_AGGR0 are hardwired to interrupt inputs 0 – 255. The VI# is zero-relative to this range.
  - In the MCU\_NAVSS, 4 event interrupt inputs from the MCRC0 are hardwired to interrupt inputs 256 – 259. The VI# is zero-relative to this range.
- Software configures an interrupt input (based on  $VI\# + IR\_IBASE$ ) to an OI#

### 10.2.1.4.1.2 Application Example

This section uses the following example to illustrate how to configure the NAVSS interrupt support for a specific use-case:

Using NAVSS0, use A72 MPU interrupt 7 for UDMA0 receive ring 16 (assumptions: use up/down event 10, NAVSS0\_UDMASS\_INTR\_AGGR0 for interrupt aggregation, and virtual interrupt 3).

Table 10-105 lists the modules' hardcoded parameters used for this example:

**Table 10-105. NAVSS Parameters for the Application Example**

Parameter	Value NAVSS0	From Table	Description
RA_UDMAP0_RX	300	<i>RINGACC Ring Mapping</i>	Starting ring number for UDMA0 receive channels
RA_RING_CNT	1024	<i>NAVSS0_RINGACC0 Configuration Parameters</i>	Number of total rings supported
IA_SEVI	4608	<i>Interrupt Aggregators Parameters</i>	NAVSS0_UDMASS_INTR_AGGR0 Steerable Event Input (SEVI) count
IA_VINTR	256	<i>Interrupt Aggregators Parameters</i>	NAVSS0_UDMASS_INTR_AGGR0 Virtual Interrupt (VINTR) count
IR_IBASE	0	<i>NAVSS Interrupt Router Input Mapping</i>	Interrupt-Router-input-interrupt base for NAVSS0_UDMASS_INTR_AGGR0
EO	0	<i>Global event Map</i>	Event offset. Destination is NAVSS0_UDMASS_INTR_AGGR0
CC_IBASE	0	<i>NAVSS0 Hardware Requests</i>	Interrupt-requests-to-CC base

Table 10-106 the variables and their respective calculations for this example.

**Table 10-106. NAVSS Software Variables for the Application Example**

Variable	Calculations/Value	Description
R#	$= (RA\_UDMAP0\_RX + 16)$ $= (300 + 16)$ $= 316$	Receive ring 16 translates to ring number
E#	10	Zero-relative event number from PSILSS to NAVSS0_UDMASS_INTR_AGGR0
GE#	$= (EO + E\#)$ $= (0 + 10)$ $= 10$	Global event number (event 10 to be sent to NAVSS0_UDMASS_INTR_AGGR0)
VI#	= 3	VI# = 3 is per example assumptions. VI# could get any value 0 – IA_VINTR-1.
SB#	= 200	Must be in the range $(VI\# \times 64) - ((VI\# \times 64) + 63) = 192 - 255$ in order to drive VI# 3. We will use 200 for this example.
OI#	$= (A72\_IBASE + 7)$ $= (0 + 7)$ $= 7$	OI# = 7 is per example assumptions. OI# could get any value 0 – 119 to drive an A72 interrupt.

Table 10-107 shows software register writes that will configure the NAVSS to produce the desired interrupt per the example description.

**Table 10-107. NAVSS Register Writes for the Application Example**

Module	Register	Value to Write	Description
RINGACC0	RING[R#]_EVT (EVENT_j)	RING[316]_EVT $= GE\#$ $= 10$	Ring event number must be specified as a global event number
INTR_AGGR0	ENTRY[E#]_INTMAP (IMAP_j)	ENTRY[10]_INTMAP $= (((SB\#/64) \& 0x1FF) < 8)   (SB\ \% 64)$ $= (3 < 8)   8$ $= 0x308$	regnum = SB# / 64 bitnum = SB# % 64
INTR_AGGR0	VINT[SB#/ 64]_ENABLE_SET (ENABLE_SET_j)	VINT[3]_ENABLE_SET $= 1 < (SB\ \% 64)$ $= 1 < 8$ $= 0x100$	Enable interrupt
INTR_ROUTER0	INTERRUPT_CONTR OL[OI#] (MUXCNTL_y)	INTERRUPT_CONTROL[7] $= IR\_IBASE + VI\#$ $= 0 + 3$ $= 3$	Route interrupt

## 10.2.2 MCU Navigator Subsystem (MCU NAVSS)

This chapter describes the integration of the MCU Navigator Subsystem (MCU NAVSS) in the device and internal module parameters.

### 10.2.2.1 MCU NAVSS Overview

MCU Navigator Subsystem (MCU NAVSS) has a subset of the modules of the main NAVSS and is instantiated in the MCU domain.

MCU Navigator Subsystem consists of DMA/Queue Management components – UDMA and Ring Accelerator (UDMASS), and Peripherals (Module subsystem [MODSS]).

**UDMASS** – UDMASS is the essential part of the TI Data Movement Architecture. UDMASS consists of:

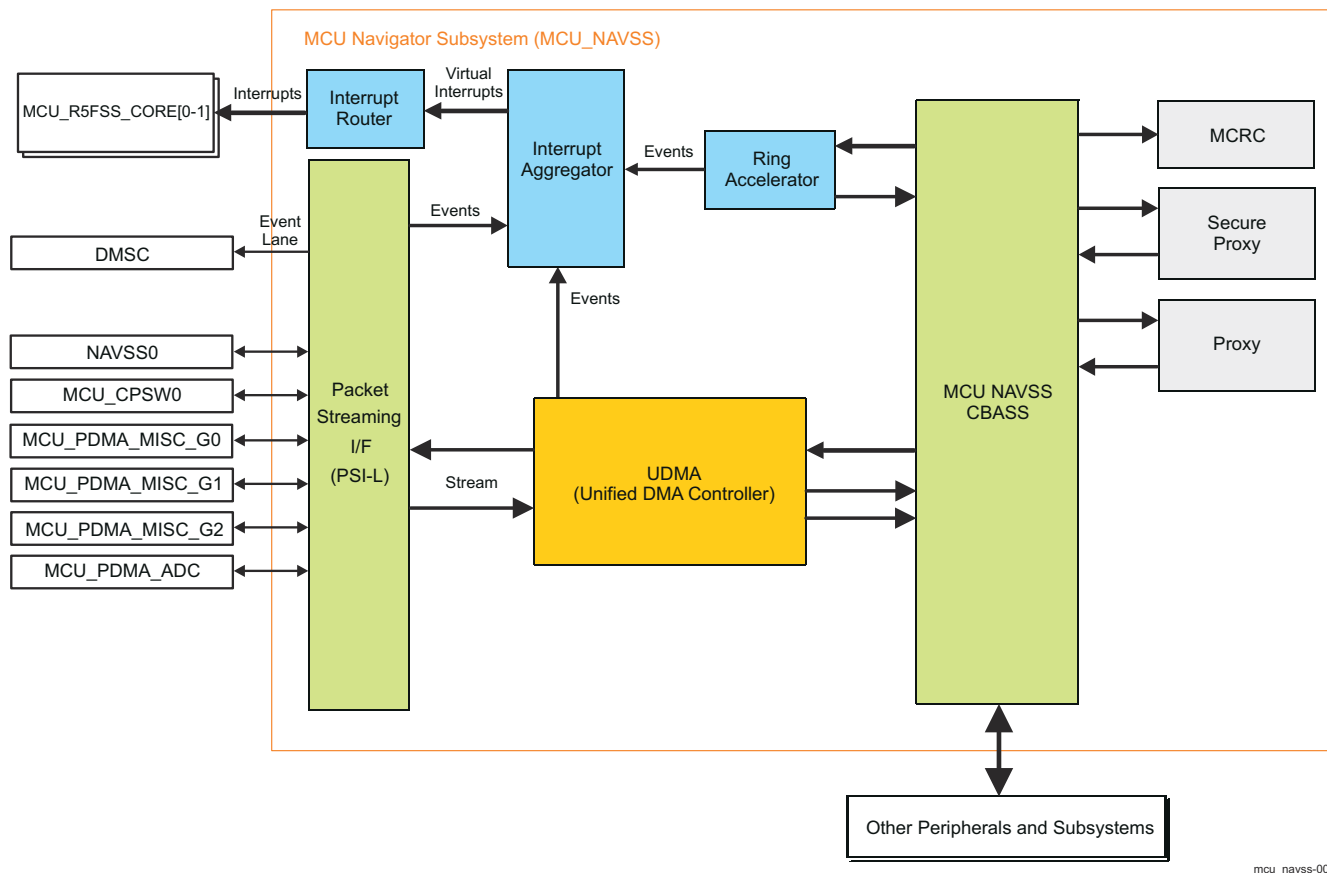
- Unified DMA Controller
- Ring Accelerator
- Packet Streaming Interface (PSILSS)

**MODSS** – MODSS is a collection of peripherals with different system-level functions. NAVSS0 contains the following modules:

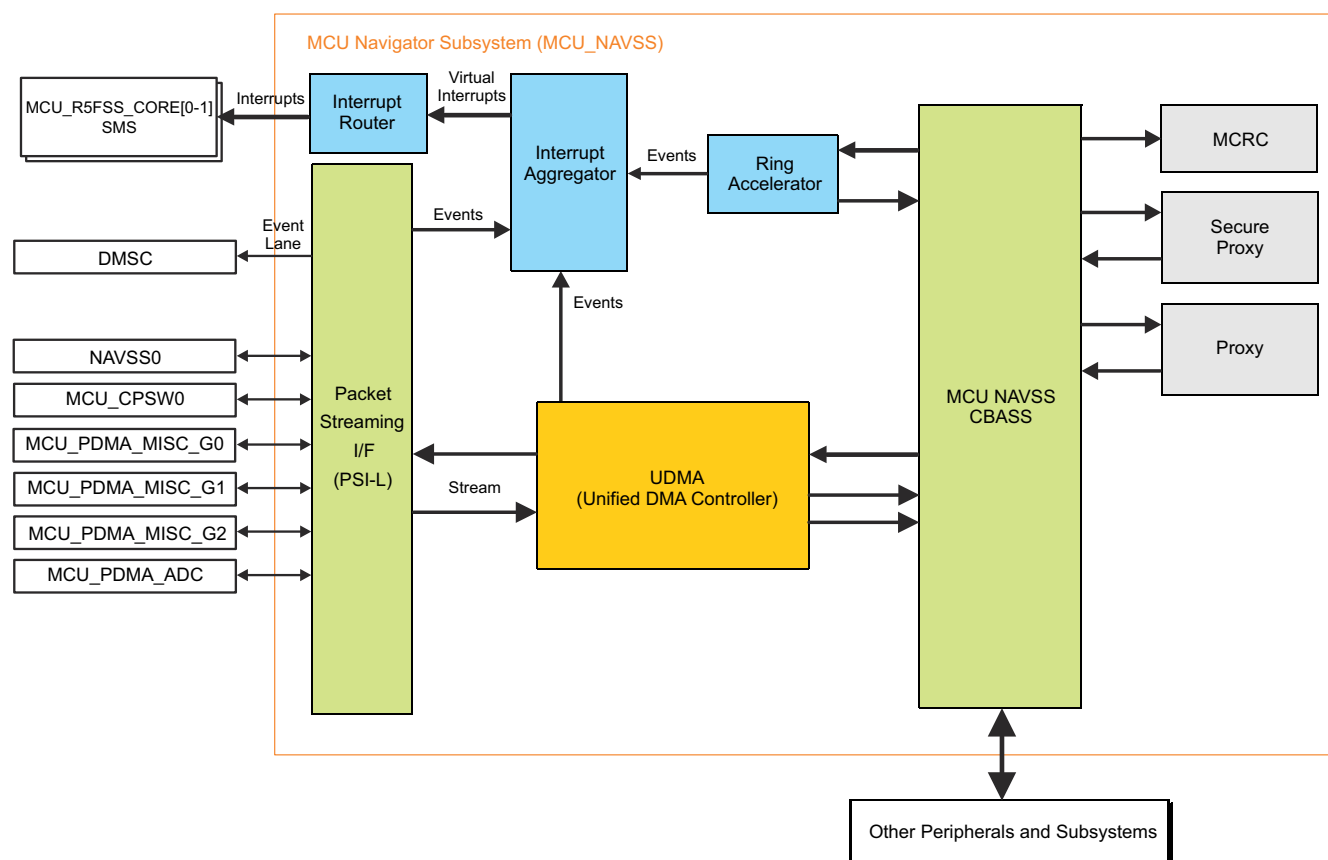
- Memory CRC module
- Infrastructure components such as CBASS, proxies, interrupt aggregators, and an interrupt router

**ECC aggregators** – for SEC/DED memory protection.

Figure 10-18 shows the MCU\_NAVSS0 top-level block diagram.



mcu\_navss-001



mcu\_navss-001

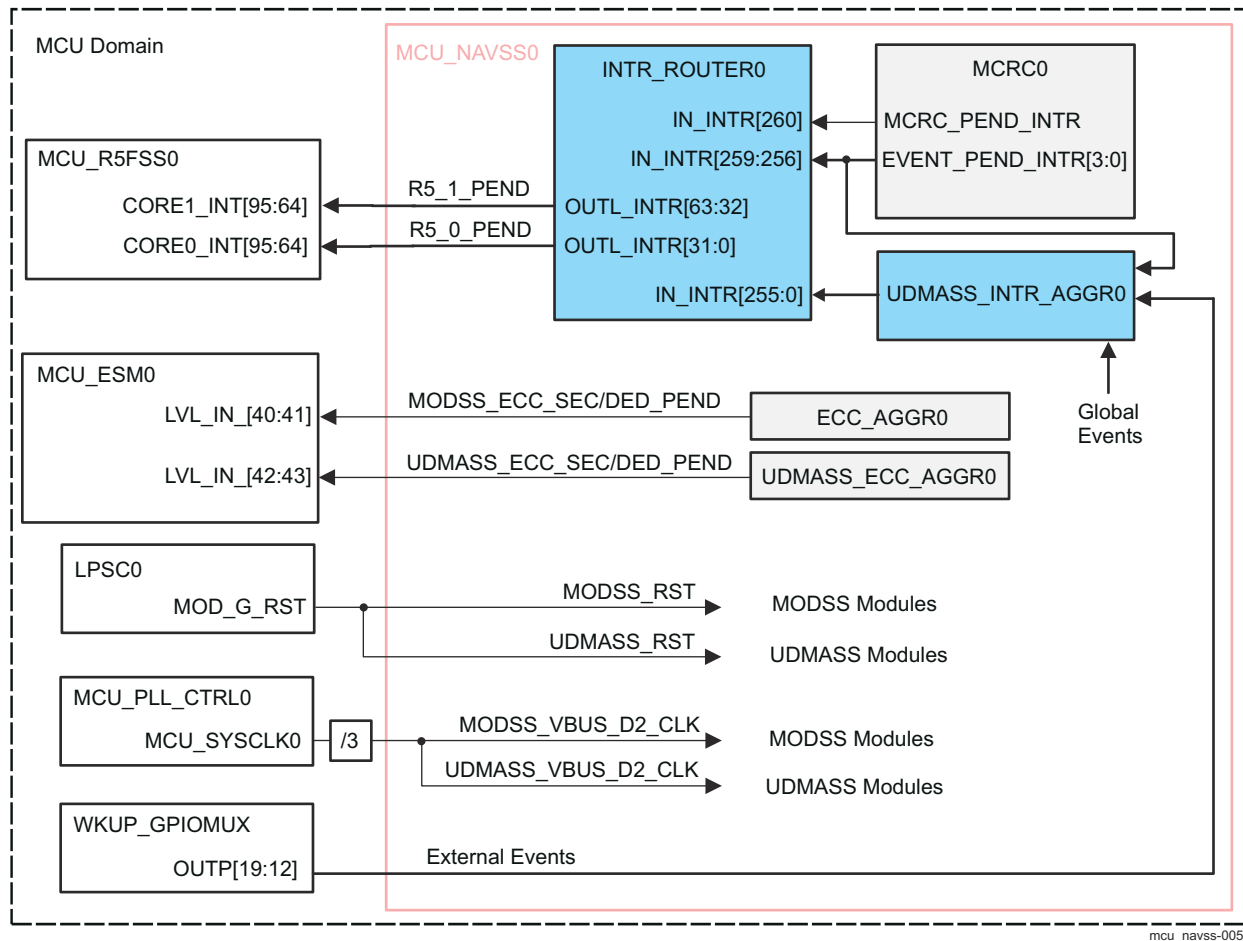
**Figure 10-18. MCU NAVSS Top-Level Block Diagram**

### Note

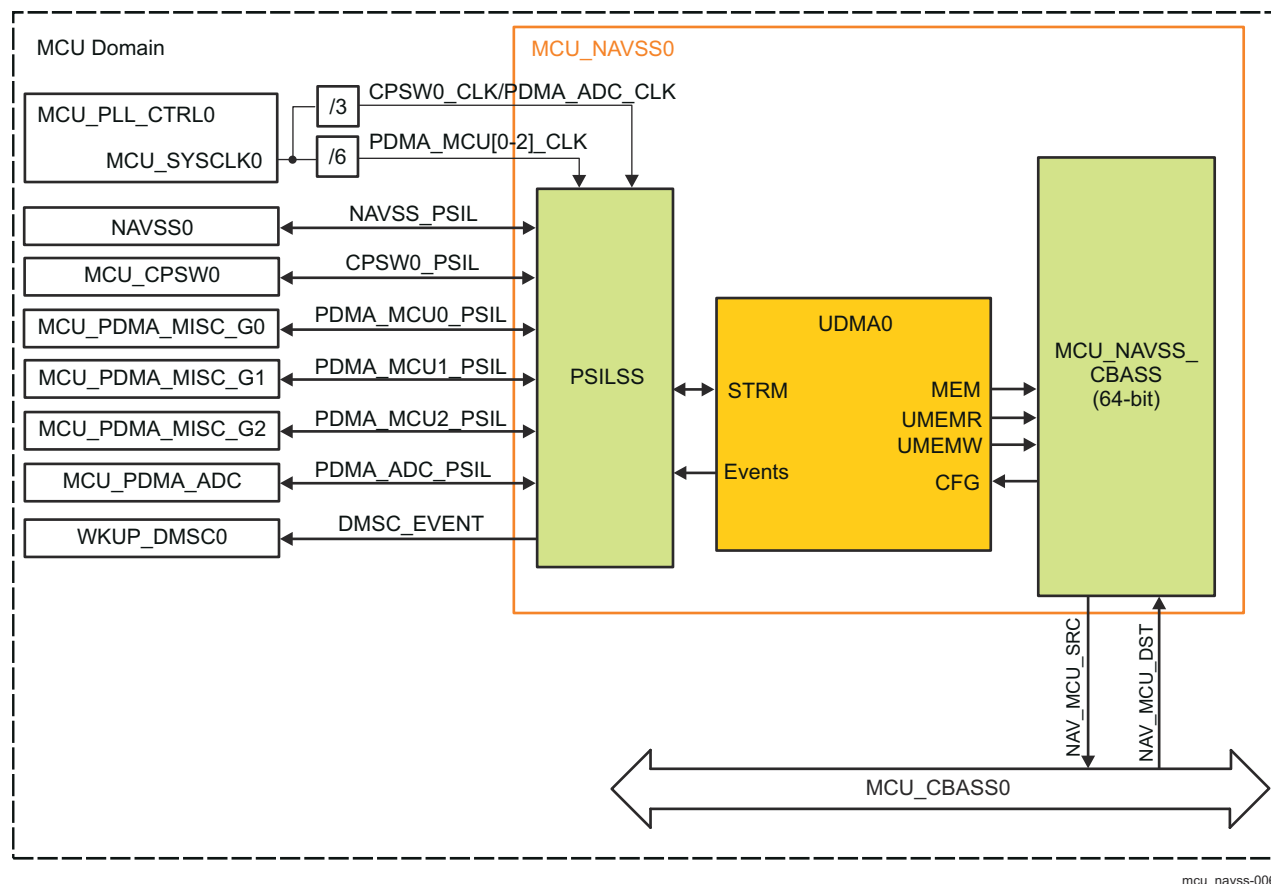
MCU NAVSS functionality is similar to that of main NAVSS, however it is composed of fewer peripherals and has reduced bandwidth. This chapter describes the integration and parameters specific for MCU NAVSS.

### 10.2.2.2 MCU NAVSS Integration

Figure 10-19 and Figure 10-20 show the top-level integration of MCU\_NAVSS0 in the device.



**Figure 10-19. MCU NAVSS Clocks, Resets, and Interrupts**



**Figure 10-20. MCU NAVSS Interconnects**

Table 10-108 and Table 10-109 summarize the integration of the module in the device.

**Table 10-108. MCU NAVSS Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
MCU_NAVSS0	WKUP_PSC0	PD0	LPSC0	MCU_CBASS0 PSI-L

**Table 10-109. MCU NAVSS Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
MCU_NAVSS0	MODSS_VBUS_D2_CLK	MCU_SYCLK0/3	MCU_PLL_CTRL0	MODSS config interface clock. This clock is used for MCU_NAVSS modules (MODSS).
	UDMASS_VBUS_D2_CLK	MCU_SYCLK0/3	MCU_PLL_CTRL0	UDMASS config interface clock. This clock is used for UDMASS modules.
	CPSW0_CLK	MCU_SYCLK0/3	MCU_PLL_CTRL0	CPSW0 PSI-L interface clock
	PDMA_MCU0_CLK	MCU_SYCLK0/6	MCU_PLL_CTRL0	PDMA_MCU0 PSI-L interface clock
	PDMA_MCU1_CLK	MCU_SYCLK0/6	MCU_PLL_CTRL0	PDMA_MCU1 PSI-L interface clock
	PDMA_MCU2_CLK	MCU_SYCLK0/6	MCU_PLL_CTRL0	PDMA_MCU2 PSI-L interface clock
	PDMA_ADC_CLK	MCU_SYCLK0/3	MCU_PLL_CTRL0	PDMA_ADC PSI-L interface clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description



**Table 10-109. MCU NAVSS Clocks and Resets (continued)**

MCU_NAVSS0_MO DSS	MODSS_RST	MOD_G_RST	WKUP_LPSC0	MODSS hardware reset
MCU_NAVSS0_UD MASS	UDMASS_RST	MOD_G_RST	WKUP_LPSC0	UDMASS hardware reset. Same as MODSS reset.

**Table 10-110. MCU NAVSS Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCU_NAVSS0	INTR_ROUTER0_OUTL_I NT[31:0]	CORE0_INT[95:64]	MCU_R5FSS_COR E0	Interrupts to MCU R5FSS Core 0	Level
	INTR_ROUTER0_OUTL_I NT[64:32]	CORE1_INT[95:64]	MCU_R5FSS_COR E1	Interrupts to MCU R5FSS Core 1	Level
	MODSS_ECC_SEC_PEN D	LVL_IN[40]	MCU_ESM0	SEC interrupt from MODSS ECC_AGGRO	Level
	MODSS_ECC_DED_PEN D	LVL_IN[41]	MCU_ESM0	DED interrupt from MODSS ECC_AGGRO	Level
	UDMASS_ECC_SEC_PE ND	LVL_IN[42]	MCU_ESM0	SEC interrupt from UDMASS ECC_AGGRO	Level
	UDMASS_ECC_DED_PE ND	LVL_IN[43]	MCU_ESM0	DED interrupt from UDMASS ECC_AGGRO	Level
Inbound Events					
Module Instance	DMA Event	DMA Event Input	Destination	Description	Type
MCU_NAVSS0	WKUP_GPIOMUX_OUTP[ 19:12]	L2G_EVENT_PEND 0[11:4]	MCU_INTR_AGGRO	External L2G inputs into INTR_AGGRO. 4 MCRC events + 8 external.	Level
	EVENT_PEND_INTR[3:0]	L2G_EVENT_PEND 0[3:0]			

#### 10.2.2.2.1 MCU NAVSS Interrupt Router Configuration

The interrupt router has the below interrupt inputs listed from the higher order bits to the lower order bits. The interrupt router is configured without an INTD (no\_intd = 1).

**Table 10-111. Interrupt Router Input Mapping**

MCU_NAVSS0_UDMASS_INTR_ROUTER0 Inputs	Interrupt Mapping (MSB to LSB)
INRTR_IN[260]	MCRC0_MCRC_PEND_INTR
INRTR_IN[259:256]	MCRC0_EVENT_PEND_INTR[3:0]
INRTR_IN[255:0]	UDMASS_INTA0_VINTR_PEND[255:0]

#### Note

Interrupt router outputs are described in [Table 10-110](#), first two rows.

Interrupt router registers are described in *INTR0\_INTR\_ROUTER\_CFG Registers*.

#### 10.2.2.2.2 MCU NAVSS UDMASS Interrupt Aggregator Configuration

[Table 10-112](#) shows the UDMASS Interrupt Aggregator 0 parameters set during design time.

**Table 10-112. Interrupt Aggregator Parameters**

Module Instance	Parameters				
	VINTR <sup>(1)</sup>	SEVI <sup>(2)</sup>	GEVI <sup>(3)</sup>	LEVI <sup>(4)</sup>	MEVI <sup>(5)</sup>
MCU_NAVSS0_UDMASS_INTR_AGGRO	256	1536	256	12 (4 + 8 external)	128

(1) VINTR – Number of Virtual Interrupt outputs. These are connected to the interrupt router inputs.

- (2) SEVI – Number of Main (Steerable) Events
- (3) GEVI – Number of Countable Global Events
- (4) LEVI – Number of Local Event inputs
- (5) MEVI – Number of Multicast Events

#### Note

4 local (LEVI) pulse interrupts are from the MCRC0 module.

If EVENT\_PEND\_INTR[3:0] needs to be converted to level interrupts, then UDMASS\_INTR\_AGGR0 can be used to generate events to the PSILSS. PSILSS would route events back to the interrupt aggregator to be turned into level output interrupts.

#### Note

For Interrupt Aggregator functional description, see [Section 10.2.7, Interrupt Aggregator](#).

### 10.2.2.2.3 MCU NAVSS UDMA Configuration

[Table 10-113](#) shows the UDMA configuration parameters set during SoC design.

**Table 10-113. UDMA Configuration Parameters**

Module Instance	Parameters <sup>(1)</sup>				
	tchan_cnt	rchan_cnt	echan_cnt <sup>(2)</sup>	hchan_cnt	rflow_cnt <sup>(3)</sup>
MCU_NAVSS0_UDMASS_UDMAP0	48	48	0	2	96

(1) Parameter values can be read from the capabilities registers. See CAP2 and CAP3 registers.

(2) External UTC channel count - specifies how many Tx channels are external UTC channels.

(3) Rx flow table entry count

#### Note

For UDMA functional description, see [Section 10.2.3, Unified DMA Controller](#).

### 10.2.2.2.4 MCU NAVSS Ring Accelerator Configuration

Each UDMA-P has an associated dedicated Ring Accelerator that is accessible by it alone (not accessible by other subsystems). The UDMA-P configuration is listed in [Table 10-114](#).

**Table 10-114. RINGACC Configuration Parameters**

Module Instance	Parameters		
	Ring Count	Number of Monitors	Proxy Target Base
MCU_NAVSS0_UDMASS_RINGACC0	286	32	0x00002B000000

[Table 10-115](#) shows the RINGACC ring mapping.

**Table 10-115. RINGACC Ring Mapping**

Mapping	Rings	Description
MCU_NAVSS0_UDMASS_UDMAP0 Transmit	ring[47:0]	48 UDMA0 transmit channels
MCU_NAVSS0_UDMASS_UDMAP0 Receive	ring[95:48]	48 UDMA0 receive channels
General purpose	ring[255:96]	General-purpose rings
MCU_NAVSS0_SEC_PROXY0	ring[285:256]	SEC_PROXY0 rings

#### Note

For Ring Accelerator functional description, see [Section 10.2.4, Ring Accelerator](#).

Table 10-116 shows the MSRAM configuration parameters set during SoC design. MSRAM is accessible only from the ring accelerator (DST port).

**Table 10-116. MSRAM Configuration Parameters**

Module Instance	Parameters		
	Depth	Width	Base Address
MCU_NAVSS0_UDMASS_MSRAM0	3594	64	0x000028000000
MCU_NAVSS0_UDMASS_MSRAM1	4096	64	0x000028010000

#### 10.2.2.2.5 MCU NAVSS Proxy Configuration

Table 10-117 shows the Proxy configuration parameters set during SoC design.

**Table 10-117. Proxy Configuration Parameters**

Module Instance	Parameters				
	Proxies <sup>(1)</sup>	Buffer Size <sup>(2)</sup>	Target	Channels <sup>(3)</sup>	Sizes <sup>(4)</sup>
MCU_NAVSS0_PROXY0	64	256	MCU_NAVSS0_RIN GACC0	286	4096

- (1) Number of proxy threads supported. Can be read off from the CONFIG read-only register
- (2) Number of bytes per proxy buffer
- (3) Number of channels for the target
- (4) Number of bytes per channel supported for the target

#### Note

For Proxy functional description, see [Section 10.2.5, Proxy](#).

#### 10.2.2.2.6 MCU NAVSS Secure Proxy Configuration

Table 10-118 shows the Secure Proxy configuration parameters set during SoC design.

**Table 10-118. Secure Proxy Configuration Parameters**

Module Instance	Parameters				
	Proxies <sup>(1)</sup>	Message Size <sup>(2)</sup>	Target	Channels <sup>(3)</sup>	Sizes <sup>(4)</sup>
MCU_NAVSS0_SEC_ PROXY0	90	64	MCU_NAVSS0_RING ACC0	30	4096

- (1) Number of proxy threads supported. Can be read off from the CONFIG read-only register
- (2) Number of bytes per message. Can be read off from the CONFIG read-only register
- (3) Number of channels for the target
- (4) Number of bytes per channel supported for the target

#### Note

For Secure Proxy functional description, see [Section 10.2.6, Secure Proxy](#).

#### 10.2.2.2.7 Global Event Map

The global event map for all navigator events is shown in [Table 10-100, Global Event Map](#).

#### 10.2.2.2.8 PSI-L System Thread Map (All NAVSS)

Endpoints in MCU\_NAVSS0 can be paired with NAVSS0 and visa versa as shown in [Table 10-101, System Thread Map for All PSI-L Threads](#) (source/destination pairs). The configuration proxy in each subsystem can configure any endpoint in the map on either subsystem without restriction.

#### 10.2.2.2.9 MCU NAVSS VBUSM Route ID Table

The VBUSM Master routelds are listed in [Table 10-119](#).

**Table 10-119. MCU NAVSS VBUSM Route IDs**

VBUSM Interface	RouteId(s)	OrderId(s)
MSMC0_MST	0-127	8-15
MSMC1_MST	255	0-15
NAV_MCU_DST0		
PROXY0	3616	0-15
RINGACC0.DST	3624	0-15
SEC_PROXY0	3640	0-15
UDMAP0.MEM0	3584	0-15
UDMAP0.MEM1	3585	0-15
UDMAP0.UMEMW	3586	0-15
UDMAP0.UMEMR	3587	0-15

#### 10.2.2.2.10

#### Note

For more information on the interconnects, see [Chapter 3, System Interconnect](#).

For more information on the power, reset, and clock management, see the corresponding sections within [Chapter 5, Device Configuration](#).

### 10.2.2.3 MCU NAVSS Functional Description

See *DMA Architecture*, for the *TI K3 Data Movement (DMA) Architecture* specification.

See the following sections for the respective module description:

UDMASS:

- Unified DMA Controller – [Section 10.2.3](#)
- Ring Accelerator – [Section 10.2.4](#)
- Packet Streaming Interface (PSI-L) – [Section 10.2.8](#)

MODSS:

- Infrastructure components:
  - CBASS – *System Interconnect*
  - Proxies – [Section 10.2.5](#) and [Section 10.2.6](#)
  - Interrupt aggregator – [Section 10.2.7](#)
  - Interrupt router – *MCU NAVSS Interrupt Router Configuration*

ECC aggregators – [Section 12.11.4](#)

## 10.2.3 Unified DMA Controller (UDMA)

This chapter describes the UDMA controller module, part of NAVSS.

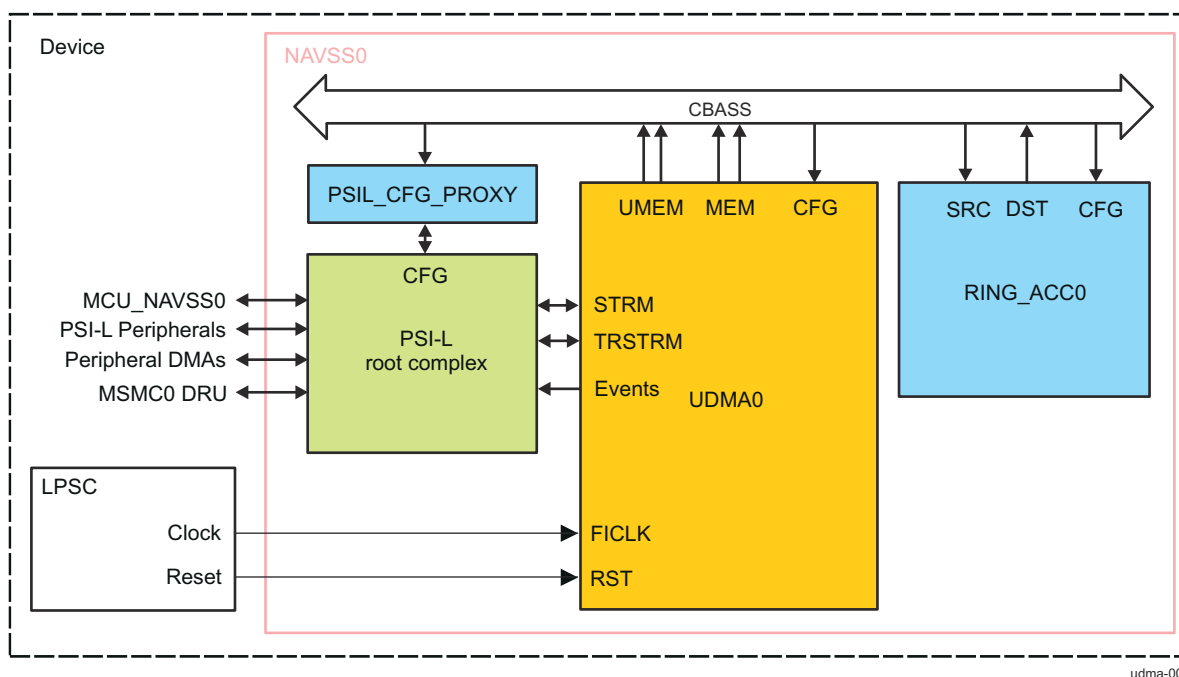
### 10.2.3.1 UDMA Overview

The UDMA-P is intended to perform similar (but significantly upgraded) functions as the packet-oriented DMA used on previous SoC devices.

**Table 10-120. UDMA Allocation Across Device Domains**

Module Instance	Domain		
	WKUP	MCU	MAIN
NAVSS0_UDMAP0	-	-	✓ (NAVSS)
MCU_NAVSS0_UDMAP0	-	✓ (MCU NAVSS)	-

Figure 10-21 shows a top-level overview of the UDMA module.



**Figure 10-21. UDMA Overview**

#### 10.2.3.1.1 UDMA Features

NAVSS0\_UDMAP0 controller module supports the following modes and features:

- *TI K3 DMA Architecture* compliant Tx/Rx port implementation (see [Section 10.1, DMA Architecture](#))
- Implements *TI DMA Architecture*-compliant Packet-Oriented DMA Functionality (UDMA-P)
  - Provides internal Tx Packet Oriented DMA processing unit
  - Provides internal Rx Packet Oriented DMA processing unit
  - Supports *rflow\_cnt* unique Rx flow table entries
  - Supports 1 transmit queue per data connection
  - Supports 1 receive queue per data connection
  - Supports physical separation of buffer control and payload information
  - Supports host and monolithic descriptor formats
  - Supports unlimited buffer scatter/gather for packets using host descriptor type
  - Supports data buffer sizes up to 4 Mbytes
  - Supports variable first valid byte offset within data buffers

- Supports packet truncation on transmit
- Provides per-channel buffering:
  - Provides 16 word deep × 128-bit Packet FIFO for each Tx channel
  - Provides 4 word deep Packet Info FIFO for each Rx channel
  - Provides 8 word deep × 128-bit Packet Data FIFO for each Rx channel
  - Supports up to 32 Protocol Specific words for Tx packets
  - Supports up to 32 Protocol Specific words for Rx packets
- Implements *TI DMA Architecture* compliant Third Party Channel Controller
  - Channel controller is built as a prefetcher engine
  - Provides the same functionality as a discrete UDMA-C/UTC combination
    - Provides a fully integrated *TI DMA Architecture* Third Party DMA compliant solution
  - Can also fetch TRs and writeback TR responses for up to *echan\_cnt* external channels which are transported via the TR PSI-L interface
- Implements *TI DMA Architecture* compliant Unified Transfer Controller
  - Provides a memory read access unit
    - Supports read bursts up to 128 bytes (limited by Tx Per Channel FIFO depth for the channel)
  - Provides a memory write access unit
    - Supports write bursts up to 128 bytes (limited by Tx Per Channel FIFO depth for the channel)
  - Supports Type 0-4, and Type 15 Transfer Request types
- Provides a set of *TI DMA Architecture* compliant Unified DMA channels which all share the execution hardware using time division multiplexing
  - Supports *tchan\_cnt* concurrent Tx channels
    - Each Tx channel can be configured at runtime to be:
      - Packet oriented Tx data channel
      - Third party combination source control and data channel (Third Party channel mode)
  - Supports *rchan\_cnt* simultaneous Rx (destination) channels
    - Each Rx channel can be configured at runtime to be:
      - Packet oriented Rx data channel
      - Third party combination destination control and data channel (Third Party channel mode)
  - Provides support for three different groups of internal channels:
    - Ultra-high capacity (UHC) channels
      - Quantity specified by: *uchan\_cnt*
      - Provide deeper Tx Per Channel FIFOs
      - Provide 16 deep descriptor/TR prefetch buffers
      - UHC channels are the first *uchan\_cnt* channels within the Tx and Rx arrays (starting with channel 0 and extending up to channel *uchan\_cnt*-1)
      - Tx and Rx UHC channels are specified as a matched pair
        - Still usable as split channels but design must have same number of Tx/Rx UHC channels
    - High-capacity (HC) channels:
      - Quantity specified by: *hchan\_cnt* - *uchan\_cnt*
      - Provide deeper Tx Per Channel FIFOs
      - Provide 16 deep descriptor/TR prefetch buffers
      - HC channels are the next *hchan\_cnt* channels (after any UHC channels) within the Tx and Rx channel arrays (starting with channel *uchan\_cnt* and extending up to *uchan\_cnt* + *hchan\_cnt*-1).
    - Normal capacity (NC) channels:
      - Quantity on Tx specified by: *tchan\_cnt* - *uchan\_cnt* - *hchan\_cnt*
      - Quantity on Rx specified by: *rchan\_cnt* - *uchan\_cnt* - *hchan\_cnt*
      - Provide standard depth Tx/Rx per channel FIFOs
      - Provide 1 deep descriptor/TR prefetch buffers

---

**Note**

User can still use ultra-high capacity and high-capacity channels as general normal-capacity channels though may not be taking advantage of the potential DMA throughput advantages.

---

- Provides per-channel buffering:
  - Provides 16 word deep data FIFO for each NC Tx (source) channel
  - Provides 8 word deep data FIFO for each NC Rx (destination) channel
- Provides single 128-bit read/write VBUSM master interface for prefetcher accesses to RINGACC and memory.
  - Supports up to 16 outstanding reads.
  - Supports up to 16 outstanding writes.
- Provides single 128-bit read/write VBUSM master interface for packet DMA and coherency unit accesses to RINGACC and memory.
  - Supports up to 16 outstanding reads.
  - Supports up to 16 outstanding writes.
- Provides single 128-bit read only VBUSM master interface for payload transfers.
  - Supports up to 16 outstanding reads.
- Provides single 128-bit write only VBUSM master interface for payload transfers.
  - Supports up to 16 outstanding writes.
- Provides 128-bit wide PSI-L compliant source interface for sending data to remote UTCs and remote peripherals
  - Includes 1 outgoing event transport lane
- Provides 128-bit wide PSI-L compliant destination interface for receiving data from remote UTCs and remote peripherals
  - Includes 1 incoming event transport lane
- Provides PSI-L interface for sending Transfer Requests to remote UTC channels and receiving back Transfer Responses
  - Provides 32 entry deep unified Transfer Response FIFO for external channels
- Provides 0 local event input buses



### 10.2.3.1.2 UDMA Parameters

Table 10-121 shows the UDMA configuration parameters in this SoC.

**Table 10-121. UDMA Configuration Parameters**

Module Instance	Parameters <sup>(1)</sup>					
	tchan_cnt <sup>(2)</sup>	rchan_cnt <sup>(3)</sup>	echan_cnt <sup>(4)</sup>	hchan_cnt <sup>(6)</sup>	uchan_cnt <sup>(7)</sup>	rflow_cnt <sup>(5)</sup>
NAVSS0_UDMAP0	140	140	160	16	4	300
MCU_NAVSS0_UDMAP0	48	48	0	2	0	96

(1) Parameter values can be read from the capabilities registers. See UDMA\_CAP2 and UDMA\_CAP3 registers.

(2) Total internal Tx channel count (Normal- + High- + Ultra-high capacities)

(3) Total internal Rx channel count (Normal- + High- + Ultra-high capacities)

(4) External UTC channel count

(5) Rx flow table entry count

(6) High-capacity + Ultra-high capacity channel count

(7) Ultra-high capacity channel count

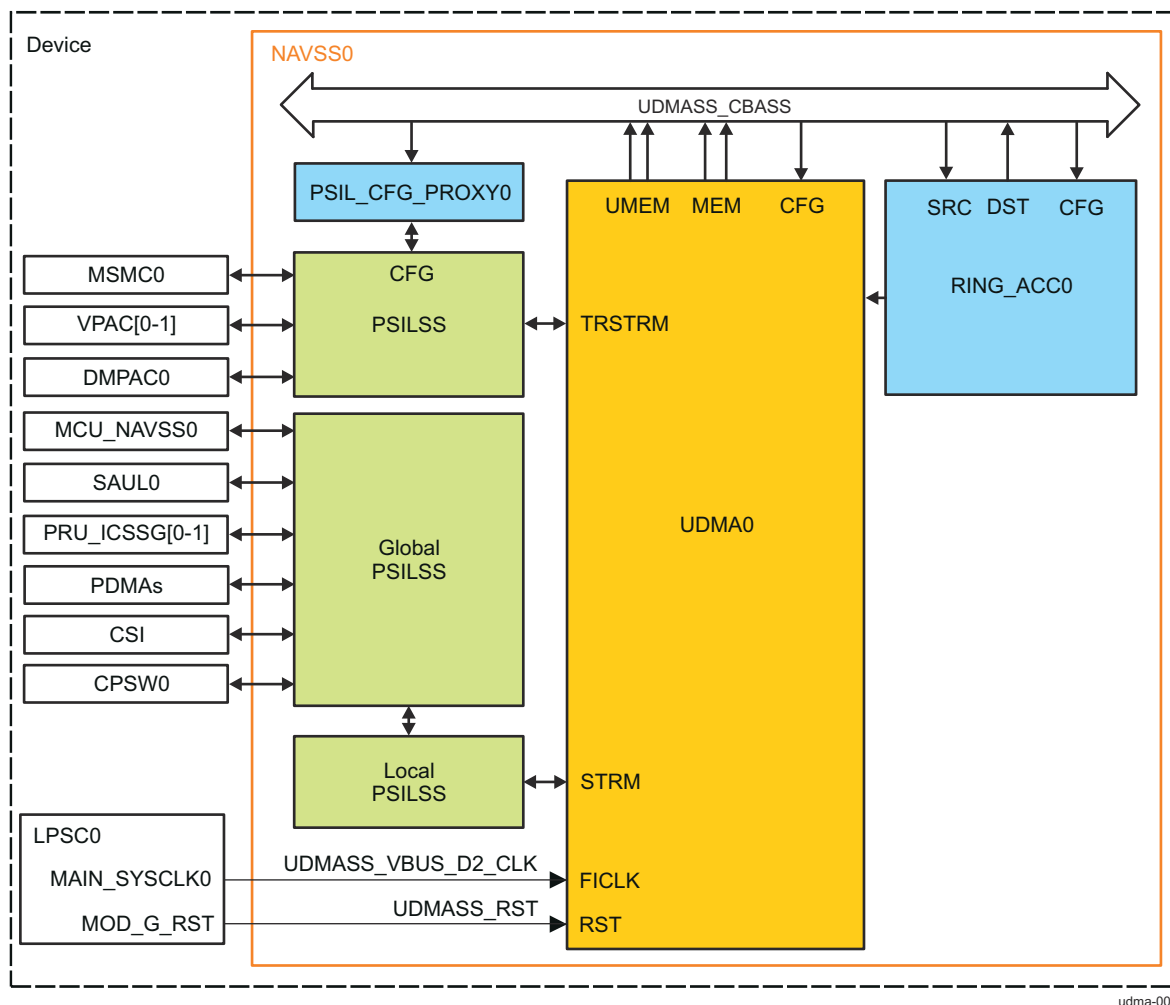
### 10.2.3.2 UDMA Integration

This section describes module integration in the device, including information about clocks, resets, and hardware requests.

#### Note

This section describes the UDMA integration in the main Navigator subsystem (NAVSS0). For MCU\_NAVSS0\_UDMA0 integration, please see *MCU Navigator Subsystem (MCU\_NAVSS)*.

Figure 10-22 shows the UDMA integration in the device.



**Figure 10-22. UDMA Integration**

Table 10-122 and Table 10-123 summarize the integration of the module in the device.

**Table 10-122. UDMA Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
NAVSS0_UDMA0	PSC0	GP	LPSC0	UDMASS_CBASS PSI-L

**Table 10-123. UDMA Clocks and Resets**

Clocks
--------

**Table 10-123. UDMA Clocks and Resets (continued)**

Module Instance	Module Clock Input	Source Clock Signal	Source	Description
NAVSS0_UDMAP0	UDMAP0_FICLK	UDMASS_VBUS_D2_CLK	MAIN_SYSCLK0	UDMA clock. This clock is used for all interface and functional operations.
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
NAVSS0_UDMAP0	UDMAP0_RST	UDMASS_RST	LPSC0	UDMA hardware reset

**Table 10-124. UDMA Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
NAVSS0_UDMAP0	-	-	-	The module does not generate traditional (local) interrupts	-
Outbound Events					
Module Instance	Module Event	Destination Event Number	Destination	Description	Type
NAVSS0_UDMAP0	TX Complete	Programmable	See (1)	To other UDMA/UTC/PDMA instances on the device for synchronization and triggering purposes.	ETL Push
	RX Complete	Programmable	See (1)	To other UDMA/UTC/PDMA instances on the device for synchronization and triggering purposes.	ETL Push
	Error	Programmable	See (1)		ETL Push
Inbound Events					
Module Instance	Module Event	Module Event Number	Source	Description	Type
NAVSS0_UDMAP0	TXCH0_TRG0	0	(1)	Tx Channel 0 Trigger 0 in third party mode	ETL Push
	TXCH0_TRG1	1	See (1)	Tx Channel 0 Trigger 1 in third party mode	ETL Push
	...	...	See (1)	...	ETL Push
	TXCH139_TRG0	278	See (1)	Tx Channel 139 Trigger 0 in third party mode	ETL Push
	TXCH139_TRG1	279	See (1)	Tx Channel 139 Trigger 1 in third party mode	ETL Push
	RXCH0_TRG0	280	See (1)	Rx Channel 0 Trigger 0 in third party mode	ETL Push
	RXCH0_TRG1	281	See (1)	Rx Channel 0 Trigger 1 in third party mode	ETL Push
	...	...	See (1)	...	ETL Push
	RXCH139_TRG0	558	See (1)	Rx Channel 139 Trigger 0 in third party mode	ETL Push
	RXCH139_TRG1	559	See (1)	Rx Channel 139 Trigger 1 in third party mode	ETL Push

(1) See [Section 10.2.1.2.2, Global Event Map](#).

### 10.2.3.3 UDMA Functional Description

The UDMA module supports the transmission and reception of various packet types. The UDMA is architected to facilitate the segmentation and reassembly of K3 DMA data structure compliant packets to/from smaller data blocks that are natively compatible with the specific requirements of each connected peripheral. Multiple Tx and Rx channels are provided within the DMA which allow multiple segmentation or reassembly operations to be ongoing. The DMA controller maintains state information for each of the channels which allows packet segmentation and reassembly operations to be time division multiplexed between channels in order to share the underlying DMA hardware.

An internal DMA scheduler is used to control the ordering and rate at which this multiplexing occurs for Transmit operations. The ordering and rate of Receive operations is indirectly controlled by the order in which blocks are pushed into the DMA on the Rx PSI-L interface.

The UDMA also supports acting as a Unified Channel Controller (UDMA-C)/Unified Transfer Controller (UTC) combined unit which accepts Transfer Request packets from Ring Accelerator and then performs the transfers which the Transfer Request (TR) specifies.

Channels in the UDMA can be configured to be either Packet-based or TR-based Third Party channels on a channel by channel basis.

#### 10.2.3.3.1 Block Diagram

Figure 10-23 shows UDMA main internal components.

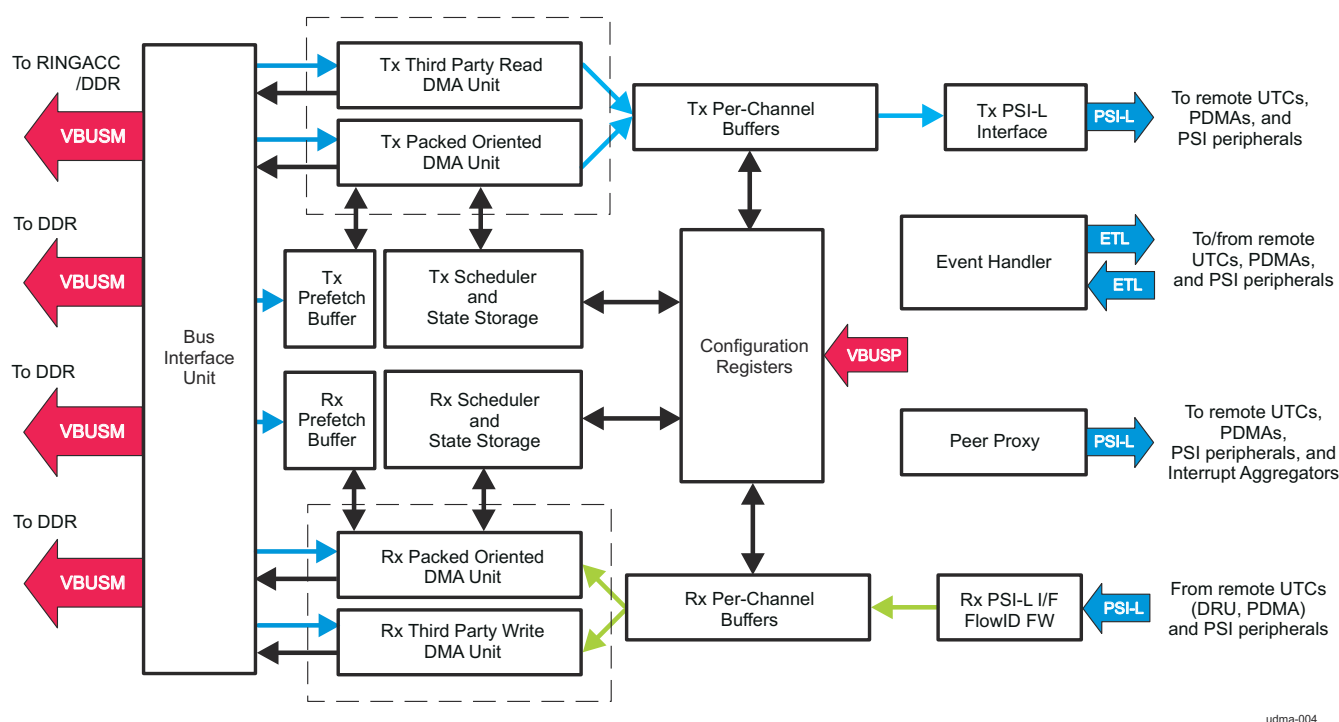


Figure 10-23. UDMA Block-Diagram

#### Bus Interface Unit

The Bus Interface Unit (BIU) is responsible for merging and buffering all of the transactions that originate from the various master blocks inside the DMA controller into the 4 separate VBUSM master interfaces. Arbitration between the blocks to a given VBUSM interface is round robin within a single priority level. A two word deep retiming buffer is provided on each sub interface of each provided VBUSM bus.

#### Tx Prefetcher Configuration Registers

The Tx Prefetcher Configuration Registers block is responsible for providing memory mapped registers for configuration of the Tx DMA function, monitoring the fullness level of the Tx prefetch buffers, maintaining Tx prefetch state information, arbitrating which channel will be allowed to perform prefetch work next, issuing scheduler commands to the Tx Prefetch Units, and writing back the updated state returned from those same prefetch units.

### **Tx Prefetch Unit(s)**

The Tx Prefetch Unit block is responsible for performing a fetch of either a Packet Descriptor (packet mode) or a TR to feed the downstream Tx DMA/Tx Read Units.

### **Tx Configuration Registers**

The Tx Configuration Registers block (*UDMASS\_UDMAP0\_CFG\_TCHAN Registers*) is responsible for monitoring the fullness level of the Tx Per Channel FIFOs, monitoring data transfer work which is pending, maintaining data movement thread state information, arbitrating which channel will be allowed to perform work next, issuing scheduler commands to the Tx Packet and TR based DMA core blocks, and writing back the updated state returned from those same DMA core blocks.

### **Tx Packet DMA Unit**

The Tx Packet DMA Unit block implements all of the state machine functionality necessary to implement the K3 DMA Host and Monolithic Tx protocol. The Tx Packet DMA unit initiates VBUSM transactions in order to read and write descriptor pointers from the Ring Accelerator, read descriptors from memory, and read data from buffers in memory.

### **Tx Packet Coherency Unit**

The Tx Packet Coherency Unit is responsible for ensuring that all control structures and data have been read (and updated if applicable) by the Tx DMA unit(s) prior to returning the packet descriptor pointer to the appropriate return queue in the Ring Accelerator. This unit ensures that the ordering of the Packet Descriptor pointer writes to the return queue directly matches the ordering in which those packets were fetched from the Tx queue. This unit is only used on channels which are in Pass-By-Reference mode.

### **Tx TR Coherency Unit**

The Tx TR Coherency Unit is responsible for ensuring that all control structures and data have been read prior to allowing the TR response to be written to either the Packet Descriptor or a Pass by Value queue in the Ring Accelerator. This unit ensures that the ordering of TR response writes also directly matches the ordering of Transfer Requests that were processed by the channel. This unit is only used on channels which are in TR mode.

### **Tx External Channel TR Coherency Unit**

The Tx External Channel TR Coherency Unit is responsible for ensuring that TRs received back from remote UTCs are written in strong order to either the Packet Descriptor or a Pass-by-Value queue in the Ring Accelerator. This unit is only used on external UTC channels.

### **Tx Event Coherency Unit**

The Tx Event Coherency Unit is responsible for ensuring that all data transfers have completed before a completion event is issued through the outgoing Event Transport Lane (ETL). This unit is only used on channels which are in TR mode.

### **Tx Per Channel Buffers**

The Tx Per Channel Buffers implement a FIFO for each Tx DMA channel that is used for buffering packet control and payload data that has been fetched by the Tx Packet DMA units or Tx Third Party Read unit modules. The buffers are byte oriented on write so that the data from the DMA units which may not be full words can be packed properly. The buffers are block oriented on read in accordance with the transport mechanism outlined in the PSI-L Interface specification. Each Tx (source) channel in the UDMA controller maps directly onto a thread in the Tx PSI-L interface. The Tx Per Channel Buffer block outputs queue fullness information to the Tx Scheduler.

block which it then uses to determine when it must initiate DMA opportunities to backfill the buffers. The Tx Per Channel Buffer will initiate transfers to the remote paired thread whenever any data is available in each channel buffer and credits are available in the corresponding thread. The block will simultaneously monitor the status of all of the threads and will perform a round robin arbitration between the different threads for the use of the Transmit PSI-L interface. Each thread for which the target is indicating it can accept data and which currently has data available in the channel buffer will be included in the arbitration.

### **Rx FlowID Firewall**

The Rx FlowID firewall checks the incoming flowID on the received packet from the Rx PSI-L interface to verify that it is either the corresponding flowID for the channel (that is, the default flow ID for the channel which is the same as the channel number) or that the flowID falls within a programmed range that is considered legal for the channel. If the received flowID does not fall within the legal range then the packet is dropped and the error is trapped.

### **Rx Per Channel Buffers**

The Rx Per Channel Buffers implement a FIFO for each Rx DMA channel that is used for buffering packet control and payload data that has been pushed into the DMA from the Rx PSI-L interface. The Rx Per Channel Buffers also includes an arbitration unit which determines which Rx DMA channel must be serviced next

### **Rx Configuration Registers**

The Rx Configuration Registers block (*UDMASS\_UDMAP0\_CFG\_RCHAN Registers*) is responsible for providing memory mapped registers for configuration of the Rx DMA functions including the default settings for the free descriptor and destination queues. For modularity and high speed pipelining reasons, the Rx traffic is looped through the Rx Configuration Registers block where the original stream information is merged with information from the configuration registers on its way to the Rx DMA unit module. This prevents the Rx DMA Core from having to spend cycles accessing the channel configuration information for the channel.

### **Rx Packet DMA Unit**

The Rx Packet DMA Unit block implements all of the state machine functionality necessary to implement the K3 DMA Rx protocol for Host and Monolithic descriptor types. The Rx Packet DMA Unit initiates VBUSM transactions in order to read descriptor pointers from the Ring Accelerator, read buffer descriptor information, write descriptors to memory, and write data to buffers in memory.

### **Rx Packet Coherency Unit**

The Rx Packet Coherency Unit is responsible for ensuring that all control structures and data have been written by the Rx DMA unit(s) prior to returning the packet descriptor pointer to the appropriate return queue in the Ring Accelerator. This unit ensures that the ordering of the Packet Descriptor pointer writes to the return queue directly matches the ordering in which those packets were fetched from the Rx free queues. Writes are not considered complete by this unit until the entire write status has been returned for all outstanding transactions for a given packet ID. This unit is only used on channels which are in Pass By Reference mode

### **Rx TR Coherency Unit**

The Rx TR Coherency Unit is responsible for ensuring that all control structures have been read and all data has been written prior to allowing the TR response to be written to either the Packet Descriptor or a Pass by Value queue in the Ring Accelerator. This unit ensures that the ordering of TR response writes also directly matches the ordering of Transfer Requests that were processed by the channel. This unit is only used on channels which are in TR mode.

### **Rx Event Coherency Unit**

The Rx Event Coherency Unit is responsible for ensuring that all data transfers have completed before a completion event is issued through the outgoing ETL. This unit is only used on channels which are in TR mode.

### **Third Party Read Unit**

The Third Party Read Unit(s) are responsible for sequencing all of the Tx side channel control and data transfers. The Third Party Read Unit is responsible for performing the actual data read operations including sequential address generation and nested loop control. Like the Tx Packet DMA Unit, the Third Party Read Unit receives state information from the Tx Configuration Registers and returns that state when a transfer opportunity is complete. The Third Party Read Unit will perform as much data transfer as is specified in the Transfer Request up until either a fixed number of bytes have been transferred, the transfer is completed, or the transfer has reached a point which requires an input trigger event to proceed.

### Third Party Write Unit

The Third Party Write Unit(s) are responsible for sequencing all of the Rx side channel data transfers. The Third Party Write Unit is responsible for performing the actual data write operations including sequential address generation and nested loop control. Like the Rx Packet DMA Unit, the Third Party Write Unit receives state information from the Rx Configuration Registers and returns that state when a transfer opportunity is complete. The Third Party Write Unit will perform as much data transfer as is specified in the Transfer Request up until either a fixed number of bytes have been transferred or the transfer is completed or the transfer has reached a point which requires an input trigger event to proceed.

### Event Handler

The Event Handler block is responsible for accepting and logging channel triggering events and for generating channel completion and error events.

### PSI-L Real Time Proxy

The PSI-L Real Time Proxy block is responsible for allowing tunneled VBUSP transactions to the Real Time Remote Peer registers to generate PSI-L configuration transactions to the paired remote peer for each Tx and Rx channel.

#### 10.2.3.3.2 General Functionality

This section is applicable to all UDMA modes.

##### 10.2.3.3.2.1 Operational States

At any given time the UDMA can be in one of three different states as described in [Table 10-125](#).

**Table 10-125. Operational States**

Operational State	Description
Init	This is the initial state of UDMA during and immediately after reset. During this state, all of the RAMs inside the UDMA will be initialized to known values including the ECC redundant parity bits. While in the Init state, the DMA will de-assert all ready signals on all applicable slave interfaces and will de-assert all request signals on all applicable master interfaces. The UDMA will automatically transition out of the Init state into the Idle state when all of the RAM initialization has been completed.
Idle	Once the UDMA leaves the Init state, it enters the Idle state whenever no outstanding transactions are pending on any of the UDMA interfaces (master or slave). The Idle state is generally a transient state and is used by the UDMA to determine when it is appropriate to allow the SoC power management complex to turn off the clock. As channels have work queued on them and transactions begin flowing into the system, the UDMA transitions to the Active state. When no more work is pending, or when the host pauses/disables active channels, or when the SoC power management desires to shut down the UDMA, the UDMA will account for any outstanding transactions and will re-enter the Idle state. The UDMA leaves the Idle state anytime it generates or receives a transactions that requires a return response as those protocols dictate that the clock must remain running to avoid faulting the handshaking protocol.
Active	The UDMA enters the Active state as soon as it issues a transaction or receives a transaction on any interface that uses a split protocol (expects a later response for a request). When all transactions have been accounted for (responses have all been either received or sent) the UDMA transitions to the Idle state.

##### 10.2.3.3.2.2 Tx Channel Allocation

In NAVSS0, a total of 140 Packet Tx channels are provided within the DMA for concurrent data transfers between memory mapped space and the Tx Per Channel Buffers. Each of these channels can be configured to operate as a packet oriented channel (uses queues/rings/descriptors/buffer) or as a Third Party DMA source



channel (uses Transfer Request packets to control read operations). Depending on which channel mode is selected, when a Tx channel comes into context the work for that channel will either be dispatched to a Tx Packet DMA Unit or a Third Party Read Unit respectively.

A total of 160 external UTC channels are also provided. These channels only perform fetches and transfers of Transfer Requests to remote UTC channels and writeback of returned Transfer Responses to TR descriptors (pass by reference) or RINGACC rings (pass by value). The external channels are at channel numbers immediately above the internal channels. The Tx channels are allocated as shown in [Table 10-126](#).

**Table 10-126. Internal Tx Channel Allocation**

DMA Channel	Function	Tx Queue (Ring)
0	Tx Channel	0 (Starting queue number in RINGACC for Tx channel 0 )
...	...	...
139	Tx Channel	139

#### 10.2.3.3.2.3 Rx Channel Allocation

In NAVSS0, a total of 140 Rx channels are provided within the DMA for concurrent transfers between the Rx Per Channel Buffers and memory mapped space. Each of these channels can be configured to operate as a packet oriented channel (uses rings/descriptors/buffer) or as a Third Party DMA destination channel (uses rings/Transfer Request packets or TR standalone records to control read operations). Depending on which channel mode is selected, when a Tx channel comes into context the work for that channel will either be dispatched to an Rx Packet DMA Unit or a Third Party Write Unit respectively. The Rx channels are allocated as shown in [Table 10-127](#).

**Table 10-127. Rx Channel Allocation**

DMA Channel	Function	Src Tag
0	Rx Channel	0
...	...	...
139	Rx Channel	139

#### 10.2.3.3.2.4 Tx Teardown

As is specified in the *TI DMA Architecture* specification, the host initiates a Tx DMA channel teardown operation by writing to the TDOWN bit in the TCHANRT UDMA\_TRT\_CTL\_j register. Once a channel teardown is requested, the appropriate Tx DMA unit will complete any packet/TRs that the channel has prefetched or is currently in the process of transferring and will then clear the EN bit, will send a Teardown packet out the outbound Payload Data PSI-L interface, and will push a Teardown Completion Record onto the Tx Default Queue for the channel. If the Tx DMA engine is not currently in a packet/TR for the channel, it will immediately clear the EN, send the Teardown packet, and push the Teardown Completion Record. Once the teardown is complete, no further packet processing will occur until the host software re-enables the channel.

#### 10.2.3.3.2.5 Rx Teardown

As is specified in the *TI DMA Architecture* specification, the host initiates an optimal Rx DMA channel teardown operation by shutting down the incoming payload stream at its source (a remote peer peripheral or the Tx side of a block copy channel). Once teardown is initiated at the source, data will eventually drain from the incoming PSI-L stream and when all data is complete, the remote peer entity will send a Teardown packet (either a standalone packet or tdown asserted with eop on the last data phase of the last packet). When the UDMA receives a teardown message, the TDOWN bit will be set in RCHANRT UDMA\_RRT\_CTL\_j. In the case where a remote peer cannot generate a teardown message, the host must ensure that all data has stopped flowing from the remote peer and will then directly set the TDOWN bit. Once the UDMA sees the TDOWN asserted it will wait for any existing packets which are currently held in the Rx Per Channel FIFO to be completed and will then de-assert the EN bit in RCHANRT UDMA\_RRT\_CTL\_j. If the back end application continues to allow new



packets to be pushed into the Rx PC FIFO, the DMA will continue to delay marking the channel as torn down as long as the Rx PC FIFO never reaches an empty state

#### 10.2.3.3.2.6 Tx Clock Stop

The Tx clock stop interface allows the Tx portion of the UDMA to be gracefully commanded to shut down its operations and enter into an IDLE state so that the main clock (FICLK) can be stopped. When the `tcs_clkstop_req` input is asserted, the Tx portion of the UDMA will stop processing transmit packets/TRs for each channel at the next packet/TR boundary. Once all of the Tx channels have gracefully stopped transmission, the UDMA will assert the `tcs_clkstop_ack` output. Once the UDMA Tx engine has entered the IDLE state, it will remain there until the `tx_clockstop_req` is de-asserted.

#### 10.2.3.3.2.7 Rx Clock Stop

The Rx clock stop interface allows the Rx portion of the UDMA to be gracefully commanded to shut down its operations and enter into an IDLE state so that the main clock (FICLK) can be stopped. When the `rcs_clkstop_req` input is asserted, the Rx portion of the UDMA will stop processing receive packets for each channel at the next packet boundary. Once all of the Rx channels have gracefully stopped reception, the UDMA will assert the `rcs_clkstop_ack` output. Once the UDMA Rx engine has entered the IDLE state, it will remain there until the `rx_clockstop_req` is de-asserted.

#### 10.2.3.3.2.8 Rx Thread Enables

PSI-L threads must first be enabled in order to transfer commands or data. Threads are enabled or disabled by setting or clearing the enable bit in the PSI-L pairing registers for the thread. When a thread is disabled, it must drop any data phases which are sent but properly return the credits for the data phases which are dropped.

#### 10.2.3.3.2.9 Events

Events are used in the *TI DMA Architecture* for triggering hardware or software (by being turned into interrupts) processes. Physical (or local) events in the *TI DMA Architecture* are implemented as active-high, single-cycle, long, synchronous pulses which are strobed to communicate that a particular condition has occurred. Virtual (or global) events are implemented as a value which is passed across a shared bus (ETL) - one value per cycle. Physical (local) events can be used by the UDMA to communicate with some local peripherals. Virtual (global) events are used by the UDMA to communicate with remote peripherals, the interrupt aggregator, and other DMA instances in the system.

##### 10.2.3.3.2.9.1 Local Event Inputs

There are no local events routed to the UDMA local event inputs in the current NAVSS implementation.

##### 10.2.3.3.2.9.2 Inbound Tx PSI-L Events

The Tx PSI-L interface includes a single inbound event transport lane (ETL) which allow events to be received from other UDMA/UTC/PDMA instances for synchronization and triggering purposes. Events passed in from this event transport lane will already have been decoded as destined for some event destination index for this particular UDMA. The appropriate LSBs of each event number are extracted and the specified event is asserted.

The event map for inbound events to the UDMA is shown in [Table 10-128](#).

**Table 10-128. Inbound Event Map**

Event Number	Target
0	Tx Channel 0 Global Trigger 0
1	Tx Channel 0 Global Trigger 1
2	Tx Channel 1 Global Trigger 0
3	Tx Channel 1 Global Trigger 1
...	...
279	Tx Channel 139 Global Trigger 1
280	Rx Channel 0 Global Trigger 0
281	Rx Channel 0 Global Trigger 1

**Table 10-128. Inbound Event Map (continued)**

Event Number	Target
282	Rx Channel 1 Global Trigger 0
283	Rx Channel 1 Global Trigger 1
...	...
559	Rx Channel 139 Global Trigger 1

#### 10.2.3.3.2.9.3 Outbound Rx PSI-L Events

The Rx PSI-L interface includes a single outbound event transport lane (ETL) which allows internal events to be sent to other UDMA/UTC/PDMA instances on the device for synchronization and triggering purposes. The destination event number is programmed for each event source within the UDMA. See registers:

- TCHAN UDMA\_TOES\_j
- TCHAN UDMA\_TEOES\_j
- RCHAN UDMA\_ROES\_j
- RCHAN UDMA\_REOES\_j

#### 10.2.3.3.2.10 Emulation Control

The emulation control input and register bits (SOFT and FREE in the UDMA\_EMU\_CTRL register) allow DMA operation to be suspended. When the emulation suspend state is entered, the DMA will stop processing receive and transmit packets for each channel at the next packet boundary. Any packet currently in reception or transmission will be completed normally without suspension. Emulation control is implemented for compatibility with other peripherals. [Table 10-129](#) shows the operations of the emulation control input and register bits.

**Table 10-129. Emulation Behavior Control**

Emulation Control Input	SOFT	FREE	Description
0	X	X	Normal Operation
1	0	0	Normal Operation
1	1	0	Emulation Suspend
1	X	1	Normal Operation

#### 10.2.3.3.3 Packet Oriented Transmit Operation

Packet transmission is accomplished within the UDMA by moving data from structures that are located in memory via the VBUSM Memory Interfaces onto the Transmit Packet Data PSI-L Interface. This movement of data is performed in blocks whose size is specified by the Tx DMA scheduler.

##### 10.2.3.3.3.1 Packet Mode VBUSM Master Interface Command ID Selection

The UDMA keeps a transaction command scoreboard for each specific direction for the Packet Mode VBUSM Master interface. The scoreboard tracks which command indexes are currently outstanding from the DMA on that interface. When a read or write is requested by one of the internal master agents in the UDMA, the scoreboard corresponding to the specified interface and for the specified direction (read/write) is queried starting at index 0 and extending up to the maximum index for that scoreboard. The lowest index that is found and which is not currently allocated is selected and is directly used as the command ID (*cid*). The index values for each scoreboard. Entries are freed for re-use in a write scoreboard when write status responses are received which account for all of the outstanding write data. Entries are freed for re-use in the read scoreboard only when all data has been returned for the read.

#### 10.2.3.3.4 Packet Oriented Receive Operation

Packet reception is accomplished within the UDMA by moving data from the Receive PSI-L Interface to data structures that are located in memory accessible via the VBUSM Memory Interface(s).

#### 10.2.3.3.4.1 Rx Packet Drop

The Rx PSI-L Interface master may assert the `rstrmp_drop` signal co-incident with the transfer of any data phase of a packet. When this signal is sampled asserted by the UDMA, the drop condition will be latched into a flag for that thread and the Rx Packet DMA unit will be notified that the packet is to be flushed and recycled rather than passed to the host. The drop signal is only required to be asserted with a single data phase and will only cause the current packet to be dropped from packet stream. The UDMA will drop all data given to it until the EOP is found on the thread and then the internal drop flag will be cleared. When the UDMA drops a packet on a channel configured for Packet mode there is no notification to the host.

#### 10.2.3.3.4.2 Rx Starvation and the Starvation Timer

As described in the *TI DMA Architecture* specification, whenever the Rx DMA engine needs to read a Free Descriptor Pointer from the Ring Accelerator, it is possible that the queue may be empty. When this occurs it is referred to as buffer starvation. The Ring Accelerator provides a message interface to the UDMA which provides increment and decrement information so that the UDMA can keep a local occupancy counter for each ring that it can access from the RINGACC. When the Rx engine requires a free descriptor it will first check to see if the local occupancy for that ring is non-zero. If the occupancy is zero, starvation has occurred. When starvation occurs the Rx DMA will respond in one of two ways depending on the value of the `rx_error_handling` bit in Rx Flow Table entry that is currently being used for that particular packet reception opportunity. If the `rx_error_handling` bit is 0, the Rx DMA will drop the packet and will return any descriptors which it may have previously allocated for that packet to the respective queues from which they were allocated. If the `rx_error_handling` bit is set, the Rx DMA is required to retry checking to see if a Free Descriptor is available at a later time.

A timer is provided in order to control the rate at which the retry attempts will occur. All channels share the same timer and the period of the timer is set using the `TIMEOUT_CNT` field in the `UDMA_PERF_CTRL` register. The `TIMEOUT_CNT` field specifies in clock cycles the minimum delay between subsequent attempts to check if a Free Descriptor is available on each individual channel.

*Example:* If the `TIMEOUT_CNT` is set to 256 and channel 1 finds its Free Descriptor queue empty on cycle 10, it will be required to wait until at least cycle 266 before it can make another attempt check the queue again. The actual number of cycles which will be waited by the DMA will typically be larger than the `TIMEOUT_CNT` value and this is not intended to be a highly precise operation. The critical requirement is that the DMA will not attempt to bring a channel into context to check if a buffer is available at a more frequent rate than is allowed.

#### 10.2.3.3.5 Third Party Mode Operation

##### 10.2.3.3.5.1 Events and Flow Control

The following sections describe the mechanisms that are provided for supporting flexible flow control between peripherals, the UDMA and other DMA instances.

##### 10.2.3.3.5.1.1 Channel Triggering

Channels which are configured in Third Party mode must be triggered in order for them to perform work. Triggering can be configured to be immediate, one-shot, or periodic at different levels of TR completion and is specified in the TR itself. Each channel in the UDMA is capable of being triggered by two separate events which are mapped at specific offsets in a global event target map (just like an address map for memory mapped slaves). The UDMA maps its channel triggers with Tx side channels first followed by Rx side channels as described in [Table 10-128, Inbound Event Map](#).

##### 10.2.3.3.5.1.2 Internal TR Completion Events

Each TR which is processed, provides the ability to assert 1 of 4 possible output events at specific completion levels for the TR. Since the output event is specified directly in the TR and the TRs are written by potentially untrusted software processes, channels are only able to assert *virtual events* when they complete various portions of the TR. These virtual events are then mapped to actual destination events in the global event map by the `TCHAN_UDMA_TOES_j` or `RCHAN_UDMA_ROES_j` event steering registers. These registers are programmed statically by a trusted software process that has determined the channels that should have access to the specified event destination slots.

### **10.2.3.3.5.2 Transmit Operation**

#### **10.2.3.3.5.2.1 Transfer Request**

After reset, all channels that are configured in Third Party transfer mode will be idle and waiting for work to be assigned to them. In order to initiate Third Party channel operation, the host software places work in the form of either a Transfer Request Descriptor or an individual Transfer Request record onto the Tx Queue of the channel. The exact mechanisms which are used and the contents of both the Transfer Request Descriptor and the Transfer Request record are given in the *TI DMA Architecture* specification.

#### **10.2.3.3.5.2.2 Transfer Response**

At the completion of each set of operations which fulfill a Transfer Request record, the UDMA will return a Transfer Response record to either the original Transfer Request Descriptor or the Tx Completion Queue, depending on the mechanism which was used to provide the Transfer Request record. If the Transfer Request record originated in a Transfer Request Descriptor then a corresponding slot in the Transfer Request Descriptor will be overwritten with the response status. If the Transfer Request record originated in a pass by value Tx Ring, the Transfer Response record will be placed in a pass by value Tx Completion Ring. The Transfer Response record will only contain a completion code which indicates if the transfer was a success or if a particular type of failure occurred. The exact mechanisms which are used and the contents of the Transfer Response Record are given in the *TI DMA Architecture* specification.

#### **10.2.3.3.5.2.3 Data Transfer**

Packet transmission is accomplished within the UDMA by moving data from structures that are located in memory via the VBUSM Memory Interface(s) onto the Transmit PSI-L Interface. On the Tx side of the UDMA, these transfers are always reads. Data is read from an attached memory mapped space and packed into the Rx Per Channel Buffer for that channel. At a later time, the data is moved from the Tx Per Channel FIFO to a remote peer DMA entity via the Tx PSI-L interface.

#### **10.2.3.3.5.2.4 Memory Interface Transactions**

The sequence of control transactions that are performed by the UDMA on the Read/Write VBUSM interface during transmit is to read a Transfer Request record from a logical work queue and to then return a corresponding Transfer Response record to a separate logical work queue when all the transfers in the request have been completed. This sequence repeats for as long as Transfer Request records are provided on the work queue. The exact details of the operations that are involved are covered in the *TI DMA Architecture* specification (see [Section 10.1](#)).

The sequences of data transactions that are performed by the UDMA on the Read Only VBUSM interface during transmit can be described as a sequence of piecewise linear read transfers whose starting addresses are each sequentially calculated based on loops and offsets given in the Transfer Request record. A TR can have up to 4 levels of nested transfer sets where each set can have a different element count and stride between elements. TR record formats are described in detail in the Transfer Request format specification.

#### **10.2.3.3.5.2.5 Error Handling**

In the case that an error occurs during any Tx operation, the UDMA will stop processing the TR at that point and will return an appropriate error code in the Transfer Response message as specified in the *TI DMA Architecture* specification. The PAUSE\_ON\_ERR bit in the TCHAN UDMA\_TCFG\_j registers controls whether or not the DMA will pause operation on that channel if an error or exception occurs. If the PAUSE\_ON\_ERR bit is set to 1 the channel will be paused so that the host can optionally read the channel state to determine where in the transfer the channel execution was. If the PAUSE\_ON\_ERR bit is 0 the DMA will flush the current Transfer Request record for channels in pass by value mode and will flush the current Transfer Request Descriptor for channels in pass by reference mode.

### **10.2.3.3.5.3 Receive Operation**

#### **10.2.3.3.5.3.1 Transfer Request**

After reset, all channels that are configured in Third Party mode will be Idle and waiting for work to be assigned to them. In order to initiate Third Party channel operation, the Host places work in the form of either a Transfer

Request Descriptor or an individual Transfer Request record onto the Rx TR Queue of the channel. The exact mechanisms which are used and the contents of both the Transfer Request Descriptor and the Transfer Request record are given in the *TI DMA Architecture* specification.

#### **10.2.3.3.5.3.2 Transfer Response**

At the completion of each set of operations which fulfill a Transfer Request, the UDMA will send a Transfer Response message back to either the Packet Descriptor (pass by reference channel mode) or to the completion queue in the Ring Accelerator (pass by value channel mode). The contents of this message are described in detail in the *TI DMA Architecture*.

#### **10.2.3.3.5.3.3 Error Handling**

In the case that an error occurs during any Rx operation, the UDMA will stop processing the TR at that point and will return an appropriate error code in the Transfer Response message as specified in the *TI DMA Architecture* specification. The PAUSE\_ON\_ERR bit in the RCHAN\_UDMA\_RCFG\_j registers controls whether or not the DMA will pause operation on that channel if an error or exception occurs. If the PAUSE\_ON\_ERR bit is set to 1 the channel will be paused so that the host can optionally read the channel state to determine where in the transfer the channel execution was. If the PAUSE\_ON\_ERR bit is 0 the DMA will flush the current Transfer Request record for channels in pass by value mode and will flush the current Transfer Request Descriptor for channels in pass by reference mode.

#### **10.2.3.3.5.4 Data Transfer**

Third Party mode packet reception is accomplished within the UDMA by unpacking and moving data from the Rx Per Channel FIFOs which were filled via the Rx PSI-L Interface to specified memory mapped address ranges via the write only VBUSM master interface. On the Rx side of the UDMA, these transfers are always writes. Each write transfer which is performed by the Third Party Write Unit is to a destination address and of a size as calculated from parameters specified in the Transfer Request packet which was previously received to control the channel behavior.

##### **10.2.3.3.5.4.1 Memory Interface Transactions**

The sequence of control transactions that are performed by the UDMA on the Read/Write VBUSM interface during transmit is to read a Transfer Request record from a logical work queue and to then return a corresponding Transfer Response record to a separate logical work queue when all the transfers in the request have been completed. This sequence repeats for as long as Transfer Request records are provided on the work queue. The exact details of the operations that are involved are covered in the *TI DMA Architecture* specification (see [Section 10.1](#)).

The sequences of data transactions that are performed by the UDMA on the Write Only Memory interface during receive can be described as a sequence of piecewise linear write transfers whose starting addresses are each sequentially calculated based on loops and offsets given in the Transfer Request record. A TR can have up to 4 levels of nested transfer sets where each set can have a different element count and stride between elements. TR record formats are described in detail in the Transfer Request format specification.

##### **10.2.3.3.5.4.2 Rx Packet Drop**

The Rx PSI-L Interface master may assert the rstmr\_drop signal co-incident with the transfer of any data phase of a packet. When this signal is sampled asserted by the UDMA, the drop condition will be latched into a flag for that thread and the Third Party write unit will be notified that the packet is to be flushed and no further writes are to be performed. The drop signal is only required to be asserted with a single data phase and will only cause the current packet to be dropped from packet stream. The UDMA will drop all data given to it until the EOP is found on the thread and then the internal drop flag will be cleared. When the UDMA drops the packet on a channel that is in Third Party mode, a 'dropped by source' error code will be placed in the Transfer Response packet which is returned to the Packet Descriptor or completion queue.

## 10.2.4 Ring Accelerator (RINGACC)

This chapter describes the RINGACC module, part of the NAVSS.

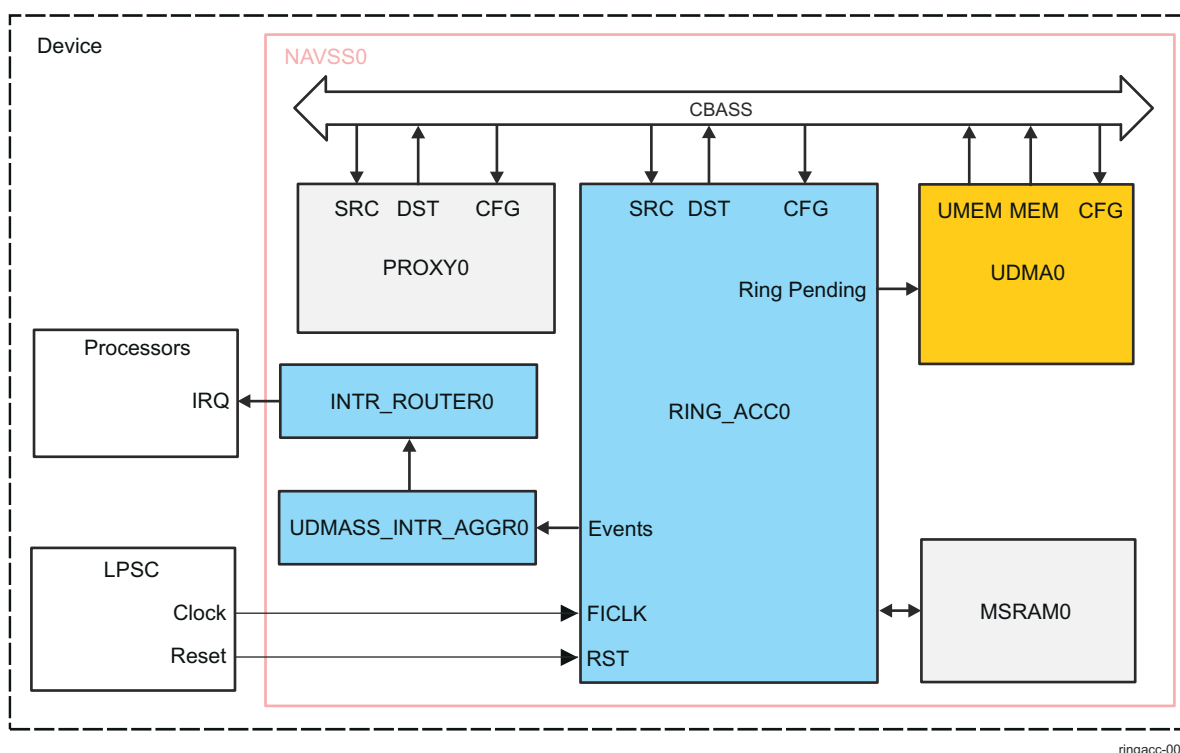
### 10.2.4.1 RINGACC Overview

The Ring Accelerator (RINGACC or RA) provides hardware acceleration to enable straightforward passing of work between a producer and a consumer. There is one RINGACC module per NAVSS.

**Table 10-130. RINGACC Allocation Across Device Domains**

Instance	Domain		
	WKUP	MCU	MAIN
NAVSS0_RINGACC0	-	-	✓ (NAVSS)
MCU_NAVSS0_RINGACC0	-	✓ (MCU NAVSS)	-

Figure 10-24 shows a top-level overview of the RINGACC module.



**Figure 10-24. RINGACC Overview**

#### 10.2.4.1.1 RINGACC Features

NAVSS0\_RINGACC0 supports the following features:

- Supports independent memory-mapped ring structures. See ring count in [Table 10-131, RINGACC Configuration Parameters](#).
- Supports various modes for each ring based on usage and compatibility
- Provides single-word deep shared incoming Transfer Response FIFO
- Provides bit-wide source VBUSM read/write slave interface for accesses from DMA controller entities.
  - Provides 2 word deep command FIFO
  - Provides 2 word deep write data FIFO
  - Provides 2 word deep read data FIFO
  - Provides 2 word deep write status FIFO
- Provides bit-wide destination VBUSM read/write master interface for accesses to ring structures in memory



- Supports up to 16 outstanding writes
- Supports up to 16 outstanding reads
- Source interface provides an array of 1024 × 512-byte long address windows (four for each ring) which are packed into a single contiguous address range
  - Read and write addresses which target a specific window are translated, in order to redirect the read or write transaction, to an effective address calculated from the base address for the ring plus current ring offset
  - Each read or write access presented on VBUSM slave interface is modified and bridged onto the VBUSM master interface

#### 10.2.4.1.2 RINGACC Not Supported Features

- Queue manager mode, which makes the queue act like a traditional queue-manager (QM) queue, is not supported.
- Peeks from tail

#### 10.2.4.1.3 RINGACC Parameters

Table 10-131 shows the RINGACC configuration parameters in this SoC.

**Table 10-131. RINGACC Configuration Parameters**

Module Instance	Parameters		
	Ring Count	Number of Monitors	Proxy Target Base
NAVSS0_RINGACC0	1024	32	0x000038000000
MCU_NAVSS0_UDMASS_RINGACC0	286	32	0x00002B000000

Table 10-132 shows the MSRAM configuration parameters set during SoC design. MSRAM0 is accessible only from the ring accelerator (DST port).

**Table 10-132. MSRAM Configuration Parameters**

Module Instance	Parameters		
	Depth	Width	Base Address
NAVSS0_MSRAM0	4096	128	0x000030000000
MCU_NAVSS0_UDMASS_MSRAM0	3594	64	0x000028000000
MCU_NAVSS0_UDMASS_MSRAM1	4096	64	0x000028010000

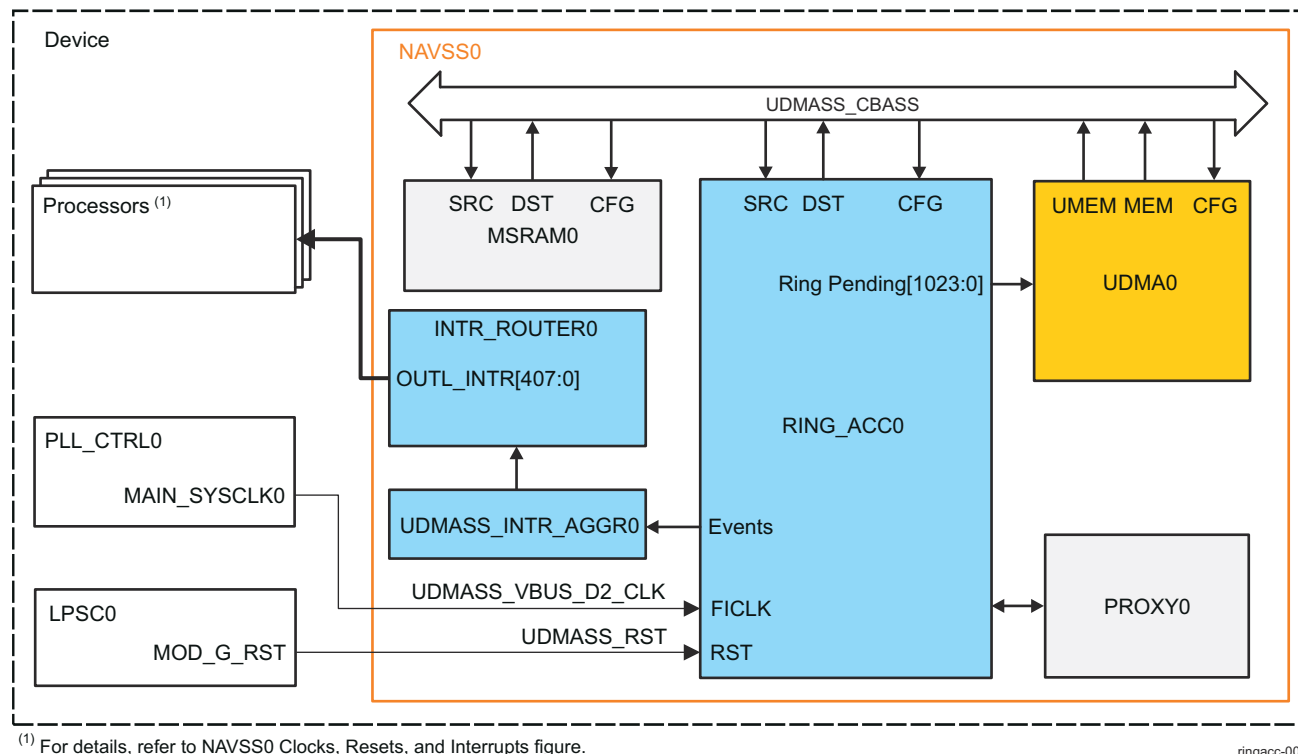
### 10.2.4.2 RINGACC Integration

This section describes module integration in the device, including information about clocks, resets, and hardware requests.

#### Note

This section describes the RINGACC integration in the main Navigator subsystem (NAVSS0). For MCU\_NAVSS0\_RINGACC0 integration, please see *MCU Navigator Subsystem (MCU\_NAVSS)*.

Figure 10-25 shows the RINGACC integration in the device.



**Figure 10-25. NAVSS0\_RINGACC0 Integration**

Table 10-133 and Table 10-135 summarize the integration of the module in the device.

**Table 10-133. RINGACC Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
NAVSS0_RINGACC0	PSC0	GP	LPSC0	CBASS_NAVSS

**Table 10-134. RINGACC Ring Mapping**

Module Instance	Rings	Mapping	Description
NAVSS0_RINGACC0	ring[299:0]	UDMAP0 Transmit	300 UDMA0 transmit channels
	ring[439:300]	UDMAP0 Receive	140 UDMA0 receive channels
	ring[973:440]	General purpose	General-purpose rings
	ring[1023:974]	NAVSS0_SEC_PROXY0	SEC_PROXY0 rings

**Table 10-135. RINGACC Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description



**Table 10-135. RINGACC Clocks and Resets (continued)**

NAVSS0_RINGAC C0	RINGACC0_FICLK	MODSS_VBUS_D2_CLK	MAIN_SYSCLK0	RINGACC clock. This clock is used for all interface and functional operations.
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
NAVSS0_RINGAC C0	RINGACC0_RST	MODSS_RST	LPSC0	RINGACC hardware reset

**Table 10-136. RINGACC Hardware Requests**

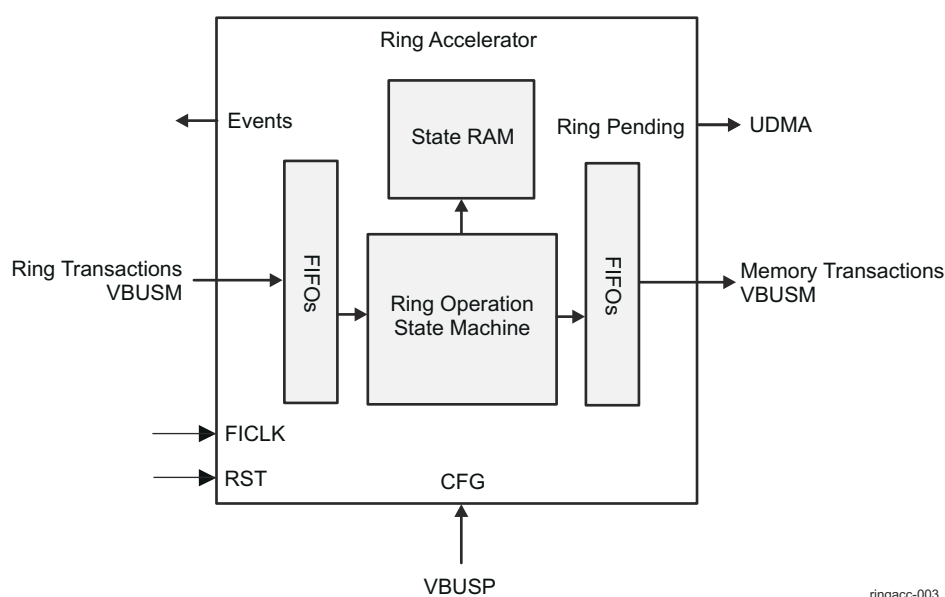
Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
NAVSS0_RINGAC C0	-	-	-	The module does not generate traditional interrupts	-
Outbound Events					
Module Instance	Module Event	Destination Event Input	Destination	Description	Type
NAVSS0_RINGAC C0	RING[0:1023]_EVT	SEVI	UDMASS_INTR_AGGR0	Ring events. Can be used by an external host for statistics or interrupts.	ETL Push
	MON[0:31]_EVT	SEVI	UDMASS_INTR_AGGR0	Monitor events. Can be used by an external host for statistics or interrupts.	ETL Push
	RINGACC_ERROR_EVT	SEVI	UDMASS_INTR_AGGR0	Bus error event for host interrupts	ETL Push

### 10.2.4.3 RINGACC Functional Description

The Ring Accelerator (RINGACC) converts constant-address read and write accesses to equivalent read or write accesses to a circular data structure in memory. The RINGACC eliminates the need for each DMA controller which needs to access ring elements from having to know the current state of the ring (base address, current offset). The DMA controller performs a read or write access to a specific address range (which maps to the source interface on the RINGACC) and the RINGACC replaces the address for the transaction with a new address which corresponds to the head or tail element of the ring (head for reads, tail for writes). Since the RINGACC maintains the state, multiple DMA controllers or channels are allowed to coherently share the same rings as applicable. The RINGACC serves a very similar function to the Queue Manager in the earlier SoCs. The RINGACC uses less memory and is able to place data which is destined towards software into cached memory directly.

#### 10.2.4.3.1 Block Diagram

Figure 10-26 shows ring accelerator's main internal components.



**Figure 10-26. Ring Accelerator Block-Diagram**

#### 10.2.4.3.1.1 Configuration Registers

The Configuration Registers block is responsible for providing memory mapped registers for configuration of the RINGACC functions. The configuration registers are divided into:

- Static configuration fields (NAVSS0\_UDMASS\_RINGACC0\_CFG Registers), which are typically initialized and then left at specified values for long time periods (or until a reset occurs)
- Real-time (RT) registers (NAVSS0\_UDMASS\_RINGACC0\_CFG\_RT Registers), which are modified frequently during runtime.

Configuration registers are listed and described in *RINGACC Register Manual*.

#### 10.2.4.3.1.2 Source Command FIFO

The Source Command FIFO buffers VBUSM command phases which are accepted from the VBUSM source interface.

#### 10.2.4.3.1.3 Source Write Data FIFO

The Source Write Data FIFO buffers VBUSM write data phases which are accepted from the VBUSM source interface.

#### **10.2.4.3.1.4 Source Read Data FIFO**

The Source Read Data FIFO buffers VBUSM read data phases which have been returned from the VBUSM destination interface (via the main state machine) and which are to be output on the VBUSM source interface.

#### **10.2.4.3.1.5 Source Write Status FIFO**

The Source Write Status FIFO buffers VBUSM write status data phases which have been returned from the VBUSM destination interface and which are to be output on the VBUSM source interface.

#### **10.2.4.3.1.6 Main State Machine**

The Main State Machine (MSM) block implements all of the control logic which is necessary to accept a command from the source interface, perform a lookup into the ring state RAM, determine the effective address for the destination interface transaction, modify the address (and byte count for reads) for the transaction, and place that modified transaction into the outgoing destination FIFOs. The MSM block is also responsible for updating the ring index and occupancy values and for producing output status to indicate changes to the ring state. The MSM also modifies the ring state whenever the read data or write status for a previous ring transaction return and forwards those responses back to the originating master.

#### **10.2.4.3.1.7 Destination Command FIFO**

The Destination Command FIFO stores VBUSM transaction commands which are output from the MSM as modified versions of transactions that were popped from the Source Command FIFO. The Destination command FIFO allocates a new command ID for each read or write command that is pushed to the FIFO. Read IDs are allocated from a scoreboard maintained in the Destination Read Data FIFO. Write IDs are allocated from a scoreboard maintained in the Destination Write Status FIFO. When a new command is allocated, the Destination Command FIFO also pushes the original *cid* and *crouteid* values from the source side transaction into a scoreboard in either the Destination Read Data or Destination Write Status FIFOs depending on the applicable transaction direction. The read interface of the Destination Command FIFO directly drives commands onto the destination VBUSM command interface.

#### **10.2.4.3.1.8 Destination Write Data FIFO**

The Destination Write Data FIFO stores write data phases for write commands that are passing through the RINGACC towards memory. The write interface of the Destination Write Data FIFO is directly connected to the read interface of the Source Write Data FIFO and the read interface directly drives write data onto the destination VBUSM write data interface.

#### **10.2.4.3.1.9 Destination Read Data FIFO**

The Destination Read Data FIFO provides a read command translation scoreboard and also stores read data phases that are passing through the RINGACC towards the originating DMA controller. The write interface of the Destination Read Data FIFO is directly connected to the destination VBUSM read data interface. As read data passes through the Destination Read Data FIFO, the *rid* and *rrouteid* are changed back to their original values using information which was stored in read command translation scoreboard.

#### **10.2.4.3.1.10 Destination Write Status FIFO**

The Destination Write Status FIFO provides a write command translation scoreboard and also stores write status phases that are passing through the RINGACC towards the originating DMA controller. The write interface of the Destination Writes Status FIFO is directly connected to the destination VBUSM write status interface. As write status words pass through the Destination Write Status FIFO, the *sid* and *srouteid* are changed back to their original values using information which was stored in the write command translation scoreboard.

### **10.2.4.3.2 RINGACC Functional Operation**

#### **10.2.4.3.2.1 Queue Modes**

Each ring or queue can be in one of four modes that determines how it must be accessed and the compatibility it has with hardware and software.

#### 10.2.4.3.2.1.1 Ring Mode

Ring mode is used when software owns one side of the ring, and hardware or another software owns the other side. This mode allows the software that owns a side of the ring to control that side including accessing the memory directly rather than going through hardware, and software updates the hardware on any changes through the Doorbell registers. These doorbell accesses indicate when software has pushed or popped entries, allowing the entity on the other side of the ring to know when there are elements ready. The ring mode has limitations that the exposed side of the ring is completely under software control, and any atomicity needed must also be performed by software, which is why the exposed side of a ring is usually owned by a single thread of software. This limitation also means that any hardware configuration that would normally access the same ring for both pushes and pops cannot use ring mode as hardware cannot access the exposed side of the ring, such as free queues that are normally read by hardware but could also be pushed by hardware on an error.

When software is popping from the ring, it must guarantee that it never pops more elements through the Doorbell register (RINGACC\_DB\_j) than what is valid in the Ring Occupancy (RINGACC\_OCC\_j) register, which holds how many elements are ready for software consumption. There could be some elements which have data written to memory but have not received status back from memory and those are not considered ready to pop yet as the ring status has not fully updated. Therefore, just reading from memory to determine the number of elements to pop is not sufficient. If software attempts to pop more than the RINGACC\_OCC\_j register value, then the occupancies could go negative resulting in missed down events and spurious interrupts. An optimization for software is to read the RINGACC\_OCC\_j register less frequently and keep a local copy which guarantees the maximum number of allowed pops. When the software copy of RINGACC\_OCC\_j reaches 0, or periodically, the software can read the register to refresh the value. This must reduce the frequency of register reads instead of reading the register for every software pop.

#### 10.2.4.3.2.1.2 Messaging Mode

Messaging mode requires that all accesses to the queue must go through RINGACC so that all accesses to the memory are controlled and ordered. RINGACC then controls the entire state of the queue, and software has no direct control, such as through doorbells and cannot access the storage memory directly. This is particularly useful when more than one software or hardware entity can be the producer and/or consumer at the same time. Software must access the data through bus accesses to the RINGACC. As with the earlier example, if there are free queues that need the hardware to both push and pop, then they must be configured as at least messaging mode (or a later mode) and not ring mode.

#### 10.2.4.3.2.1.3 Credentials Mode

Credentials mode adds per element credentials storage to messaging mode. This allows multiple producers with their own credentials to have those credentials stored with the element. This allows the consumer to inherit the credentials of each element rather than a single set for the entire queue, such as for DMA TR credential inheritance. This mode requires each operation to use two elements of storage since the credentials are stored in the second element, so the element count must be scaled appropriately. The credentials field is located in the same location as all modes, the word just before the first word of the message.

#### 10.2.4.3.2.1.4 Queue Manager Mode

Queue manager mode makes the queue act like a traditional queue-manager (QM) queue in previous devices.

---

#### Note

Queue Manager mode is no longer supported in this device. This section was left for completeness only.

---

#### 10.2.4.3.2.1.5 Peek Support

All modes except ring mode allow the peek operations so that software can view the next element without actually popping it from the queue. This is done by reading from a different offset from the normal pop operation. All the same fields are read but the element is not actually popped off of the queue. This is useful for algorithms

that need to know about the overall queue and head element to make decisions but have not yet decided to pop from the queue.

#### 10.2.4.3.2.1.6 Index Register Operation

In ring mode, the Index (RINGACC\_IND<sub>X</sub>\_j) register follows the software control of the ring through the doorbell registers, while the hardware index (RINGACC\_HWIND<sub>X</sub>\_j) register follows the hardware access of the ring through bus transactions. It works for a ring in either direction as the doorbell register update indicates whether elements were produced or consumed. But in the other queue modes, there is no simple manner to determine which index is for software or hardware since they both use bus transactions which the module cannot differentiate. So for these other queue modes, the index register is always the read index, and the hardware index register is the write index.

#### 10.2.4.3.2.2 VBUSM Slave Ring Operations

Each ring contains four memory spaces on the VBUSM slave bus for access, each of which is 512 bytes long, with a total of 4 kB per ring. Each ring space starts immediately after the preceding ring. The allocation per ring is as in [Table 10-137](#).

**Table 10-137. Ring Memory Partition**

Offset	Operation	Supported Ring Modes
0x0 - 0x1FF	Reads pop from head. Writes push to head.	Read supported in all modes, write supported in all modes except Ring mode
0x200 - 0x3FF	Writes push to tail. Reads pop from tail.	Write supported in all modes, read supported in all modes except Ring mode
0x400 - 0x5FF	Reads peek from head. Writes ignored.	All modes except for Ring mode.
0x600 - 0x7FF	Reserved	Peeks from tail are not supported in this device
0x800 - 0xFFF	Reserved	Reserved

Within each 512-byte (0x200) space, the message data is right justified so that the last byte is always the 511-th byte. The element size of the ring defines which words are valid. The word just before the first valid message word is the credentials field. Access must not be made before the credentials register .

#### 10.2.4.3.2.3 VBUSM Master Interface Command ID Selection

The RINGACC keeps a transaction command scoreboard for reads and writes. Each scoreboard tracks which command indexes are currently outstanding from the RINGACC and which are free to use. When a read or write is requested by the Main State Machine, the scoreboard corresponding to the specified direction (read/write) is queried starting at index 0 and extending up to the maximum index for that scoreboard. The lowest index that is found and which is not currently allocated is selected and is directly used as the command ID (*cid*) for the outgoing transaction on the destination interface. 15 is the maximum index value for each scoreboard. Entries are freed for re-use in a write scoreboard when write status responses are received which account for all of the outstanding write data. Entries are freed for re-use in the read scoreboard only when all data has been returned for the read.

#### 10.2.4.3.2.4 Ring Push Operation (VBUSM Write to Source Interface)

The following sequence will occur for each VBUSM write which is sent to the source interface:

1. RINGACC extracts ring number from incoming write transaction address
2. RINGACC looks up ring state using ring number
3. RINGACC calculates effective write address using *ring base + ring\_index*
4. RINGACC re-evaluates pending bit for ring
5. RINGACC increments hardware index and hardware occupancy for ring
6. RINGACC allocates *cid* from scoreboard and places original *routeid*, *cid*, and *ring number* into scoreboard
7. RINGACC pushes altered *caddress*, *cid* and unaltered remainder of command attributes to output FIFO (note *routeid* is not included)

At a later time when write status returns:

1. RINGACC recovers original *cid*, *crouteid*, and *ring number* from scoreboard
2. RINGACC increments software index and occupancy for ring
3. RINGACC pushes restored *srouteid*, *sid*, and unaltered write status to output write status FIFO

#### 10.2.4.3.2.5 Ring Pop Operation (VBUSM Read from Source Interface)

The following sequence will occur for each VBUSM read which is sent to the source interface:

1. RINGACC extracts ring number from incoming read transaction address
2. RINGACC looks up ring state using ring number
3. RINGACC calculates effective read address using ring base + ring\_index
4. RINGACC increments hardware index and decrements hardware occupancy for ring
5. RINGACC re-evaluates pending bit for ring
6. RINGACC munges *cid* to add indicator in 4 MSBs to reconstitute *routeid* for returning read data
7. RINGACC pushes altered *address*, *cid* and unaltered remainder of command attributes to output FIFO (note *routeid* is not included)

At a later time when read data returns:

1. RINGACC recovers original *cid*, *crouteid*, and ring number from scoreboard
2. RINGACC increments software index and decrements software occupancy for ring
3. RINGACC pushes restored *routeid*, *rid*, and unaltered read data, status, and other control signals to output read FIFO

#### 10.2.4.3.2.6 Host Doorbell Access

The following sequence will occur for each VBUSP write to the doorbell register (RINGACC\_DB\_j) for a ring:

1. RINGACC extracts ring number from config address
2. RINGACC looks up ring state using ring number
3. RINGACC adds doorbell ring value to both hardware and software occupancies for ring
4. RINGACC re-evaluates pending bit for ring

#### 10.2.4.3.2.7 Queue Push Operation (VBUSM Write to Source Interface)

The following sequence will occur for each VBUSM write which is sent to the source interface:

1. RINGACC extracts ring number from incoming write transaction address
2. RINGACC looks up ring state using ring number
3. RINGACC calculates effective write address using ring base + ring\_index
4. RINGACC re-evaluates pending bit for ring
5. RINGACC increments WR index and RD- and WR-occupancy for ring
6. RINGACC allocates *cid* from scoreboard and places original *crouteid*, *cid*, and ring number into scoreboard
7. RINGACC pushes altered *address*, *cid* and unaltered remainder of command attributes to output FIFO (note *routeid* is not included)

At a later time when write status returns:

1. RINGACC recovers original *cid*, *crouteid*, and ring number from scoreboard
2. RINGACC pushes restored *srouteid*, *sid*, and unaltered write status to output write status FIFO

#### 10.2.4.3.2.8 Queue Pop Operation (VBUSM Read from Source Interface)

The following sequence will occur for each VBUSM read which is sent to the source interface:

1. RINGACC extracts ring number from incoming read transaction address
2. RINGACC looks up ring state using ring number
3. RINGACC calculates effective read address using *ring base + ring\_index*
4. RINGACC increments RD index and decrements RD- and WR-occupancy for ring
5. RINGACC re-evaluates pending bit for ring
6. RINGACC munges *cid* to add indicator in 4 MSBs to reconstitute *routeid* for returning read data



7. RINGACC pushes altered *caddress*, *cid* and unaltered remainder of command attributes to output FIFO (note *routeid* is not included)

At a later time when read data returns:

1. RINGACC recovers original *cid*, *crouteid*, and ring number from scoreboard
2. RINGACC pushes restored *rrouteid*, *rid*, and unaltered read data, status, and other control signals to output read FIFO

#### 10.2.4.3.2.9 Mismatched Element Size Handling

When less data is written than defined by the element size for the ring for RING or MESSAGE modes, only the written data is stored in the memory and any remaining bytes maintain their old values. The index and *occ* are still incremented normally, and any special fields that are unwritten are assumed to be 0 .

Writing less data for CREDENTIALS mode is illegal, as those types need the full data to append the additional words to the correct offsets in memory.

When more data is written than defined by the element size for the ring, it will be an error and the write will be ignored and no index or *occ* increments occur. The only exception is for writes to an 8-byte element size ring, where the additional fields up to another 8 bytes are allowed but not written (for old Queue Manager compatability).

When less data is read than defined by the element size for the ring, only the read data is fetched from memory, and any remaining bytes are lost. The index and *occ* are still incremented normally.

When more data is read than defined by the element size for the ring, it will be an error and the read will not affect the ring, the read data will be 0s, and no index or *occ* decrements occur. Access beyond the credentials word must not be attempted and can result in unpredictable behavior.

All accesses must be for multiples of 32-bit words, and partial bytes are not supported for all ring modes.

#### 10.2.4.3.3 Events

The module outputs events about queue levels that can be used by an external module for statistics or interrupts.

Each queue has

- an up event when the queue goes from 0 elements to 1 or more elements
- and a down event when the queue goes from 1 or more elements to 0 elements.

The event number is programmable per queue in the RINGACC\_EVENT\_j register. This event occurs when the state machine updates the final occupancy for the queue. In ring mode this is when the response from the memory access returns, while in other queue modes this is when the operation is processed.

An event number programmed to 0xFFFF indicates to not produce an event for that ring when an event is not necessary. The register event number fields reset to 0xFFFF, so they must be programmed to a valid value before events will be generated. This applies to the monitor event number fields as well.

#### 10.2.4.3.4 Bus Error Handling

If a bus error is detected on a response to a ring memory transaction, an error event is triggered to alert software. Since the ring contents may be corrupted as the push or pop was not completed correctly, software should consider resetting the ring and any software or hardware elements using the ring. A log register (RINGACC\_ERROR\_LOG) captures the ring that had the error and whether it was from a push or pop, and an error up event is sent to the programmed event number. After an error is logged, another will not be captured until the first is read out. When read, if an error is pending an error down event is sent to clear any interrupt, and then the next bus error log can be captured. A read of the RINGACC\_ERROR\_LOG register when there is no error log pending will not produce any down events. The event that is triggered is also programmable. Only legal push and pop operations will capture this error, and not for any peek, emudbg read, flush command, or any operations that is illegal and causes the resulting memory access to be flushed.

#### 10.2.4.3.5 Monitors

Monitors can be configured that allow various functions to be tracked of programmed queues. Each monitor is programmed to monitor a particular queue, the function to provide, and the data source to operate on. The supported functions are: to check the queue against programmed thresholds, to keep track of watermarks of the queue, to keep track of starvation occurrences (a read of an empty queue), or statistics capture. The supported data sources are: the queue element count, the head element size value, or the accumulated size value. Each monitor can also be programmed to produce an event should it have a triggering condition.

##### 10.2.4.3.5.1 Threshold Monitor

A threshold monitor uses a programmed low threshold value and a programmed high threshold value to track the data source in the queue and cause an interrupt or status when a threshold is broken, either below the low value or above the high value. This is useful for software to replenish empty buffers when a queue has too few, or for a queue to accumulate enough elements so that software must start processing them, or for debug. The threshold being broken will cause the up or down event.

##### 10.2.4.3.5.2 Watermark Monitor

A watermark monitor tracks the low and high values of the data source since the last read of the monitor. After initial setup or clearing, the watermark values will load the current value of the data source as the low and high. Then for successive operations if the resulting data source of the queue is below the low watermark, or above the high watermark, the associated watermark will be updated. When the monitor value registers are read, the value is cleared and starts tracking again using the current data source value. This can be useful for tracking low and high values of queue element counts or sizes for data tracking, debug or future optimizations, such as allocating items to a queue or buffer sizes based on counts.

##### 10.2.4.3.5.3 Starvation Monitor

A starvation monitor tracks the number of starvation events (a read to an empty queue) that occurred on the queue. The data source is ignored, and the starvation count is always incremented whenever there is a read to the queue and the queue is empty. The count is cleared when the monitor value is read. This can be useful to trigger if any starvation event occurs, or to track how many of these events occur in a timeframe, and can be used for future optimizations or debug.

##### 10.2.4.3.5.4 Statistics Monitor

The statistics monitor can track some simple statistics on the queue operations. The data source is ignored. The first value is the count of the number of writes to the queue, and the second value is the count of the number of reads from the queue. The counts are cleared when the monitor values are read. This can be useful to track the activity on a queue, and can be used for future optimizations or debug.

##### 10.2.4.3.5.5 Overflow

RINGACC provides an overflow function where a push to a full ring will result in a push to a special overflow ring, rather than just losing the pushed data. The overflow ring is defined through the RINGACC\_OVRFLOW register, and that ring must be defined with a large enough element size to cover the element size of any other ring, as well as enough elements to hold enough overflow data (as an overflow to the overflow ring will be lost). When there is a push to a full ring, there will be two elements written to the overflow ring, first the push data, and then the ring number that was full. A supervisor entity can own the overflow ring and perform what actions are needed from the overflow data. The size of the overflow ring must be an even number since two elements will be pushed for each overflow event. The overflow ring must be configured in *ring* or *message* modes.

An overflow queue value of 0xFFFF will disable the overflow function and will not record queue overflows.

##### 10.2.4.3.5.6 Ring Update Port

To enhance performance for DMAs, the RINGACC provides a bus that indicates when ring updates occur, allowing the DMA to track their own rings and detect updates before the responses arrive. The bus will go valid whenever there is an update to the state of a ring, which can be from a push, pop, or doorbell write. It will not go valid for a peek, a emudbg pop, a push to a full ring, as those do not update the ring state. When the bus is



valid, the other signals are valid and indicate whether the operation was a push or a pop, whether the resulting state of the ring is empty, the ring number, and the number of elements updated to the ring (the push determines whether it is subtracted or added to the current count). One special case is a pop from an empty ring, which indicates an update but the element count is 0 since no items were actually popped, but a DMA using the cnt to add or subtract must handle this correctly by default. The bus will go valid once the operation command has been completed, and not after the response has been received, even for ring mode rings, allowing the DMA to see the update as soon as it is known by the RINGACC even before it is known to software.

#### 10.2.4.3.5.7 Tracing

This module provides a trace output of all operations so that the traffic can be viewed at a later time. The trace uses a simple write-only 32-bit VBUSP bus, with a defined packet header. The *msgtype* field defines the type of operation, and the message data is simply the data involved in the operation. It is likely the trace output will be read slower than new operations could provide data, so there will be a trace buffer able to hold two complete messages plus trace headers. If a new operation starts while the trace buffer does not have enough space, then that trace message will not be sent. And RINGACC\_TRACE\_CTL register controls whether to enable the trace output, whether to trace a single queue or all queues, and whether to include the message data in the trace output.

Table 10-138 shows the trace message for a push.

**Table 10-138. Trace Message for a Push**

Word	Bits	Data	Comment
1	31:16	0x0	Time to be replaced by trace hardware
1	15:9	0x1	Msgtype for push
1	8:0	0x009	Standard Trace Header
2	31	push_to_head	Push to head flag, 0 = to tail, 1 = to head
2	30:16	0x0	Unused
2	15:0	queue	Queue for push
3-N	31:0	message data	Optional message data for push, length dependent on message size

Table 10-139 shows the trace message for a pop.

**Table 10-139. Trace Message for a Pop**

Word	Bits	Data	Comment
1	31:16	0x0	Time to be replaced by trace hardware
1	15:9	0x2	Msgtype for pop
1	8:0	0x009	Standard Trace Header
2	31	empty	empty pop, 0 = valid message, 1 = empty
2	30:16	0x0	unused
2	15:0	queue	queue for pop
3-N	31:0	message data	Optional message Data for pop, length dependent on message size

Table 10-140 shows the trace message for a peek.

**Table 10-140. Trace Message for a Peek**

Word	Bits	Data	Comment
1	31:16	0x0	Time to be replaced by trace hardware
1	15:9	0x3	Msgtype for peek
1	8:0	0x009	Standard Trace Header
2	31	empty	empty peek, 0 = valid message, 1 = empty
2	30:16	0x0	unused
2	15:0	queue	queue for peek

**Table 10-140. Trace Message for a Peek (continued)**

Word	Bits	Data	Comment
3-N	31:0	message data	Optional message Data for peek, length dependent on message size

## 10.2.5 Proxy

This chapter describes the Proxy module in a NAVSS.

### 10.2.5.1 Proxy Overview

The Proxy module provides proxy buffers for hosts that need to create large data bursts but can only perform small accesses.

Figure 10-27 shows a top-level overview of the Proxy module.

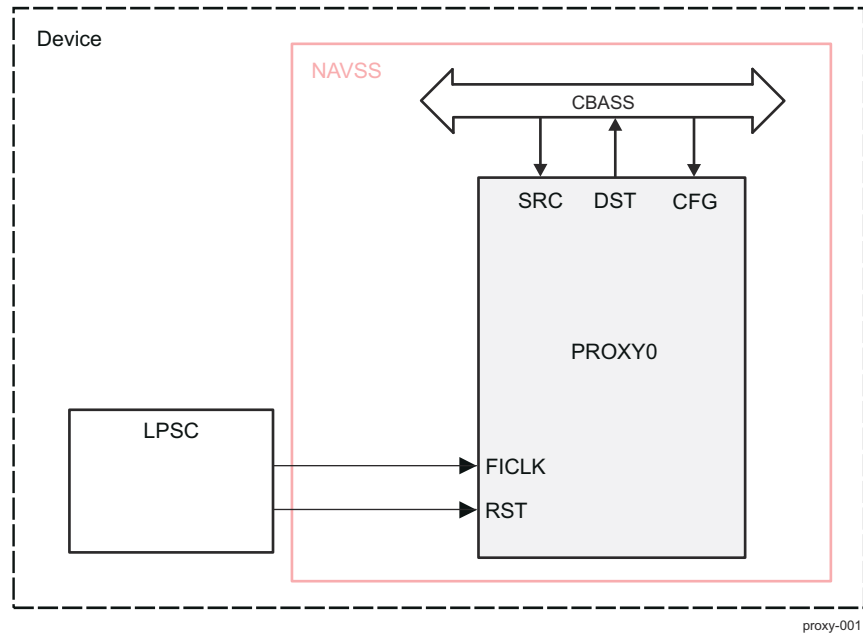


Figure 10-27. Proxy Overview

Table 10-141. Proxy Allocation Across Device Domains

Instance	Domain		
	WKUP	MCU	MAIN
NAVSS0_PROXY0	-	-	✓ (NAVSS)
MCU_NAVSS0_PROXY0	-	✓ (MCU NAVSS)	-

#### 10.2.5.1.1 Proxy Features

Each Proxy supports the following features:

- Provides proxy buffers to store large data bursts that a host can only access in smaller amounts.
- Keeps the large data coherent until the complete data has been accessed.
- Allows for interleaved access between multiple hosts using multiple proxies.
- Each target has pre-configured number of channels, size of each channel, and address offset. See [Table 10-142](#).

#### 10.2.5.1.2 Proxy Parameters

Table 10-142 shows the Proxy configuration parameters set during SoC design.

Table 10-142. Proxy Configuration Parameters

Module Instance	Parameters				
	Proxies <sup>(1)</sup>	Buffer Size <sup>(2)</sup>	Target	Channels <sup>(3)</sup>	Sizes <sup>(4)</sup>
NAVSS0_PROXY0	64	256	NAVSS0_RINGACC0	1024	4096

**Table 10-142. Proxy Configuration Parameters (continued)**

MCU_NAVSS0_PROXY0	64	256	MCU_NAVSS0_RING ACC0	286	4096
-------------------	----	-----	-------------------------	-----	------

- (1) Number of proxy threads supported. Can be read off from the PROXY\_CONFIG read-only register
- (2) Number of bytes per proxy buffer
- (3) Number of channels for the target
- (4) Number of bytes per channel supported for the target

#### **10.2.5.1.3 Proxy Not Supported Features**

The following features are not supported by the module:

- Does not support proxy sharing. Each host must use its own proxy within the module.

### 10.2.5.2 Proxy Integration

This section describes module integration in the device, including information about clocks, resets, and hardware requests.

#### Note

This section describes the Proxy integration in the main Navigator subsystem (NAVSS0). For MCU\_NAVSS0\_PROXY0 integration, please see *MCU Navigator Subsystem (MCU\_NAVSS)*.

Figure 10-28 shows the Proxy integration in the NAVSS

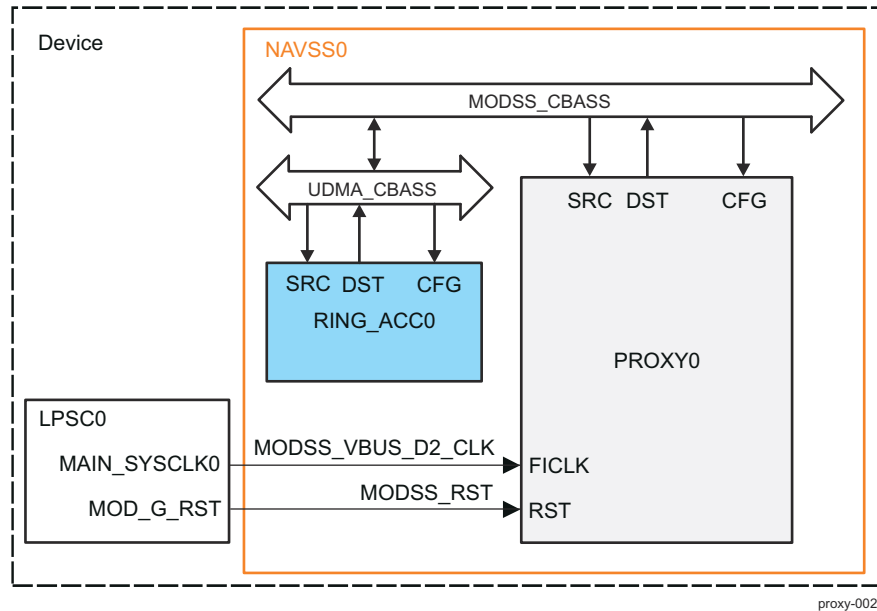


Figure 10-28. Proxy Integration

Table 10-143 and Table 10-144 summarize the integration of the module in the device.

Table 10-143. Proxy Integration Attributes

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
NAVSS0_PROXY0	PSC0	GP	LPSC0	MODSS_CBASS

Table 10-144. Proxy Clocks and Resets

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
NAVSS0_PROXY0	PROXY0_FICLK	MODSS_VBUS_D2_CLK	MAIN_SYSCCLK0	Proxy clock. This clock is used for all interface and functional operations.

Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
NAVSS0_PROXY0	PROXY0_RST	MODSS_RST	LPSC0	Proxy hardware reset

Table 10-145. Proxy Hardware Requests

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
NAVSS0_PROXY0	-	-	-	No interrupts to external interrupt controllers	-

**Table 10-145. Proxy Hardware Requests (continued)**

DMA Events					
Module Instance	Module DMA Event	Destination DMA Event	Destination Input	Description	Type
NAVSS0_PROXY0	-	-	-	No PDMA channels to external DMA engines	-

### 10.2.5.3 Proxy Functional Description

This module provides proxy buffers for hosts that need to create large data bursts but can only perform small accesses. Examples of this need are for large Message Manager messages, Queue Manager data that is more than a pointer, or DMA Transfer Requests Messages. All of these require larger data sets, from 12 to 128 bytes, and the modules require the entire data on the bus in a single burst. Unfortunately, many CPUs cannot produce bursts to device memory, such as hardware registers, and the largest CPU access is typically 8 bytes. So the proxy provides the ability to access the module with a single burst, and at the same time allows the CPU to access the same data in smaller accesses. The proxy itself does not contain the resources, but it stays in front of other module (target module) that does contain the resources. The proxy will access the target module with the entire data in a single burst, once the host has completed the data.

#### 10.2.5.3.1 Targets

##### 10.2.5.3.1.1 Ring Accelerator

The RINGACC provides four distinct offsets to use per ring/queue shown in [Table 10-146](#), totaling 4 kB space per ring/queue. Each channel has access modes (enabled via PROXY\_CTL\_j[17-16] MODE field):

- head access
- tail access
- peeks from head

**Table 10-146. RINGACC Offsets per Ring/Queue**

Offset	Operation
0x0 - 0x1FF	Reads pop from head. Writes push to head.
0x200 - 0x3FF	Writes push to tail. Reads pop from tail.
0x400 - 0x5FF	Reads peek from head. Writes ignored.
0x600 - 0x7FF	Reads peek from tail. Writes ignored. This mode is not supported in this device.
0x800 - 0xFFFF	Reserved

#### 10.2.5.3.2 Proxy Sizes

The number of proxies in the module is configured during SoC design. See [Table 10-142](#). Each proxy has its own address space to the module and determining which proxy is being accessed is simply based on the address bits above the final size of each proxy space. The size of the proxy space is always 4 KB. This is enough to contain the data message, along with controls for which channel, access mode, and element size for the access.

#### 10.2.5.3.3 Proxy Interleaving

Each proxy must have a single owner, as each proxy only has a single buffer to work on the current data. If there are multiple owners in the system, they must each use their own proxy. Then they will have their own buffer and each can access a different resource at the same time. Each proxy is completely independent and kept coherent regardless of accesses to other proxies or regardless of the order of accesses. So the proxies support interleaving of multiple hosts' accesses.

#### 10.2.5.3.4 Proxy Host States

Each individual proxy per host uses a simple state machine to track the currently usage by that host. This state machine is used to track when a proxy is in use or not, but also for error detection. The proxy always stays in

idle state, when there is no activity. There is a write state for when the proxy has been written and contains data that has yet to be written to a resource. There is a read state for when the proxy has read data from a resource and that entire data has not been fully read by the host. When a proxy access completes, final write or final read, then the state will go back to idle. The following sections show the utilization of these states.

#### 10.2.5.3.5 Proxy Host Channel Selection

The host must program the PROXY\_CTL\_j (where j = 0h to 3Fh) prior to accessing the PROXY\_DATA\_j\_y (where j = 0h to 3Fh, y = 0h to 7Fh) unless the previous values are to be reused. This sets the channel within the target, the access mode (head, tail, peek), and the element size. These values will be used when writing the full message to the target, or reading a new message from the target. The target address used will be based on the target base address, the programmed channel, the programmed access mode (for the offset within the 4 kB of the target channel), and the element size (for the offset within the 512 Bytes max size message).

$$\text{target address} = \text{target base} + (\text{channel} \times 4\text{KB}) + (\text{mode} \times 512\text{B}) + (512 - \text{element size in bytes})$$

, for element size in bytes length.

The proxy will detect an error if these fields are changed while in the middle of a message.

#### 10.2.5.3.6 Proxy Host Access

A proxy begins in an idle state, where there is no data in the buffer. In this state the proxy will accept any access to a new resource. If the proxy is in another state, representing a resource is in progress, and the host attempts to access another target or resource, or switches from read to write, then the proxy will detect that as an error and flush the buffer with no final target access and the data is lost and the proxy goes back to idle state.

##### 10.2.5.3.6.1 Proxy Host Writes

The first write to an idle proxy puts the proxy into write mode for the selected target and channel. Further legal writes are stored in the buffer for the enabled bytes and position within the buffer. A write to a previously written location will overwrite the previous data. When there is a legal write to the last byte of the resource, then the entire data is written by the proxy to the target and channel as a single write burst. The proxy will wait for the write status from the target write before it sends the write status for this final byte host write. This allows the host to use the write status to guarantee the entire write data from the proxy has landed at the target. The proxy is then put back into idle state.

##### 10.2.5.3.6.2 Proxy Host Reads

When in idle state, the first read to the proxy will put the proxy into read state, and the proxy will read the target and channel requested in a single burst of the target's channel size. When the data is returned it is stored in the buffer and the accessed part is returned to the host. Further legal reads are allowed and return the accessed data in the buffer. When there is a legal read to the last byte of the resource, the accessed data is returned, and the proxy is put back into idle state.

#### 10.2.5.3.7 Permission Inheritance

The proxy will automatically inherit the permissions from the access and pass these to the target upon the target burst access. The exact transactions that are inherited from are the final byte write or the first read, and these exact permissions are used on the output bus transaction that accesses the target. This will copy the secure, debug, priv, privid, and virtid signals so that they can be firewalled during the target access for the resource.

#### 10.2.5.3.8 Buffer Size

Buffer size is shown in [Table 10-142](#). An attempt to access a larger message will result in the proxy going into error mode. Software must only create messages up to the supported target size.

#### 10.2.5.3.9 Error Events

Each proxy can produce an event when it has detected an error. This error occurs when the host changes the target, resource, or access direction before completing the previous data. An error can also occur for an access to an illegal target, illegal offset within a target, such as beyond the target size or buffer size, or if the

access spans multiple channels. This error does not include accessing reserved register locations within the proxy thread, as that results in normal reserved register behavior of ignoring write data and reading 0s. There is a separate status bit per proxy for these errors. The event number is programmed via `PROXY_EVT_REG_j` (where `j = 0h to 3Fh`) for each proxy. The event will not be sent if the event is programmed to `0xFFFF`.

#### **10.2.5.3.10 Debug Reads**

Since debug reads with `emudbg=1` must not cause any side effects or bus errors, any debug read will just read the buffer and never cause a target read. It will also not return any bus errors.



## 10.2.6 Secure Proxy

This chapter describes the Secure Proxy module in NAVSS.

### 10.2.6.1 Secure Proxy Overview

The Secure Proxy module provides proxy buffers for hosts that need to create large data bursts but can only perform small accesses. Secure Proxy is a modified version of the Proxy module in NAVSS.

**Table 10-147. Secure Proxy Allocation Across Device Domains**

Instance	Domain		
	WKUP	MCU	MAIN
NAVSS0_SEC_PROXY0	-	-	✓ (NAVSS)
MCU_NAVSS0_SEC_PROXY0	-	✓ (MCU NAVSS)	-

#### 10.2.6.1.1 Secure Proxy Features

Secure Proxy supports the following features:

- Provides proxy function to store large data bursts that a host can only access in smaller amounts
- Supports a number of threads, where each has their own independent proxy function
- Keeps the large data burst coherent until the complete data has been accessed
- Allows for interleaved access between multiple hosts or tasks using multiple proxy threads
- Supports a programmable fixed queue for each proxy thread
- Supports multiple producers all writing to the same queue
- Supports a max message count for outbound proxy threads limiting the number of messages a thread can produce
- Supports programmable thresholds for when to generate events.

#### 10.2.6.1.2 Secure Proxy Parameters

Table 10-148 shows the Secure Proxy configuration parameters set during SoC design.

**Table 10-148. Secure Proxy Configuration Parameters**

Module Instance	Parameters				
	Proxies <sup>(1)</sup>	Message Size <sup>(2)</sup>	Target	Channels <sup>(3)</sup>	Sizes <sup>(4)</sup>
NAVSS0_SEC_PROXY0	160	64	NAVSS0_RINGACC0	50	4096
MCU_NAVSS0_SEC_PROXY0	90	64	MCU_NAVSS0_RINGACC0	30	4096

(1) Number of proxy threads supported. Can be read off from the SEC\_PROXY\_CONFIG read-only register

(2) Number of bytes for message. Can be read off from the SEC\_PROXY\_CONFIG read-only register

(3) Number of channels for the target

(4) Number of bytes per channel supported for the target

#### 10.2.6.1.3 Secure Proxy Not Supported Features

The following features are not supported by the module:

- Does not support proxy thread sharing. Each host or task must use its own proxy thread within the module if they can access simultaneously or independently. This includes when the task can be interrupted by another task that also uses the proxy
- In the system, a proxy must be the only method to access messages that use the same set of queues (to keep credits local), which means all hosts using the same set of message queues must also use the same proxy for access and not local proxys
- Does not support multiple proxy threads reading from the same queue. For tracking purposes, there can be only 1 consumer proxy thread per queue.

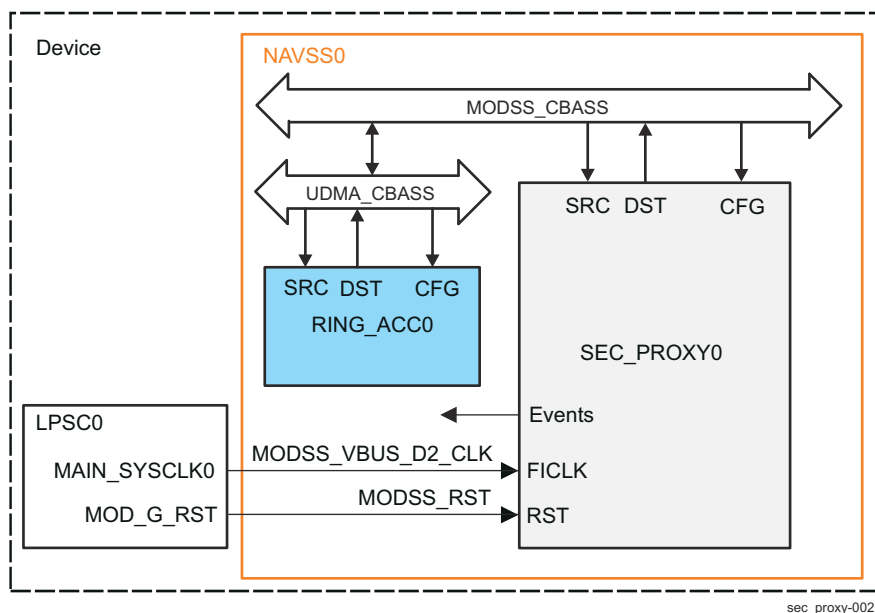
### 10.2.6.2 Secure Proxy Integration

This section describes module integration in the device, including information about clocks, resets, and hardware requests.

#### Note

This section describes the Secure Proxy integration in the main Navigator subsystem (NAVSS0). For MCU\_NAVSS0\_SEC\_PROXY0 integration, please see *MCU Navigator Subsystem (MCU\_NAVSS)*.

Figure 10-29 shows the Secure Proxy integration in the NAVSS



**Figure 10-29. Secure Proxy Integration**

Table 10-149 and Table 10-150 summarize the integration of the module in the device.

**Table 10-149. Secure Proxy Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
NAVSS0_SEC_PROXY0	PSC0	GP	LPSC0	MODSS_CBASS

**Table 10-150. Proxy Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
NAVSS0_SEC_PROXY0	SEC_PROXY0_FICLK	MODSS_VBUS_D2_CLK	MAIN_SYSCCLK0	Proxy clock. This clock is used for all interface and functional operations.
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
NAVSS0_SEC_PROXY0	SEC_PROXY0_RST	MODSS_RST	LPSC0	Proxy hardware reset

**Table 10-151. Proxy Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type

**Table 10-151. Proxy Hardware Requests (continued)**

NAVSS0_SEC_PR - OXY0	-	-	-	No interrupts to external interrupt controllers	-
DMA Events					
Module Instance	Module DMA Event	Destination DMA Event	Destination Input	Description	Type
NAVSS0_SEC_PR - OXY0	-	-	-	No PDMA channels to external DMA engines	-

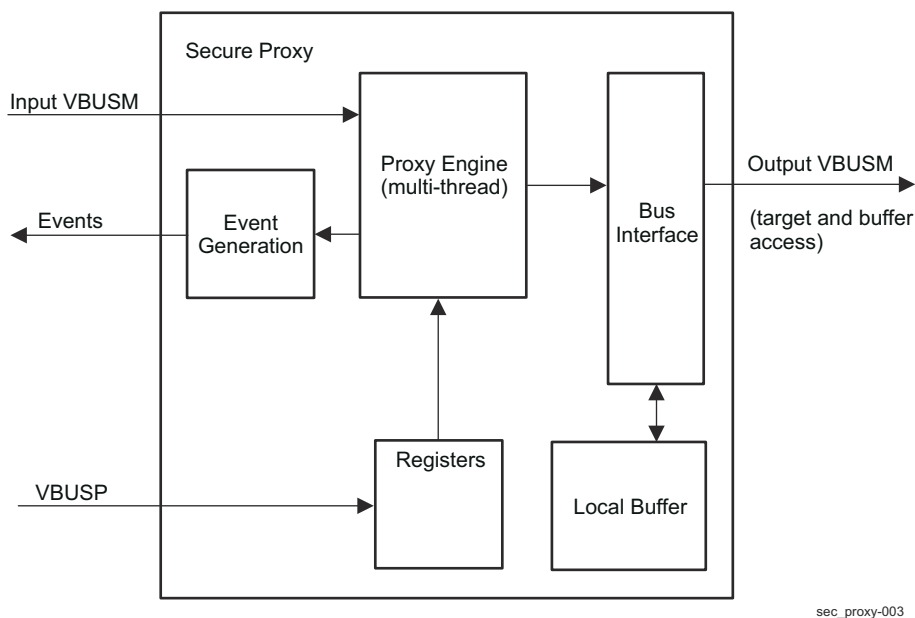
### 10.2.6.3 Secure Proxy Functional Description

This module provides proxy functions for hosts that need to create large data bursts but can only perform small accesses. Examples of this need are:

- large messages (for power control, security control, or IPC)
- data descriptors that are more than a pointer
- DMA Transfer Requests (TR)

All of these require larger data sets, from 12 to 128 bytes, and the storage hardware requires the entire data on the bus in a single burst. Unfortunately, many CPUs cannot produce bursts to device memory, such as registers, and the largest CPU access is typically 8 bytes. Therefore the proxy provides the ability to access the hardware with a single burst, and at the same time allows the CPU to access the same data in smaller accesses.

Figure 10-30 describes Secure Proxy major building blocks.


**Figure 10-30. Secure Proxy Block Diagram**

#### 10.2.6.3.1 Targets

##### 10.2.6.3.1.1 Ring Accelerator

The RING\_ACC provides four distinct offsets to use per ring/queue shown in Table 10-152, totaling 4 KB space per ring/queue.

Secure Proxy always writes to tail, which is the offsets from 0x200 to 0x3FF. Secure Proxy always reads from the head, which is the offsets from 0x000 to 0x1FF. The latter two offsets in RING\_ACC are not used by this proxy. And if the proxy message size is less than 512 bytes, then it always access the upper bytes of those regions, so that the message always ends on the last byte of each region, byte 511.

**Table 10-152. RING\_ACC Offsets per Ring/Queue**

Offset	Operation
0x0 - 0x1FF	Reads pop from head. Writes push to head.
0x200 - 0x3FF	Writes push to tail. Reads pop from tail.
0x400 - 0x5FF	Reads peek from head. Writes ignored. Not used by Secure Proxy
0x600 - 0x7FF	Reads peek from tail. Writes ignored. Not used by Secure Proxy
0x800 - 0xFFFF	Reserved

This means that target writes will use the following calculation:

$$\text{target base address} + (\text{queue} \times 4\text{KB}) + 0x400 - \text{msg\_size} \quad (17)$$

,and burst until offset 0x3FF.

And target reads will use the following calculation:

$$\text{target base address} + (\text{queue} \times 4\text{KB}) + 0x200 - \text{msg\_size} \quad (18)$$

,and burst until offset 0x1FF.

### 10.2.6.3.2 Buffers

The proxy does not store all the buffers internally. It is programmed with a buffer base address (SEC\_PROXY\_BUFFER\_L, SEC\_PROXY\_BUFFER\_H), and all accesses to the buffer are made through the output VBUSM port using that base. The external buffer must be enough to store one *msg\_size* per proxy thread. The proxy only includes a small temporary buffer to save one entire message when accessing the target. Otherwise all host accesses are made to the external buffer. This buffer storage must be considered private for the proxy hardware, so any other access can cause unpredictable results.

#### 10.2.6.3.2.1 Proxy Credits

To support outbound proxy limits, the proxy will use internal credits for tracking resource usage. SEC\_PROXY\_CTL\_j register sets the max credits each outbound proxy thread contains. When a proxy thread sends a message, the current credit count (SEC\_PROXY\_STATUS\_j) decrements by 1. If the current credit count is 0, then the proxy thread is not allowed to send any more messages, and they just remain inside the proxy (so that the host can come back later when a credit has become available and quickly send the same message). When a message that was send by this proxy thread is read by an inbound proxy thread then the credit is returned to this proxy thread and the current credit count increments by 1.

For inbound proxy threads, the credits are used to track pending messages. When an outbound proxy thread sends a message that is read by this proxy thread, then the current credit count of this proxy thread is incremented by 1. A current credit count that is non zero indicates a message is pending and can be read. When the proxy thread completes the read of a message the current credit count is decremented by 1. If the current credit count is zero, then the proxy thread will read an empty message where all the data is 0s. The credit count saturates at 255, so if there can be more messages in the system for that proxy thread than 255 then multiple proxy threads should be used, or the count can become mismatched after a saturate.

#### 10.2.6.3.2.2 Proxy Private Word

To support tracking of credits from each proxy thread, the proxy will extract the first word from the total message size for private tracking usage. This will include saving the proxy thread ID used within the proxy. This is necessary so that when a proxy reads a new message from the target it can inspect this word and return the credit to the source proxy thread. This allows the outbound proxy threads to have a set number of message credits, which decrement when a new message is written, and increment when a message from that proxy thread is read, thus keeping the credits in the system coherent. Any data written to this first word of the message will be ignored, so software should not write to the first word of messages through the proxy. Consumer software can use the first word to identify the source proxy thread of the message (SEC\_PROXY\_DATA\_j).

#### **10.2.6.3.2.3 Completion Byte**

The proxy completes a message when the completion byte is accessed. This completion byte is the final byte in the full size of the message. This byte can be written by any size access, whether byte, 32-bit, or 64-bit word. The write of a message is not complete until this final byte of the message is written, and only then will the message be sent to the target. The read of a message is not complete until this final byte of the message is read by the host, and only then will reading a new message be allowed. Even if the final byte of the message is not needed, it must be accessed to complete the message inside the proxy hardware.

#### **10.2.6.3.3 Proxy Thread Sizes**

There are a configurable number of proxy threads in the module. Each thread has its own address space to the module and determining which proxy thread is being accessed is simply based on the address bits above the final size of each proxy thread space. Since each proxy thread will be programmed to a single resource, the only space requirement is that for using 4 KB to match typically used MMU page sizes. Only the first N bytes, matching the target channel size, is usable.

#### **10.2.6.3.4 Proxy Thread Interleaving**

Each proxy thread must have a single owner, as each proxy thread only has a single buffer to work on the current data. If there are multiple owners in the system, they must each use their own proxy thread. Then they will have their own buffer and each can access a different resource at the same time. Each proxy thread is completely independent and kept coherent regardless of accesses to other proxy threads or regardless of the order of accesses. So the proxy support interleaving of multiple hosts' accesses.

#### **10.2.6.3.5 Proxy States**

Each proxy thread uses a simple state machine to track the currently usage by that host. This state machine is used to track when a proxy thread is in use or not, but also for error detection. The proxy thread always starts in idle state, when there is no currently activity. There is a write state for when the proxy thread has been written and contains data that has yet to be written to a resource. There is a read state for when the proxy thread has read data from a resource and that entire data has not been fully read by the host. When a proxy thread access completes, final write or final read, then the state will go back to idle. The following sections show the utilization of these states.

#### **10.2.6.3.6 Proxy Host Access**

A proxy thread begins in an idle state, where there is no data in the buffer. In this state the proxy thread will accept an access to the resource.

##### **10.2.6.3.6.1 Proxy Host Writes**

The first write to an idle proxy thread puts the proxy thread into write mode for the selected channel. Further legal writes are stored in the buffer for the enabled bytes and offset within the buffer. A write to a previously written location will overwrite the previous data. The buffer is not cleared initially, so if a byte is not written, the message will include previous buffer data. When there is a legal write to the last byte of the resource, then the entire data is written by the proxy to the target and channel as a single write burst. The proxy will wait for the write status from the target write before it sends the write status for this final byte host write. This allows the host to use the write status to guarantee the entire write data from the proxy has landed at the target. The proxy thread is then put back into idle state. A write completion when there are no credits available results in the message not being written to the resource and the proxy thread remains in write mode (so that the message can be quickly completed when a credit becomes available).

##### **10.2.6.3.6.2 Proxy Host Reads**

When in idle state, the first read to the proxy thread will put the proxy into read state, and the proxy will read the target and channel requested in a single burst of the target's channel size. When the data is returned it is stored in the buffer and the accessed part is returned to the host. Further legal reads are allowed and return the accessed data in the buffer. When there is a legal read to the last byte of the resource, the accessed data is returned, and the proxy thread is put back into idle state. If an initial read is made and there are no pending messages, the proxy thread will return an empty message, where the data is all 0s.

#### **10.2.6.3.6.3 Buffer Accesses**

Each read or write will actually trigger a read or write to the external buffer space allocated for that proxy thread, where the real message data is stored. In addition, when a write message completes, the external buffer will be read for the entire message for that proxy thread so that it can then be written to the target. Similarly, after a read causes a read from the target to get a new message, the entire message is written to the external buffer to store the message to the buffer.

#### **10.2.6.3.6.4 Target Access**

After the write has completed and the external buffer read for the entire message, the entire message is then written to the target as a single burst. Similarly, for a read that needs a new message, the target is read for a full message as a single burst.

#### **10.2.6.3.6.5 Error State**

If an error is detected in a proxy thread, the proxy will put that proxy thread into error state, and the current buffer will be lost. A status bit will be set that indicates the proxy thread is in error. All accesses to that proxy thread will be ignored until the status bit is cleared. Once cleared the proxy thread is put back into idle state. Errors are: an access to an offset beyond the length of the message, or a mismatch in the direction of access against what was programmed such as reading an outbound thread or writing an inbound thread.

#### **10.2.6.3.7 Permission Inheritance**

The proxy will automatically inherit the permissions from the access and pass these to the target upon the target burst access. The exact transactions that are inherited from are the final byte write or the first read, and these exact permissions are used on the output bus transaction that accesses the target.

#### **10.2.6.3.8 Resource Association**

Each proxy thread has a register (SEC\_PROXY\_CTL\_j) to define the resource within the target that the proxy thread is mapped to. The host has no ability to access any other resource in the target. This allows the setup of the proxy to determine the resources that all the hosts use.

#### **10.2.6.3.9 Direction**

Each proxy thread has a register (SEC\_PROXY\_CTL\_j) to define the direction for access. It can either be an outbound proxy thread for writing messages to resources. Or it can be an inbound proxy thread for reading messages from resources. The host must match the direction programmed or it will result in a proxy thread error.

#### **10.2.6.3.10 Threshold Events**

Each proxy thread can produce events based on the state of the resource it uses. For outbound proxy threads, a threshold can be set so that an event is produced when there are now at least N messages available to write. For inbound proxy threads, a threshold can be set so that an event is produced when there are now at least N messages available to read. These threshold values are programmed in SEC\_PROXY\_EVT\_MAP\_j, allowing the host to determine how often it gets events. Programming the count values while the threshold is set to 0 will not trigger events, but if the threshold is not 0 then programming the counts will trigger events based on whether the threshold is reached or not. No event is generated if the event is programmed to 0xFFFF.

#### **10.2.6.3.11 Error Events**

Each proxy thread can produce an event when it has detected an error. This error occurs when the host violates the programmed setup of the proxy thread or accesses beyond the message size. There is a separate status bit per proxy thread for these errors. No event is generated if the event is programmed to 0xFFFF.

#### **10.2.6.3.12 Bus Errors and Credits**

If a target read for a message returns a bus error, this indicates the read was bad and the data is corrupt and usually 0s. Because the private word is contained in the message, if the private word is corrupted then the source proxy thread is not valid and the credit return will not be correct. Since the data is usually 0, this means the credit would be returned to proxy thread 0 incorrectly most of the time. And the credit will be lost for the actual source proxy thread of the message that was not read correctly. The hardware has no way to correct for

this, as the message from the target is corrupt so it does not know the correct owner of the credit. If software detects that a read message is corrupt, it can attempt to correct the situation by resetting the affected proxy threads using that queue, to restore their credit counts back to the default. Software may also want to reset proxy thread 0 since it may be getting extra credits.

#### **10.2.6.3.13 Debug**

When the transaction has the *emudbg* set for a read, all side effects will not occur. This means that the read will not cause a target read since that would affect the target, and instead current buffer will be read even if the previous message has completed.



## 10.2.7 Interrupt Aggregator (INTR\_AGGR)

This chapter describes the INTR\_AGGR module, part of NAVSS.

### 10.2.7.1 INTR\_AGGR Overview

The Interrupt Aggregator (INTR\_AGGR or INTA) provides a centralized machine which handles the termination of system events to that they can be coherently processed by the host(s) in the system. The main function of the module is to convert between 'global events' (assertion and de-assertion events transmitted on the event-transport-lane (ETL) bus) and 'local events' (level sensitive signals on output, and level or edge sensitive signals on input).

**Table 10-153. INTR\_AGGR Allocation Across Device Domains**

Instance	Domain		
	WKUP	MCU	MAIN
NAVSS0_INTR_AGGR0	-	-	✓ (NAVSS)
NAVSS0_INTR_AGGR1	-	-	✓ (NAVSS)
NAVSS0_UDMASS_INTR_AGGR0	-	-	✓ (NAVSS)
MCU_NAVSS0_UDMASS_INTR_AGGR0	-	✓ (MCU NAVSS)	-

#### 10.2.7.1.1 INTR\_AGGR Features

NAVSS0\_INTR\_AGGR0 supports the following features:

- 64-bit VBUSP slave using 64-bit registers
- Provides a set of *TI Interrupt Architecture* compliant interrupt status and mask registers which are used to pass specific event status to one or more host blocks.
  - Maps a collection of "DMA Messaged Events" which are input through an Event Transport Lane (ETL) into specific bit locations in one or more interrupt cause registers
  - Mapping is performed based on a programmable table which provides a single location for each ordinal input event number (0 through sevt\_cnt-1)
    - Table specifies a specific bit in a specific cause register that the event should set or clear depending on the type of event (up vs down)
    - Tracks a single 'event up'/'event down' condition. There is no event counting. (The 'cnt' field of the ETL is ignored)
  - Tracked status interrupts are presented to the user through a standard interrupt register interface
    - Provides separate 'enable set' and 'enable clear' registers
    - Provides both masked and unmasked interrupt status
    - Interrupt status bits can be manually cleared using 'write 1 to clear'
- Provides a set of Global Event Input (GEVI) counters which can count events which were delivered via an ingress Event Transport Lane (ETL)
  - Each counted event provides an register by which the count can be read and an register by which a specified amount can be decremented by the host
  - Each counted event generates a egress Global event on a zero to non-zero and non-zero to zero transition, controllable through a mapping register
- Provides a set of Local Event Input (LEVI) to Global event registers which can be used to convert pulsed discrete interrupt inputs or clock synchronous rising edge events into Global events on an egress ETL
  - Each ingress event index provides a configurable 'pulse' or 'rising edge' bit, plus the configurable index of the generated Global event
- Provides a set of Global Event Input (GEVI) 'Multicast' registers which can take a Global event from an ingress ETL and generate two egress Global events on two egress ETL interfaces.



### 10.2.7.1.2 INTR\_AGGR Parameters

Table 10-154 shows the MODSS Interrupt Aggregators 0 and 1, and UDMASS Interrupt Aggregator 0 parameters set during design time. 4 local (LEVI) interrupts are from the MCRC0 module. 48 LEVI interrupts are from the MAILBOX0 module. See INTR\_AGGR Integration.

**Table 10-154. Interrupt Aggregators Parameters**

Module Instance	Parameters				
	VINTR <sup>(1)</sup>	SEVI <sup>(2)</sup>	GEVI <sup>(3)</sup>	LEVI <sup>(4)</sup>	MEVI <sup>(5)</sup>
NAVSS0_INTR_AGGR0	64	1024	0	0	0
NAVSS0_INTR_AGGR1	64	1024	0	0	0
NAVSS0_UDMASS_INTR_AGGR0	256	4608	512	84	512
MCU_NAVSS0_UDMASS_INTR_AGGR0	256	1536	256	12	128

- (1) VINTR – Number of Virtual Interrupt outputs. These are connected to the interrupt router inputs. Value can be read from INTA\_INTCAP/UDMA\_INTA\_INTCAP register.
- (2) SEVI – Number of Main (Steerable) Events. Value can be read from INTA\_INTCAP/UDMA\_INTA\_INTCAP register.
- (3) GEVI – Number of Countable Events. Value can be read from INTA\_AUXCAP/UDMA\_INTA\_AUXCAP register.
- (4) LEVI – Number of Local Event inputs. Value can be read from INTA\_AUXCAP/UDMA\_INTA\_AUXCAP register.
- (5) MEVI – Number of Multicast Events. Value can be read from INTA\_AUXCAP/UDMA\_INTA\_AUXCAP register.

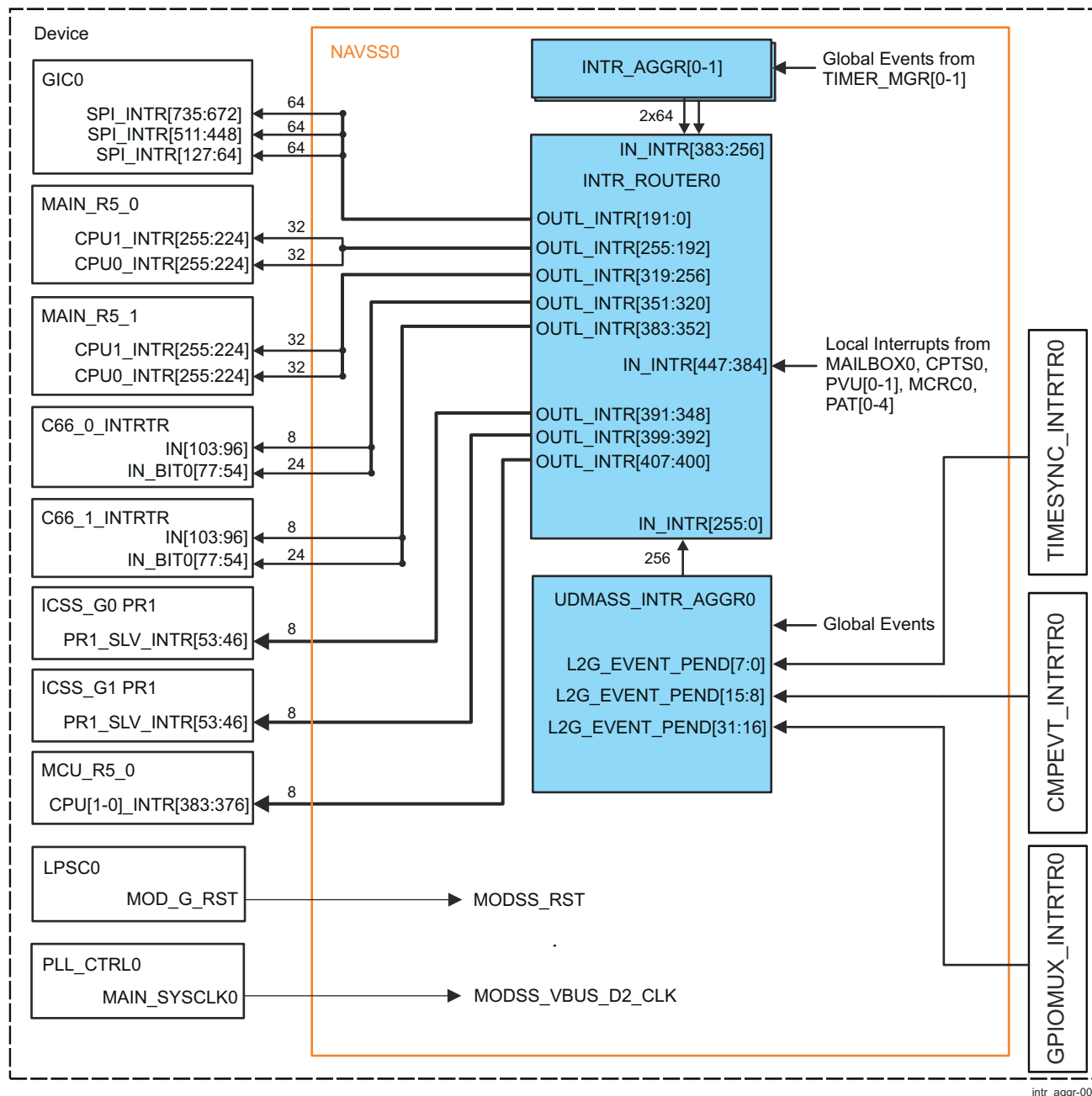
### 10.2.7.2 INTR\_AGGR Integration

This section describes module integration in the device, including information about clocks, resets, and hardware requests.

#### Note

This section describes the INTR\_AGGR integration in the main Navigator subsystem (NAVSS0). For MCU\_NAVSS0\_INTR\_AGGR0 integration, please see *MCU Navigator Subsystem (MCU\_NAVSS)*.

Figure 10-31 shows the INTR\_AGGR integration in the device.



intr\_aggr-002

**Figure 10-31. NAVSS0\_INTR\_AGGR0 Integration**

Table 10-155 and Table 10-156 summarize the integration of the module in the device.

**Table 10-155. INTR\_AGGR Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
NAVSS0_UDMASS_INTR_AGGR0	PSC0	GP	LPSC0	UDMASS_CBASS
NAVSS0_INTR_AGGR0	PSC0	GP	LPSC0	MODSS_CBASS
NAVSS0_INTR_AGGR1	PSC0	GP	LPSC0	MODSS_CBASS

**Table 10-156. INTR\_AGGR Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
NAVSS0_UDMASS_INTR_AGGR0	INTR_AGGR0_FICLK	MODSS_VBUS_D2_CLK	MAIN_SYSCLK0	INTR_AGGR clock. This clock is used for all interface and functional operations.
NAVSS0_INTR_AGGR0	INTR_AGGR0_FICLK	MODSS_VBUS_D2_CLK	MAIN_SYSCLK0	INTR_AGGR clock. This clock is used for all interface and functional operations.
NAVSS0_INTR_AGGR1	INTR_AGGR0_FICLK	MODSS_VBUS_D2_CLK	MAIN_SYSCLK0	INTR_AGGR clock. This clock is used for all interface and functional operations.
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
NAVSS0_UDMASS_INTR_AGGR0	INTR_AGGR0_RST	MODSS_RST	LPSC0	INTR_AGGR hardware reset
NAVSS0_INTR_AGGR0	INTR_AGGR0_RST	MODSS_RST	LPSC0	INTR_AGGR hardware reset
NAVSS0_INTR_AGGR1	INTR_AGGR0_RST	MODSS_RST	LPSC0	INTR_AGGR hardware reset

**Table 10-157. INTR\_AGGR Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
NAVSS0_UDMASS_INTR_AGGR0	UDMASS_INTA0_VINTR_PEND[255:0]	INRTR_IN[255:0]	INTR_ROUTER0	Global/local events	Level
NAVSS0_INTR_AGGR0	MODSS_INTA0_VINTR_PEND[63:0]	INRTR_IN[383:320]	INTR_ROUTER0	Global events from TIMER_MGR0	Level
NAVSS0_INTR_AGGR1	MODSS_INTA1_VINTR_PEND[63:0]	INRTR_IN[319:256]	INTR_ROUTER0	Global events from TIMER_MGR1	Level
L2G Interrupt Request Inputs					
Module Instance	Module Interrupt Signal	Source Interrupt Input	Source	Description	Type
NAVSS0_UDMASS_INTR_AGGR0	L2G_EVENT_PEND[7:0]	TIMESYNC_INTRTR0_OUTL[47:40]	TIMESYNC_INTRTR0	Interrupts from TIMESYNC_INTRTR0	Level
	L2G_EVENT_PEND[15:8]	CMPEVT_INTRTR0_OUTP[31:24]	CMPEVT_INTRTR0	Interrupts from CMPEVT_INTRTR0	Level
	L2G_EVENT_PEND[31:16]	GPIOMUX_INTRTR0_OUTP[31:16]	GPIOMUX_INTRTR0	Interrupts from GPIOMUX_INTRTR0	Level
DMA Events					
Module Instance	Module DMA Event	Destination DMA Event Input	Destination	Description	Type
NAVSS0_UDMASS_INTR_AGGR0	-	-	-	No PDMA channels to external DMA engines	-
NAVSS0_INTR_AGGR0	-	-	-	No PDMA channels to external DMA engines	-
NAVSS0_INTR_AGGR1	-	-	-	No PDMA channels to external DMA engines	-

Table 10-158 shows the local interrupt inputs (LEVI) and external L2G inputs.

**Table 10-158. NAVSS0\_UDMASS\_INTR\_AGGR0 LEVI and L2G Local Interrupts**

Interrupt Mapping	Local Interrupt Bits
MCRC Events	0-3
Mailbox 0 Cluster 0 Events	4-7
Mailbox 0 Cluster 1 Events	8-11
Mailbox 0 Cluster 2 Events	12-15
Mailbox 0 Cluster 3 Events	16-19
Mailbox 0 Cluster 4 Events	20-23
Mailbox 0 Cluster 5 Events	24-27
Mailbox 0 Cluster 6 Events	28-31
Mailbox 0 Cluster 7 Events	32-35
Mailbox 0 Cluster 8 Events	36-39
Mailbox 0 Cluster 9 Events	40-43
Mailbox 0 Cluster 10 Events	44-47
Mailbox 0 Cluster 11 Events	48-51
External L2G Events	52-83

### 10.2.7.3 INTR\_AGGR Functional Description

The Interrupt Aggregator provides a centralized machine which handles the termination of system events to that they can be coherently processed by the host(s) in the system. Both the signaling and content of K3 system events are incompatible with standard interrupt controllers. The INTR\_AGGR provides mechanisms which convert the TI proprietary system events to standard level sensitive pending bits which can be used by the Interrupt Router and all downstream interrupt infrastructure. Events can be presented to the INTR\_AGGR in two different forms:

- Events may be active high pulses, or clock synchronous rising edges, which are input on separate orthogonal pins. These are called 'Local Events'.
  - Local event signals that are used in pulse counting mode must be sourced by the same clock as the interrupt aggregator.
  - Local event signals that are used in edge counting mode must be sourced by pseudo-synchronous sources (and be a slower clock multiple) to the interrupt aggregator clock, and they must remain high for at least 1 'slow' clock cycle.
- Events may be messages which are input in time division multiplexed sequential order from an Event Transport Lane. These are called 'Global Events'.

In both cases, these events need to be conditioned to transition them from a transient indicator that something occurred to a persistent and reliable indication of the current state of the system. The INTR\_AGGR is architected to perform this conversion in an efficient and cost effective manner.

#### 10.2.7.3.1 Submodule Descriptions

##### 10.2.7.3.1.1 Status/Mask Registers

The Status and Mask Registers block is responsible for providing persistent storage for the interrupt status and mask bits and for formatting those in a way that is compliant to the *TI Interrupt Architecture* requirements. These requirements include the ability to set and clear bits orthogonally and to provide a masked version of the status register that corresponds to the supplied bit mask for each register.

##### 10.2.7.3.1.2 Interrupt Mapping Block

The Interrupt Mapping Block accepts ordinally numbered events (0-N) and converts those ordinal event numbers into a status register number and bit number pair that is then used to manipulate the specified bit in the Status Registers block.

##### 10.2.7.3.1.3 Global Event Input (GEVI) Counters

The GEVI Counters block is responsible for accepting an Event Transport Lane events and summing the total message counts for each received event index. The module creates global egress events for zero to non-zero count up' events and non-zero to zero 'count down' events. The egress global event index is configured via GEVI UDMA\_INTA\_MCMAP\_j register, so count status egress events can optionally be fed back into the INT\_AGGR, via the ETL switch fabric.

##### 10.2.7.3.1.4 Local Event Input (LEVI) to Global Event Conversion

The Local to Global event block is responsible for accepting an array of input pins and independently counting the number of cycles for which those pins are asserted high, or by counting individual clock synchronous rising edge events. The counting mode is configurable on a 'per pin' basis. The events are converted to egress Global events on an egress ETL. Global events can optionally be fed back into the INT\_AGGR, via the ETL switch fabric.

##### 10.2.7.3.1.5 Global Event Multicast

The Global event multicast block takes an ingress global event from an ingress ETL, and maps it into two egress Global events on two egress ETL interfaces. Global events can optionally be fed back into the INT\_AGGR, via the ETL switch fabric.

### 10.2.7.3.2 General Functionality

#### 10.2.7.3.2.1 Event to Interrupt Bit Steering

Each time an event message is received on the Main Event Transport Lane interface, the interrupt mapping block performs a direct lookup into an SRAM using the event number as the address. The SRAM stores a corresponding raw interrupt status register and bit number within that register which are to be manipulated anytime a message is received indicating something occurred on that specific event. When an *up* event is received, the specified bit in the *j* Status register will be set. When a *down* event is received, that same bit will be cleared. This block is what allows flexible aggregation of various system events into an array of bits in the interrupt status registers. It is intended that this mapping is essentially static - set up when a resource is allocated and left untouched until the resource is no longer needed. Note that the 'cnt' field of the ETL is ignored and no interrupt counting is performed here. When 'UpDn=1', the interrupt flag bit is set, and when 'UpDn=0', the flag bit is cleared.

#### 10.2.7.3.2.2 Interrupt Status

The event messages which are generated from the event to interrupt bit steering logic are input to the Interrupt Status registers. Each time an event is received, the interrupt status registers machine will assert or de-assert the specified bit in the specified register. The assertion and de-assertion in the interrupt status register is unaffected by the interrupt enable state. When an up event is received, the corresponding bit is set and when a down event is received, the corresponding bit is cleared. Some sources of input events will not include the ability to send a down event. In these cases, the interrupt router provides the ability to clear the status bits through the Interrupt Source Clear register. The host will write a one to the specific bit in the register which is to be cleared and the interrupt status machine will clear the bit internally. It is not intended that the Host directly clear bits which are automatically cleared via down events from the source itself.

#### 10.2.7.3.2.3 Interrupt Masked Status

Interrupt enable bits are used in conjunction with the interrupt status bits to create the interrupt masked status register values. The interrupt masked status register (INTA\_STATUSM\_j) contains the value of the interrupt status ANDed with the value of the interrupt enable register. Each time a new event message is received from the event to interrupt bit steering logic or the interrupt enable register is modified, the interrupt masked status register is re-evaluated.

#### 10.2.7.3.2.4 Enabling/Disabling Individual Interrupt Source Bits

Separate (INTA\_ENABLE\_SET\_j and INTA\_ENABLE\_CLEAR\_j) registers are provided to allow individual enable bits to be enabled or disabled without the need for a read-modify-write operation. When the INTA\_ENABLE\_SET\_j register is written, all bits within written bytes which are 1 will cause corresponding bits in the internal enable register to be set. When the INTA\_ENABLE\_CLEAR\_j register is written, all bits within written bytes which are 1 will cause corresponding bits in the internal enable register to be cleared.

#### 10.2.7.3.2.5 Interrupt Output Generation

Each interrupt masked status register (INTA\_STATUSM\_j) is accompanied by a single pending bit which indicates if any enabled interrupt source within the corresponding interrupt status register is currently asserted. These pending bits from the interrupt masked status registers are collectively output from the INTR\_AGGR as the VINTR\_PEND bus. The interrupt status outputs will always be updated to reflect the current values of the INTA\_STATUSM\_j register. This is accomplished by triggering output updates on write operations to the internal RAM that holds the current raw status and enable masks.

#### 10.2.7.3.2.6 Global Event Counting

When enabled, the INTR\_AGGR will provide a set of counters which will track how many outstanding messages have been observed for each ingress event index from the ingress Counted Global Event Transport Lane (ETL) interface. For each message received where the 'UpDn' flag is set, the corresponding counter will be incremented by value of the 'cnt' field of ingress message. Events where the 'UpDn' flag is cleared are ignored. The counter is made visible to software in the UDMA\_INTA\_COUNT\_j register. When software has read the count, it can acknowledge that count has been seen and processed by writing back an integer value specifying the amount by which the counter should be decremented, and the counter will subtract that value from the

current count (which may have updated since it was read). The count will saturate at a value of 0xFFFFFFFF. Writing a count 'ack' value higher than the one read is not supported and will produce non-deterministic results.

When a count transitions from zero to a non-zero value, a Global Up 'UpDn=1' event is sent out an egress ETL interface. When a count transitions from a non-zero value to zero, a down 'UpDn=0' event will be sent. The index of the Global event is mappable on a 'per-counter' basis, using the UDMA\_INTA\_MAP\_j register. The event may be mapped such that it then re-enters INTR\_AGGR's 'Event to Interrupt Bit Steering Block' to generate an interrupt to the host. The event routing is handled by an external event switch fabric. Note that writing a new egress event index via the UDMA\_INTA\_MAP\_j register does not alter the stored event count for the ingress event register index.

#### **10.2.7.3.2.7 Local Event to Global Event Conversion**

When enabled, the INTR\_AGGR will provide a set of Local to Global event conversion registers. Local events may be discrete clock synchronous pulses or clock synchronous rising edges. The counting mode is configurable on a 'per pin' basis. In pulsed mode, the module will track how many outstanding clock cycle long pulses have been observed on each of the provided LEVI interface pins. For each cycle in which a LEVI input pending bit is asserted the corresponding counter will be incremented by 1. In 'rising edge' mode, the number of rising edge transitions on the pin is counted.

The module will generate egress Global events on its egress ETL, with a different global index configured for each ingress LEVI pin. The LEVI pins numbers (1-N) are converted to arbitrary global event indices via the global event mapping UDMA\_INTA\_MAP\_j registers. The egress Global events can themselves be mapped to re-enter the INTR\_AGGR's Global Event Counting block, so that the counts can then be made visible to software in the GEVI count registers as described in [Section 10.2.7.3.1.3](#). The event routing is handled by an external event switch fabric.

#### **10.2.7.3.2.8 Global Event Multicast**

In UDMASS\_INTR\_AGGR0, the INTR\_AGGR will provide a set of Global event multicast (UDMA\_INTA\_MCMAP\_j) registers. These registers allow an ingress Global event on its ingress ETL interface to be mapped into two egress Global events on two separate egress ETL interfaces. A set of registers map the ingress Global event index (1-N) into two arbitrary egress event indices.

## 10.2.8 Packet Streaming Interface Link (PSI-L)

This section describes the Packet Streaming Interface Link (PSI-L) for the device.

### 10.2.8.1 PSI-L Overview

PSI-L is a packet based protocol used to transfer data between several device modules. Each transaction is routed based on thread ID value instead of physical address. All PSI-L interfaces are point to point direct connections to NAVSS0 or MCU\_NAVSS0. All switching functions among the PSI-L based interfaces are done inside NAVSS0 and MCU\_NAVSS0.

The PSI-L also has an event interface referred to as ETL, which is purely used to transfer event information. Only MCU\_NAVSS0 has such event interface. It is connected to WKUP\_DMSC0 and allows MCU\_NAVSS0 to send event information to the interrupt aggregator inside WKUP\_DMSC0.

[Figure 10-32](#) shows an overview of all PSI-L connections within the device.

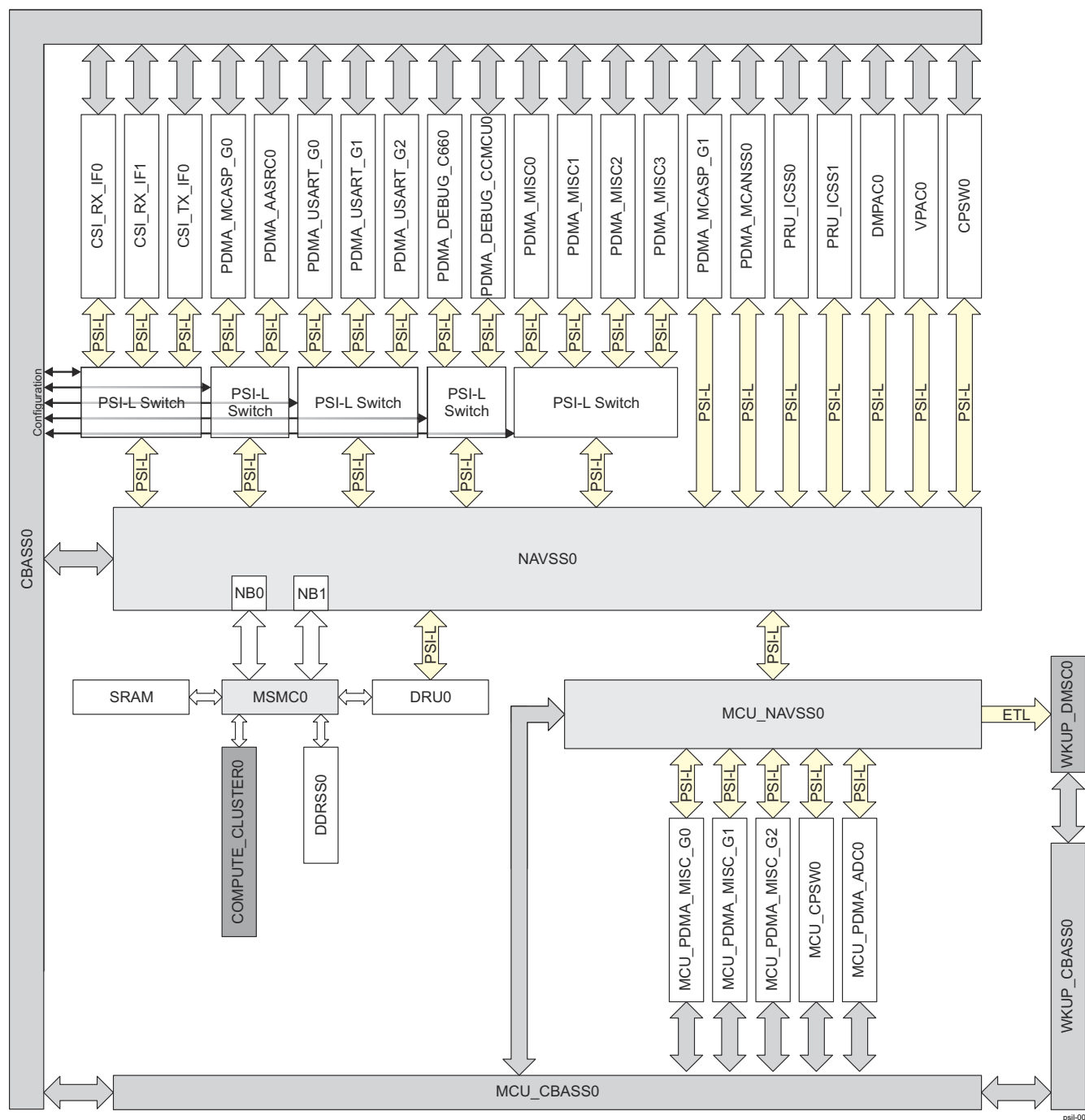
---

#### Note

For more information about the PSI-L switch (PSILSS), see *PSIL Subsystem (PSILSS)*.

---





### Figure 10-32. PSI-L Overview

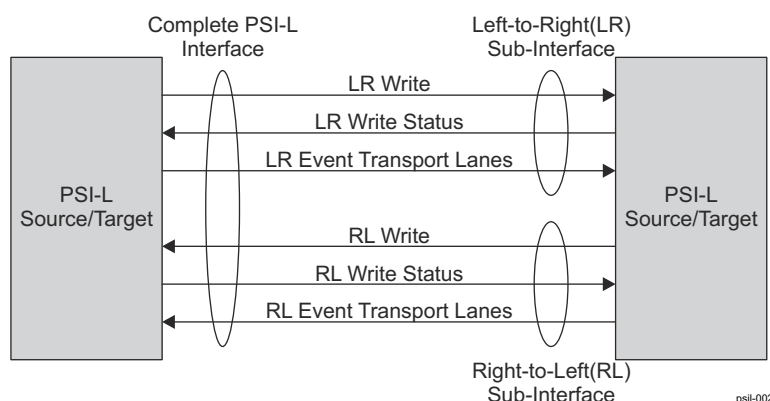
## 10.2.8.2 PSI-L Functional Description

### 10.2.8.2.1 PSI-L Introduction

The PSI-L is used to transfer words of packet data and control information between two peer entities via direct connection. There are credits used for flow control to guarantee that blocking does not occur between threads. Multiple packet transfers can be ongoing simultaneously across a PSI-L interface each on a separate logical thread but all sharing a single datapath through time division multiplexing. Each packet which is transferred is accompanied by a destination thread ID which indicates the logical destination thread to which the packet is being sent.

There is low level hardware handshake between PSI-L master and slave used for transferring data. As it is blocking by nature, a higher level protocol is also used in which credits are maintained for each thread connection. Moments of non-readiness are allowed on the interface but there is no possibility that one thread can block another for an indefinite time since a thread must have credit for the destination before it is allowed to use the interface.

Figure 10-33 illustrates the point to point PSI-L connection.



**Figure 10-33. PSI-L Connection**

PSI-L is intended to be a versatile interface which can carry various types of communications. Table 10-159 describes several types of threads which have been defined.

**Table 10-159. Types of PSI-L Threads**

Type	Description	Example Endpoints
Queue Management (QM) Messaging	Provides transport for queue push, pop, and divert messages to support distributed queue based work passing	Centralized queue manager or message manager and distributed DMA clients
Direct IO	Allows for tunneling CBASS compatible bus transactions through a PSI interconnect	Module which has no CBASS master to a remote proxy block
DMA Control Transport	Provides transport for DMA transfer descriptions from a source thread to a destination thread	DMA channel controller to DMA transfer controller
DMA Data Transport	Provides transport for packed data from a source thread to a destination thread	Packet oriented module like networking adapters to or from centralized DMA

Each thread on PSI-L is established between a single thread master and a single thread slave using a connection oriented transfer mechanism. A thread master and a thread slave are "paired" to create a logical connection between them referred to as a thread. Pairing is accomplished using a Pair Configuration Message which is valid in all thread types. Pairing can only occur between a thread master and a thread slave of the same type.

#### 10.2.8.2.2 PSI-L Operation

The PSI-L allows multiple independent streams of packet data (threads) to share a single interface using data phase granularity time division multiplexing. This time division multiplexing is accomplished by dividing each stream into single data transfers. In this way a stream is comprised of a sequence of strongly ordered data transfers but packet transfers between different streams are allowed to be interleaved. Each data phase is qualified with a type identifier which indicates whether the data is packet info, direct IO write or read info, timestamp, software info, control data, payload data, or status.

##### 10.2.8.2.2.1 Event Transport

PSI-L provides a mechanism by which events can be transported within one or more optional sub-interfaces on a link. These sub-interfaces are referred to as event transport lanes (ETLs) and are independent of any PSI-L packet or message transfers which may be ongoing on the other interfaces within the PSI-L. A maximum of one event can be transferred on each event transport lane in each clock cycle. The event transport protocol is a simple request-ready type of hardware handshake.

##### 10.2.8.2.2.2 Threads

A thread is a complete flow-controlled stream of communication. The PSI-L thread space is divided into a 32K contiguous region representing all of the source threads (0x0000 - 0x7FFF) and a 32K contiguous region representing all of the destination threads (0x8000 - 0xFFFF).

Source threads are responsible for sending request transactions, accepting response transactions, and sending data transfer transactions. Destination threads are responsible for accepting request transactions, sending response transactions, and accepting data transfer transactions. A given thread generally performs only a single class of transactions (see [Table 10-159](#)). Both source and destination threads may act as masters for transfers on one of the PSI-L write sub-interfaces but in this case source threads are actually initiating data transfers while destination threads are only responding to a previous request issued by a source request. Source threads only transfer to destination threads and destination threads only transfer to source threads. Transfers between same thread types do not occur.

The following types of message transfers always originate from source threads and terminate in destination threads:

- Configuration write message
- Configuration read message
- QM push message
- QM pop message
- QM divert message
- Direct read operation message
- Direct write operation message
- DMA transfer request message
- DMA data message

The following types of message transfers always originate in destination threads and terminate in source threads:

- Configuration write response message
- Configuration read response message
- QM push response message
- QM pop response message
- QM divert response message
- DMA transfer response message

##### 10.2.8.2.2.3 Arbitration Protocol

On each cycle, an arbiter built into the master side of each interface decides on which thread to transfer data in the next clock cycle. Both forward (from source threads) and reverse (from destination threads) direction transfers can occur across the same physical interface so the arbiter must ensure that sustained blocking of any thread can occur.

The decision about which thread to allow using the interface is based on the following factors:

1. Whether data is ready at the source to be sent
2. Whether credit exists for the thread such that it is guaranteed there is a place for the data to land in the destination endpoint. For reverse direction transfers (response messages), it is assumed that credit always exists.
3. The relative priority of the traffic that the thread is carrying for the given packet duration
4. Whether the transfer is for a forward or reverse direction thread (requests versus responses).

No thread can be considered for inclusion in arbitration unless both data is available and credit exists in the destination endpoint. For each thread which can be considered in the current arbitration cycle, the arbiter should then pick the thread with the highest priority. Reverse transfers should be considered higher priority than forward transfers.

#### **10.2.8.2.2.4 Thread Configuration**

Each source thread has programmable registers for configuring the pairing with a corresponding destination thread and for enabling data flow. Each destination thread has programmable registers for reporting the buffering capability of the thread and for enabling data flow. These registers are accessed using configuration read and write messages from a configuration host.

Each configuration write message sent from a configuration host is acknowledged with a configuration write response message from the addressed endpoint. Each configuration read message sent from a configuration host is acknowledged with a configuration read response message from the addressed endpoint.

##### **10.2.8.2.2.4.1 Thread Pairing**

Depending on the type of traffic that each source thread carries it may or may not have a fixed pairing with a destination thread.

##### **10.2.8.2.2.4.1.1 Configuration Transaction Pairing**

Configuration write and read transactions are only initiated by a special module called configuration proxy (CFG\_PROXY). Each configuration proxy in the system uses a single source thread which can initiate configuration write and read transactions to the configuration registers of every other source and destination thread in the PSI-L system. The CFG\_PROXY registers are described in *PSI-L CFG\_PROXY Registers*.

Each source thread routes to a single target thread and this pairing is specified by programming a required pairing register set for each thread. A source has no ability to change what target thread it talks to. If a source must be able to talk to different targets (or target threads), then it can either have multiple threads or a streaming switch that can be programmed to route the transfers appropriately based on specific packet fields.

##### **10.2.8.2.2.4.2 Configuration Registers Region**

The configuration registers region contains static configuration information including the settings for linking a thread source to a thread destination. These registers are described in *PSI-L Configuration Registers*. The thread ID mapping is shown in *Navigator Subsystem (NAVSS)*.

## 10.2.9 PSIL Subsystem (PSILSS)

This chapter describes the PSIL subsystem (PSILSS) in the device.

### 10.2.9.1 PSILSS Overview

The PSILSS is a PSI-L compliant hardware switch that makes the connection between PSI-L master and slave endpoints. The SoC implements the following PSILSS instances:

- PDMA\_USART\_PSILSS0
  - [3:1] switch
  - Master endpoint: NAVSS0
  - Slave endpoints: PDMA\_USART\_G0, PDMA\_USART\_G1 and PDMA\_USART\_G2
- PDMA\_MISC\_PSILSS0
  - [4:1] switch
  - Master endpoint: NAVSS0
  - Slave endpoints: PDMA\_MISC\_G0, PDMA\_MISC\_G1, PDMA\_MISC\_G2 and PDMA\_MISC\_G3
- PDMA\_DEBUG\_PSILSS0
  - [2:1] switch
  - Master endpoint: NAVSS0
  - Slave endpoints: PDMA\_DEBUG\_CCMCU and PDMA\_DEBUG\_C66
- PDMA\_AASRC\_PSILSS0
  - [2:1] switch
  - Master endpoint: NAVSS0
  - Slave endpoints: PDMA\_AASRC and PDMA\_MCASP\_G0
- CSI\_PSILSS0
  - [3:1] switch
  - Master endpoint: NAVSS0
  - Slave endpoints: CSI\_TX\_IF0, CSI\_RX\_IF0 and CSI\_RX\_IF1

**Table 10-160. PSILSS Modules Allocation within Device Domains**

PSILSS Instance	Domain		
	WKUP	MCU	MAIN
PDMA_USART_PSILSS0	–	–	✓
PDMA_MISC_PSILSS0	–	–	✓
PDMA_DEBUG_PSILSS0	–	–	✓
PDMA_AASRC_PSILSS0	–	–	✓
CSI_PSILSS0	–	–	✓

#### 10.2.9.1.1 PSILSS Features

Each PSILSS module supports the following features:

- Implements a PSI-L 1.2 compliant interconnect
- Integrates the configured endpoints, either master or slave, and applies clock, data width or protocol conversion when necessary
- Integrates a single level CBASS for switching traffic between multiple masters and slaves using VBUSP
  - It uses the configured data width, clock and reset for the main SCR
- Provides full connectivity between all master and all slave endpoints
- Transactions are routed based on the configured thread map to the defined slave endpoint
- Events are routed based on the configured event map to the defined slave endpoints, or the default event output for an event aggregator
- Support event endpoints that are not PSI-L but still connect to the event interconnect

## 10.2.9.2 PSILSS Functional Description

### 10.2.9.2.1 PSILSS Basic Operation

The PSILSS implements a PSI-L 1.2 interconnect between endpoints. Each endpoint connects to a CBASS that provides a CBA 4.0 VBUSP transport which is used to transport the PSI-L streams. The CBASS can implement any bridging needed for clock conversion, but any data width conversion is handled in the PSILSS as data width conversion in PSI-L requires special bridges for packing and unpacking data. In this context, a master endpoint is one that creates streams that go into the PSILSS, and a slave endpoint is one that consumes streams from the PSILSS and returns status.

The PSILSS converts the PSI-L signals to VBUSP signals in order to use the CBASS to perform the routing based on the thread maps and the requested destination thread. The CBASS performs the routing and switching of the traffic from a number of masters to a number of slaves. The masters send transactions to the CBASS. The CBASS decodes which slave is being accessed based on the signals and the map. The CBASS routes that transaction to the slave. The CBASS also arbitrates for a slave when there are multiple transactions being requested simultaneously, which can be based on transaction priority. Once a transaction is selected, the CBASS sends the transaction to the slave.

### 10.2.9.2.2 PSILSS Event Routing

The PSIL events are also routed based on the event number. An event map defines which event number range maps to the destination ETL events of that endpoint. All source events from endpoints will be routed to a destination endpoint based on this event map. When the event number does not map to any endpoint, then it is routed to a default output event interface that connects to an event aggregator to map them to interrupts.

In addition to PSI-L endpoints, event endpoints are supported. These events connect to the same event interconnect as the PSI-L endpoints and pass events in the same way. An event endpoint can be a producer or consumer of events, and the same event mapping can apply to an event or PSI-L endpoint.

### 10.2.9.2.3 PSILSS Link Down Detection

The PSILSS provides MMRs that show the current link status of all the PSIL endpoints. It also tracks and provides status MMRs of any links that went down, from being active and running to a down state when the PSILSS is not in reset. This detection can generate an event so that software can remedy the situation. The hardware does not cleanup the traffic, so software must take care to idle any traffic before an endpoint is reset, which causes the link to go down, as well as to cleanup any connections after an endpoint has gone down. Software must guarantee the endpoint link is up again before setting up new connections.

### 10.2.9.2.4 PSILSS Configuration

Table 10-161 shows the thread numbers for the PSILSS slave endpoints.

**Table 10-161. PSILSS Endpoint Thread Map**

PSILSS Instance	Endpoint	Thread Number
PDMA_DEBUG_PSILSS0	PDMA_DEBUG_CCMCU	0x4300
	PDMA_DEBUG_C66	0x4304
PDMA_AASRC_PSILSS0	PDMA_MCASP_G0	0x4400
	PDMA_AASRC	0x4404
PDMA_MISC_PSILSS0	PDMA_MISC_G0	0x4600
	PDMA_MISC_G1	0x460C
	PDMA_MISC_G2	0x4618
	PDMA_MISC_G3	0x4624
PDMA_USART_PSILSS0	PDMA_USART_G0	0x4700
	PDMA_USART_G1	0x4702
	PDMA_USART_G2	0x4704
	PDMA_MCAN	0x470C

**Table 10-161. PSILSS Endpoint Thread Map (continued)**

PSILSS Instance	Endpoint	Thread Number
CSI_PSILSS0	CSI_TX_IF0	0x4900
	CSI_RX_IF0	0x4940
	CSI_RX_IF1	0x4960

Table 10-162 through Table 10-166 provide further details about the source (src) and destination (dst) configuration parameters for the various PSILSS instances, including number of threads per endpoint. Note that the last two columns in each table present the associated source and destination thread numbers in decimal value.

**Table 10-162. PDMA\_USART\_PSILSS0 Configuration Parameters**

Endpoint	Data Width (Src / Dst)	ETL Count (Src / Dst)	Thread Count (Src / Dst)	Source Thread Map	Destination Thread Map
PDMA_STRM	128-bit / 128-bit	1 / 1	32704 / 32704	0 through 18175; 18240 through 32767	32K + 0 through 18175; 32K + 18240 through 32767
USART_G0_STRM	128-bit / 128-bit	1 / 1	2 / 2	18176 through 18177	32K + 18176 through 18177
USART_G1_STRM	128-bit / 128-bit	1 / 1	2 / 2	18178 through 18179	32K + 18178 through 18179
USART_G2_STRM	128-bit / 128-bit	1 / 1	6 / 6	18180 through 18185	32K + 18180 through 18185
MCAN_STRM	128-bit / 128-bit	1 / 1	30 / 30	18188 through 18217	32K + 18188 through 18217

**Table 10-163. PDMA\_MISC\_PSILSS0 Configuration Parameters**

Endpoint	Data Width (Src / Dst)	ETL Count (Src / Dst)	Thread Count (Src / Dst)	Source Thread Map	Destination Thread Map
PDMA_STRM	128-bit / 128-bit	1 / 1	32704 / 32704	0 through 17919; 17984 through 32767	32K + 0 through 17919; 32K + 17984 through 32767
MISC_G0_STRM	128-bit / 128-bit	1 / 1	11 / 11	17920 through 17930	32K + 17920 through 17930
MISC_G1_STRM	128-bit / 128-bit	1 / 1	11 / 11	17932 through 17942	32K + 17932 through 17942
MISC_G2_STRM	128-bit / 128-bit	1 / 1	11 / 11	17944 through 17954	32K + 17944 through 17954
MISC_G3_STRM	128-bit / 128-bit	1 / 1	11 / 11	17956 through 17966	32K + 17956 through 17966

**Table 10-164. PDMA\_DEBUG\_PSILSS0 Configuration Parameters**

Endpoint	Data Width (Src / Dst)	ETL Count (Src / Dst)	Thread Count (Src / Dst)	Source Thread Map	Destination Thread Map
PDMA_STRM	128-bit / 128-bit	1 / 1	32704 / 32704	0 through 17151; 17216 through 32767	32K + 0 through 17151; 32K + 17216 through 32767
CCMCU_STRM	128-bit / 128-bit	1 / 1	3 / 0	17152 through 17154	32K + 17152 through 17154
MAINC66_STRM	128-bit / 128-bit	1 / 1	2 / 0	17156 through 17157	32K + 17156 through 17157

**Table 10-165. PDMA\_AASRC\_PSILSS0 Configuration Parameters**

Endpoint	Data Width (Src / Dst)	ETL Count (Src / Dst)	Thread Count (Src / Dst)	Source Thread Map	Destination Thread Map
PDMA_STRM	128-bit / 128-bit	1 / 1	32704 / 32704	0 through 17407; 17472 through 32767	32K + 0 through 17407; 32K + 17472 through 32767
MCASP_G0_STRM	128-bit / 128-bit	1 / 1	3 / 3	17408 through 17410	32K + 17408 through 17410
AASRC_STRM	128-bit / 128-bit	1 / 1	8 / 8	17412 through 17419	32K + 17412 through 17419

**Table 10-166. CSI\_PSILSS0 Configuration Parameters**

Endpoint	Data Width (Src / Dst)	ETL Count (Src / Dst)	Thread Count (Src / Dst)	Source Thread Map	Destination Thread Map
CSI_STRM	128-bit / 128-bit	0 / 0	32512 / 32512	0 through 18687; 18944 through 32767	32K + 0 through 18687; 32K + 18944 through 32767
TX0_STRM	128-bit / 128-bit	0 / 0	0 / 32	18688 through 18719	32K + 18688 through 18719
RX0_STRM	128-bit / 128-bit	0 / 0	32 / 0	18752 through 18783	32K + 18752 through 18783

**Table 10-166. CSI\_PSILSS0 Configuration Parameters (continued)**

Endpoint	Data Width (Src / Dst)	ETL Count (Src / Dst)	Thread Count (Src / Dst)	Source Thread Map	Destination Thread Map
RX1_STRM	128-bit / 128-bit	0 / 0	32 / 0	18784 through 18815	32K + 18784 through 18815



## 10.2.10 NAVSS North Bridge (NB)

This section describes the North Bridge (NB) in the main Navigator Subsystem.

### 10.2.10.1 NB Overview

The North bridge bridges between CBA 4.0 compliant VBUSM interfaces and a three-threaded CBA 4.0 compliant VBUSM.C interface.

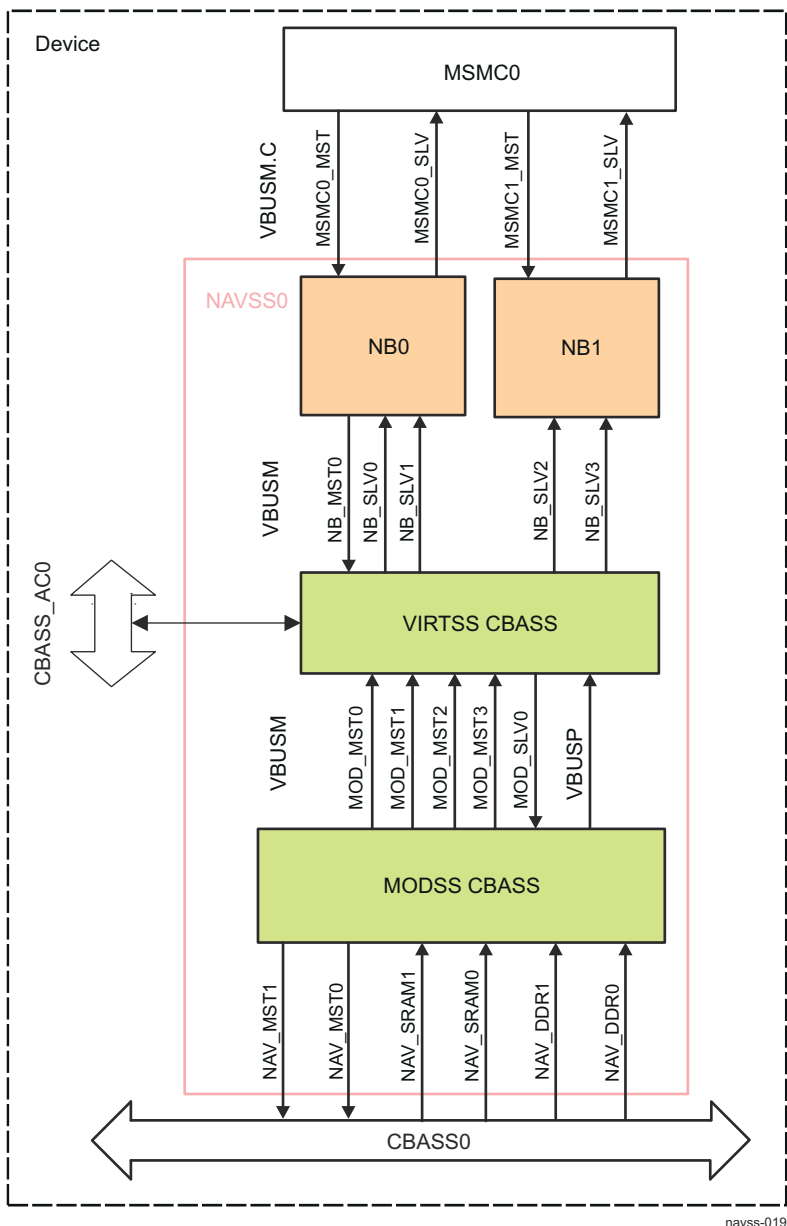


Figure 10-34. NB Overview

#### 10.2.10.1.1 Features Supported

- Implements a CBA 4.0 VBUSM to/from VBUSM.C compliant bridge
- Implements CBA 4.0 VBUSM compliant slave interface(s) to receive traffic from a VBUSM source
- Implements a CBA 4.0 VBUSM compliant master interface to create traffic to a VBUSM destination

- Implements a CBA 4.0 VBUSM.C compliant interface to receive and create traffic to and from a VBUSM.C endpoint using 3 threads
- Can optionally support fragmentation of VBUSM write commands into defined burst aligned sizes
- Can optionally support fragmentation and reassembly of VBUSM read commands into defined burst aligned sizes
- Can optionally support synchronous or asynchronous clock conversion
- Can optionally support data width conversion
- Can optionally support using the CBA 4.0 VBUSM credit interface for the VBUSM interfaces
- Allows for priority escalation through the bridge, either for higher priority pending commands inside the bridge or requesting into the bridge
- Supports the K3 clkstop idle signal
- Can optionally support a maximum outstanding read byte count to limit the total read data
- Can optionally insert memory attributes for commands
- Can optionally maintain VBUSM order based on orderid
- Allows programming the mapping of the VBUSM source interfaces to VBUSM.C threads

#### 10.2.10.1.2 NB Parameters

North Bridges are implemented in the SoC with the parameters shown in [Table 10-167](#) and [Table 10-168](#).

**Table 10-167. NB0 Configuration**

Parameter	Value	Description
Number of sources	2	The number of VBUSM source interfaces
Source 0	CBASS MSMC_SLV0 to MSMC0	Source 0
Source 1	CBASS MSMC_SLV1 to MSMC0	Source 1
source_oidmap[0]	0-7	This defines the orderid mapping to the VBUSM sources. The index represents the source number. The values are integer.
source_oidmap[1]	8-15	
start64k	0x0060000000	This defines the starting address of the memory in VBUSM.C that uses the 64-KB table.
num64k	8192	This defines the number of entries in the 64-KB table.

**Table 10-168. NB1 Configuration**

Parameter	Value	Description
Number of sources	2	The number of VBUSM source interfaces
Source 0	CBASS MSMC_DDR_0 to MSMC1	Source 0
Source 1	CBASS MSMC_DDR_1 to MSMC1	Source 1
source_oidmap[0]	0-7	This defines the orderid mapping to the VBUSM sources. The index represents the source number. The values are integer.
source_oidmap[1]	8-15	
start16m[0]	0x0080000000	This defines the starting address of the memory in VBUSM.C that uses the 16-MB table. The index is for each of the 16-MB regions.
start16m[1]	0x0800000000	
num16m[0]	128	This defines the number of entries in the 16-MB table. The index is for each 16-MB region.
num16m[1]	8192	

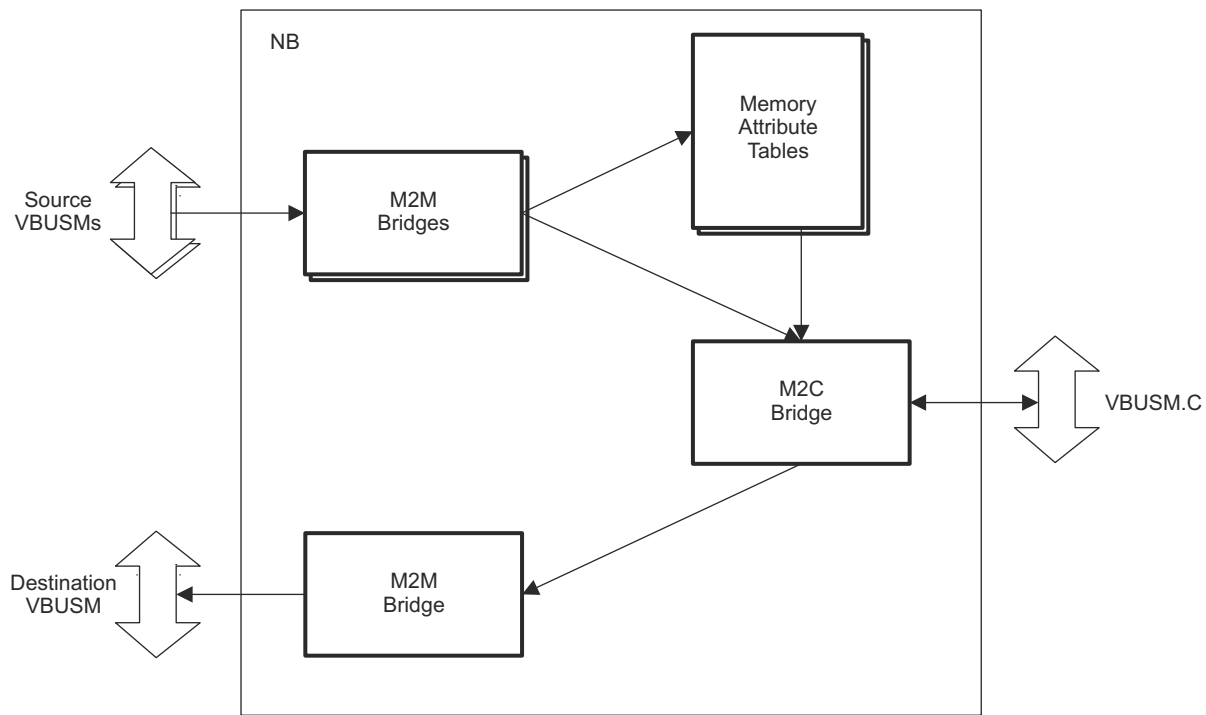
#### 10.2.10.1.2.1 Compliance to Standards

The module is compliant to the CBA 4.0 VBUSM specification and VBUSM.C annex.

#### 10.2.10.1.2.2 Features Not Supported

- Does not support fragmentation of VBUSM.C commands
- Does not support VBUSM.C coherence traffic

### 10.2.10.2 NB Functional Description



**Figure 10-35. NB Block Diagram**

The CBA 4.0 North bridge, NB, bridges between CBA 4.0 compliant VBUSM interfaces and a three threaded CBA 4.0 compliant VBUSM.C interface. This is achieved by using M2M bridges to support all the M2M features and then an M2C bridge to convert the VBUSM interfaces to a VBUSM.C interface, which support duplex traffic. The M2M bridges allow for clock frequency conversion, data width conversion and fragmentation for the Source M2M bridge(s). This allows the M2C bridge to remain simple and operate at the VBUSM.C clock frequency and data width, as well as assume all required burst limitations have already been handled. Some additional features needed for the north bridge are implemented, including: adding memory attributes to source VBUSM transactions, and forcing a maximum outstanding read data limit.

#### 10.2.10.2.1 VBUSM Slave Interfaces

The slave interfaces on the bridge is the source VBUSM port to receive commands and return data and status. These ports have the defined prefix, smX (SoC Master, input to bridge), to identify them, where the X is the number to identify each port starting from zero. The number of these interfaces and bridges is shown in [Table 10-167](#) and [Table 10-168](#).

#### 10.2.10.2.2 VBUSM Master Interface

The master interface on the bridge is the destination VBUSM port to send commands and receive data and status. These ports have the defined prefix, ss (SoC Slave, output of bridge), to identify them.

It is important to note that any feature not supported in the optional signals for this bus that is supported on the VBUSM.C interface will not be corrected inside the bridge. Examples are excluding the emudbg pin on this bus but expecting VBUSM.C transactions with emudbg set to not create bus status errors on this VBUSM bus outside the bridge (as the bridge will just pass the returned status which could be an error if it does not support the emudbg pin).

#### 10.2.10.2.3 VBUSM.C Interfaces

The pair of interfaces on the bridge is the VBUSM.C port to send and receive commands and send and receive data and status. These ports will have the defined prefix, mm (Memory Master, output of bridge) and

ms (Memory Slave, input to bridge), to identify them. These interfaces support three threads for sending and receiving traffic.

#### **10.2.10.2.3.1 Multi-Threading**

The VBUSM.C interfaces use multi-threading to support both input and output traffic on the same data signals. The threads are used to separate that traffic. Thread 0 is used for commands to VBUSM.C, thread 1 is used for commands from VBUSM.C, and thread 2 is used for realtime commands to VBUSM.C. Any responses due to commands must use the same thread when returned.

#### **10.2.10.2.3.2 Write Command Crediting**

VBUSM.C interfaces use credits for sending data phases. For writes and read responses to the MSMC (VBUSM.C consumer), there is a requirement that all credits for the write, including the command and all data phases, should be available before sending the command. Since the VBUSM.C data width is always half the max burst size on VBUSM.C, the write command must wait for a TAC credit available and 1 or 2 TDC credits available before the write can be sent.

#### **10.2.10.2.3.3 Early Credit Response**

The VBUSM.C interfaces use the credit mechanism available. They do not check for full burst available credits before sending any piece of the burst (command or data). To prevent deadlocks, the connected VBUSM.C device must support sending early credit responses, before the entire burst is received.

#### **10.2.10.2.3.4 Priority Escalation**

The VBUSM.C interface does not natively define a mechanism for priority escalation, since the cepriority signal is only valid when a command is sent, and can otherwise be ignored. To support this feature this bridge supports updating the cepriority output even if there is no valid command, so identify the highest priority of any pending commands when the bridge cannot sent a command, such as due to a lack of credits. The VBUSM.C consumer can assume the cepriority output from this bridge is always valid every cycle, and use the cepriority along with the priorities of all pending commands. When there are no pending commands, the cepriority output will be the lowest priority, 7.

#### **10.2.10.2.4 Source M2M Bridges**

The source M2M bridges perform all the usual bridging functions from the source VBUSM interface to an internal VBUSM interface running at the same clock frequency and data width of the VBUSM.C interface. It also performs the fragmentation to sizes the VBUSM.C interface requires. It additionally performs full read data reassembly, orderid based ordering, and read data cacheline conversion.

#### **10.2.10.2.5 Destination M2M Bridge**

The destination M2M bridge performs all the usual bridging functions to the destination VBUSM interface from an internal VBUSM interface running at the same clock frequency and data width of the VBUSM.C interface.

#### **10.2.10.2.6 M2C Bridge**

The M2C bridge performs the protocol conversion and multi-threading of the VBUSM.C interface from the internal VBUSM interfaces. It also handles the cacheline conversion of the write data.

A VBUSM.C write with a dtype of 1 (CPU instruction) will be converted to a VBUSM write with a dtype of 0 (CPU data), as this condition is allowed in VBUSM.C but not in VBUSM.

#### **10.2.10.2.7 Memory Attribute Tables**

The bridge can optionally include a memory attribute lookup table to add the coherence memory attributes required for VBUSM.C for VBUSM commands missing them. It is implemented using up to 4 tables, a table of 64-KB regions and 1 to 3 tables of 16-MB regions. The first table is meant to cover an internal memory with smaller granularity, while the second to fourth tables are meant to cover the external memory with a much larger size and larger granularity. The external memory has one to three regions as its address space might be fragmented in the SoC memory map. Starting address and number of table entries for each region are shown in [Table 10-167](#) and [Table 10-168](#).

When a command without memory attributes, an `atype = 0`, is input to the M2C bridge, this table performs the lookup in parallel and returns the memory attributes to the M2C bridge in the next cycle. If the `atype` is another value, it is assumed the memory attributes on the VBUSM interface are valid. The lookup is performed by matching the address against the starting addresses, assuming the successive regions always starts at later address, including that the up to three 16-MB regions are given in increasing order. Then the offset from the starting address of the region and the region size, 64-KB or 16-MB, is used to calculate the entry of attributes for that address. Then a memory is read to retrieve the attributes and return them to the M2C bridge. An output FIFO is implemented to capture the attributes if they are not accepted immediately.

There is a memory attribute table module for each VBUSM slave interface. If the MAT is enabled, at least one of the 64-KB region and/or the first 16-MB region must be populated, or both. The second and third 16-MB regions are optional.

#### **10.2.10.2.8 Outstanding Read Data Limiter**

If a param defines a maximum outstanding read data byte count, then this function will track the requested read bytes against the limit. If a read command does not violate the total read bytes outstanding limit then the read is allowed to the VBUSM.C, otherwise it is stalled. When read data arrives from the VBUSM.C the outstanding count is reduced, allowing more read commands to be sent.

#### **10.2.10.2.9 Ordering**

Ordering becomes a concern due to the fact that the VBUSM commands are fragmented and not guaranteed to be returned in order by the VBUSM.C interface. This violates the CBA ordering rules for a single slave endpoint. So the bridge will force read data ordering based on the `orderid` signal across all masters, as a balance between cost and performance. This means that each read command received from the VBUSM interface with a particular `orderid` value will have their read data returned in exactly the same order. This means that even if the reads are from different masters, the data will still be returned in order, to reduce the amount of tracking the bridge has to perform. But, if the reads use different `orderid` values then that read data can be returned in any order, whichever is received first on the VBUSM.C interface.

#### **10.2.10.2.10 Quality of Service**

To support quality of service, QoS, the north bridge allows for configuring multiple VBUSM source interfaces. This not only allows defining particular sources to be realtime to achieve better qos, but also to load balance and effectively utilize any extra bandwidth provided by the VBUSM.C interface. There are registers (`NB_THREADMAP`) allowing programming of which source interfaces map to which of the two VBUSM.C threads, normal or realtime traffic. All traffic on the source will always map to the defined thread.

Any sources mapped to the realtime thread will be arbitrated first, before any sources mapped to the normal thread, to provide the QoS. If there are multiple sources mapped to the same thread then they are arbitrated based on priority and if they are the same priority then by round robin.

To route return traffic from the VBUSM.C back to the correct VBUSM source, the `orderid` is used. As such, each source mapped to each thread must use a different `orderid` value in the SoC. This would mean that if there are 8 sources total, with 4 mapped to each thread, then each group of 4 should not share the same `orderid` for any traffic routed to it in the SoC to keep their `pathid` values unique. The bridge is configured to identify which VBUSM source uses which `orderid` value(s).

#### **10.2.10.2.11 IDLE Behavior**

The north bridge IDLE output is a combination of all the IDLE outputs from the components. These are combined and registered as the NB IDLE on the Destination VBUSM bus clock.

#### **10.2.10.2.12 Clock Power Management**

The north bridge provides internal clock power management for all major components: bridges and register modules. Each component will track its own activity and shut off the clock tree when there has been no activity for 12 cycles. There is a 1 cycle latency for starting up the clock once off, so bus transactions may be initially stalled. The bus early wakeup signals can be used to wakeup the clocks before the transactions arrives to remove some of the latency impact.

## 10.3 Peripheral DMA (PDMA)

This chapter describes the PDMA architecture in the device.

### 10.3.1 PDMA Controller

#### 10.3.1.1 PDMA Overview

The Peripheral DMA is a simple DMA which has been architected to specifically meet the data transfer needs of peripherals, which perform data transfers using memory mapped registers (MMRs) accessed via a standard non-coherent bus fabric. The PDMA module is located close to one or more peripherals which require an external DMA for data movement and is architected to reduce cost by using VBUSP interfaces and supporting only statically configured Transfer Request (TR) operations.

The PDMA is only responsible for performing the data movement transactions which interact with the peripherals themselves. Data which is read from a given peripheral is packed by a PDMA source channel into a PSI-L data stream which is then sent to a remote peer UDMA-P destination channel which then performs the movement of the data into memory. Likewise, a remote UDMA-P source channel fetches data from memory and transfers it to a peer PDMA destination channel over PSI-L which then performs the writes to the peripheral.

The PDMA architecture is intentionally heterogeneous (UDMA-P + PDMA) to right size the data transfer complexity at each point in the system to match the requirements of whatever is being transferred to or from. Peripherals are typically FIFO based and do not require multi-dimensional transfers beyond their FIFO dimensioning requirements, so the PDMA transfer engines are kept simple with only a few dimensions (typically for sample size and FIFO depth), hardcoded address maps, and simple triggering capabilities.

Multiple source and destination channels are provided within the PDMA which allow multiple simultaneous transfer operations to be ongoing. The DMA controller maintains state information for each of the channels and employs round-robin scheduling between channels in order to share the underlying DMA hardware.

There are several PDMA modules in the device. [Table 10-169](#) shows PDMA allocation across device domains.

**Table 10-169. PDMA Allocation Across Device Domains**

PDMA Instance	Serviced Peripherals	Domain		
		WKUP	MCU	MAIN
MCU_PDMA0 (MCU_PDMA_MISC_G0)	MCU_SPI0, MCU_MCAN0	–	✓	–
MCU_PDMA1 (MCU_PDMA_MISC_G1)	MCU_SPI1, MCU_SPI2	–	✓	–
MCU_PDMA2 (MCU_PDMA_MISC_G2)	MCU_USART0, MCU_MCAN1	–	✓	–
MCU_PDMA3 (MCU_PDMA_ADC)	MCU_ADC0, MCU_ADC1	–	✓	–
PDMA0 (PDMA_AASRC)	AASRC0	–	–	✓
PDMA2 (PDMA_DEBUG_CCMCU)	Debug Cells (CC, MCU, C66_1)	–	–	✓
PDMA3 (PDMA_DEBUG_C66)	Debug Cells (SoC, C66_0)	–	–	✓
PDMA5 (PDMA_MCAN)	MCAN4 to MCAN13	–	–	✓
PDMA6 (PDMA_MCASP_G0)	MCASP0 to MCASP2	–	–	✓
PDMA7 (PDMA_MCASP_G1)	MCASP3 to MCASP11	–	–	✓
PDMA8 (PDMA_MISC_G0)	MCAN0, SPI0, SPI1	–	–	✓
PDMA9 (PDMA_MISC_G1)	MCAN1, SPI2, SPI3	–	–	✓
PDMA10 (PDMA_MISC_G2)	MCAN2, SPI4, SPI5	–	–	✓
PDMA11 (PDMA_MISC_G3)	MCAN3, SPI6, SPI7	–	–	✓
PDMA13 (PDMA_USART_G0)	UART0, UART1	–	–	✓
PDMA14 (PDMA_USART_G1)	UART2, UART3	–	–	✓
PDMA15 (PDMA_USART_G2)	UART4 to UART9	–	–	✓

The PDMA is linked to the UDMA-P (NAVSS) either through a direct PSI-L connection, or through a PSI-L switch (PSILSS). For more details, refer to:



- [Section 10.2, Navigator Subsystem \(NAVSS\)](#)
- [Section 10.2.8, Packet Streaming Interface Link \(PSI-L\)](#)
- [Section 10.2.9, PSIL Subsystem \(PSILSS\)](#)

#### **10.3.1.1.1 PDMA Features**

##### **10.3.1.1.1.1 MCU\_PDMA0 (MCU\_PDMA\_MISC\_G0) Features**

The MCU domain PDMA0 supports the following features:

- Implements CPPI 5.0 compliant third-party Unified Transfer Controller (UTC)
- Provides 1 memory write access units (write unit 0)
  - Provides a 32-bit wide VBUSP write only master interface for peripheral accesses
  - Supports write bursts up to 64 bytes (on selected channel types)
- Provides 1 memory read access unit (read unit 0):
  - Provides a 32-bit wide VBUSP read-only master interface for peripheral accesses
  - Supports read burst up to 64 bytes (on selected channel types)
  - Supports 1 outstanding read
- Supports up to 7 simultaneous destination (Tx) channels
- Supports up to 7 simultaneous source (Rx) channels
- Supports static transfer requests (TR) only
- Supports X-Y and MCAN transfer modes (does not support AASRC)
- Provides per-channel buffering:
  - Provides 8×128-bit word deep data FIFO for each destination channel
  - Provides 8×128-bit word deep data FIFO for each source channel
- Provides 128-bit wide PSI-L compliant data interface to remote UDMA-P and remote peripherals
- Provides 128-bit wide PSI-L compliant data interface from remote UDMA-P and remote peripherals

##### **10.3.1.1.1.2 MCU\_PDMA1 (MCU\_PDMA\_MISC\_G1) Features**

The MCU domain PDMA1 supports the following features:

- Implements CPPI 5.0 compliant third-party Unified Transfer Controller (UTC)
- Provides 1 memory write access unit (write unit 0):
  - Provides a 32-bit wide VBUSP write-only master interface for peripheral accesses.
  - Supports write bursts up to 64 bytes (on selected channel types)
- Provides 1 memory read access unit (read unit 0)
  - Provides a 32-bit wide VBUSP read-only master interface for peripheral accesses
  - Supports read burst up to 64 bytes (on selected channel types)
  - Supports 1 outstanding read
- Supports up to 8 simultaneous destination (Tx) channels
- Supports up to 8 simultaneous source (Rx) channels
- Supports static transfer requests (TR) only
- Supports X-Y transfer mode only (does not support MCAN and AASRC)
- Provides per-channel buffering:
  - Provides 8×128-bit word deep data FIFO for each destination channel
  - Provides 8×128-bit word deep data FIFO for each source channel
- Provides 128-bit wide PSI-L compliant data interface to remote UDMA-P and remote peripherals
- Provides 128-bit wide PSI-L compliant data interface from remote UDMA-P and remote peripherals

##### **10.3.1.1.1.3 MCU\_PDMA2 (MCU\_PDMA\_MISC\_G2) Features**

The MCU domain PDMA2 supports the following features:

- Implements CPPI 5.0 compliant third-party Unified Transfer Controller (UTC)
- Provides 1 memory write access unit (write unit 0):
  - Provides a 32-bit wide VBUSP write-only master interface for peripheral accesses.
  - Supports write bursts up to 64 bytes (on selected channel types)
- Provides 1 memory read access unit (read unit 0)

- Provides a 32-bit wide VBUSP read-only master interface for peripheral accesses
- Supports read burst up to 64 bytes (on selected channel types)
- Supports 1 outstanding read
- Supports up to 4 simultaneous destination (Tx) channels
- Supports up to 4 simultaneous source (Rx) channels
- Supports static transfer requests (TR) only
- Supports X-Y and MCAN transfer modes (does not support AASRC)
- Provides per-channel buffering:
  - Provides 8×128-bit word deep data FIFO for each destination channel
  - Provides 8×128-bit word deep data FIFO for each source channel
- Provides 128-bit wide PSI-L compliant data interface to remote UDMA-P and remote peripherals
- Provides 128-bit wide PSI-L compliant data interface from remote UDMA-P and remote peripherals

#### 10.3.1.1.1.4 MCU\_PDMA3 (MCU\_PDMA\_ADC) Features

The MCU domain PDMA3 supports the following features:

- Implements CPPI 5.0 compliant third-party Unified Transfer Controller (UTC)
- Provides 0 memory write access units
- Provides 1 memory read access unit (read unit 0)
  - Provides a 32-bit wide VBUSP read-only master interface for peripheral accesses
  - Supports read burst up to 64 bytes (on selected channel types)
  - Supports 1 outstanding read
- Supports up to 0 simultaneous destination (Tx) channels
- Supports up to 4 simultaneous source (Rx) channels
- Supports static transfer requests (TR) only
- Supports X-Y transfer mode only (does not support MCAN and AASRC)
- Provides per-channel buffering:
  - Provides 8×128-bit word deep data FIFO for each destination channel
  - Provides 8×128-bit word deep data FIFO for each source channel
- Provides 128-bit wide PSI-L compliant data interface to remote UDMA-P and remote peripherals
- Provides 128-bit wide PSI-L compliant data interface from remote UDMA-P and remote peripherals

#### 10.3.1.1.1.5 PDMA0 (PDMA\_AASRC) Features

The MAIN domain PDMA0 supports the following features:

- Implements CPPI 5.0 compliant third-party Unified Transfer Controller (UTC)
- Provides 1 memory write access unit (write unit 0):
  - Provides a 32-bit wide VBUSP write-only master interface for peripheral accesses.
  - Supports write bursts up to 64 bytes (on selected channel types)
- Provides 1 memory read access unit (read unit 0)
  - Provides a 32-bit wide VBUSP read-only master interface for peripheral accesses
  - Supports read burst up to 64 bytes (on selected channel types)
  - Supports 1 outstanding read
- Supports up to 8 simultaneous destination (Tx) channels
- Supports up to 8 simultaneous source (Rx) channels
- Supports static transfer requests (TR) only
- Supports AASRC transfer mode only (does not support X-Y and MCAN)
- Provides per-channel buffering:
  - Provides 8×128-bit word deep data FIFO for each destination channel
  - Provides 8×128-bit word deep data FIFO for each source channel
- Provides 128-bit wide PSI-L compliant data interface to remote UDMA-P and remote peripherals
- Provides 128-bit wide PSI-L compliant data interface from remote UDMA-P and remote peripherals

#### 10.3.1.1.1.6 PDMA2 (PDMA\_DEBUG\_CCMCU) Features

The MAIN domain PDMA2 supports the following features:



- Implements CPPI 5.0 compliant third-party Unified Transfer Controller (UTC)
- Provides 0 memory write access units
- Provides 1 memory read access unit (read unit 0)
  - Provides a 32-bit wide VBUSP read-only master interface for peripheral accesses
  - Supports read burst up to 64 bytes (on selected channel types)
  - Supports 1 outstanding read
- Supports up to 0 simultaneous destination (Tx) channels
- Supports up to 3 simultaneous source (Rx) channels
- Supports static transfer requests (TR) only
- Supports X-Y transfer mode only (does not support MCAN and AASRC)
- Provides per-channel buffering:
  - Provides 8×128-bit word deep data FIFO for each destination channel
  - Provides 8×128-bit word deep data FIFO for each source channel
- Provides 128-bit wide PSI-L compliant data interface to remote UDMA-P and remote peripherals
- Provides 128-bit wide PSI-L compliant data interface from remote UDMA-P and remote peripherals

#### 10.3.1.1.7 PDMA3 (PDMA\_DEBUG\_C66) Features

The MAIN domain PDMA3 supports the following features:

- Implements CPPI 5.0 compliant third-party Unified Transfer Controller (UTC)
- Provides 0 memory write access units
- Provides 1 memory read access unit (read unit 0)
  - Provides a 32-bit wide VBUSP read-only master interface for peripheral accesses
  - Supports read burst up to 64 bytes (on selected channel types)
  - Supports 1 outstanding read
- Supports up to 0 simultaneous destination (Tx) channels
- Supports up to 2 simultaneous source (Rx) channels
- Supports static transfer requests (TR) only
- Supports X-Y transfer mode only (does not support MCAN and AASRC)
- Provides per-channel buffering:
  - Provides 8×128-bit word deep data FIFO for each destination channel
  - Provides 8×128-bit word deep data FIFO for each source channel
- Provides 128-bit wide PSI-L compliant data interface to remote UDMA-P and remote peripherals
- Provides 128-bit wide PSI-L compliant data interface from remote UDMA-P and remote peripherals

#### 10.3.1.1.8 PDMA5 (PDMA\_MCAN) Features

The MAIN domain PDMA5 supports the following features:

- Implements CPPI 5.0 compliant third-party Unified Transfer Controller (UTC)
- Provides 1 memory write access unit (write unit 0):
  - Provides a 32-bit wide VBUSP write-only master interface for peripheral accesses.
  - Supports write bursts up to 64 bytes (on selected channel types)
- Provides 1 memory read access unit (read unit 0)
  - Provides a 32-bit wide VBUSP read-only master interface for peripheral accesses
  - Supports read burst up to 64 bytes (on selected channel types)
  - Supports 1 outstanding read
- Supports up to 30 simultaneous destination (Tx) channels
- Supports up to 30 simultaneous source (Rx) channels
- Supports static transfer requests (TR) only
- Supports MCAN transfer mode only (does not support X-Y and AASRC)
- Provides per-channel buffering:
  - Provides 8×128-bit word deep data FIFO for each destination channel
  - Provides 8×128-bit word deep data FIFO for each source channel
- Provides 128-bit wide PSI-L compliant data interface to remote UDMA-P and remote peripherals
- Provides 128-bit wide PSI-L compliant data interface from remote UDMA-P and remote peripherals

- Provides ECC support

#### **10.3.1.1.1.9 PDMA6 (PDMA\_MCASP\_G0) Features**

The MAIN domain PDMA6 supports the following features:

- Implements CPPI 5.0 compliant third-party Unified Transfer Controller (UTC)
- Provides 1 memory write access unit (write unit 0):
  - Provides a 32-bit wide VBUSP write-only master interface for peripheral accesses.
  - Supports write bursts up to 64 bytes (on selected channel types)
- Provides 1 memory read access unit (read unit 0)
  - Provides a 32-bit wide VBUSP read-only master interface for peripheral accesses
  - Supports read burst up to 64 bytes (on selected channel types)
  - Supports 1 outstanding read
- Supports up to 3 simultaneous destination (Tx) channels
- Supports up to 3 simultaneous source (Rx) channels
- Supports static transfer requests (TR) only
- Supports X-Y transfer mode only (does not support MCAN and AASRC)
- Provides per-channel buffering:
  - Provides 8×128-bit word deep data FIFO for each destination channel
  - Provides 8×128-bit word deep data FIFO for each source channel
- Provides 128-bit wide PSI-L compliant data interface to remote UDMA-P and remote peripherals
- Provides 128-bit wide PSI-L compliant data interface from remote UDMA-P and remote peripherals

#### **10.3.1.1.1.10 PDMA7 (PDMA\_MCASP\_G1) Features**

The MAIN domain PDMA7 supports the following features:

- Implements CPPI 5.0 compliant third-party Unified Transfer Controller (UTC)
- Provides 1 memory write access unit (write unit 0):
  - Provides a 32-bit wide VBUSP write-only master interface for peripheral accesses.
  - Supports write bursts up to 64 bytes (on selected channel types)
- Provides 1 memory read access unit (read unit 0)
  - Provides a 32-bit wide VBUSP read-only master interface for peripheral accesses
  - Supports read burst up to 64 bytes (on selected channel types)
  - Supports 1 outstanding read
- Supports up to 9 simultaneous destination (Tx) channels
- Supports up to 9 simultaneous source (Rx) channels
- Supports static transfer requests (TR) only
- Supports X-Y transfer mode only (does not support MCAN and AASRC)
- Provides per-channel buffering:
  - Provides 8×128-bit word deep data FIFO for each destination channel
  - Provides 8×128-bit word deep data FIFO for each source channel
- Provides 128-bit wide PSI-L compliant data interface to remote UDMA-P and remote peripherals
- Provides 128-bit wide PSI-L compliant data interface from remote UDMA-P and remote peripherals

#### **10.3.1.1.1.11 PDMA8 (PDMA\_MISC\_G0) Features**

The MAIN domain PDMA8 supports the following features:

- Implements CPPI 5.0 compliant third-party Unified Transfer Controller (UTC)
- Provides 1 memory write access unit (write unit 0):
  - Provides a 32-bit wide VBUSP write-only master interface for peripheral accesses.
  - Supports write bursts up to 64 bytes (on selected channel types)
- Provides 1 memory read access unit (read unit 0)
  - Provides a 32-bit wide VBUSP read-only master interface for peripheral accesses
  - Supports read burst up to 64 bytes (on selected channel types)
  - Supports 1 outstanding read

- Supports up to 11 simultaneous destination (Tx) channels
- Supports up to 11 simultaneous source (Rx) channels
- Supports static transfer requests (TR) only
- Supports X-Y and MCAN transfer modes (does not support AASRC)
- Provides per-channel buffering:
  - Provides 8×128-bit word deep data FIFO for each destination channel
  - Provides 8×128-bit word deep data FIFO for each source channel
- Provides 128-bit wide PSI-L compliant data interface to remote UDMA-P and remote peripherals
- Provides 128-bit wide PSI-L compliant data interface from remote UDMA-P and remote peripherals

#### **10.3.1.1.1.12 PDMA9 (PDMA\_MISC\_G1) Features**

The MAIN domain PDMA9 supports the following features:

- Implements CPPI 5.0 compliant third-party Unified Transfer Controller (UTC)
- Provides 1 memory write access unit (write unit 0):
  - Provides a 32-bit wide VBUSP write-only master interface for peripheral accesses.
  - Supports write bursts up to 64 bytes (on selected channel types)
- Provides 1 memory read access unit (read unit 0)
  - Provides a 32-bit wide VBUSP read-only master interface for peripheral accesses
  - Supports read burst up to 64 bytes (on selected channel types)
  - Supports 1 outstanding read
- Supports up to 11 simultaneous destination (Tx) channels
- Supports up to 11 simultaneous source (Rx) channels
- Supports static transfer requests (TR) only
- Supports X-Y and MCAN transfer modes (does not support AASRC)
- Provides per-channel buffering:
  - Provides 8×128-bit word deep data FIFO for each destination channel
  - Provides 8×128-bit word deep data FIFO for each source channel
- Provides 128-bit wide PSI-L compliant data interface to remote UDMA-P and remote peripherals
- Provides 128-bit wide PSI-L compliant data interface from remote UDMA-P and remote peripherals

#### **10.3.1.1.1.13 PDMA10 (PDMA\_MISC\_G2) Features**

The MAIN domain PDMA10 supports the following features:

- Implements CPPI 5.0 compliant third-party Unified Transfer Controller (UTC)
- Provides 1 memory write access unit (write unit 0):
  - Provides a 32-bit wide VBUSP write-only master interface for peripheral accesses.
  - Supports write bursts up to 64 bytes (on selected channel types)
- Provides 1 memory read access unit (read unit 0)
  - Provides a 32-bit wide VBUSP read-only master interface for peripheral accesses
  - Supports read burst up to 64 bytes (on selected channel types)
  - Supports 1 outstanding read
- Supports up to 11 simultaneous destination (Tx) channels
- Supports up to 11 simultaneous source (Rx) channels
- Supports static transfer requests (TR) only
- Supports X-Y and MCAN transfer modes (does not support AASRC)
- Provides per-channel buffering:
  - Provides 8×128-bit word deep data FIFO for each destination channel
  - Provides 8×128-bit word deep data FIFO for each source channel
- Provides 128-bit wide PSI-L compliant data interface to remote UDMA-P and remote peripherals
- Provides 128-bit wide PSI-L compliant data interface from remote UDMA-P and remote peripherals

#### **10.3.1.1.1.14 PDMA11 (PDMA\_MISC\_G3) Features**

The MAIN domain PDMA11 supports the following features:

- Implements CPPI 5.0 compliant third-party Unified Transfer Controller (UTC)
- Provides 1 memory write access unit (write unit 0):
  - Provides a 32-bit wide VBUSP write-only master interface for peripheral accesses.
  - Supports write bursts up to 64 bytes (on selected channel types)
- Provides 1 memory read access unit (read unit 0)
  - Provides a 32-bit wide VBUSP read-only master interface for peripheral accesses
  - Supports read burst up to 64 bytes (on selected channel types)
  - Supports 1 outstanding read
- Supports up to 11 simultaneous destination (Tx) channels
- Supports up to 11 simultaneous source (Rx) channels
- Supports static transfer requests (TR) only
- Supports X-Y and MCAN transfer modes (does not support AASRC)
- Provides per-channel buffering:
  - Provides 8×128-bit word deep data FIFO for each destination channel
  - Provides 8×128-bit word deep data FIFO for each source channel
- Provides 128-bit wide PSI-L compliant data interface to remote UDMA-P and remote peripherals
- Provides 128-bit wide PSI-L compliant data interface from remote UDMA-P and remote peripherals

#### 10.3.1.1.15 PDMA13 (PDMA\_USART\_G0) Features

The MAIN domain PDMA13 supports the following features:

- Implements CPPI 5.0 compliant third-party Unified Transfer Controller (UTC)
- Provides 1 memory write access unit (write unit 0):
  - Provides a 32-bit wide VBUSP write-only master interface for peripheral accesses.
  - Supports write bursts up to 64 bytes (on selected channel types)
- Provides 1 memory read access unit (read unit 0)
  - Provides a 32-bit wide VBUSP read-only master interface for peripheral accesses
  - Supports read burst up to 64 bytes (on selected channel types)
  - Supports 1 outstanding read
- Supports up to 2 simultaneous destination (Tx) channels
- Supports up to 2 simultaneous source (Rx) channels
- Supports static transfer requests (TR) only
- Supports X-Y transfer mode only (does not support MCAN and AASRC)
- Provides per-channel buffering:
  - Provides 8×128-bit word deep data FIFO for each destination channel
  - Provides 8×128-bit word deep data FIFO for each source channel
- Provides 128-bit wide PSI-L compliant data interface to remote UDMA-P and remote peripherals
- Provides 128-bit wide PSI-L compliant data interface from remote UDMA-P and remote peripherals

#### 10.3.1.1.16 PDMA14 (PDMA\_USART\_G1) Features

The MAIN domain PDMA14 supports the following features:

- Implements CPPI 5.0 compliant third-party Unified Transfer Controller (UTC)
- Provides 1 memory write access unit (write unit 0):
  - Provides a 32-bit wide VBUSP write-only master interface for peripheral accesses.
  - Supports write bursts up to 64 bytes (on selected channel types)
- Provides 1 memory read access unit (read unit 0)
  - Provides a 32-bit wide VBUSP read-only master interface for peripheral accesses
  - Supports read burst up to 64 bytes (on selected channel types)
  - Supports 1 outstanding read
- Supports up to 2 simultaneous destination (Tx) channels
- Supports up to 2 simultaneous source (Rx) channels
- Supports static transfer requests (TR) only
- Supports X-Y transfer mode only (does not support MCAN and AASRC)
- Provides per-channel buffering:

- Provides 8×128-bit word deep data FIFO for each destination channel
- Provides 8×128-bit word deep data FIFO for each source channel
- Provides 128-bit wide PSI-L compliant data interface to remote UDMA-P and remote peripherals
- Provides 128-bit wide PSI-L compliant data interface from remote UDMA-P and remote peripherals

#### **10.3.1.1.1.17 PDMA15 (PDMA\_USART\_G2) Features**

The MAIN domain PDMA15 supports the following features:

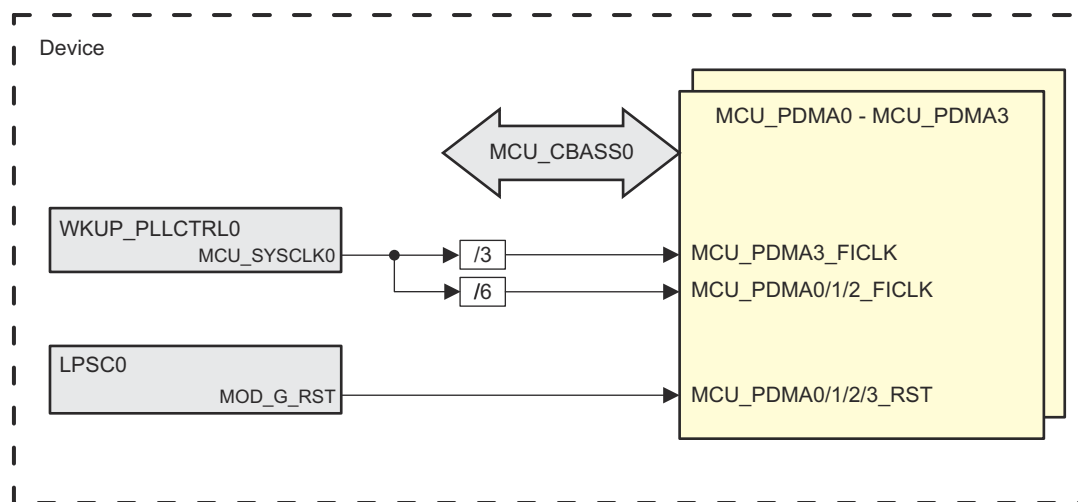
- Implements CPPI 5.0 compliant third-party Unified Transfer Controller (UTC)
- Provides 1 memory write access unit (write unit 0):
  - Provides a 32-bit wide VBUSP write-only master interface for peripheral accesses.
  - Supports write bursts up to 64 bytes (on selected channel types)
- Provides 1 memory read access unit (read unit 0)
  - Provides a 32-bit wide VBUSP read-only master interface for peripheral accesses
  - Supports read burst up to 64 bytes (on selected channel types)
  - Supports 1 outstanding read
- Supports up to 6 simultaneous destination (Tx) channels
- Supports up to 6 simultaneous source (Rx) channels
- Supports static transfer requests (TR) only
- Supports X-Y transfer mode only (does not support MCAN and AASRC)
- Provides per-channel buffering:
  - Provides 8×128-bit word deep data FIFO for each destination channel
  - Provides 8×128-bit word deep data FIFO for each source channel
- Provides 128-bit wide PSI-L compliant data interface to remote UDMA-P and remote peripherals
- Provides 128-bit wide PSI-L compliant data interface from remote UDMA-P and remote peripherals

### 10.3.1.2 PDMA Integration

This section describes the PDMA integration in the device, including information about clocks, resets, and hardware requests.

#### 10.3.1.2.1 PDMA Integration in MCU Domain

Figure 10-36 shows the integration of the PDMA modules in the device MCU domain.



**Figure 10-36. MCU Domain PDMA Integration**

Table 10-170 through summarize the integration of PDMA modules in the device MCU domain.

**Table 10-170. MCU Domain PDMA Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
MCU_PDMA0 (MCU_PDMA_MISC_G0)	WKUP_PSC0	PD0	LPSC0	MCU_CBASS0
MCU_PDMA1 (MCU_PDMA_MISC_G1)	WKUP_PSC0	PD0	LPSC0	MCU_CBASS0
MCU_PDMA2 (MCU_PDMA_MISC_G2)	WKUP_PSC0	PD0	LPSC0	MCU_CBASS0
MCU_PDMA3 (MCU_PDMA_ADC)	WKUP_PSC0	PD0	LPSC0	MCU_CBASS0

**Table 10-171. MCU Domain PDMA Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
MCU_PDMA0	MCU_PDMA0_FICLK	MCU_SYSCCLK0/3	WKUP_PLLCTRL0	MCU_PDMA0 functional and interface clock
MCU_PDMA1	MCU_PDMA1_FICLK	MCU_SYSCCLK0/3	WKUP_PLLCTRL0	MCU_PDMA1 functional and interface clock
MCU_PDMA2	MCU_PDMA2_FICLK	MCU_SYSCCLK0/3	WKUP_PLLCTRL0	MCU_PDMA2 functional and interface clock
MCU_PDMA3	MCU_PDMA3_FICLK	MCU_SYSCCLK0/6	WKUP_PLLCTRL0	MCU_PDMA3 functional and interface clock

Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
MCU_PDMA0	MCU_PDMA0_RST	MOD_G_RST	LPSC0	MCU_PDMA0 hardware reset
MCU_PDMA1	MCU_PDMA1_RST	MOD_G_RST	LPSC0	MCU_PDMA1 hardware reset

**Table 10-171. MCU Domain PDMA Clocks and Resets (continued)**

MCU_PDMA2	MCU_PDMA2_RST	MOD_G_RST	LPSC0	MCU_PDMA2 hardware reset
MCU_PDMA3	MCU_PDMA3_RST	MOD_G_RST	LPSC0	MCU_PDMA3 hardware reset

### 10.3.1.2.2 PDMA Integration in MAIN Domain

Figure 10-37 shows the integration of the PDMA modules in the device MAIN domain.

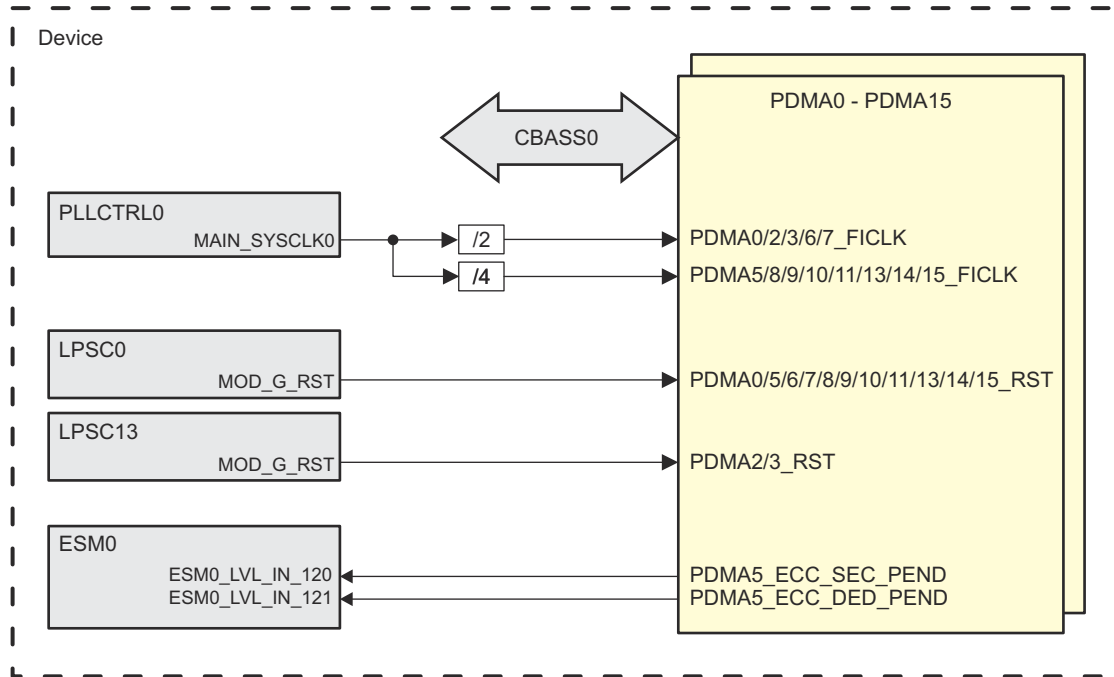

**Figure 10-37. MAIN Domain PDMA Integration**

Table 10-172 through Table 10-174 summarize the integration of the PDMA modules in the device MAIN domain.

**Table 10-172. MAIN Domain PDMA Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
PDMA0 (PDMA_AASRC)	PSC0	PD0	LPSC0	CBASS0
PDMA2 (PDMA_DEBUG_CCMCU)	PSC0	PD0	LPSC13	CBASS0
PDMA3 (PDMA_DEBUG_C66)	PSC0	PD0	LPSC13	CBASS0
PDMA5 (PDMA_MCAN)	PSC0	PD0	LPSC0	CBASS0
PDMA6 (PDMA_MCASP_G0)	PSC0	PD0	LPSC0	CBASS0
PDMA7 (PDMA_MCASP_G1)	PSC0	PD0	LPSC0	CBASS0
PDMA8 (PDMA_MISC_G0)	PSC0	PD0	LPSC0	CBASS0
PDMA9 (PDMA_MISC_G1)	PSC0	PD0	LPSC0	CBASS0
PDMA10 (PDMA_MISC_G2)	PSC0	PD0	LPSC0	CBASS0
PDMA11 (PDMA_MISC_G3)	PSC0	PD0	LPSC0	CBASS0
PDMA13 (PDMA_USART_G0)	PSC0	PD0	LPSC0	CBASS0
PDMA14 (PDMA_USART_G1)	PSC0	PD0	LPSC0	CBASS0
PDMA15 (PDMA_USART_G2)	PSC0	PD0	LPSC0	CBASS0

**Table 10-173. MAIN Domain PDMA Clocks and Resets**

Clocks
--------



**Table 10-173. MAIN Domain PDMA Clocks and Resets (continued)**

Module Instance	Module Clock Input	Source Clock Signal	Source	Description
PDMA0	PDMA0_FICLK	MAIN_SYSCLK0/2	PLLCTRL0	PDMA0 functional and interface clock
PDMA2	PDMA2_FICLK	MAIN_SYSCLK0/2	PLLCTRL0	PDMA2 functional and interface clock
PDMA3	PDMA3_FICLK	MAIN_SYSCLK0/2	PLLCTRL0	PDMA3 functional and interface clock
PDMA5	PDMA5_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	PDMA5 functional and interface clock
PDMA6	PDMA6_FICLK	MAIN_SYSCLK0/2	PLLCTRL0	PDMA6 functional and interface clock
PDMA7	PDMA7_FICLK	MAIN_SYSCLK0/2	PLLCTRL0	PDMA7 functional and interface clock
PDMA8	PDMA8_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	PDMA8 functional and interface clock
PDMA9	PDMA9_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	PDMA9 functional and interface clock
PDMA10	PDMA10_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	PDMA10 functional and interface clock
PDMA11	PDMA11_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	PDMA11 functional and interface clock
PDMA13	PDMA13_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	PDMA13 functional and interface clock
PDMA14	PDMA14_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	PDMA14 functional and interface clock
PDMA15	PDMA15_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	PDMA15 functional and interface clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
PDMA0	PDMA0_RST	MOD_G_RST	LPSC0	PDMA0 hardware reset
PDMA2	PDMA2_RST	MOD_G_RST	LPSC13	PDMA2 hardware reset
PDMA3	PDMA3_RST	MOD_G_RST	LPSC13	PDMA3 hardware reset
PDMA5	PDMA5_RST	MOD_G_RST	LPSC0	PDMA5 hardware reset
PDMA6	PDMA6_RST	MOD_G_RST	LPSC0	PDMA6 hardware reset
PDMA7	PDMA7_RST	MOD_G_RST	LPSC0	PDMA7 hardware reset
PDMA8	PDMA8_RST	MOD_G_RST	LPSC0	PDMA8 hardware reset
PDMA9	PDMA9_RST	MOD_G_RST	LPSC0	PDMA9 hardware reset
PDMA10	PDMA10_RST	MOD_G_RST	LPSC0	PDMA10 hardware reset
PDMA11	PDMA11_RST	MOD_G_RST	LPSC0	PDMA11 hardware reset
PDMA13	PDMA13_RST	MOD_G_RST	LPSC0	PDMA13 hardware reset
PDMA14	PDMA14_RST	MOD_G_RST	LPSC0	PDMA14 hardware reset
PDMA15	PDMA15_RST	MOD_G_RST	LPSC0	PDMA15 hardware reset

**Table 10-174. MAIN Domain PDMA Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
PDMA5	PDMA5_ECC_SEC_PEND	ESM0_LVL_IN_120	ESM0	PDMA5 SEC ECC error interrupt	Level
	PDMA5_ECC_DED_PEND	ESM0_LVL_IN_121	ESM0	PDMA5 DED ECC error interrupt	Level



### 10.3.1.3 PDMA Functional Description

#### 10.3.1.3.1 PDMA Functional Blocks

Figure 10-38 shows the PDMA block diagram.

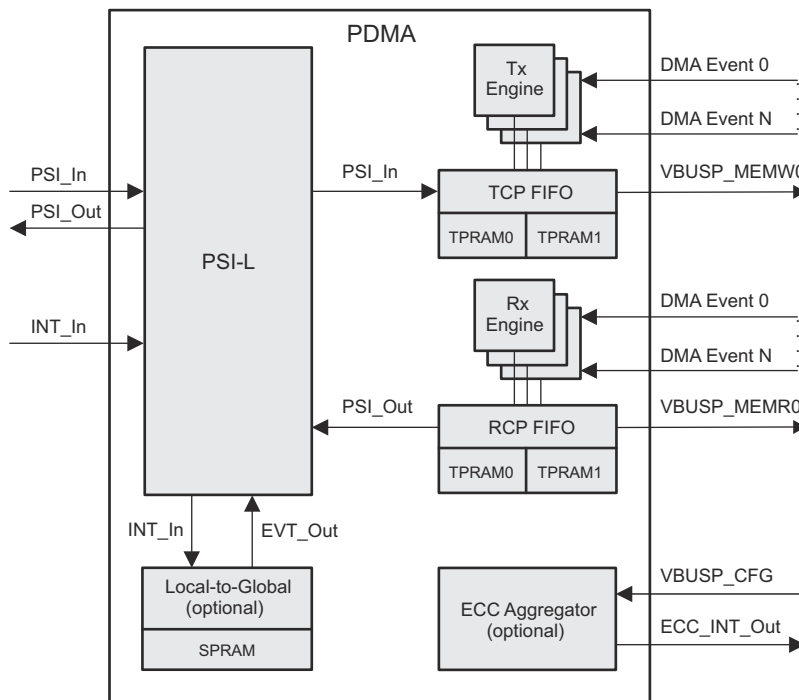


Figure 10-38. PDMA Block Diagram

#### Note

For detailed description of the PSI-L interface, see [Section 10.2.8, Packet Streaming Interface Link \(PSI-L\)](#).

##### 10.3.1.3.1.1 Scheduler

The scheduler block is responsible for monitoring the fullness level of the various FIFOs, monitoring input DMA triggering events, maintaining channel state information, and issuing credits to the Tx and Rx DMA units when it is time to perform each low-level read or write operation.

##### 10.3.1.3.1.2 Tx Per-Channel Buffers (TCP FIFO)

The Tx per-channel buffers implement channelized FIFOs for each DMA channel. A single logical data FIFO is provided for each destination DMA channel that is used for buffering payload data that has been pushed into the PDMA from the Tx PSI-L interface. The Tx per-channel buffers are always written with full Tx PSI-L wide words (except for EOL/EOP data phases) but are read on byte granularity from the Tx DMA units.

##### 10.3.1.3.1.3 Tx DMA Unit (Tx Engine)

The Tx DMA unit block implements all of the state machine functionality necessary to implement static TR type UTC destination channels. The Tx DMA unit waits until it is triggered by an incoming DMA event and then writes data from the Tx per-channel data FIFO associated with the channel to an external memory mapped target via a VBUSP master interface. The number and width of the writes that are performed is in accordance with the parameters, which have been programmed via PSI-L into the static TR for the channel.

#### 10.3.1.3.1.4 Rx Per-Channel Buffers (RCP FIFO)

The Rx per-channel buffers implement a logical FIFO for each source DMA channel that is used for buffering payload data that has been fetched by the Rx DMA units. The buffers are byte-oriented on write so that the data from the Rx DMA units, which may not be full words, can be packed properly. The buffers are word-oriented on read in accordance with the transport mechanism outlined in the PSI-L interface specification. Each channel in the Rx DMA controller maps directly onto a thread in the Rx PSI-L interface.

The Rx per-channel buffers block outputs queue fullness information to the scheduler block, which it then uses to determine when it should initiate DMA opportunities to backfill the buffers. The Rx per-channel buffers will initiate transfers to the remote paired thread whenever any data is available in each channel buffer and credits are available in the corresponding thread. The block will simultaneously monitor the status of all of the threads and will perform a round-robin arbitration between the different threads for the use of the Rx PSI-L interface. Each thread for which the target is indicating it can accept data and which currently has data available in the channel buffer will be included in the arbitration.

#### 10.3.1.3.1.5 Rx DMA Unit (Rx Engine)

The Rx DMA unit block implements all of the state machine functionality necessary to implement static TR type UTC source channels. The Rx DMA unit waits until it is triggered by an incoming DMA event and then reads data from an external memory mapped source via a VBUSP master read interface into the Rx per-channel data FIFO associated with the channel. The number and width of the writes that are performed is in accordance with the parameters, which have been programmed via PSI-L into the static TR for the channel.

#### 10.3.1.3.2 PDMA General Functionality

##### 10.3.1.3.2.1 Operational States

At any given time the PDMA can be in one of three different states as follows:

- **INIT:** This is the initial state of the machine during and immediately after reset. During this state, all of the RAMs inside the PDMA will be initialized to known values including the ECC redundant parity bits. While in the INIT state, the DMA will de-assert all 'ready' signals on all applicable slave interfaces and will de-assert all 'request' signals on all applicable master interfaces. The PDMA will automatically transition out of the INIT state into the IDLE state when all of the RAM initialization has been completed.
- **IDLE:** Once the PDMA leaves the INIT state, it enters the IDLE state whenever no outstanding transactions are pending on any of the PDMA interfaces (master or slave). The IDLE state is generally a transient state and is used by the PDMA to determine when it is appropriate to allow the SoC power management complex to turn off the clock. As channels have work queued on them and transactions begin flowing into the system, the PDMA transitions to the ACTIVE state. When no more work is pending, or when the host pauses/disables active channels, or when the power management complex on the SoC desires to shut down the PDMA and asserts 'clock stop request', the PDMA will account for any outstanding transactions and will re-enter the IDLE state. The PDMA leaves the IDLE state anytime it generates or receives a transactions that requires a return response as those protocols dictate that the clock must remain running to avoid faulting the handshaking protocol.
- **ACTIVE:** The PDMA enters the ACTIVE state as soon as it issues a transaction or receives a transaction on any interface that uses a split protocol (expects a later response for a request). When all transactions have been accounted for (responses have all been either received or sent), the PDMA transitions to the IDLE state.

##### 10.3.1.3.2.2 Clock Stop

The clock stop interface allows the PDMA to be gracefully commanded to shut down its operations and enter into an IDLE state so that the main clock can be stopped. When the 'clock stop request' input is asserted, the PDMA will stop processing TRs for each channel at the next TR boundary. Once all of the channels have gracefully reached the end of their current TRs, the PDMA will assert the 'clock stop acknowledge' output. Once the PDMA has entered the IDLE state, it will remain there until the 'clock stop request' is de-asserted.

### 10.3.1.3.2.3 Emulation Control

The emulation control input (EMUSUSP) and the per-channel emulation control FREE bit allow PDMA operation to be suspended on a per-channel basis. When the emulation suspend state is entered, the PDMA will stop processing receive and transmit TRs for each channel at the next TR boundary. Any TR currently in reception or transmission will be halted at its next FIFO boundary as configured through the 'X' and 'Y' parameters in the static TR.

Emulation control is implemented for compatibility with other peripherals. Each source and destination channel can be configured to either honor the suspend signal, or to 'free run' without regard to suspend. This is controlled via the PDMA\_PSILCFG\_TX\_RT\_ENABLE / PDMA\_PSILCFG\_RX\_RT\_ENABLE[0] FREE bit.

### 10.3.1.3.3 PDMA Events and Flow Control

The following sections describe the simplified mechanism that is provided on the PDMA for triggering and completion event generation.

#### 10.3.1.3.3.1 Channel Types

Each channel in the PDMA is configured at design time (and cannot be changed by software) to be either standard X-Y FIFO mode or MCAN channel or AASRC channel. Either of these two sections provide information about the channel type support in the device:

- [Section 10.3.1.1.1, PDMA Features](#)
- [Section 10.3.2, PDMA Sources](#)

When reading the sections below, ignore any information that concerns unsupported channel type.

#### 10.3.1.3.3.1.1 X-Y FIFO Mode

Most peripherals which are serviced by the PDMA follow the pattern of transferring 'X' bytes from a fixed, non-changing address in a burst 'Y' times for each triggering event that is received. The 'X' parameter is typically 1-16 bytes and the 'Y' parameter can be any integer up to 2048.

#### 10.3.1.3.3.1.2 MCAN Mode

The MCAN module is found to require a few minor differences than standard X-Y mode. First, buffer ownership on transmit is different in that the PDMA will start out owning the TX buffer space (instead of waiting for an initial DMA event from the MCAN). There is also a requirement to perform a write of a fixed value to a fixed address after each DMA event is serviced to pass ownership of a buffer back to the MCAN. Finally, there is an MCAN packet fragmentation requirement to place an 8-byte header on each 64-byte chunk that is transmitted and to remove the header from each 64-byte chunk (other than the first block) that is received.

#### 10.3.1.3.3.1.3 AASRC Mode

The AASRC mode of the PDMA is similar to the X-Y mode in that X specifies the sample width, any Y specifies the number of FIFO samples to transfer per DMA request, but in addition to this, a programmable FIFO list is supplied that allows a single PDMA channel to transfer data to multiple FIFOs. The FIFO list specifies which FIFOs to access, and in which order to access them. Both AASRC 'stream' and 'group' modes are supported in terms of DMA signalling and the FIFO memory map. When in AASRC mode, the entire PDMA operates exclusively in this mode. AASRC channels do not share the same PDMA as do XY and MCAN channels.

The following additional details apply to the PDMA in AASRC mode:

- Main features:
  - Up to 16 independent RX channels and 16 TX channels
    - Each channel represents a different data stream to either the UDMA-P or a McASP PDMA
  - Each channel can be configured to read or write any of the AASRC FIFOs in any order
  - Packing support for 1, 2, 3, and 4 byte samples
    - Can directly talk to a McASP using any McASP data format
    - Packing sample size is fixed for a given channel

- Supports both 'group' and 'stream' AASRC signaling and FIFO memory maps
  - There is a single stream/group mode setting for each PDMA channel
- Additional details:
  - Each channel has a 'mask' of which DMA requests must fire to activate the channel
    - In stream mode, the mask specifies the all the FIFO DMA requests that must fire
    - In group mode, the mask specifies the one group DMA request that must fire
  - Each channel specifies an ordered list of how FIFOs should be accessed
    - All channels use a common list, specifying a range of entry indices within that list
      - This allow each channel to use a varying number of list elements
    - The order list can be processed multiple times per DMA request
      - The number of samples to be transferred per request must match the configured threshold for the DMA request in the AASRC
  - All FIFOs are specified by index only
    - In the TX ordering table, the indices are assumed to be TX FIFOs
    - In the RX ordering table, the indices are assumed to be RX FIFOs
    - If a DMA channel is in group mode, the PDMA assumes the indices referenced by that channel represent group mode FIFOs; otherwise, it assumes stream mode FIFOs
  - The PDMA has configured base addressed and inter-FIFO spans such that is can always convert from a FIFO index to the proper Stream or Group mode FIFO address

#### 10.3.1.3.3.1.4

#### 10.3.1.3.3.2 Channel Triggering

Channels must be triggered in order for them to perform work. A local event input bus is provided on the PDMA and each bit in the input bus corresponds to the trigger for the channel with the same channel index as the bit index in the bus (bit 0 triggers channel 0, bit 1 triggers channel 1, etc.).

The PDMA provides a 2-bit counter per event input to accommodate startup latency in the channel.

#### 10.3.1.3.3.3 Completion Events

All transfers on PDMA are split TRs in that they have a UDMA-P half and a (static) PDMA half. Completion events are designed to be triggered from the UDMA-P half of the split TR.

#### 10.3.1.3.4 PDMA Transmit Operation

##### 10.3.1.3.4.1 Destination (Tx) Channel Allocation

A total of  $N$  destination channels are provided within the PDMA for concurrent transfers from Tx per-channel buffers to the various attached peripherals, where  $N$  is a design-time configurable paramater. Each Tx channel requires a single PSI-L thread. See [Section 10.3.2](#) for Tx channel allocation for each PDMA.

##### 10.3.1.3.4.2 Destination (Tx) Channel Out-of-Band Signals

[Table 10-175](#) shows how PSIL signals are handled for destination channels.

**Table 10-175. Tx Channel Out-of-Band Signals**

PSI-L Signal	XY/AASRC Mode	MCAN Mode
SOP	Ignored	Ignored
EOP	Ignored	Closes current MCAN packet
SOL	Ignored	Ignored
EOL	Ignored	Ignored
DROP	Ignored	Ignored
PKT_ERROR	Ignored	Ignored
PAUSE	Ignored	Ignored
TDOWN	Reflect in status; teardown when data marker reached	Reflect in status; teardown when data marker reached

#### 10.3.1.3.4.3 Destination Channel Initialization

After reset, all threads/channels in the PDMA will be idle and waiting for work to be assigned to them. In order to initiate PDMA operation, the host will need to first pair the channel with a remote data source (normally a UDMA-P channel), setup the static TR for the channel, and enable the thread.

##### 10.3.1.3.4.3.1 PSI-L Destination Thread Pairing

After reset, all threads on PSI-L are uninitialized and unpaired. The host will pair each destination PDMA channel with (typically) a corresponding source channel in the UDMA-P. The PDMA thread is configured to point to the UDMA-P thread and the UDMA-P thread is configured to point to the PDMA thread. Once paired, the source channels in the UDMA-P will send all their data to the destination channel in the PDMA and the destination channel in the PDMA will never see any data other than data from the source channel in the UDMA-P.

The following PSI-L registers are related to destination thread pairing:

- PDMA\_PSILCFG\_TX\_ENABLE
- PDMA\_PSILCFG\_TX\_CAPABILITIES
- PDMA\_PSILCFG\_TX\_BYTE\_COUNT
- PDMA\_PSILCFG\_TX\_RT\_ENABLE

Refer to their descriptions for further details.

##### 10.3.1.3.4.3.2 Static Transfer Request Setup

After reset, all channels in the PDMA will be idle and waiting for work to be assigned to them. In order to initiate PDMA operation, the host will send configuration transactions across PSI-L and will setup the static TR for each destination channel. The format of the TR is identical for each DMA mode, consisting of 'X' and 'Y' parameters, but the field interpretation varies somewhat. The following PSI-L register bit fields are used to setup the static TR:

##### 10.3.1.3.4.3.3

- PDMA\_PSILCFG\_TX\_STATIC\_TR[26-24] X
- PDMA\_PSILCFG\_TX\_STATIC\_TR[11-0] Y

Refer to their descriptions for further details.

##### 10.3.1.3.4.3.4 PSI-L Destination Thread Enables

PSI-L destination threads must first be enabled in order to accept data. Threads are enabled or disabled by setting or clearing the ENABLE bit in the PSI-L pairing registers for the thread. When a thread is disabled, it must drop any data phases which are sent but properly return the credits for the data phases which are dropped.

Once a thread is paired, the TR has been set up and the enable is asserted, the channel is armed and ready to be triggered to perform the specified set of write transactions.

##### 10.3.1.3.4.4 Data Transfer

Packet transmission is accomplished within the PDMA by unpacking and moving data from the Tx per-channel FIFOs which were filled via the transmit PSI-L interface to specified memory mapped address ranges via the VBUSP master interfaces. On the Tx side of the PDMA, these transfers are always writes. Each write transfer which is performed by the Tx DMA unit is to a destination address that is hardcoded in the channel at design time and of a size as specified in the static transfer request.

The sequences of logical transactions that are performed by the PDMA on the memory interface during transmit is dependent on the channel type. The following sections describe what will happen for the two different channel types.

##### 10.3.1.3.4.4.1 X-Y FIFO Mode Channel

The PDMA channel will remain idle until a pulse is detected on the associated input DMA request event pin. Once the pulse is detected, the DMA will sequentially issue a total of 'Y' parameter writes of 'X' parameter bytes to the data FIFO address specified for the channel. Each write that the DMA performs will be a single

'X' element in size (no large bursts). Once the total specified number of transactions has been completed the channel will return to an idle state and wait until it is triggered again. The write transfers that are performed will be accomplished as quickly as possible given availability of data in the Tx channelized FIFO and given the arbitration that may occur as a result of other channels also using the same write unit.

#### 10.3.1.3.4.4.1.1 X-Y FIFO Burst Mode

The burst mode for XY is designed to allow the PDMA to burst across a FIFO region, while extracting out the 'sample size' from each data phase. So for example, it can read a "burst" of multiple 32-bit words from a McASP, and grab only 24-bits from each data phase.

It uses the same registers as standard XY mode, where X is the encoded sample size, and Y is the number of samples to read/write per DMA request. Setting the PDMA\_PSILCFG\_TX\_STATIC\_TR[31] BURST bit enables burst.

The following are the rules for enabling burst mode. Incompatible peripherals do not need to enable burst, or can enable it on RX without enabling it on TX.

- The VBUSP address will increment throughout the burst
- Bursts will be sub-divided to fit into a 64 byte transfer window
- One sample is transferred for every bus data phase
  - Cannot burst one byte samples from a 16-bit peripheral on a 32-bit bus
  - Cannot burst 64-bit samples on a 32-bit bus (configure for 2×32-bit samples instead)
- PDMA will 'gap' the byte enables on writes as needed

#### 10.3.1.3.4.4.2 MCAN Mode Channel

The PDMA channel will remain idle until data is received from the UDMA-P over the PSIL interface. At this time, the PDMA will immediately start copying packet bytes into the MCAN TX buffer corresponding the PDMA/MCAN channel. The number of bytes copied is smaller of the remaining bytes in the packet or the value of the 'Y' parameter. Once a TX buffer has been filled, the PDMA will transfer ownership of the buffer to MCAN by performing an MCAN register write. The PDMA will then wait to regain ownership of the TX buffer by waiting for the corresponding MCAN TX event. At this time, the PDMA will continue with refilling the TX buffer for the next packet fragment. On subsequent fills (until the end of packet is reached), the PDMA will skip over the 8-byte MCAN header, leaving the original contents from the initial fragment copy in place.

#### 10.3.1.3.4.4.2.1 MCAN Burst Mode

Since MCAN buffers are stored in linear memory, the burst mode for MCAN is a simple linear burst across the transfer window. The max burst size is set to the 72 byte size of the MCAN buffer. This will allow a full MCAN packet to be read out as a single burst.

#### 10.3.1.3.4.4.3 AASRC Mode Channel

The AASRC channel is controlled primarily via the PDMA\_PSILCFG\_TX\_AASRC\_TX\_FIFO\_CONFIG register for the channel. This register holds three basic pieces of information:

- Whether the channel uses AASRC stream mode or group mode
- The slot range in the order table used (from designated first slot to designated last slot)
- The AASRC DMA request event(s) that must fire before the channel becomes active

The channel will remain idle until a pulse is detected on ALL associated input DMA request event pins required by the PDMA\_PSILCFG\_TX\_AASRC\_TX\_FIFO\_CONFIG setting. The pulses for the individual events are latched and held by the PDMA until they all arrive. Once the channel activates, it will start reading FIFO index values from TX order table (PDMA\_PSILCFG\_TX\_AASRC\_TX\_ORDER\_TABLE0, PDMA\_PSILCFG\_TX\_AASRC\_TX\_ORDER\_TABLE1), starting at the configured FIRSTSLOT and ending with the LASTSLOT. The actual FIFO indices used are obtained from the ordering table. For example, say FIRSTSLOT=3 and LASTSLOT=5. If the first 6 slots of the ordering table were: 0, 2, 4, 6, 8, 10, the FIFOs written for the event would be 6, 8, and 10, because 'slot 3' of the table contains 6, and it would proceed through 'slot 5' of the table which contains 10.



The X and Y registers are still used as they are in a normal X-Y mode. When accessing each FIFO, the setting of X determines the sample byte width written to the FIFO. The value of 'Y' determines how many times the entire FIFO list is processed for each activation of the channel. Once the total specified number of transactions has been completed the channel will return to an idle state and wait until it is triggered again. The write transfers that are performed will be accomplished as quickly as possible given availability of data in the TX channelized FIFO and given the arbitration that may occur as a result of other channels also using the same write unit.

#### **10.3.1.3.4.5 Tx Pause**

The host initiates a channel pause by setting the PDMA\_PSILCFG\_TX\_RT\_ENABLE[9] PAUSE bit. The paused channel can be resumed by clearing the register bit.

#### **10.3.1.3.4.6 Tx Teardown**

The host initiates a channel teardown by setting the TDOWN bit in the UDMA-P channel that is paired with the PDMA. The UDMA-P communicates the teardown state through the PSI-L data channel, to ensure that the teardown is not seen by the PDMA until all the previous UDMA-P data for the channel has been flushed. At this time, the teardown state will be reflected in the PDMA\_PSILCFG\_TX\_RT\_ENABLE[30] TDOWN bit. Note that a non-synchronized teardown can also be initiated by directly clearing the PDMA\_PSILCFG\_TX\_RT\_ENABLE[31] ENABLE.

Once all data has been flushed from the PDMA to the peripheral, the enable state of the PDMA channel will be cleared in the PDMA\_PSILCFG\_TX\_RT\_ENABLE register, but the teardown bit will remain high. Upon completion, no further packet processing will occur until the host re-configures the channel. If the channel fails to teardown because the peripheral has stopped responding, or if the UDMA-P transmission stops on a data boundary that is not compatible with the static TR configuration, the PDMA\_PSILCFG\_TX\_RT\_ENABLE[28] FLUSH bit can be set to guarantee that all data can be properly flushed from the internal pipe.

#### **10.3.1.3.4.7 Tx Channel Reset**

In the unlikely event that channel synchronization is corrupted, a channel may fail to teardown gracefully, even with flush enabled. If this occurs, the channel may be reset by clearing the PDMA\_PSILCFG\_TX\_ENABLE[31] ENABLE bit. This will cause a local reset of the entire channel, including TR and pairing registers. Note that it does not reset the UDMA-P peer. Resetting the UDMA-P peer is also required before re-initializing and re-pairing the channel.

#### **10.3.1.3.4.8 Tx Debug/State Registers**

The debug/state registers are supplied to give software applications additional information about the PDMA than they would need in regular operation, but which may be useful in debug situations. These registers appear on the PSI-L bus, near the static TR registers. For transmit, they are defined as follows:

- PDMA\_PSILCFG\_TX\_DEBUG\_1
- PDMA\_PSILCFG\_TX\_DEBUG\_2

Refer to their descriptions for further details.

#### **10.3.1.3.5 PDMA Receive Operation**

##### **10.3.1.3.5.1 Source (Rx) Channel Allocation**

A total of  $M$  destination channels are provided within the PDMA for concurrent transfers from the various attached peripherals into the Rx per-channel buffers and on to the PSI-L Rx Interface, where  $M$  is a design-time configurable parameter. Each Rx channel requires a single PSI-L thread. See [Section 10.3.2](#) for Rx channel allocation for each PDMA.

##### **10.3.1.3.5.2 Source Channel Initialization**

After reset, all threads/channels in the PDMA will be idle and waiting for work to be assigned to them. In order to initiate PDMA operation, the host will need to first pair the channel with a remote data source (normally a UDMA-P channel), setup the static TR for the channel, and enable the thread.

### 10.3.1.3.5.2.1 PSI-L Source Thread Pairing

After reset, all threads on PSI-L are uninitialized and unpaired. The host will pair each source PDMA channel with (typically) a corresponding destination channel in the UDMA-P. The PDMA thread is configured to point to the UDMA-P thread and the UDMA-P thread is configured to point to the PDMA thread. Once paired, the source channels in the PDMA will send all their data to the source channel in the UDMA-P and the destination channel in the UDMA-P will never see any data other than data from the source channel in the PDMA.

The following PSI-L registers are related to source thread pairing:

- PDMA\_PSILCFG\_RX\_PEER\_THREAD\_ID
- PDMA\_PSILCFG\_RX\_PEER\_CREDIT
- PDMA\_PSILCFG\_RX\_ENABLE
- PDMA\_PSILCFG\_RX\_BYTE\_COUNT
- PDMA\_PSILCFG\_RX\_RT\_ENABLE

Refer to their descriptions for further details.

### 10.3.1.3.5.2.2 Static Transfer Request Setup

After reset, all channels in the PDMA will be idle and waiting for work to be assigned to them. In order to initiate PDMA operation, the Host will send configuration transactions across PSI-L and will set up the static TR for each destination channel. The format of the TR is identical for each DMA mode, consisting of 'X', 'Y', and 'Z' parameters, but the field interpretation varies somewhat. The following PSI-L register bit fields are used to setup the static TR:

- PDMA\_PSILCFG\_RX\_STATIC\_TR[26-24] X
- PDMA\_PSILCFG\_RX\_STATIC\_TR[11-0] Y
- PDMA\_PSILCFG\_RX\_STATIC\_TR\_Z[11-0] Z

Refer to their descriptions for further details.

### 10.3.1.3.5.2.3 PSI-L Source Thread Enables

PSI-L source threads must be enabled in order to process peripheral DMA requests and send data. When disabled, DMA requests are ignored.

Once a thread is paired, the TR has been set up and the enable is asserted, the channel is armed and ready to be triggered to perform the specified set of read transactions.

### 10.3.1.3.5.3 Data Transfer

Packet reception is accomplished within the PDMA by moving data from structures that are located in memory accessible via the VBUSP Memory Interface(s) onto the receive PSI-L interface. On the Rx side of the PDMA, these transfers are always reads. Data is read from an attached memory mapped space and packed into the Rx per-channel buffer for that channel. At a later time, the data is moved from the Rx per-channel FIFO to a remote peer DMA entity via the Rx PSI-L interface.

The sequences of logical transactions that are performed by the PDMA on the memory interface during receive is dependent on the channel type. The following sections describe what will happen for the two different channel types.

#### 10.3.1.3.5.3.1 X-Y FIFO Mode Channel

The PDMA channel will remain idle until a pulse is detected on the associated input DMA request event pin. Once the pulse is detected, the DMA will sequentially issue a total of 'Y' parameter reads of 'X' parameter bytes from the data FIFO address specified for the channel. Each read that the DMA performs will be a single 'X' element in size (no large bursts). Once the total specified number of transactions has been completed the channel will return to an idle state and wait until it is triggered again. The read transfers that are performed will be accomplished as quickly as possible given availability of data in the Rx channelized FIFO and given the arbitration that may occur as a result of other channels also using the same read unit.



### 10.3.1.3.5.3.2 MCAN Mode Channel

The PDMA channel will remain idle until a pulse is detected on the associated input DMA request event pin. The DMA event pins are mapped from the CAN filter event bits. The event bits determine the channel and one of two CAN buffers to use for the RX operation. This allows the RX buffers on the CAN to be configured in a ping/pong fashion, preventing the loss of CAN packet data. When an event is received, the PDMA will start copying bytes out of the corresponding CAN buffer. The number of bytes copied is equal to the value of the 'Y' parameter in the channel configuration. Once all data has been copied from the buffer, the PDMA transfers ownership of the buffer back to the CAN by writing a CAN register. It then waits for another RX DMA event. The PDMA can copy multiple CAN RX buffers to the same CPPI packet. It will not close out the CPPI packet until it has copied out a number of RX buffers equal to the 'Z' parameter in the channel configuration. On subsequent RX buffer copies, the MCAN will skip the first 8 bytes of the RX buffer, which is the MCAN packet header.

The mapping of MCAN filter events to MCAN channels and buffer is important for properly configuring the DMA. The mapping is defined as shown in [Table 10-176](#).

**Table 10-176. Mapping of MCAN Filter Events to MCAN Channels**

CAN FE2	CAN FE1	CAN FE0	Description
0	0	0	Not used
0	0	1	Not used
0	1	0	PDMA CAN channel offset 0, RX buffer 0
0	1	1	PDMA CAN channel offset 0, RX buffer 1
1	0	0	PDMA CAN channel offset 1, RX buffer 2
1	0	1	PDMA CAN channel offset 1, RX buffer 3
1	1	0	PDMA CAN channel offset 2, RX buffer 4
1	1	1	PDMA CAN channel offset 2, RX buffer 5

By using two filter entries in the CAN, software can setup a ping-pong buffer for a particular RX packet ID. For example, the following two filter entries will setup a ping-pong using RX buffers 0 and 1 for packet ID = 5 on CAN RX channel 0 ([Table 10-177](#)).

**Table 10-177. Ping-Pong Buffer Example in MCAN Mode**

Filter Entry	Packet Match ID	Filter Event Bits
0	5	0x2
1	5	0x3

### 10.3.1.3.5.3.2.1 MCAN Burst Mode

Since MCAN buffers are stored in linear memory, the burst mode for MCAN is a simple linear burst across the transfer window. The max burst size is set to the 72 byte size of the MCAN buffer. This will allow a full MCAN packet to be read out as a single burst.

### 10.3.1.3.5.3.3 AASRC Mode Channel

The AASRC channel is controlled primarily via the PDMA\_PSILCFG\_RX\_AASRC\_RX\_FIFO\_CONFIG register for the channel. This register holds three basic pieces of information:

- Whether the channel uses AASRC stream mode or group mode
- The slot range in the order table used (from designated first slot to designated last slot)
- The AASRC DMA request event(s) that must fire before the channel becomes active

The channel will remain idle until a pulse is detected on ALL associated input DMA request event pins required by the PDMA\_PSILCFG\_RX\_AASRC\_RX\_FIFO\_CONFIG setting. The pulses for the individual events are latched and held by the PDMA until they all arrive. Once the channel activates, it will start reading FIFO index values from RX order table (PDMA\_PSILCFG\_RX\_AASRC\_RX\_ORDER\_TABLE0, PDMA\_PSILCFG\_RX\_AASRC\_RX\_ORDER\_TABLE1), starting at the configured FIRSTSLOT and ending with

the LASTSLOT. The actual FIFO indices used are obtained from the ordering table. For example, say FIRSTSLOT=3 and LASTSLOT=5. If the first 6 slots of the ordering table were: 0, 2, 4, 6, 8, 10, the FIFOs read for the event would be 6, 8, and 10, because 'slot 3' of the table contains 6, and it would proceed through 'slot 5' of the table which contains 10.

The X and Y registers are still used as they are in a normal X-Y mode. When accessing each FIFO, the setting of X determines the sample byte width read from the FIFO. The value of 'Y' determines how many times the entire FIFO list is processed for each activation of the channel. Once the total specified number of transactions has been completed the channel will return to an idle state and wait until it is triggered again. The read transfers that are performed will be accomplished as quickly as possible given availability of data in the TX channelized FIFO and given the arbitration that may occur as a result of other channels also using the same write unit.

#### 10.3.1.3.5.4 Rx Pause

The host initiates a channel pause by setting the PDMA\_PSILCFG\_RX\_RT\_ENABLE[29] PAUSE bit. The paused channel can be resumed by clearing the register bit.

#### 10.3.1.3.5.5 Rx Teardown

The host initiates a RX channel teardown by setting the PDMA\_PSILCFG\_RX\_RT\_ENABLE[30] TDOWN bit for the target RX channel. The PDMA communicates the teardown state to the UDMA-P through the PSI-L data channel, to ensure that the teardown is not seen by the UDMA-P until all the previous PDMA data for the channel has been flushed.

The PDMA will not stop reading peripheral data until it reaches a FIFO boundary as configured through the 'X' and 'Y' parameters in the static TR. It will always attempt to complete the 'Y' count for the current event being processed. Upon reaching a stopping point, the PDMA will then clear the ENABLE bit in the pairing register, however the TDOWN bit will remain set. No further packet processing will occur until the host re-configures the channel. The teardown process will propagate to the UDMA-P and its final status can be checked there.

#### 10.3.1.3.5.6 Rx Channel Reset

In the unlikely event that channel synchronization is corrupted, a channel may fail to teardown gracefully. If this occurs, the channel may be reset by clearing the PDMA\_PSILCFG\_RX\_ENABLE[31] ENABLE bit. This will cause a local reset of the entire channel, including TR and pairing registers. Note that it does not reset the UDMA-P peer. Resetting the UDMA-P peer is also required before re-initializing and re-pairing the channel.

#### 10.3.1.3.5.7 Rx Debug/State Register

The debug/state registers are supplied to give software applications additional information about the PDMA than they would need in regular operation, but which may be useful in debug situations. These registers appear on the PSI-L bus, near the static TR registers. For receive, they are defined as follows:

- PDMA\_PSILCFG\_RX\_DEBUG\_1
- PDMA\_PSILCFG\_RX\_DEBUG\_2
- PDMA\_PSILCFG\_RX\_DEBUG\_3

Refer to their descriptions for further details.

#### 10.3.1.3.6 PDMA ECC Support

The PDMA optionally integrates an ECC aggregator module that consolidates the ECC configuration and status bits for all the ECC-supported memories within the PDMA. The ECC aggregator provides a single end-of-interrupt (EOI) handshake based interrupt to the host (for both single and double error detections), and a standard 32-bit VBUSP interface for configuring and querying the various ECC wrappers via their respective register set.

In this device, only PDMA5 includes an ECC aggregator.

For detailed description of the generic ECC aggregator functionality, see *ECC Aggregator*. For register descriptions of PDMA ECC aggregators, see *PDMA Registers*.

### 10.3.2 PDMA Sources

#### 10.3.2.1 MCU Domain PDMA Event Maps

##### 10.3.2.1.1 MCU\_PDMA\_MISC\_G0 Event Map

**Table 10-178. MCU\_PDMA\_MISC\_G0 Tx Channel Allocation**

Tx PDMA Channel	PDMA Input	Source Event	Channel Type	Trigger Type	Data FIFO Address
8000	SPI_MCU_0_TX_0	MCU_MCSPI0_DMA_WRITE_EVENT_0	XY	Edge	4030 0138h
8001	SPI_MCU_0_TX_1	MCU_MCSPI0_DMA_WRITE_EVENT_1	XY	Edge	4030 014Ch
8002	SPI_MCU_0_TX_2	MCU_MCSPI0_DMA_WRITE_EVENT_2	XY	Edge	4030 0160h
8003	SPI_MCU_0_TX_3	MCU_MCSPI0_DMA_WRITE_EVENT_3	XY	Edge	4030 0174h
8004	MCANSS_MCU_0_TX_0	MCU_MCAN0_MCANSS_TX_DMA_0	MCAN	Pulse	4050 0000h
8005	MCANSS_MCU_0_TX_1	MCU_MCAN0_MCANSS_TX_DMA_1	MCAN	Pulse	4050 0048h
8006	MCANSS_MCU_0_TX_2	MCU_MCAN0_MCANSS_TX_DMA_2	MCAN	Pulse	4050 0090h

**Table 10-179. MCU\_PDMA\_MISC\_G0 Rx Channel Allocation**

Rx PDMA Channel	PDMA Input	Source Event	Channel Type	Trigger Type	Data FIFO Address
0	SPI_MCU_0_RX_0	MCU_MCSPI0_DMA_READ_EVENT_0	XY	Edge	4030 013Ch
1	SPI_MCU_0_RX_1	MCU_MCSPI0_DMA_READ_EVENT_1	XY	Edge	4030 0150h
2	SPI_MCU_0_RX_2	MCU_MCSPI0_DMA_READ_EVENT_2	XY	Edge	4030 0164h
3	SPI_MCU_0_RX_3	MCU_MCSPI0_DMA_READ_EVENT_3	XY	Edge	4030 0178h
4	MCANSS_MCU_0_FE_0	MCU_MCAN0_MCANSS_FE_0	MCAN	Pulse	4050 0900h
5	MCANSS_MCU_0_FE_1	MCU_MCAN0_MCANSS_FE_1	MCAN	Pulse	4050 0990h
6	MCANSS_MCU_0_FE_2	MCU_MCAN0_MCANSS_FE_2	MCAN	Pulse	4050 0A20h

##### 10.3.2.1.2 MCU\_PDMA\_MISC\_G1 Event Map

**Table 10-180. MCU\_PDMA\_MISC\_G1 Tx Channel Allocation**

Tx PDMA Channel	PDMA Input	Source Event	Function	Trigger Type	Data FIFO Address
8000	SPI_MCU_1_TX_0	MCU_MCSPI1_DMA_WRITE_EVENT_0	XY	Edge	4031 0138h
8001	SPI_MCU_1_TX_1	MCU_MCSPI1_DMA_WRITE_EVENT_1	XY	Edge	4031 014Ch
8002	SPI_MCU_1_TX_2	MCU_MCSPI1_DMA_WRITE_EVENT_2	XY	Edge	4031 0160h
8003	SPI_MCU_1_TX_3	MCU_MCSPI1_DMA_WRITE_EVENT_3	XY	Edge	4031 0174h
8004	SPI_MCU_2_TX_0	MCU_MCSPI1_DMA_WRITE_EVENT_0	XY	Edge	4032 0138h
8005	SPI_MCU_2_TX_1	MCU_MCSPI1_DMA_WRITE_EVENT_1	XY	Edge	4032 014Ch
8006	SPI_MCU_2_TX_2	MCU_MCSPI1_DMA_WRITE_EVENT_2	XY	Edge	4032 0160h
8007	SPI_MCU_2_TX_3	MCU_MCSPI1_DMA_WRITE_EVENT_3	XY	Edge	4032 0174h

**Table 10-181. MCU\_PDMA\_MISC\_G1 Rx Channel Allocation**

Rx PDMA Channel	PDMA Input	Source Event	Channel Type	Trigger Type	Data FIFO Address
0	SPI_MCU_1_RX_0	MCU_MCSPI1_DMA_READ_EVENT_0	XY	Edge	4031 013Ch
1	SPI_MCU_1_RX_1	MCU_MCSPI1_DMA_READ_EVENT_1	XY	Edge	4031 0150h
2	SPI_MCU_1_RX_2	MCU_MCSPI1_DMA_READ_EVENT_2	XY	Edge	4031 0164h
3	SPI_MCU_1_RX_3	MCU_MCSPI1_DMA_READ_EVENT_3	XY	Edge	4031 0178h
4	SPI_MCU_2_RX_0	MCU_MCSPI1_DMA_READ_EVENT_0	XY	Edge	4032 013Ch
5	SPI_MCU_2_RX_1	MCU_MCSPI1_DMA_READ_EVENT_1	XY	Edge	4032 0150h
6	SPI_MCU_2_RX_2	MCU_MCSPI1_DMA_READ_EVENT_2	XY	Edge	4032 0164h

**Table 10-181. MCU\_PDMA\_MISC\_G1 Rx Channel Allocation (continued)**

Rx PDMA Channel	PDMA Input	Source Event	Channel Type	Trigger Type	Data FIFO Address
7	SPI_MCU_2_RX_3	MCU_MCSPI1_DMA_READ_EVENT_3	XY	Edge	4032 0178h

### 10.3.2.1.3 MCU\_PDMA\_MISC\_G2 Event Map

**Table 10-182. MCU\_PDMA\_MISC\_G2 Tx Channel Allocation**

Tx PDMA Channel	PDMA Input	Source Event	Channel Type	Trigger Type	Data FIFO Address
8000	USART_MCU_0_TX_0	MCU_UART0_USART_DMA_0	XY	Edge	40A0 0000h
8001	MCANSS_MCU_1_TX_0	MCU_MCAN1_MCANSS_TX_DMA_0	MCAN	Pulse	4054 0000h
8002	MCANSS_MCU_1_TX_1	MCU_MCAN1_MCANSS_TX_DMA_1	MCAN	Pulse	4054 0048h
8003	MCANSS_MCU_1_TX_2	MCU_MCAN1_MCANSS_TX_DMA_2	MCAN	Pulse	4054 0090h

**Table 10-183. MCU\_PDMA\_MISC\_G2 Rx Channel Allocation**

Rx PDMA Channel	PDMA Input	Source Event	Channel Type	Trigger Type	Data FIFO Address
0	USART_MCU_0_RX_0	MCU_UART0_USART_DMA_1	XY	Edge	40A0 0000h
1	MCANSS_MCU_1_FE_0	MCU_MCAN1_MCANSS_FE_0	MCAN	Pulse	4054 0900h
2	MCANSS_MCU_1_FE_1	MCU_MCAN1_MCANSS_FE_1	MCAN	Pulse	4054 0990h
3	MCANSS_MCU_1_FE_2	MCU_MCAN1_MCANSS_FE_2	MCAN	Pulse	4054 0A20h

### 10.3.2.1.4 MCU\_PDMA\_ADC Event Map

**Table 10-184. MCU\_PDMA\_ADC Rx Channel Allocation**

Rx PDMA Channel	PDMA Input	Source Event	Channel Type	Trigger Type	Data FIFO Address
0	ADC12_MCU_0_RX_0	MCU_ADC0_FIFO0_PULSE_0	XY	Pulse	4020 8100h
1	ADC12_MCU_0_RX_1	MCU_ADC0_FIFO1_PULSE_0	XY	Pulse	4020 8200h
2	ADC12_MCU_1_RX_0	MCU_ADC1_FIFO0_PULSE_0	XY	Pulse	4021 8100h
3	ADC12_MCU_1_RX_1	MCU_ADC1_FIFO1_PULSE_0	XY	Pulse	4021 8200h

### 10.3.2.2 MAIN Domain PDMA Event Maps

#### 10.3.2.2.1 PDMA\_AASRC Event Map

**Table 10-185. PDMA\_AASRC Tx Channel Allocation**

Tx PDMA Channel	Function	Channel Type	Trigger Type	Stream FIFO Base Address	Group FIFO Base Address
8000	AASRC Tx Channel 0	AASRC	Pulse	02D2 0000	02D1 0000
8001	AASRC Tx Channel 1	AASRC	Pulse	02D2 0000	02D1 0000
8002	AASRC Tx Channel 2	AASRC	Pulse	02D2 0000	02D1 0000
8003	AASRC Tx Channel 3	AASRC	Pulse	02D2 0000	02D1 0000
8004	AASRC Tx Channel 4	AASRC	Pulse	02D2 0000	02D1 0000
8005	AASRC Tx Channel 5	AASRC	Pulse	02D2 0000	02D1 0000
8006	AASRC Tx Channel 6	AASRC	Pulse	02D2 0000	02D1 0000
8007	AASRC Tx Channel 7	AASRC	Pulse	02D2 0000	02D1 0000

Any of the eight PDMA\_AASRC Tx channels can serve any of the AASRC Tx events listed in [Table 10-186](#).

**Table 10-186. PDMA\_AASRC Tx Events**

PDMA Input	Source Event
OUTFIFO_EVT_[15:0]	AASRC0_OUTFIFO_DMA_[15:0]
OUTGRP_EVT_[3:0]	AASRC0_OUTGRP_DMA_[3:0]

**Table 10-187. PDMA\_AASRC Rx Channel Allocation**

Rx PDMA Channel	Function	Channel Type	Trigger Type	Stream FIFO Base Address	Group FIFO Base Address
0	AASRC Rx Channel 0	AASRC	Pulse	02D2 1000	02D1 0080
1	AASRC Rx Channel 1	AASRC	Pulse	02D2 1000	02D1 0080
2	AASRC Rx Channel 2	AASRC	Pulse	02D2 1000	02D1 0080
3	AASRC Rx Channel 3	AASRC	Pulse	02D2 1000	02D1 0080
4	AASRC Rx Channel 4	AASRC	Pulse	02D2 1000	02D1 0080
5	AASRC Rx Channel 5	AASRC	Pulse	02D2 1000	02D1 0080
6	AASRC Rx Channel 6	AASRC	Pulse	02D2 1000	02D1 0080
7	AASRC Rx Channel 7	AASRC	Pulse	02D2 1000	02D1 0080

Any of the eight PDMA\_AASRC Rx channels can serve any of the AASRC Rx events listed in [Table 10-188](#).

**Table 10-188. PDMA\_AASRC Rx Events**

PDMA Input	Source Event
INFIFO_EVT_[15:0]	AASRC0_INFIFO_DMA_[15:0]
INGRP_EVT_[3:0]	AASRC0_INGRP_DMA_[3:0]

#### 10.3.2.2.2 PDMA\_DEBUG\_CCMCU Event Map

**Table 10-189. PDMA\_DEBUG\_CCMCU Rx Channel Allocation**

Rx PDMA Channel	PDMA Input	Source Event	Channel Type	Trigger Type	Data FIFO Address
0	MCU_DEBUG_CELL_RX_0	DEBUGSS1_DAVDMA_PULSE_0	XY	Edge	0800 0000h
1	CC_DEBUG_CELL_RX_0	CCDEBUGSS0_DAVDMA_PULSE_0	XY	Edge	0800 8000h
2	C66X2_DEBUG_CELL_RX_0	C66DEBUGSS1_DAVDMA_PULSE_0	XY	Edge	0802 0000h

### 10.3.2.2.3 PDMA\_DEBUG\_C66 Event Map

**Table 10-190. PDMA\_DEBUG\_C66 Rx Channel Allocation**

Rx PDMA Channel	PDMA Input	Source Event	Channel Type	Trigger Type	Data FIFO Address
0	MAIN_DEBUG_CELL_RX_0	DEBUGSS0_DAVDMA_PULSE_0	XY	Edge	0800 4000h
1	C66_DEBUG_CELL_RX_0	C66DEBUGSS0_DAVDMA_PULSE_0	XY	Edge	0801 0000h

### 10.3.2.2.4 PDMA\_MCAN Event Map

**Table 10-191. PDMA\_MCAN Tx Channel Allocation**

Tx PDMA Channel	PDMA Input	Source Event	Channel Type	Trigger Type	Data FIFO Address
8000	MCANSS_MAIN_4_TX_0	MCAN4_MCANSS_TX_DMA_0	MCAN	Pulse	0274 8000h
8001	MCANSS_MAIN_4_TX_1	MCAN4_MCANSS_TX_DMA_1	MCAN	Pulse	0274 8048h
8002	MCANSS_MAIN_4_TX_2	MCAN4_MCANSS_TX_DMA_2	MCAN	Pulse	0274 8090h
8003	MCANSS_MAIN_5_TX_0	MCAN5_MCANSS_TX_DMA_0	MCAN	Pulse	0275 8000h
8004	MCANSS_MAIN_5_TX_1	MCAN5_MCANSS_TX_DMA_1	MCAN	Pulse	0275 8048h
8005	MCANSS_MAIN_5_TX_2	MCAN5_MCANSS_TX_DMA_2	MCAN	Pulse	0275 8090h
8006	MCANSS_MAIN_6_TX_0	MCAN6_MCANSS_TX_DMA_0	MCAN	Pulse	0276 8000h
8007	MCANSS_MAIN_6_TX_1	MCAN6_MCANSS_TX_DMA_1	MCAN	Pulse	0276 8048h
8008	MCANSS_MAIN_6_TX_2	MCAN6_MCANSS_TX_DMA_2	MCAN	Pulse	0276 8090h
8009	MCANSS_MAIN_7_TX_0	MCAN7_MCANSS_TX_DMA_0	MCAN	Pulse	0277 8000h
800A	MCANSS_MAIN_7_TX_1	MCAN7_MCANSS_TX_DMA_1	MCAN	Pulse	0277 8048h
800B	MCANSS_MAIN_7_TX_2	MCAN7_MCANSS_TX_DMA_2	MCAN	Pulse	0277 8090h
800C	MCANSS_MAIN_8_TX_0	MCAN8_MCANSS_TX_DMA_0	MCAN	Pulse	0278 8000h
800D	MCANSS_MAIN_8_TX_1	MCAN8_MCANSS_TX_DMA_1	MCAN	Pulse	0278 8048h
800E	MCANSS_MAIN_8_TX_2	MCAN8_MCANSS_TX_DMA_2	MCAN	Pulse	0278 8090h
800F	MCANSS_MAIN_9_TX_0	MCAN9_MCANSS_TX_DMA_0	MCAN	Pulse	0279 8000h
8010	MCANSS_MAIN_9_TX_1	MCAN9_MCANSS_TX_DMA_1	MCAN	Pulse	0279 8048h
8011	MCANSS_MAIN_9_TX_2	MCAN9_MCANSS_TX_DMA_2	MCAN	Pulse	0279 8090h
8012	MCANSS_MAIN_10_TX_0	MCAN10_MCANSS_TX_DMA_0	MCAN	Pulse	027A 8000h
8013	MCANSS_MAIN_10_TX_1	MCAN10_MCANSS_TX_DMA_1	MCAN	Pulse	027A 8048h
8014	MCANSS_MAIN_10_TX_2	MCAN10_MCANSS_TX_DMA_2	MCAN	Pulse	027A 8090h
8015	MCANSS_MAIN_11_TX_0	MCAN11_MCANSS_TX_DMA_0	MCAN	Pulse	027B 8000h
8016	MCANSS_MAIN_11_TX_1	MCAN11_MCANSS_TX_DMA_1	MCAN	Pulse	027B 8048h
8017	MCANSS_MAIN_11_TX_2	MCAN11_MCANSS_TX_DMA_2	MCAN	Pulse	027B 8090h
8018	MCANSS_MAIN_12_TX_0	MCAN12_MCANSS_TX_DMA_0	MCAN	Pulse	027C 8000h
8019	MCANSS_MAIN_12_TX_1	MCAN12_MCANSS_TX_DMA_1	MCAN	Pulse	027C 8048h
801A	MCANSS_MAIN_12_TX_2	MCAN12_MCANSS_TX_DMA_2	MCAN	Pulse	027C 8090h
801B	MCANSS_MAIN_13_TX_0	MCAN13_MCANSS_TX_DMA_0	MCAN	Pulse	027D 8000h
801C	MCANSS_MAIN_13_TX_1	MCAN13_MCANSS_TX_DMA_1	MCAN	Pulse	027D 8048h
801D	MCANSS_MAIN_13_TX_2	MCAN13_MCANSS_TX_DMA_2	MCAN	Pulse	027D 8090h

**Table 10-192. PDMA\_MCAN Rx Channel Allocation**

Rx PDMA Channel	PDMA Input	Source Event	Channel Type	Trigger Type	Data FIFO Address
0	MCANSS_MAIN_4_FE_0	MCAN4_MCANSS_FE_0	MCAN	Pulse	0274 8900h
1	MCANSS_MAIN_4_FE_1	MCAN4_MCANSS_FE_1	MCAN	Pulse	0274 8990h
2	MCANSS_MAIN_4_FE_2	MCAN4_MCANSS_FE_2	MCAN	Pulse	0274 8A20h



**Table 10-192. PDMA\_MCAN Rx Channel Allocation (continued)**

Rx PDMA Channel	PDMA Input	Source Event	Channel Type	Trigger Type	Data FIFO Address
3	MCANSS_MAIN_5_FE_0	MCAN5_MCANSS_FE_0	MCAN	Pulse	0275 8900h
4	MCANSS_MAIN_5_FE_1	MCAN5_MCANSS_FE_1	MCAN	Pulse	0275 8990h
5	MCANSS_MAIN_5_FE_2	MCAN5_MCANSS_FE_2	MCAN	Pulse	0275 8A20h
6	MCANSS_MAIN_6_FE_0	MCAN6_MCANSS_FE_0	MCAN	Pulse	0276 8900h
7	MCANSS_MAIN_6_FE_1	MCAN6_MCANSS_FE_1	MCAN	Pulse	0276 8990h
8	MCANSS_MAIN_6_FE_2	MCAN6_MCANSS_FE_2	MCAN	Pulse	0276 8A20h
9	MCANSS_MAIN_7_FE_0	MCAN7_MCANSS_FE_0	MCAN	Pulse	0277 8900h
10	MCANSS_MAIN_7_FE_1	MCAN7_MCANSS_FE_1	MCAN	Pulse	0277 8990h
11	MCANSS_MAIN_7_FE_2	MCAN7_MCANSS_FE_2	MCAN	Pulse	0277 8A20h
12	MCANSS_MAIN_8_FE_0	MCAN8_MCANSS_FE_0	MCAN	Pulse	0278 8900h
13	MCANSS_MAIN_8_FE_1	MCAN8_MCANSS_FE_1	MCAN	Pulse	0278 8990h
14	MCANSS_MAIN_8_FE_2	MCAN8_MCANSS_FE_2	MCAN	Pulse	0278 8A20h
15	MCANSS_MAIN_9_FE_0	MCAN9_MCANSS_FE_0	MCAN	Pulse	0279 8900h
16	MCANSS_MAIN_9_FE_1	MCAN9_MCANSS_FE_1	MCAN	Pulse	0279 8990h
17	MCANSS_MAIN_9_FE_2	MCAN9_MCANSS_FE_2	MCAN	Pulse	0279 8A20h
18	MCANSS_MAIN_10_FE_0	MCAN10_MCANSS_FE_0	MCAN	Pulse	027A 8900h
19	MCANSS_MAIN_10_FE_1	MCAN10_MCANSS_FE_1	MCAN	Pulse	027A 8990h
20	MCANSS_MAIN_10_FE_2	MCAN10_MCANSS_FE_2	MCAN	Pulse	027A 8A20h
21	MCANSS_MAIN_11_FE_0	MCAN11_MCANSS_FE_0	MCAN	Pulse	027B 8900h
22	MCANSS_MAIN_11_FE_1	MCAN11_MCANSS_FE_1	MCAN	Pulse	027B 8990h
23	MCANSS_MAIN_11_FE_2	MCAN11_MCANSS_FE_2	MCAN	Pulse	027B 8A20h
24	MCANSS_MAIN_12_FE_0	MCAN12_MCANSS_FE_0	MCAN	Pulse	027C 8900h
25	MCANSS_MAIN_12_FE_1	MCAN12_MCANSS_FE_1	MCAN	Pulse	027C 8990h
26	MCANSS_MAIN_12_FE_2	MCAN12_MCANSS_FE_2	MCAN	Pulse	027C 8A20h
27	MCANSS_MAIN_13_FE_0	MCAN13_MCANSS_FE_0	MCAN	Pulse	027D 8900h
28	MCANSS_MAIN_13_FE_1	MCAN13_MCANSS_FE_1	MCAN	Pulse	027D 8990h
29	MCANSS_MAIN_13_FE_2	MCAN13_MCANSS_FE_2	MCAN	Pulse	027D 8A20h

### 10.3.2.2.5 PDMA\_MCASP\_G0 Event Map

**Table 10-193. PDMA\_MCASP\_G0 Tx Channel Allocation**

Tx PDMA Channel	PDMA Input	Source Event	Channel Type	Trigger Type	Data FIFO Address
8000	MCASP_MAIN_0_TX_0	MCASP0_XMIT_DMA_EVENT_REQ_0	XY	Edge	02B0 8000h
8001	MCASP_MAIN_1_TX_0	MCASP1_XMIT_DMA_EVENT_REQ_0	XY	Edge	02B1 8000h
8002	MCASP_MAIN_2_TX_0	MCASP2_XMIT_DMA_EVENT_REQ_0	XY	Edge	02B2 8000h

**Table 10-194. PDMA\_MCASP\_G0 Rx Channel Allocation**

Rx PDMA Channel	PDMA Input	Source Event	Channel Type	Trigger Type	Data FIFO Address
0	MCASP_MAIN_0_RX_0	MCASP0_REC_DMA_EVENT_REQ_0	XY	Edge	02B0 8000h
1	MCASP_MAIN_1_RX_0	MCASP1_REC_DMA_EVENT_REQ_0	XY	Edge	02B1 8000h
2	MCASP_MAIN_2_RX_0	MCASP2_REC_DMA_EVENT_REQ_0	XY	Edge	02B2 8000h

### 10.3.2.2.6 PDMA\_MCASP\_G1 Event Map

**Table 10-195. PDMA\_MCASP\_G1 Tx Channel Allocation**

Tx PDMA Channel	PDMA Input	Source Event	Channel Type	Trigger Type	Data FIFO Address
8000	MCASP_MAIN_3_TX_0	MCASP3_XMIT_DMA_EVENT_REQ_0	XY	Edge	02B3 8000h
8001	MCASP_MAIN_4_TX_0	MCASP4_XMIT_DMA_EVENT_REQ_0	XY	Edge	02B4 8000h
8002	MCASP_MAIN_5_TX_0	MCASP5_XMIT_DMA_EVENT_REQ_0	XY	Edge	02B5 8000h
8003	MCASP_MAIN_6_TX_0	MCASP6_XMIT_DMA_EVENT_REQ_0	XY	Edge	02B6 8000h
8004	MCASP_MAIN_7_TX_0	MCASP7_XMIT_DMA_EVENT_REQ_0	XY	Edge	02B7 8000h
8005	MCASP_MAIN_8_TX_0	MCASP8_XMIT_DMA_EVENT_REQ_0	XY	Edge	02B8 8000h
8006	MCASP_MAIN_9_TX_0	MCASP9_XMIT_DMA_EVENT_REQ_0	XY	Edge	02B9 8000h
8007	MCASP_MAIN_10_TX_0	MCASP10_XMIT_DMA_EVENT_REQ_0	XY	Edge	02BA 8000h
8008	MCASP_MAIN_11_TX_0	MCASP11_XMIT_DMA_EVENT_REQ_0	XY	Edge	02BB 8000h

**Table 10-196. PDMA\_MCASP\_G1 Rx Channel Allocation**

Rx PDMA Channel	PDMA Input	Source Event	Channel Type	Trigger Type	Data FIFO Address
0	MCASP_MAIN_3_RX_0	MCASP3_REC_DMA_EVENT_REQ_0	XY	Edge	02B3 8000h
1	MCASP_MAIN_4_RX_0	MCASP4_REC_DMA_EVENT_REQ_0	XY	Edge	02B4 8000h
2	MCASP_MAIN_5_RX_0	MCASP5_REC_DMA_EVENT_REQ_0	XY	Edge	02B5 8000h
3	MCASP_MAIN_6_RX_0	MCASP6_REC_DMA_EVENT_REQ_0	XY	Edge	02B6 8000h
4	MCASP_MAIN_7_RX_0	MCASP7_REC_DMA_EVENT_REQ_0	XY	Edge	02B7 8000h
5	MCASP_MAIN_8_RX_0	MCASP8_REC_DMA_EVENT_REQ_0	XY	Edge	02B8 8000h
6	MCASP_MAIN_9_RX_0	MCASP9_REC_DMA_EVENT_REQ_0	XY	Edge	02B9 8000h
7	MCASP_MAIN_10_RX_0	MCASP10_REC_DMA_EVENT_REQ_0	XY	Edge	02BA 8000h
8	MCASP_MAIN_11_RX_0	MCASP11_REC_DMA_EVENT_REQ_0	XY	Edge	02BB 8000h

### 10.3.2.2.7 PDMA\_MISC\_G0 Event Map

**Table 10-197. PDMA\_MISC\_G0 Tx Channel Allocation**

Tx PDMA Channel	PDMA Input	Source Event	Channel Type	Trigger Type	Data FIFO Address
8000	SPI_MAIN_0_TX_0	MCSPi0_DMA_WRITE_EVENT_0	XY	Edge	0210 0138h
8001	SPI_MAIN_0_TX_1	MCSPi0_DMA_WRITE_EVENT_1	XY	Edge	0210 014Ch
8002	SPI_MAIN_0_TX_2	MCSPi0_DMA_WRITE_EVENT_2	XY	Edge	0210 0160h
8003	SPI_MAIN_0_TX_3	MCSPi0_DMA_WRITE_EVENT_3	XY	Edge	0210 0174h
8004	SPI_MAIN_1_TX_0	MCSPi1_DMA_WRITE_EVENT_0	XY	Edge	0211 0138h
8005	SPI_MAIN_1_TX_1	MCSPi1_DMA_WRITE_EVENT_1	XY	Edge	0211 014Ch
8006	SPI_MAIN_1_TX_2	MCSPi1_DMA_WRITE_EVENT_2	XY	Edge	0211 0160h
8007	SPI_MAIN_1_TX_3	MCSPi1_DMA_WRITE_EVENT_3	XY	Edge	0211 0174h
8008	MCANSS_MAIN_0_TX_0	MCAN0_MCANSS_TX_DMA_0	MCAN	Pulse	0270 8000h
8009	MCANSS_MAIN_0_TX_1	MCAN0_MCANSS_TX_DMA_1	MCAN	Pulse	0270 8048h
800A	MCANSS_MAIN_0_TX_2	MCAN0_MCANSS_TX_DMA_2	MCAN	Pulse	0270 8090h

**Table 10-198. PDMA\_MISC\_G0 Rx Channel Allocation**

Rx PDMA Channel	PDMA Input	Source Event	Channel Type	Trigger Type	Data FIFO Address
0	SPI_MAIN_0_RX_0	MCSPi0_DMA_READ_EVENT_0	XY	Edge	0210 013Ch
1	SPI_MAIN_0_RX_1	MCSPi0_DMA_READ_EVENT_1	XY	Edge	0210 0150h
2	SPI_MAIN_0_RX_2	MCSPi0_DMA_READ_EVENT_2	XY	Edge	0210 0164h



**Table 10-198. PDMA\_MISC\_G0 Rx Channel Allocation (continued)**

Rx PDMA Channel	PDMA Input	Source Event	Channel Type	Trigger Type	Data FIFO Address
3	SPI_MAIN_0_RX_3	MCSP10_DMA_READ_EVENT_3	XY	Edge	0210 0178h
4	SPI_MAIN_1_RX_0	MCSP11_DMA_READ_EVENT_0	XY	Edge	0211 013Ch
5	SPI_MAIN_1_RX_1	MCSP11_DMA_READ_EVENT_1	XY	Edge	0211 0150h
6	SPI_MAIN_1_RX_2	MCSP11_DMA_READ_EVENT_2	XY	Edge	0211 0164h
7	SPI_MAIN_1_RX_3	MCSP11_DMA_READ_EVENT_3	XY	Edge	0211 0178h
8	MCANSS_MAIN_0_FE_0	MCAN0_MCANSS_FE_0	MCAN	Pulse	0270 8900h
9	MCANSS_MAIN_0_FE_1	MCAN0_MCANSS_FE_1	MCAN	Pulse	0270 8990h
10	MCANSS_MAIN_0_FE_2	MCAN0_MCANSS_FE_2	MCAN	Pulse	0270 8A20h

#### 10.3.2.2.8 PDMA\_MISC\_G1 Event Map

**Table 10-199. PDMA\_MISC\_G1 Tx Channel Allocation**

Tx PDMA Channel	PDMA Input	Source Event	Channel Type	Trigger Type	Data FIFO Address
8000	SPI_MAIN_2_TX_0	MCSP12_DMA_WRITE_EVENT_0	XY	Edge	0212 0138h
8001	SPI_MAIN_2_TX_1	MCSP12_DMA_WRITE_EVENT_1	XY	Edge	0212 014Ch
8002	SPI_MAIN_2_TX_2	MCSP12_DMA_WRITE_EVENT_2	XY	Edge	0212 0160h
8003	SPI_MAIN_2_TX_3	MCSP12_DMA_WRITE_EVENT_3	XY	Edge	0212 0174h
8004	SPI_MAIN_3_TX_0	MCSP13_DMA_WRITE_EVENT_0	XY	Edge	0213 0138h
8005	SPI_MAIN_3_TX_1	MCSP13_DMA_WRITE_EVENT_1	XY	Edge	0213 014Ch
8006	SPI_MAIN_3_TX_2	MCSP13_DMA_WRITE_EVENT_2	XY	Edge	0213 0160h
8007	SPI_MAIN_3_TX_3	MCSP13_DMA_WRITE_EVENT_3	XY	Edge	0213 0174h
8008	MCANSS_MAIN_1_TX_0	MCAN1_MCANSS_TX_DMA_0	MCAN	Pulse	0271 8000h
8009	MCANSS_MAIN_1_TX_1	MCAN1_MCANSS_TX_DMA_1	MCAN	Pulse	0271 8048h
800A	MCANSS_MAIN_1_TX_2	MCAN1_MCANSS_TX_DMA_2	MCAN	Pulse	0271 8090h

**Table 10-200. PDMA\_MISC\_G1 Rx Channel Allocation**

Rx PDMA Channel	PDMA Input	Source Event	Channel Type	Trigger Type	Data FIFO Address
0	SPI_MAIN_2_RX_0	MCSP12_DMA_READ_EVENT_0	XY	Edge	0212 013Ch
1	SPI_MAIN_2_RX_1	MCSP12_DMA_READ_EVENT_1	XY	Edge	0212 0150h
2	SPI_MAIN_2_RX_2	MCSP12_DMA_READ_EVENT_2	XY	Edge	0212 0164h
3	SPI_MAIN_2_RX_3	MCSP12_DMA_READ_EVENT_3	XY	Edge	0212 0178h
4	SPI_MAIN_3_RX_0	MCSP13_DMA_READ_EVENT_0	XY	Edge	0213 013Ch
5	SPI_MAIN_3_RX_1	MCSP13_DMA_READ_EVENT_1	XY	Edge	0213 0150h
6	SPI_MAIN_3_RX_2	MCSP13_DMA_READ_EVENT_2	XY	Edge	0213 0164h
7	SPI_MAIN_3_RX_3	MCSP13_DMA_READ_EVENT_3	XY	Edge	0213 0178h
8	MCANSS_MAIN_1_FE_0	MCAN1_MCANSS_FE_0	MCAN	Pulse	0271 8900h
9	MCANSS_MAIN_1_FE_1	MCAN1_MCANSS_FE_1	MCAN	Pulse	0271 8990h
10	MCANSS_MAIN_1_FE_2	MCAN1_MCANSS_FE_2	MCAN	Pulse	0271 8A20h

#### 10.3.2.2.9 PDMA\_MISC\_G2 Event Map

**Table 10-201. PDMA\_MISC\_G2 Tx Channel Allocation**

Tx PDMA Channel	PDMA Input	Source Event	Channel Type	Trigger Type	Data FIFO Address
8000	SPI_MAIN_4_TX_0	MCSP14_DMA_WRITE_EVENT_0	XY	Edge	0214 0138h
8001	SPI_MAIN_4_TX_1	MCSP14_DMA_WRITE_EVENT_1	XY	Edge	0214 014Ch

**Table 10-201. PDMA\_MISC\_G2 Tx Channel Allocation (continued)**

Tx PDMA Channel	PDMA Input	Source Event	Channel Type	Trigger Type	Data FIFO Address
8002	SPI_MAIN_4_TX_2	MCSP14_DMA_WRITE_EVENT_2	XY	Edge	0214 0160h
8003	SPI_MAIN_4_TX_3	MCSP14_DMA_WRITE_EVENT_3	XY	Edge	0214 0174h
8004	SPI_MAIN_5_TX_0	MCSP15_DMA_WRITE_EVENT_0	XY	Edge	0215 0138h
8005	SPI_MAIN_5_TX_1	MCSP15_DMA_WRITE_EVENT_1	XY	Edge	0215 014Ch
8006	SPI_MAIN_5_TX_2	MCSP15_DMA_WRITE_EVENT_2	XY	Edge	0215 0160h
8007	SPI_MAIN_5_TX_3	MCSP15_DMA_WRITE_EVENT_3	XY	Edge	0215 0174h
8008	MCANSS_MAIN_2_TX_0	MCAN2_MCANSS_TX_DMA_0	MCAN	Pulse	0272 8000h
8009	MCANSS_MAIN_2_TX_1	MCAN2_MCANSS_TX_DMA_1	MCAN	Pulse	0272 8048h
800A	MCANSS_MAIN_2_TX_2	MCAN2_MCANSS_TX_DMA_2	MCAN	Pulse	0272 8090h

**Table 10-202. PDMA\_MISC\_G2 Rx Channel Allocation**

Rx PDMA Channel	PDMA Input	Source Event	Channel Type	Trigger Type	Data FIFO Address
0	SPI_MAIN_4_RX_0	MCSP14_DMA_READ_EVENT_0	XY	Edge	0214 013Ch
1	SPI_MAIN_4_RX_1	MCSP14_DMA_READ_EVENT_1	XY	Edge	0214 0150h
2	SPI_MAIN_4_RX_2	MCSP14_DMA_READ_EVENT_2	XY	Edge	0214 0164h
3	SPI_MAIN_4_RX_3	MCSP14_DMA_READ_EVENT_3	XY	Edge	0214 0178h
4	SPI_MAIN_5_RX_0	MCSP15_DMA_READ_EVENT_0	XY	Edge	0215 013Ch
5	SPI_MAIN_5_RX_1	MCSP15_DMA_READ_EVENT_1	XY	Edge	0215 0150h
6	SPI_MAIN_5_RX_2	MCSP15_DMA_READ_EVENT_2	XY	Edge	0215 0164h
7	SPI_MAIN_5_RX_3	MCSP15_DMA_READ_EVENT_3	XY	Edge	0215 0178h
8	MCANSS_MAIN_2_FE_0	MCAN2_MCANSS_FE_0	MCAN	Pulse	0272 8900h
9	MCANSS_MAIN_2_FE_1	MCAN2_MCANSS_FE_1	MCAN	Pulse	0272 8990h
10	MCANSS_MAIN_2_FE_2	MCAN2_MCANSS_FE_2	MCAN	Pulse	0272 8A20h

### 10.3.2.2.10 PDMA\_MISC\_G3 Event Map

**Table 10-203. PDMA\_MISC\_G3 Tx Channel Allocation**

Tx PDMA Channel	PDMA Input	Source Event	Channel Type	Trigger Type	Data FIFO Address
8000	SPI_MAIN_6_TX_0	MCSP16_DMA_WRITE_EVENT_0	XY	Edge	0216 0138h
8001	SPI_MAIN_6_TX_1	MCSP16_DMA_WRITE_EVENT_1	XY	Edge	0216 014Ch
8002	SPI_MAIN_6_TX_2	MCSP16_DMA_WRITE_EVENT_2	XY	Edge	0216 0160h
8003	SPI_MAIN_6_TX_3	MCSP16_DMA_WRITE_EVENT_3	XY	Edge	0216 0174h
8004	SPI_MAIN_7_TX_0	MCSP17_DMA_WRITE_EVENT_0	XY	Edge	0217 0138h
8005	SPI_MAIN_7_TX_1	MCSP17_DMA_WRITE_EVENT_1	XY	Edge	0217 014Ch
8006	SPI_MAIN_7_TX_2	MCSP17_DMA_WRITE_EVENT_2	XY	Edge	0217 0160h
8007	SPI_MAIN_7_TX_3	MCSP17_DMA_WRITE_EVENT_3	XY	Edge	0217 0174h
8008	MCANSS_MAIN_3_TX_0	MCAN3_MCANSS_TX_DMA_0	MCAN	Pulse	0273 8000h
8009	MCANSS_MAIN_3_TX_1	MCAN3_MCANSS_TX_DMA_1	MCAN	Pulse	0273 8048h
800A	MCANSS_MAIN_3_TX_2	MCAN3_MCANSS_TX_DMA_2	MCAN	Pulse	0273 8090h

**Table 10-204. PDMA\_MISC\_G3 Rx Channel Allocation**

Rx PDMA Channel	PDMA Input	Source Event	Channel Type	Trigger Type	Data FIFO Address
0	SPI_MAIN_6_RX_0	MCSP16_DMA_READ_EVENT_0	XY	Edge	0216 013Ch
1	SPI_MAIN_6_RX_1	MCSP16_DMA_READ_EVENT_1	XY	Edge	0216 0150h

**Table 10-204. PDMA\_MISC\_G3 Rx Channel Allocation (continued)**

Rx PDMA Channel	PDMA Input	Source Event	Channel Type	Trigger Type	Data FIFO Address
2	SPI_MAIN_6_RX_2	MCSPi6_DMA_READ_EVENT_2	XY	Edge	0216 0164h
3	SPI_MAIN_6_RX_3	MCSPi6_DMA_READ_EVENT_3	XY	Edge	0216 0178h
4	SPI_MAIN_7_RX_0	MCSPi7_DMA_READ_EVENT_0	XY	Edge	0217 013Ch
5	SPI_MAIN_7_RX_1	MCSPi7_DMA_READ_EVENT_1	XY	Edge	0217 0150h
6	SPI_MAIN_7_RX_2	MCSPi7_DMA_READ_EVENT_2	XY	Edge	0217 0164h
7	SPI_MAIN_7_RX_3	MCSPi7_DMA_READ_EVENT_3	XY	Edge	0217 0178h
8	MCANSS_MAIN_3_FE_0	MCAN3_MCANSS_FE_0	MCAN	Pulse	0273 8900h
9	MCANSS_MAIN_3_FE_1	MCAN3_MCANSS_FE_1	MCAN	Pulse	0273 8990h
10	MCANSS_MAIN_3_FE_2	MCAN3_MCANSS_FE_2	MCAN	Pulse	0273 8A20h

#### 10.3.2.2.11 PDMA\_USART\_G0 Event Map

**Table 10-205. PDMA\_USART\_G0 Tx Channel Allocation**

Tx PDMA Channel	PDMA Input	Source Event	Channel Type	Trigger Type	Data FIFO Address
8000	UART_MAIN_0_TX_0	UART0_USART_DMA_0	XY	Edge	0280 0000h
8001	UART_MAIN_1_TX_0	UART1_USART_DMA_0	XY	Edge	0281 0000h

**Table 10-206. PDMA\_USART\_G0 Rx Channel Allocation**

Rx PDMA Channel	PDMA Input	Source Event	Channel Type	Trigger Type	Data FIFO Address
0	UART_MAIN_0_RX_0	UART0_USART_DMA_1	XY	Edge	0280 0000h
1	UART_MAIN_1_RX_0	UART1_USART_DMA_1	XY	Edge	0281 0000h

#### 10.3.2.2.12 PDMA\_USART\_G1 Event Map

**Table 10-207. PDMA\_USART\_G1 Tx Channel Allocation**

Tx PDMA Channel	PDMA Input	Source Event	Channel Type	Trigger Type	Data FIFO Address
8000	UART_MAIN_2_TX_0	UART2_USART_DMA_0	XY	Edge	0282 0000h
8001	UART_MAIN_3_TX_0	UART3_USART_DMA_0	XY	Edge	0283 0000h

**Table 10-208. PDMA\_USART\_G1 Rx Channel Allocation**

Rx PDMA Channel	PDMA Input	Source Event	Channel Type	Trigger Type	Data FIFO Address
0	UART_MAIN_2_RX_0	UART2_USART_DMA_1	XY	Edge	0282 0000h
1	UART_MAIN_3_RX_0	UART3_USART_DMA_1	XY	Edge	0283 0000h

#### 10.3.2.2.13 PDMA\_USART\_G2 Event Map

**Table 10-209. PDMA\_USART\_G2 Tx Channel Allocation**

Tx PDMA Channel	PDMA Input	Source Event	Channel Type	Trigger Type	Data FIFO Address
8000	UART_MAIN_4_TX_0	UART4_USART_DMA_0	XY	Edge	0284 0000h
8001	UART_MAIN_5_TX_0	UART5_USART_DMA_0	XY	Edge	0285 0000h
8002	UART_MAIN_6_TX_0	UART6_USART_DMA_0	XY	Edge	0286 0000h
8003	UART_MAIN_7_TX_0	UART7_USART_DMA_0	XY	Edge	0287 0000h
8004	UART_MAIN_8_TX_0	UART8_USART_DMA_0	XY	Edge	0288 0000h
8005	UART_MAIN_9_TX_0	UART9_USART_DMA_0	XY	Edge	0289 0000h

**Table 10-210. PDMA\_USART\_G2 Rx Channel Allocation**

Rx PDMA Channel	PDMA Input	Source Event	Channel Type	Trigger Type	Data FIFO Address
0	UART_MAIN_4_RX_0	UART4_USART_DMA_1	XY	Edge	0284 0000h
1	UART_MAIN_5_RX_0	UART5_USART_DMA_1	XY	Edge	0285 0000h
2	UART_MAIN_6_RX_0	UART6_USART_DMA_1	XY	Edge	0286 0000h
3	UART_MAIN_7_RX_0	UART7_USART_DMA_1	XY	Edge	0287 0000h
4	UART_MAIN_8_RX_0	UART8_USART_DMA_1	XY	Edge	0288 0000h
5	UART_MAIN_9_RX_0	UART9_USART_DMA_1	XY	Edge	0289 0000h

## 10.4 Data Routing Unit (DRU)

This section describes the SoC level Data Routing Unit (DRU) for the device.

### Note

There are also DRUs in VPAC0 and DMPAC0. Their functionality is almost same as described in this section. For the differences between all DRUs, see [Section 10.4.3.6](#).

### 10.4.1 DRU Overview

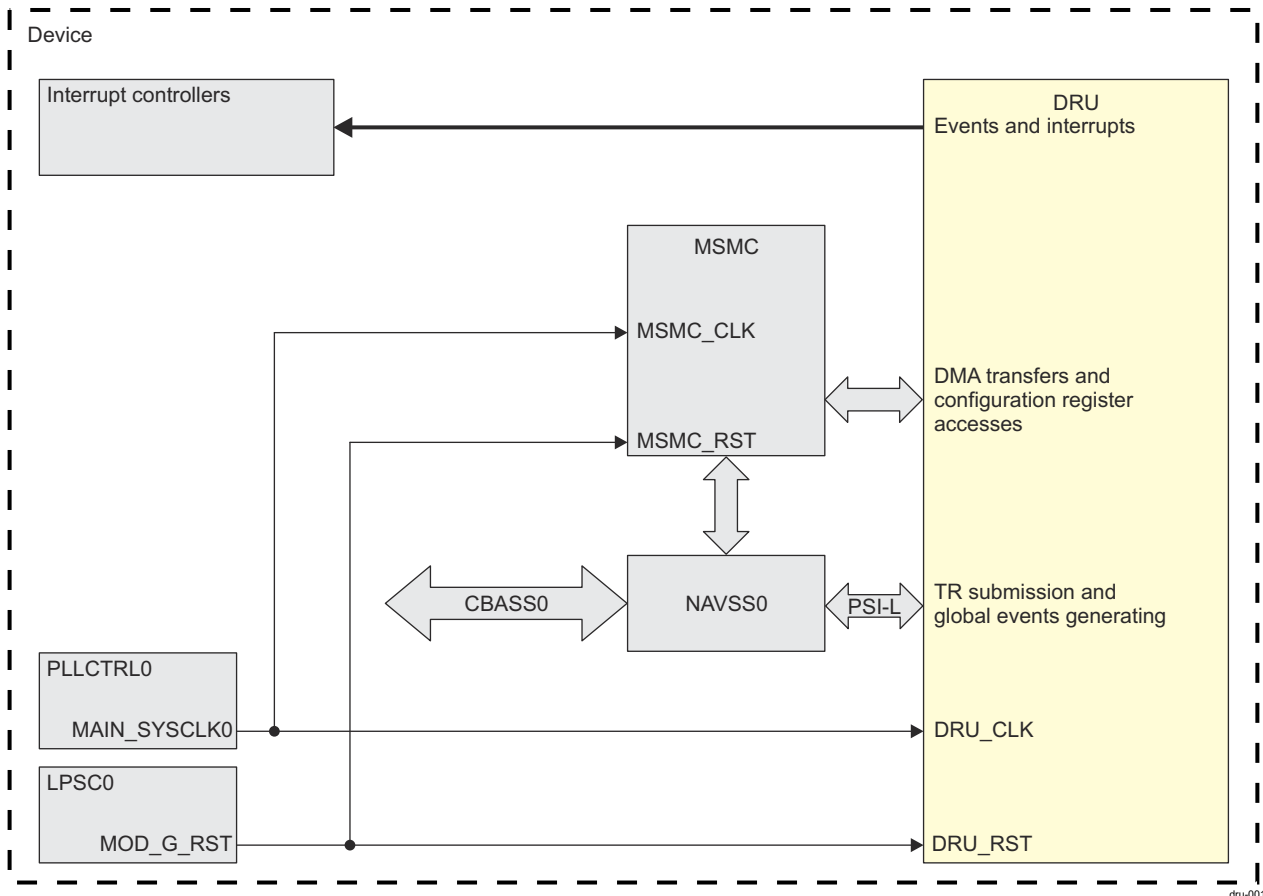
The data routing unit (DRU) is a high bandwidth, flexible routing engine with programmable DMA transfer requests. It enables user to perform high speed data transfers between memory mapped slave endpoints, processor caches and shared caches. DRU behaves like a DMA transfer controller, moving data at CPU frequency.

[Table 10-211](#) shows DRU modules allocation within device domains.

**Table 10-211. DRU Modules Allocation within Device Domains**

Module Instance	Domain		
	WKUP	MCU	MAIN
DRU	-	-	✓

[Figure 10-39](#) shows an overview of the DRU and its surrounding modules.



**Figure 10-39. DRU Overview**

The DRU supports the following features:

- Programmable configuration registers for direct transfer request submission
  - A dedicated atomic register set that requires entire TR sent in a single burst
  - Three nonatomic register sets that allow TR to be written in separate burst and a write to the SUBMIT\_WORD0 register triggers the TR
- Read and write command queues
  - Five different queues with a split arbitration between fixed priority and round robin arbitration
  - Each channel can be configured to go to one of these queues
- Programmable priority for each queue
  - Allows setting priority for commands relative to other masters in the system
- One dedicated read port and one dedicated write port to generate independent read and write commands
  - Generates one read command every cycle
  - Generates one write data phase every cycle
  - Moves the data at CPU frequency
  - Optimizes transfers up to 128 byte boundaries
- Support for triggers to gate levels of the DMA loop
  - Supports one dedicated local trigger
  - Supports two dedicated external global triggers from the PSI-L
  - All triggers can be overwritten and controlled by software instead of hardware based event
- Support for region based and channelized firewall
  - Region based firewall intended to protect DRU configuration registers within a programmatically specified range
  - Channelized firewall that protects:
    - The real time configuration registers for each channel
    - The TR submission registers for each channel
    - The data transfers for each channel
- Data formatting networks
  - Transpose data between source and destination memory to memory transfers
  - Circular addressing support for one side of the transfer
- Independent 48-bit address fields for source and destinations
- Up to four dimensional data transfers
  - Has independent source and destination index for up to four dimensional transfers
- Error Handling
  - Debug support through memory mapped registers
  - Data transfer error reports
- Error detection and Correction
  - Full SEC/DED support for the incoming data
  - Full SEC/DED protection each entry in the TR queues
  - Single bit parity support for each 32-bit word in the data buffer
  - Parity protection for the address
- PSI-L Interface
  - Receive requests for pre-warm of L2 or L3 Cache
  - Receive TR requests from the UDMA
  - Send response TRs
  - Receive data from a remote UTC
  - Send data to a remote UTC
  - Maintains 2D formatting across PSI-L if requested in TR

### 10.4.2 DRU Integration

This section describes the SoC level DRU integration in the device, including information about clocks, resets, and hardware requests.

---

#### Note

There are also DRUs in VPAC0 and DMPAC0. Their functionality is almost same as the SoC level DRU. For the differences between all DRUs, see [Section 10.4.3.6](#). For integration details about the VPAC0 and DMPAC0 DRUs, see [Section 6.9 Vision Pre-processing Accelerator \(VPAC\)](#) and [Section 6.10 Depth and Motion Perception Accelerator \(DMPAC\)](#).

---

#### 10.4.2.1 DRU Integration in MAIN Domain

A single DRU module is integrated in the device MAIN domain. [Figure 10-40](#) shows the integration of DRU.

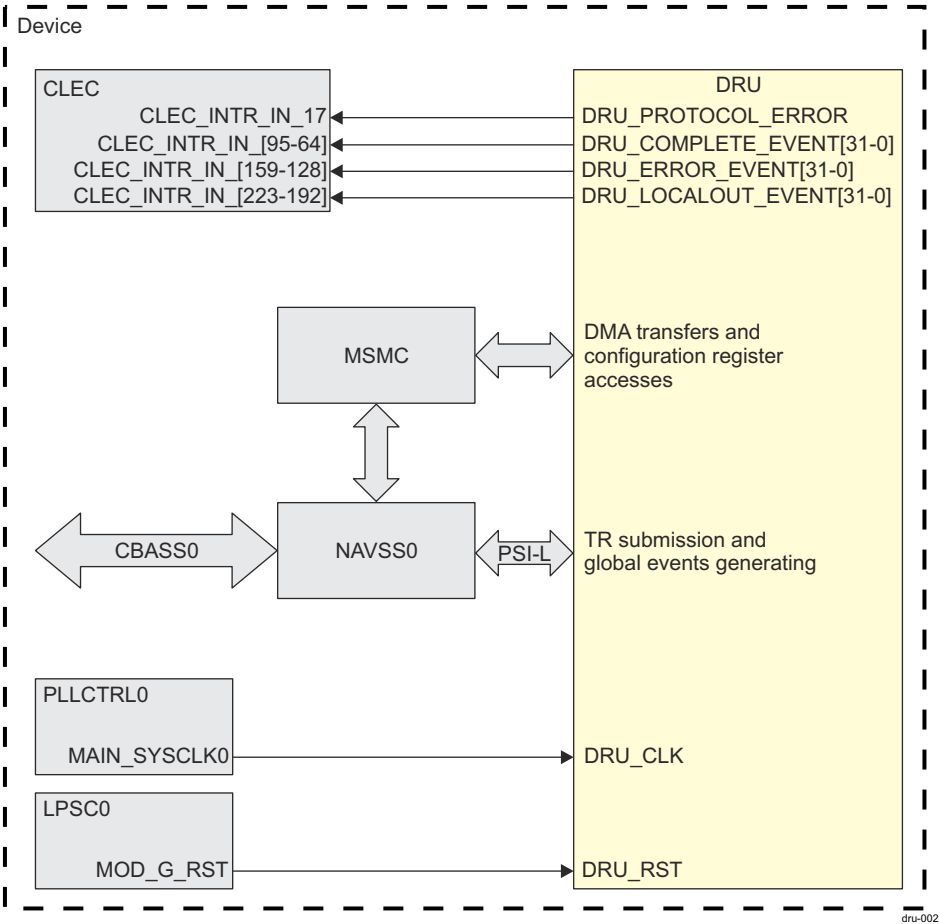


Figure 10-40. DRU Integration

Table 10-212 through Table 10-214 summarize the integration of DRU in device MAIN domain.

Table 10-212. DRU Integration Attributes

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect



**Table 10-212. DRU Integration Attributes (continued)**

DRU	PSC0	PD0	LPSC0	CBASS0 (Accessed through NAVSS0 and then through MSMC) PSI-L
-----	------	-----	-------	---

**Table 10-213. DRU Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
DRU	DRU_CLK	MAIN_SYSCLK0	PLLCTRL0	DRU clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
DRU	DRU_RST	MOD_G_RST	LPSC0	DRU reset

**Table 10-214. DRU Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DRU	DRU_PROTOCOL_ERROR	CLEC_INTR_IN_17	CLEC	DRU protocol violation event	Level
	DRU_COMPLETE_EVENT[31-0]	CLEC_INTR_IN_[95-64]	CLEC	DRU completion events	Level
	DRU_ERROR_EVENT[31-0]	CLEC_INTR_IN_[159-128]	CLEC	DRU error completion events	Level
	DRU_LOCALOUT_EVENT[31-0]	CLEC_INTR_IN_[223-192]	CLEC	DRU local output events	Level
DMA Events					
Module Instance	Module DMA Event	Destination DMA Event Input	Destination	Description	Type
DRU	-	-	-	-	-

### Note

For more information on the DRU events, see [Section 10.4.3.2](#).

For more information on the interconnects, see [Chapter 3, System Interconnect](#).

For more information on the power, reset and clock management, see the corresponding sections within [Chapter 5, Device Configuration](#).

For more information on the device interrupt controllers, see [Section 9.2, Interrupt Controllers](#).

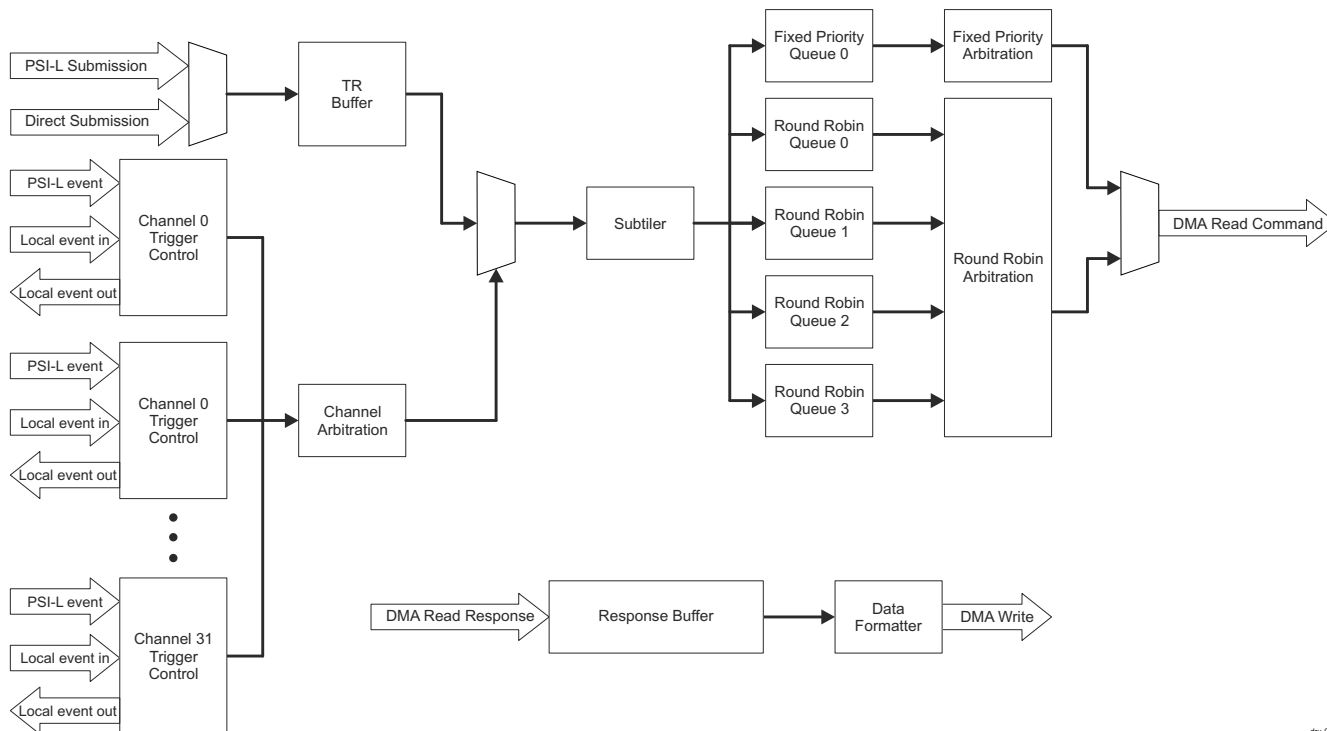
### 10.4.3 DRU Functional Description

#### Note

This section describes the SoC level Data Routing Unit (DRU) for the device. There are also DRUs in VPAC0 and DMPAC0. Their functionality is almost same as described in this section and its subsections. For the differences between all DRUs, see [Section 10.4.3.6](#).

#### 10.4.3.1 DRU Basic Functionality

Figure 10-40 shows the DRU functional diagram.



**Figure 10-41. DRU Functional Diagram**

The DRU receives commands to transfer data from one location to another called Transfer Requests (TRs) or to send messages to CPUs or MMUs to pre-warm logic called Cache Requests (CRs). The DRU can receive these commands through the following mechanisms:

- A direct write to the DRU submission registers
- TR submission over PSI-L from external UDMA-C

The previously described mechanisms of receiving TRs and CRs are also referred to as TR submission (see, [Section 10.4.3.1.3](#)). In other words, the DRU memory-to-memory transfers work by receiving TR or CR through TR submission. Then the corresponding request (TR or CR) is placed in a queue to be processed.

Once the TR has been received the Channel Trigger Control block detects if the triggers specified in the TR have been received. If the trigger is not specified (value of 0x0 in the TRIGGER0 or TRIGGER1 fields of the TR FLAGS field), the treatment is as it is always received. Once received the channel is considered active. All active channels are involved in arbitration process for using the subtiler. The channel arbitration is based on the channel queue. There are queues using a fixed priority arbitration - the lowest channel has the highest priority. There are also round robin queues that use the same algorithm as in the fixed priority queue arbitration (lowest channel - highest priority) but on a round robin basis.

The subtiler takes the winning channel (that is, channel with highest priority) and generates the next 1D or 2D transfer (called also sub-TR) based on the trigger size, transfer type and event size specified in the TR. Then the subtiler places that sub-TR into the queue specified for the channel. Once the sub-TR is generated an event control is notified and any active triggers are cleared if the sub-TR crosses the boundary of one of the triggers. For example, if a TR has a trigger size for a 3D block and the ICNT2 value is 16, then the active triggers are cleared after 16 sub-TRs are sent to the queue. ICNT2 can be specified through one of the following:

- DRU\_ATOMIC\_SUBMIT\_CURR\_TR\_WORD4\_5\_j[47-32] ICNT2 field
- DRU\_SUBMIT\_WORD4\_5\_j\_k[47-32] ICNT2 field
- ICNT2 field of the UDMA-C TR received over PSI-L

Upon reaching the top of the queue the TR is processed. Based on the information in the TR a read of the specified location and size is performed followed by writing the data to the specified destination. Upon sending the last write of the entire TR loop a completion event is generated. The completion event output also places a TR response in the PSI-L response FIFO in case of UDMA-C TR. For information about the TR response format, see *Transfer Response Record* in *DMA Architecture*.

---

#### Note

The channel is an independent data flow that can be given requests for data movement. Each channel operates independently of the others but they all share common resources.

The queue is a shared unit between several channels. It holds the order of the channel data movement requests until the shared data engine can perform the actual data move. A channel is assigned a queue where its data movement requests are placed.

---



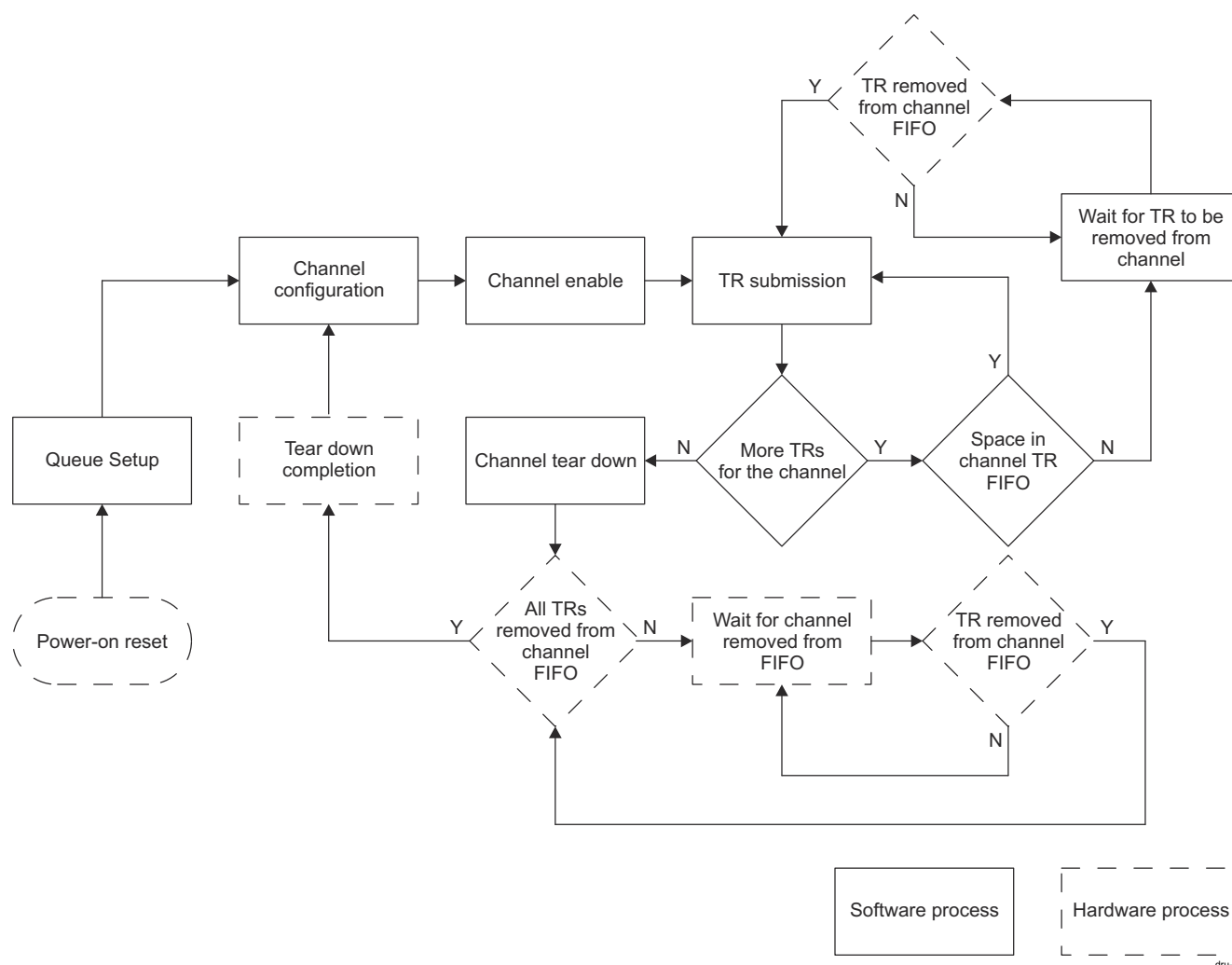
---

#### Note

The CHRT\_SWTRIG register can only be accessed directly through the DRU configuration interface and is used if software wants to control the triggers for the TR.

---

Figure 10-42 shows the interaction between software and hardware to setup DRU, submit TRs and then release the channel to be used by a different resource. Section 10.4.3.1.1 through Section 10.4.3.1.5 explain in more detail what occurs in each process or decision.


**Figure 10-42. DRU Software Managed Flow Diagram**

#### 10.4.3.1.1 Queues

Every TR that is active is placed in one of the DRU queues. Each queue operates independently of the others. In a given queue the TRs complete in the order they are placed by the subtiler. Only one read and one write queue may exist at a time. The queues use same resources and need arbitration which is performed based on fixed priority or round robin scheme. In both cases the highest priority queue is the one with the lowest number. The queue priority is specified through the DRU\_CFG\_y[2-0] PRI field.

Whenever arbitration occurs if there is a TR in a fixed priority queue that TR always goes and the round robin queues are held off. If the fixed priority queues are empty then the round robin queues are arbitrated on a round robin basis, that is, each queue sends its specified number of transactions and then arbitration starts again. On the write side an arbitration ends if the read response has not been received for the next command when the current one completes.

The arbitration happens the last phase of a transfer chunk. A transfer chunk is defined as sending the number of read transactions specified by the DRU\_CFG\_y[23-16] CONSECUTIVE\_TRANS field. After CONSECUTIVE\_TRANS number of commands elapse the queue releases the bus and waits for the number of commands specified by the DRU\_CFG\_y[31-24] REARB\_WAIT field before trying to request the bus and involve in arbitration again. Thus software can tune the data flow from each queue to prevent starvation.

If a queue is running and no other queues have a TR then the running one starts its next chunk without stopping. Each time arbitration is performed for round robin queues the queue with the next priority becomes the highest priority and the current one the lowest priority queue. If a fixed priority queue is active then the current round robin queue remains the highest priority one until the CONSECUTIVE\_TRANS count expires or queue is empty.

The DMA bus QoS and order ID attributes for each queue can also be configured through the DRU\_CFG\_y[10-8] QOS and DRU\_CFG\_y[7-4] ORDERID fields.

---

#### Note

Queue configuration must be done before setting up any channel as queue changes while channels are active can cause unknown behavior.

---

#### 10.4.3.1.2 Channel Configuration

The DRU operates on having multiple channels that can run in parallel on the shared hardware but are independently triggered by events as needed. Each channel must be configured before it is used. The channel configuration has the following parts:

- Non-realtime configuration intended to be controlled by a single master in the system for all software processes and processors
- Real time configuration controlled by the software process that owns the channel

##### 10.4.3.1.2.1 Non-realtime Channel Configuration

The non-realtime configuration consists of the following:

- Assigning the channel to a submission mechanism via the DRU\_CFG\_j[19] CHAN\_TYPE\_OWNER bit
- Assigning the channel via the DRU\_CHST\_SCHED\_j[2-0] QUEUE field to a hardware queue to be used for channel transactions
- Assigning the channel via the DRU\_CHOES0\_j[15-0] EVT\_NUM field to an event to be used for channel notifications

The DRU\_CFG\_j[31] PAUSE\_ON\_ERR bit can be used to control the channel behavior in case of error or exception.

The DRU\_CFG\_j registers must be written to before the channel is enabled via the DRU\_CHRT\_CTL\_j[31] ENABLE bit as once a channel is enabled any write to the DRU\_CFG\_j, DRU\_CHST\_SCHED\_j and DRU\_CHOES0\_j registers will result in write failure and an address error. If a channel does not exist an address error is generated upon write too.

##### 10.4.3.1.2.2 Realtime Channel Configuration

Once the non-realtime channel configuration has been done the DRU\_CHRT\_CTL\_j[31] ENABLE bit should be written. It must be written before any TRs can be written. The mechanism to write this bit is based on the ownership of the channel as specified via the DRU\_CFG\_j[19] CHAN\_TYPE\_OWNER bit. If the owner is a CPU, than this bit must be written directly through the DRU configuration interface. If the owner is UDMA, than the PSI-L interface must be used to write the ENABLE bit through the PSI-L register configuration write process as described in *PSI-L*.

---

#### Note

Once a channel has been enabled the channel configuration registers cannot be modified. The channel can only be disabled through writing 0x1 to the DRU\_CHRT\_CTL\_j[30] TEARDOWN bit.

---

#### 10.4.3.1.3 TR Submission

The DRU supports the following TR methods:

- A direct write to the DRU submission registers
- TR submission over PSI-L from external UDMA-C

The TR submission checks the following:

- If a legal TR has been submitted
- If the correct submission method has been used
- If there is space in the channel TR FIFO for the new TR

#### 10.4.3.1.3.1 Direct TR Submission

The direct TR submission can be atomic or nonatomic. There is one atomic register set that allows for a TR to be submitted with a single burst write. There are also three nonatomic register sets for nonatomic direct TR submission. Each of these three sets allows direct TR submission without requiring submission in a single burst.

The direct atomic TR submission is handled through DRU register writes to the DRU\_ATOMIC\_SUBMIT\_CURR\_TR\_WORD0\_1\_j to DRU\_ATOMIC\_SUBMIT\_CURR\_TR\_WORD14\_15\_j registers which require write in a single 64-byte burst. Any write other than this size returns an address error. All burst values are checked if they are legal values and space in the channel FIFO is checked too. Then the TR is pushed into the FIFO and the write status is returned with success. If any of the legal TR format checks or FIFO space check fails or if the channel is not owned and enabled the result is an address error returned.

The direct nonatomic TR submission is handled through DRU register writes to the DRU\_SUBMIT\_WORD0\_1\_j\_k to DRU\_SUBMIT\_WORD14\_15\_j\_k registers which are intended to be used for cores not being able to perform a large burst write in a single cycle. There is a single set for each core regardless of the channel number. A write to the DRU\_SUBMIT\_WORD0\_1\_j\_k register triggers the TR.

#### Note

The following should be taken into account:

- If a channel is full any write to the DRU\_SUBMIT\_WORD0\_1\_j\_k register results in an address error.
- The DRU does not check anything to ensure that the same core writes to the same nonatomic register set. Software should perform the check procedure.

#### 10.4.3.1.3.2 PSI-L TR Submission

The DRU PSI-L TR submission is used for TR submissions from an external UDMA-C. The DRU has a single submission thread for each channel. The channel must be configured as UDMA-C type channel before PSI-L pairing.

The PSI-L pairing, TR submission and response follow the mechanisms as described in *PSI-L*. For every TR a TR response is sent. If the TR fails a legal check or channel ownership or FIFO fullness check, an immediate TR response is sent back with the cause of the submission error provided in the response.

#### 10.4.3.1.4 TR Removal from Channel

After all requested write responses are received from a TR the DRU sends completion event for the channel that completed the TR. Software can use this event to determine when a TR can be added to a channel FIFO. Each channel has a small FIFO allowing multiple TRs to be preloaded into a channel thus maximizing the channel throughput. The completion event signifies that an entry has become available in the FIFO.

#### 10.4.3.1.5 Channel Tear Down

A channel must be torn down through setting the DRU\_CHRT\_CTL\_j[30] TEARDOWN bit to 0x1. This allows for channel reconfiguration. Setting the TEARDOWN bit to 0x1 forces the channel to stop sending any new requests not already in the pipeline and when all outstanding requests have been received all other bits in the DRU\_CHRT\_CTL\_j register are cleared. If the DRU\_CFG\_j[31] PAUSE\_ON\_ERR bit is set and an error occurs the tear down mechanism can be used to restart the channel. The TEARDOWN bit being set to 0x1 also makes the TR STATIC field in any future reads from the TR memory to be set to 0x0 no matter the value when the TR was submitted. This is the mechanism used to stop a static TR.

#### 10.4.3.1.5.1 Tear Down Completion

The tear down completion process is when software has set the DRU\_CHRT\_CTL\_j[30] TEARDOWN bit to 0x1 so no new TRs can be added to the channel but when previous TRs that had been added to the channel are still running. Once the TR FIFO for the channel is empty hardware clears the DRU\_CHRT\_CTL\_j[30] TEARDOWN and DRU\_CHRT\_CTL\_j[31] ENABLE bits. The channel is now back at the channel configuration process as described in [Section 10.4.3.1.2.1](#) and can be reused as needed by the system.

#### 10.4.3.2 DRU Output Events

The DRU generates the following events:

- Local output events
- PSI-L global events
- Protocol violation event
- Completion events
- Error completion events

The EVENT\_SIZE field of each TR specifies how often a TR generates an output event. That event is provided on the local output event interface as shown in [Figure 10-41](#) and as global event on the PSI-L interface. There is one event per channel. The global event number used by a PSI-L event transport lane can be configured for each channel through the DRU\_CHOES0\_j[15-0] EVT\_NUM field.

As shown in [Table 10-215](#), if the EVENT\_SIZE value is in range from 1 to 3, then the output event is generated when the write command for the given loop level (ICNT1, ICNT2 or ICNT3 decremented) has been sent. If EVENT\_SIZE = 0, then the output event is generated when all responses have been received.

**Table 10-215. EVENT\_SIZE Encoding**

EVENT_SIZE Value	Event Encoding	Description
0	TR complete	The TR has completed and all responses have been received
1	ICNT1 loop is decremented	The last write command of a row has been sent
2	ICNT2 loop is decremented	The last write command of an array has been sent
3	ICNT3 loop is decremented	The last write command of an array of an array has been sent

For more information about PSI-L interface and its event transport lane, see *PSI-L*.

For more information about the TR EVENT\_SIZE field, see *Transfer Request Record* of *DMA Architecture*.

The protocol violation event is not channel associated.

The completion events are defined for each channel. The completion event fires whenever a channel completes a TR successfully. It is only generated when all responses are received from the write portion of the block copy. A single write completion event can occur in a given cycle since it is based on the write responses being received to generate the response.

The error completion events are defined for each channel. The error completion event fires whenever a channel completes a TR with an error and it has flushed out all data related to that TR in the response buffer. Upon determining that a TR is not able to complete successfully, the DRU enters a flush mode to remove all read data received after the error is determined as well as make sure all write responses are received. Once all of this criteria has been matched the write engine fires a completion event. If multiple error completions occur at the same time the DRU will still send them out one at a time using an arbitration scheme used for the original arbitration for the queue, either fixed priority or round robin based on the queue that has the error.

#### 10.4.3.3 DRU Address Fetch Algorithm, TR and CR Formats

The address fetch algorithm, TR and CR formats are defined in *Transfer Request Record* in *DMA Architecture*. It documents all features that can be supported by a Universal Transfer Controller (UTC) like the DRU but it may not support all of them. The DRU\_CAPABILITIES register must be read to check what features of the TR are supported by the DRU.



The TR format allows to perform some kinds of data reformatting or to alter from the standard addressing formats. The DRU supports the transpose data reformatting function and the circular buffering alternate addressing algorithm.

#### 10.4.3.3.1 Transpose

Linear streams work for large classes of algorithms, but not all. In some cases it is desired to transpose rows to columns and columns to rows in the data structure. To help in these cases the DRU supports a transpose function.

The transpose type works on element sizes of 32, 64 or 128 bits by sending sub-TRs in the size of 32 elements by 32 lines for 32-bit, 16 elements by 16 lines for 64-bit and 8 elements by 8 lines for 128-bit element. The sub-TR that is being sent always results in the maximum transfer size. For the write requests the data in the response buffer always looks as a buffer aligned to the zeroth offset in the address.

#### 10.4.3.3.2 Circular Buffering

Circular addressing modifies how the DRU performs address arithmetic so that addresses remain within a power-of-two sized window. Circular addressing works by holding upper address bits constant during an address update, while allowing the lower address bits to vary. This contrasts with the default linear addressing which allows all of the address bits to vary.

The TR provides address mode selection for each level of loop nest. Each level can select between linear addressing or circular addressing with one of two circular block sizes.

If circular addressing mode is selected in the AMODE field of the TR, the definition of the upper 16 bits of the FMTFLAGS field of the TR select the type of addressing for each entry in the loop and define two unique masks that can be used for the circular buffering. Additionally, the DIR field in the TR determines if the circular addressing flags apply for the source or the destination of the block move. The other direction always follows the standard linear addressing that is used if AMODE is not set.

For more information about circular buffering and the associated TR fields, see *Circular Address Mode Specific Flags* of *DMA Architecture*.

#### 10.4.3.4 DRU Firewalls

Unlike most of the device modules whose associated firewalls reside on system interconnect (CBASS) level the DRU has its own embedded firewalls. They are the following:

- Region based firewall intended to protect DRU configuration registers within a programmatically specified range.
- Channelized firewall that protects:
  - The real time configuration registers for each channel:
    - DRU\_CHRT\_CTL\_j
    - DRU\_CHRT\_SWTRIG\_j
  - The TR submission registers for each channel:
    - DRU\_ATOMIC\_SUBMIT\_CURR\_TR\_WORD0\_1\_j to DRU\_ATOMIC\_SUBMIT\_CURR\_TR\_WORD14\_15\_j
    - DRU\_SUBMIT\_WORD0\_1\_j\_k to DRU\_SUBMIT\_WORD14\_15\_j\_k
  - The data transfers for each channel

The region based firewall should be configured to cover the range from 0x6D00 4000 to 0x6D0C FFFF. This allows for a single master to control the memory attribute channels and channel ownership. If the region based firewall is configured to overlap the channelized firewall then both firewall checks have to pass for the register update to occur. If the firewall check fails the DRU returns protection error status to the requestor.

#### Note

Protection error takes precedence over address error. If the address falls into the firewall protected range then a protection error is returned.



The DRU region based and channelized firewalls are same as the other device firewalls. For more information about their functionality, see [Section 3.3.4 Interconnect Firewalls](#) in *System Interconnect*. The DRU firewall associated registers are described in the following sections:

- *DRU\_FW Registers*
- *DRU\_FW\_GLB Registers*
- *DRU\_MMR\_FW Registers*
- *DRU\_MMR\_FW\_GLB Registers*

#### 10.4.3.5 DRU Errors

In all error cases the DRU reports the error to the system, ensures that all outstanding commands for the TR have completed and proceeds with the next TR in the queue as if the previous one has finished normally. In most error cases the read transaction completes as normal but the responses are flushed. This ensures that both the read and the write start fresh on the next TR. The error details are logged in the DRU\_CHRT\_STATUS\_DET\_j and DRU\_CHRT\_STATUS\_CNT\_j registers. The DRU\_CAUSE\_y registers can be read to check if an error for the corresponding channel has occurred.

#### 10.4.3.6 DRU Configurations

[Table 10-216](#) shows the differences between the SoC level DRU and the rest DRUs in the device. The rest of the features and functionality is same as described in [Section 10.4.1](#) and [Section 10.4.3](#).

**Table 10-216. Differences between DRU Configurations**

Feature	SoC Level DRU	DMPAC.UTC_DRU	VPAC.UTC0_RT_DRU, VPAC.UTC1_NRT_DRU
Cache warm support	Y	Y	N
Atomic direct TR submission	Y	N	N
Nonatomic TR submission register sets	3	0	0
Data reformat functions	Transpose	None	None
Minimum transpose size	32 bits	N/A	N/A
Embedded firewall	Y	Y	N
Event generation	Encoded	Bit vector	Bit vector



This chapter describes the time sync modules in the device.

11.1 Time Sync Module (CPTS).....	1321
11.2 Timer Manager.....	1332
11.3 Time Sync and Compare Events.....	1340

## 11.1 Time Sync Module (CPTS)

This chapter describes the Time Synchronization (CPTS) module.

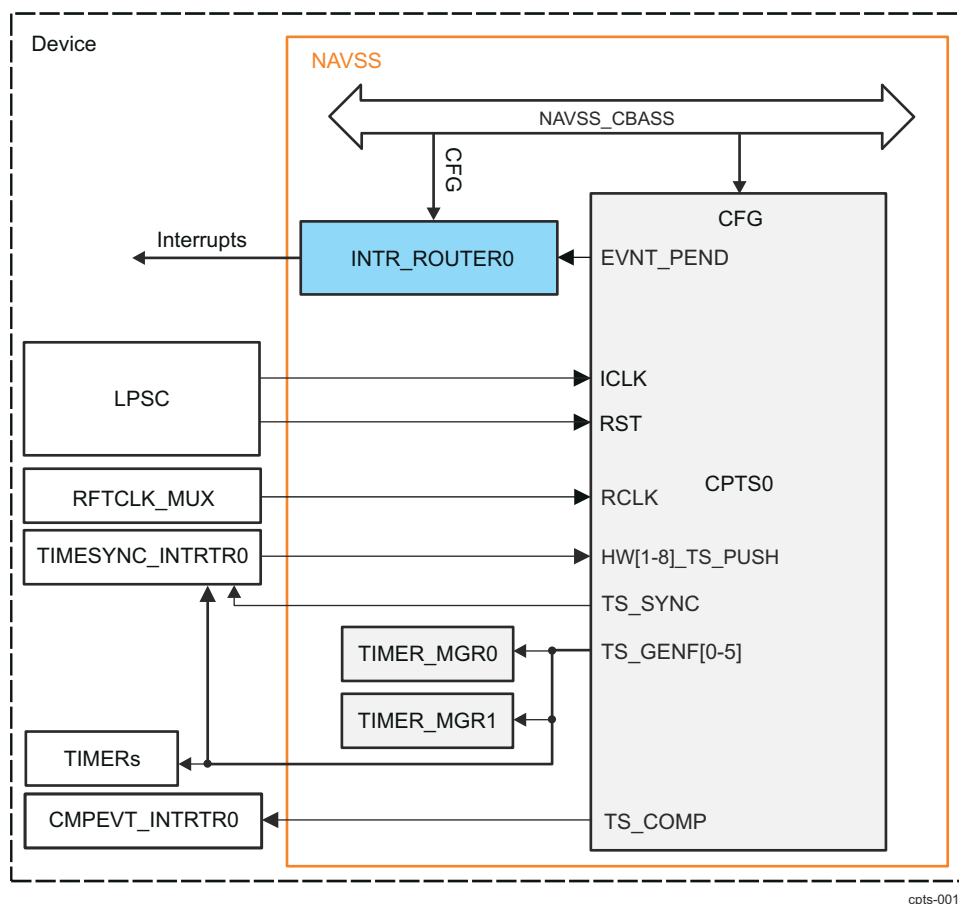
### 11.1.1 CPTS Overview

The Common Platform Time Sync (CPTS) module is used to facilitate host control of time sync operations.

### Table 11-1. CPTS Allocation Across Device Domains

Instance	Domain		
	WKUP	MCU	MAIN
NAVSS0_CPTS0	-	-	✓ (NAVSS)
PCIE0_CPTS0	-	-	✓ (PCIE0)
PCIE1_CPTS0	-	-	✓ (PCIE1)
PCIE2_CPTS0	-	-	✓ (PCIE2)
PCIE3_CPTS0	-	-	✓ (PCIE3)
MCU_CPSW0_CPTS0	-	✓ (CPSW)	-

Figure 11-1 shows an overview of the NAVSS0 CPTS0 module.



### Figure 11-1. NAVSS0\_CPTS0 Overview

#### 11.1.1.1 CPTS Features

Main features of CPTS module are:

- Supports the selection of multiple external clock sources
- Software control of time sync events via interrupt or polling

- Supports 8 hardware timestamp push inputs
- Supports timestamp counter compare output (TS\_COMP)
- Supports timestamp counter bit output (TS\_SYNC)
- Supports 6 timestamp Generator function outputs (TS\_GENF0 to TS\_GENF5)
- 32-bit and 64-bit timestamp modes.

#### **11.1.1.2 CPTS Not Supported Features**

The following features are not supported by the module:

- Oscillator adjustment by module

### 11.1.2 CPTS Integration

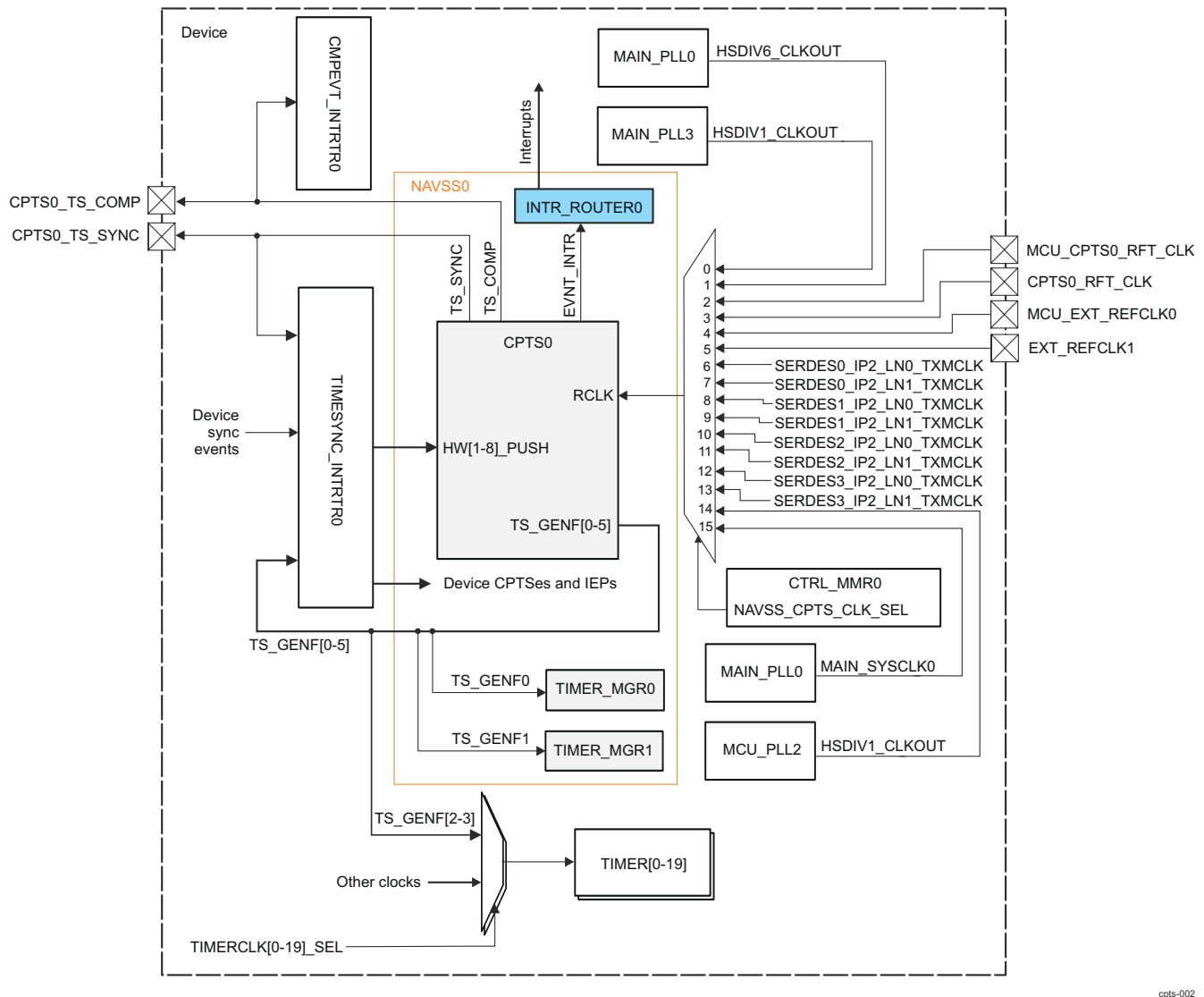
This section describes module integration in the device, including information about clocks, resets, and hardware requests.

#### Note

This section describes the CPTS in the main Navigator subsystem (NAVSS0). For CPTS in PCIe integration, please see *PCIe Subsystem Precision Time Measurement (PTM) in Peripheral Component Interconnect Express (PCIe) Subsystem*.

For CPTS in CPSW integration, please refer to *Gigabit Ethernet Switch (CPSW)*.

Figure 11-2 shows the NAVSS0\_CPTS0 integration.



**Figure 11-2. NAVSS0\_CPTS0 Integration**

Table 11-2 and Table 11-3 summarize the integration of the module in the device.

**Table 11-2. CPTS Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
NAVSS0_CPTS0	PSC0	GP	LPSC0	NAVSS_CBASS

**Table 11-3. CPTS Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
NAVSS0_CPTS0	CPTS0_ICLK	MODSS_VBUS_D2_CLK	MAIN_SYSCLK0	CPTS0 interface clock
	CPTS0_RCLK	HSDIV1_CLKOUT	MAIN_PLL3	CPTS0 reference clock. Selectable in CTRL_MMR CTRLMMR_NAVSS_CLKSEL register. The recommended RCLK frequency is greater than or equal to VBUS clock frequency.
		HSDIV6_CLKOUT	MAIN_PLL0	
		MCU_CPTS0_RFT_CLK	Pin	
		CPTS0_RFT_CLK	Pin	
		MCU_EXT_REFCLK0	Pin	
		EXT_REFCLK1	Pin	
		SERDES0_IP2_LN0_TX MCLK <sup>(1)</sup>	SERDES0	
		SERDES0_IP2_LN1_TX MCLK	SERDES0	
		SERDES1_IP2_LN0_TX MCLK	SERDES1	
		SERDES1_IP2_LN1_TX MCLK	SERDES1	
		SERDES2_IP2_LN0_TX MCLK	SERDES2	
		SERDES2_IP2_LN1_TX MCLK	SERDES2	
		SERDES3_IP2_LN0_TX MCLK	SERDES3	
		SERDES3_IP2_LN1_TX MCLK	SERDES3	
		HSDIV1_CLKOUT	MCU_PLL2	
		MAIN_SYSCLK0	MAIN_PLL0	
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
NAVSS0_CPTS0	CPTS0_RST	MODSS_RST	LPSC0	CPTS0 hardware reset

(1) IP2 in SERDES clock names corresponds to PCIe.

**Table 11-4. CPTS Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
NAVSS0_CPTS0	CPTS0_EVNT_PEND_INT	IN_INTR[391]	INTR_ROUTER0	Event pending interrupt	Level
DMA Events					
Module Instance	Module DMA Event	Destination DMA Event Input	Destination	Description	Type
NAVSS0_CPTS0	-	-	-	No PDMA channels to external DMA engines	-
Time Sync Event Inputs					
Module Instance	Module Sync Input	Sync Source Signal	Source	Description	Type

**Table 11-4. CPTS Hardware Requests (continued)**

NAVSS0_CPTS0	CPTS0_HW1_PUSH	TIMESYNC_INTRTR0_OUTL_0	TIMESYNC_INTRTR0	Asynchronous hardware timestamp 1 push input	Pulse
	CPTS0_HW2_PUSH	TIMESYNC_INTRTR0_OUTL_1	TIMESYNC_INTRTR0	Asynchronous hardware timestamp 2 push input	Pulse
	CPTS0_HW3_PUSH	TIMESYNC_INTRTR0_OUTL_2	TIMESYNC_INTRTR0	Asynchronous hardware timestamp 3 push input	Pulse
	CPTS0_HW4_PUSH	TIMESYNC_INTRTR0_OUTL_3	TIMESYNC_INTRTR0	Asynchronous hardware timestamp 4 push input	Pulse
	CPTS0_HW5_PUSH	TIMESYNC_INTRTR0_OUTL_4	TIMESYNC_INTRTR0	Asynchronous hardware timestamp 5 push input	Pulse
	CPTS0_HW6_PUSH	TIMESYNC_INTRTR0_OUTL_5	TIMESYNC_INTRTR0	Asynchronous hardware timestamp 6 push input	Pulse
	CPTS0_HW7_PUSH	TIMESYNC_INTRTR0_OUTL_6	TIMESYNC_INTRTR0	Asynchronous hardware timestamp 7 push input	Pulse
	CPTS0_HW8_PUSH	TIMESYNC_INTRTR0_OUTL_7	TIMESYNC_INTRTR0	Asynchronous hardware timestamp 8 push input	Pulse
Time Sync Event Outputs					
Module Instance	Module Sync Output	Destination Sync Input	Destination	Description	Type
NAVSS0_CPTS0	CPTS0_TS_GENF0	TIMESYNC_INTRTR0_I N4 EON_TICK_EVT	TIMESYNC_INTRTR0 TIMER_MGR0	Generation Function Output 0	Edge
	CPTS0_TS_GENF1	TIMESYNC_INTRTR0_I N5 EON_TICK_EVT	TIMESYNC_INTRTR0 TIMER_MGR1	Generation Function Output 1	Edge
	CPTS0_TS_GENF2	TIMESYNC_INTRTR0_I N6 0xC	TIMESYNC_INTRTR0 TIMER[0:19]_CLKMUX	Generation Function Output 2	Edge
	CPTS0_TS_GENF3	TIMESYNC_INTRTR0_I N7 0xD	TIMESYNC_INTRTR0 TIMER[0:19]_CLKMUX	Generation Function Output 3	Edge
	CPTS0_TS_GENF4	TIMESYNC_INTRTR0_I N8	TIMESYNC_INTRTR0	Generation Function Output 4	Edge
	CPTS0_TS_GENF5	TIMESYNC_INTRTR0_I N9	TIMESYNC_INTRTR0	Generation Function Output 5	Edge
	CPTS0_TS_SYNC	TIMESYNC_INTRTR0_I N36 CPTS0_TS_SYNC	TIMESYNC_INTRTR0 Pin	Sync Output	Edge
Compare Event Outputs					
Module Instance	Module Comp Output	Destination Comp Input	Destination	Description	Type
NAVSS0_CPTS0	CPTS0_TS_COMP	CMPEVT_INTRTR0_IN 8	CMPEVT_INTRTR0	Comparison Output	Edge
		CPTS0_TS_COMP	Pin		

### 11.1.3 CPTS Functional Description

#### 11.1.3.1 CPTS Architecture

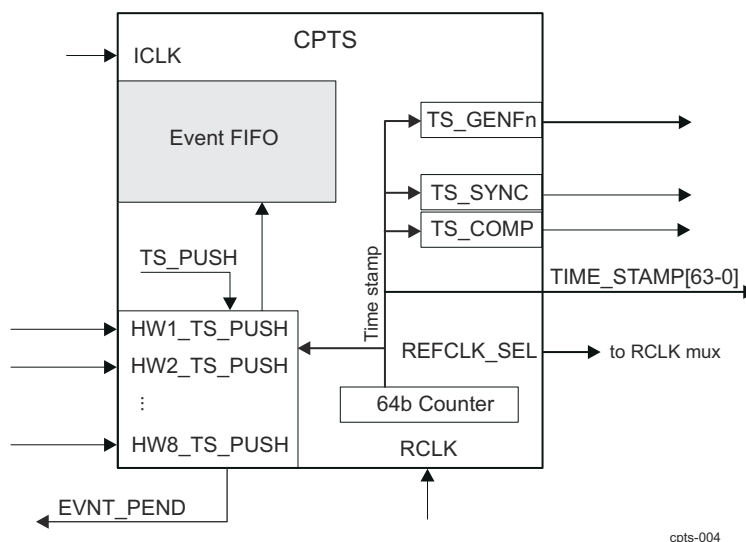
Figure 11-3 shows the architecture of the CPTS module.

The CPTS module is used to facilitate host control of time sync operations. The CPTS collects time sync events and then presents them to the host for processing. The types of time sync events are as follows:

- Host Transmit Event (CPSW0\_CPTS0)

- Ethernet receive event (CPSW0\_CPTS0)
- Ethernet transmit event (CPSW0\_CPTS0)
- Time stamp push event
- Time stamp counter rollover event (32-bit mode only)
- Time stamp counter half-rollover event (32-bit mode only)
- Hardware Time Stamp Push Event
- Time Stamp Compare Event

The reference clock used for the time stamp (RCLK) can be derived from several sources.



**Figure 11-3. CPTS Block Diagram**

#### Note

See *CPTS Integration* for CPTS integration in the NAVSS0.

#### Note

This section describes the CPTS in the main Navigator subsystem (NAVSS0). For CPTS in PCIe integration, please see *PCIe Subsystem Precision Time Measurement (PTM) in Peripheral Component Interconnect Express (PCIe) Subsystem*.

For CPTS in CPSW integration, please refer to in *Gigabit Ethernet Switch (CPSW)*.

#### 11.1.3.2 CPTS Initialization

The CPTS module must be configured as follows:

1. Reset the CPTS module
2. Clear the CPTS\_EN bit in the CPTS\_CONTROL\_REG
3. Write the RFTCLK\_SEL value in the CPTS\_RFTCLK\_SEL\_REG with the desired reference clock selection
4. Set the CPTS\_EN bit in the CPTS\_CONTROL\_REG
5. If using interrupts and not polling, enable the interrupt by setting the TS\_PEND\_EN bit in the CPTS\_INT\_ENABLE\_REG

#### 11.1.3.3 32-bit Time Stamp Value

The time stamp value is a 32-bit value that increments on each RCLK rising edge when CPTS\_EN is set to 1. When CPTS\_EN is cleared to 0, the time stamp value is reset to 0.



If more than 32-bits of time stamp are required by the application, the host software must maintain the necessary number of upper bits. The upper time stamp value should be incremented by the host when the rollover event is detected.

For test purposes, the time stamp can be written via the time stamp load function (CPTS\_TS\_LOAD\_VAL\_REG and CPTS\_TS\_LOAD\_EN\_REG).

The TS\_INC\_VAL[2:0] value must be zero in 32-bit mode. Nudge and PPM adjustments are not supported in 32-bit mode.

#### 11.1.3.4 64-bit Time Stamp Value

The time stamp value is a 64-bit value that increments on each RCLK rising edge when CPTS\_EN is set to 1. When CPTS\_EN is cleared to 0, the time stamp value is reset to 0.

64-bit mode is selected via CPTS\_CONTROL\_REG[5] MODE\_64BIT bit set to 1.

For test purposes, the time stamp can be written via the time stamp load function (CPTS\_TS\_LOAD\_VAL\_REG, CPTS\_TS\_LOAD\_HIGH\_VAL\_REG, and CPTS\_TS\_LOAD\_EN\_REG).

The TS\_ADD\_VAL feature is included to allow 1-ns timestamp operations with an RCLK rate less than 1 GHz. Table 11-5 shows the RCLK and TS\_ADD\_VAL values for 1-ns operations. The highest RCLK frequency possible should be used as allowed by the silicon technology.

**Table 11-5. TS\_ADD\_VAL**

RCLK (MHz)	TS_ADD_VAL
1000	0
500	1
333.33	2
250	3
200	4
166.66	5
142.85714	6
125	7

##### 11.1.3.4.1 64-Bit Timestamp Nudge

The 64-bit TIME\_STAMP value can be adjusted by writing the CPTS\_TS\_NUDGE\_VAL\_REG[7:0] register value which is a two's complement value. A value of 0xFF will subtract one RCLK clock from the next incremented TIME\_STAMP[63:0] value. A nudge value of 0x01 will add one RCLK clock to the next incremented TIME\_STAMP[63:0] value. For example, if the current TIME\_STAMP value is 0x0F06, and TS\_ADD\_VAL[2:0]=3, the next incremented timestamp value would be 0x0F0A without a nudge and 0x0F0A +/- TS\_NUDGE[7:0] with a nudge. The TS\_NUDGE value is cleared to zero when the nudge has occurred.

##### 11.1.3.4.2 64-bit Timestamp PPM

The 64-bit TIME\_STAMP can be adjusted by parts per million (PPM) or by parts per hour (PPH). Writing a non-zero value to the TS\_PPM[41:0] value enables PPM operations. The adjustment is up or down depending on the TS\_PPM\_DIR bit. The TIME\_STAMP value is increased by the PPM value when TS\_PPM\_DIR is cleared and decreased by the PPM value when TS\_PPM\_DIR is set.

- To adjust for 100 parts per million the configured value for TS\_PPM[41:0] is:  
 $1\,000\,000/100 = 10\,000$  (DEC)
- To adjust for 1 part per hour at 1 GHz RCLK, the configured value for TS\_PPM[41:0] is:  
 $(1\,000\,000\,000\text{Hz}/1\text{pph}) \times (3600\text{ seconds}/\text{hour}) = 366\,30B8\,A000$  (HEX)

### 11.1.3.5 Event FIFO

All time sync events are push onto the Event FIFO. No overrun indication supported. Software must service the event FIFO in a timely manner to prevent FIFO overrun.

### 11.1.3.6 Timestamp Compare Output

CPTS features one Time Stamp Compare (TS\_COMP) output.

---

#### Note

64-bit mode operation is identical to 32-bit mode except that all 64 bits of the TIME\_STAMP[63-0] are used instead of only the lower 32 bits.

---

#### 11.1.3.6.1 Non-Toggle Mode

The TS\_COMP output is asserted for CPTS\_TS\_COMP\_LEN\_REG[31-0] RCLK periods when the TIME\_STAMP[31-0] value compares with the CPTS\_TS\_COMP\_VAL\_REG[31-0] and the length value is non-zero. The TS\_COMP rising edge occurs three RCLK periods after the values compare. A timestamp compare event is pushed into the event FIFO when TS\_COMP is asserted. The polarity of the TS\_COMP output is determined by the CPTS\_CONTROL\_REG[2] TS\_POLARITY bit. The output is asserted low when the polarity bit is 0.

#### 11.1.3.6.2 Toggle Mode

The TS\_COMP output is asserted for CPTS\_TS\_COMP\_LEN\_REG RCLK periods when the TIME\_STAMP[31-0] value compares with the CPTS\_TS\_COMP\_VAL\_REG[31-0] and the length value is non-zero. The TS\_COMP toggles thereafter on CPTS\_TS\_COMP\_LEN\_REG[31-0] RCLK periods. The length high or low can be adjusted by writing the CPTS\_TS\_COMP\_NUDGE\_REG[7-0] register value which is a two's complement value. A value of 0xFF will subtract one RCLK from the CPTS\_TS\_COMP\_LEN\_REG value. A value of 0x01 will add one RCLK to the CPTS\_TS\_COMP\_LEN\_REG value. Only a single high or low time is adjusted (nudged) and the CPTS\_TS\_COMP\_NUDGE\_REG value is cleared to zero when the nudge has occurred. The TS\_COMP output is asserted low when the CPTS\_CONTROL\_REG[2] TS\_COMP\_POLARITY bit is 0.

No compare events and no CPTS\_EVNT interrupts are generated in toggle mode.

The CPTS\_CONTROL\_REG[6] TS\_COMP\_TOG bit must be set for toggle mode, and must be set before writing a non-zero value to CPTS\_TS\_COMP\_LEN\_REG.

### 11.1.3.7 Timestamp Sync Output

The TS\_SYNC output is a selected bit of the TIME\_STAMP counter value. One of the counter bits 17-31 can be selected in CPTS\_CONTROL\_REG[31-28] TS\_SYNC\_SEL. The TS\_SYNC output is disabled when CPTS\_CONTROL\_REG[31-28] TS\_SYNC\_SEL is zero.

If the selected counter bit is 1 at the time when TS\_SYNC\_SEL value is written then a rising edge will not occur on the TS\_SYNC output. A rising edge will occur on the TS\_SYNC output upon the next transition-to-1 of the selected counter bit. The TS\_SYNC\_SEL value must be written to zero before changing to a different non-zero value. No events are generated due to the TS\_SYNC operation. The TS\_SYNC output is two RCLK periods after the actual count value.

### 11.1.3.8 Timestamp GENF Output

There are six TS Generate Function (TS\_GENFn) outputs (n = 0 to 5). The TS\_GENFn outputs have a programmable cycle (frequency) with a PPM feature and a software nudge feature. The TS\_GENFn output cycle is CPTS\_LENGTH\_REG\_j[31:0] RCLK periods (which is different than TS\_COMP operation).

The TS\_GENFn output cycle is CPTS\_LENGTH\_REG\_j[31:0] RCLK periods beginning when the TIME\_STAMP[63:0] value compares with the CPTS\_COMP\_LOW\_REG\_j[63:0] and the length value is non-zero. The TS\_GENFn output cycle repeats thereafter every CPTS\_LENGTH\_REG\_j[31:0] RCLK periods. The upper 32-bit word should be written first for 64-bit values. The length should be zero while the comparison

value and other configuration parameters are being configured. The length should be written non-zero to enable operations last. The first cycle after comparison is active high when the TS\_GENFn\_POLARITY bit is low. No compare events and no CPTS\_EVNT interrupts are generated.

#### 11.1.3.8.1 GENFn Nudge

The cycle length can be adjusted by writing the CPTS\_NUDGE\_REG\_j[7:0] register value which is a two's complement value. A value of 0xFF will subtract one RCLK clock from the CPTS\_LENGTH\_REG\_j[31:0] value. A value of 0x01 will add one RCLK clock to the CPTS\_LENGTH\_REG\_j value. The CPTS\_NUDGE\_REG\_j value is cleared to zero when the nudge has occurred.

#### 11.1.3.8.2 GENFn PPM

The TS\_GENFn output cycle can be adjusted by parts per million (PPM) or by parts per hour (PPH). Writing a non-zero CPTS\_PPM\_LOW\_REG\_j/CPTS\_PPM\_HIGH\_REG\_j value enables PPM operations. The PPM counter continually loads and decrements to zero and then loads again. A single RCLK adjustment is made when the PPM counter decrements to zero. The adjustment is up or down depending on the CPTS\_CONTROL\_REG\_j[0] PPM\_DIR bit. When PPM\_DIR is set a single RCLK time is subtracted from the generate function counter which has the effect of increasing the generate function frequency by the PPM amount. When PPM\_DIR is clear a single RCLK time is added to the generate function counter which has the effect of decreasing the generate function frequency by the PPM amount.

- To adjust for 100 parts per million the configured value for TS\_GENFn\_PPM[41:0] is:  
 $1\ 000\ 000/100 = 10\ 000\ (DEC)$
- To adjust for 1 part per hour at 1 GHz RCLK the configured value for TS\_GENFn\_PPM[41:0] is:  
 $(1\ 000\ 000\ 000\text{Hz}/1\text{pph}) \times (3600\ \text{seconds}/\text{hour}) = 346\ 30B8\ A000\ (HEX)$

#### 11.1.3.9 Time Sync Events

Time Sync events are 96-bit or 128-bit (for 32-bit and 64-bit time stamp respectively) values that are pushed onto the event FIFO and read by software in 32-bit reads. Four 32-bit registers, CPTS\_EVENT\_0\_REG through CPTS\_EVENT\_3\_REG hold the data of a time sync event.

##### 11.1.3.9.1 Time Stamp Push Event

Software can obtain the current time stamp value (at the time of the write) by initiating a time stamp push event. The push event is initiated by setting the TS\_PUSH bit of the CPTS\_TS\_PUSH\_REG. The time stamp value is returned in the event, along with a time stamp push event code.

##### 11.1.3.9.2 Time Stamp Counter Rollover Event (32-bit mode only)

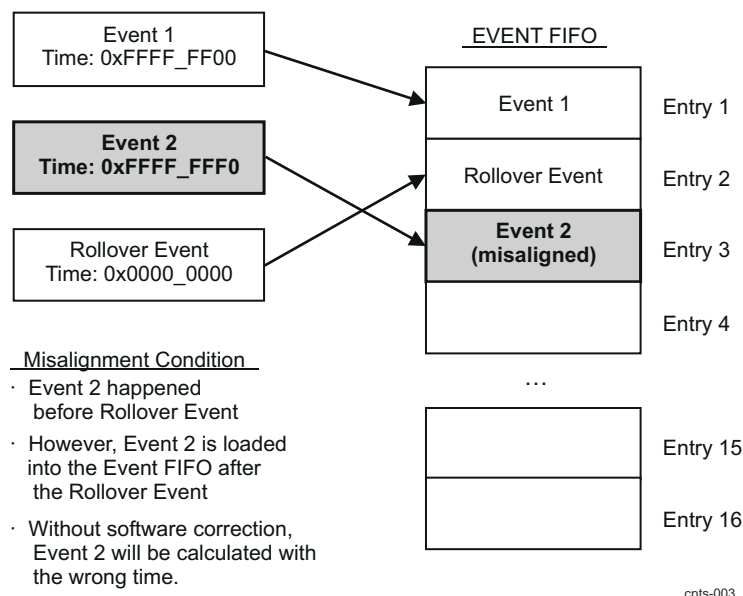
The CPTS module contains a 32-bit time stamp value. The counter upper bits are maintained by host software. The rollover event indicates to software that the time stamp counter has rolled over from 0xFFFF FFFF to 0x0000 0000 and the software-maintained upper count value should be incremented.

##### 11.1.3.9.3 Time Stamp Counter Half-rollover Event (32-bit mode only)

The CPTS includes a time stamp counter half-rollover event. The half-rollover event indicates to software that the time stamp value has incremented from 0x7FFF FFFF to 0x8000 0000. The half-rollover event is included to enable software to correct a misaligned event condition. The half-rollover event is included to enable software to determine the correct time for each event that contains a valid time stamp value, such as an Ethernet event. If an Ethernet event occurs around a counter rollover (full rollover), the rollover event could possibly be loaded into the event FIFO before the Ethernet event, even though the Ethernet event time was actually taken before the rollover. [Figure 11-4](#) shows a misalignment condition.

Host software must detect and correct for misaligned event conditions. For every event after a rollover and before a half-rollover, software must examine the time stamp most significant bit. If bit 31 of the time stamp value is low (0x0000 0000 through 0x7FFF FFFF), then the event time stamp was taken after the rollover and no correction is required. If the value is high (0x8000 0000 through 0xFFFF FFFF), the time stamp value was taken before the rollover and a misalignment is detected. The misaligned case indicates to software that it must subtract one from the upper count value stored in software to calculate the correct time for the misaligned event.

The misaligned event occurs only on the rollover boundary and not on the half-rollover boundary. Software only needs to check for misalignment from a rollover event to a half-rollover event.



**Figure 11-4. Event FIFO Misalignment Condition**

#### 11.1.3.9.4 Hardware Time Stamp Push Event

There are eight hardware time stamp inputs (HW[1-8]\_TS\_PUSH) that can cause hardware time stamp push events to be loaded into the Event FIFO. Each time stamp input is mapped in the device as shown in *CPTS Integration*.

The event is loaded into the event FIFO on the rising edge of the timer, and the PORT\_NUMBER field in the CPTS\_EVENT\_1\_REG register indicates the hardware time stamp input that caused the event.

Each hardware time stamp input must be asserted for at least 10 periods of the selected RCLK clock. Each input can be enabled or disabled by setting the respective bits in the CPTS\_CONTROL\_REG register.

Hardware time stamps are intended to be an extremely low frequency signals, such that the event FIFO does not overrun. Software must keep up with the event FIFO and ensure that there is no overrun, or events will be lost.

#### 11.1.3.10 Timestamp Compare Event

##### Note

Timestamp compare events are generated for non-toggle mode only.

The CPTS can generate an event for a time stamp comparison in 32-bit or 64-bit mode. The TS\_COMP output is also asserted when the event is generated. The event is generated when the TIME\_STAMP value compares with the CPTS\_TS\_COMP\_VAL\_REG/CPTS\_TS\_COMP\_HIGH\_VAL\_REG and the CPTS\_TS\_COMP\_LEN\_REG value is non-zero. The CPTS\_TS\_COMP\_LEN\_REG value should be written by software after the CPTS\_TS\_COMP\_VAL\_REG/CPTS\_TS\_COMP\_HIGH\_VAL\_REG is written and should be zero when the comparison value is written.

#### 11.1.3.11 CPTS Interrupt Handling

When an event is push onto the Event FIFO, an interrupt can be generated to indicate to software that a time sync event occurred. The following steps should be taken to process time sync events using interrupts:

1. Enable the TS\_PEND interrupt by setting the TS\_PEND\_EN bit of the CPTS\_INT\_ENABLE\_REG.
2. Upon interrupt, read the CPTS\_EVENT\_0\_REG through CPTS\_EVENT\_3\_REG register values.

3. Set the CPTS\_EVENT\_POP\_REG[0] EVENT\_POP bit to 1 to pop the previously read value off of the event FIFO.
4. Process the interrupt as required by the application software.

Software has the option of processing more than a single event from the event FIFO in the interrupt service routine in the following way:

1. Enable the TS\_PEND interrupt by setting the TS\_PEND\_EN bit of the CPTS\_INT\_ENABLE\_REG.
2. Upon interrupt, read the CPTS\_EVENT\_0\_REG through CPTS\_EVENT\_3\_REG register values.
3. Set the CPTS\_EVENT\_POP\_REG[0] bit to 1 to pop the previously read value off of the event FIFO.
4. Wait for an amount of time greater than four RCLK periods plus four ICLK periods.
5. Read the TS\_PEND\_RAW bit in the CPTS\_INTSTAT\_RAW\_REG register to determine if another valid event is in the event FIFO. If it is asserted, go to step 2; otherwise, proceed to step 6.
6. Process the interrupt(s) as required by the application software.

Software also has the option of disabling the interrupt and polling the TS\_PEND\_RAW bit of the CPTS\_INTSTAT\_RAW\_REG to determine if a valid event is on the event FIFO.

## 11.2 Timer Manager

This chapter describes the Timer Manager module of the device.

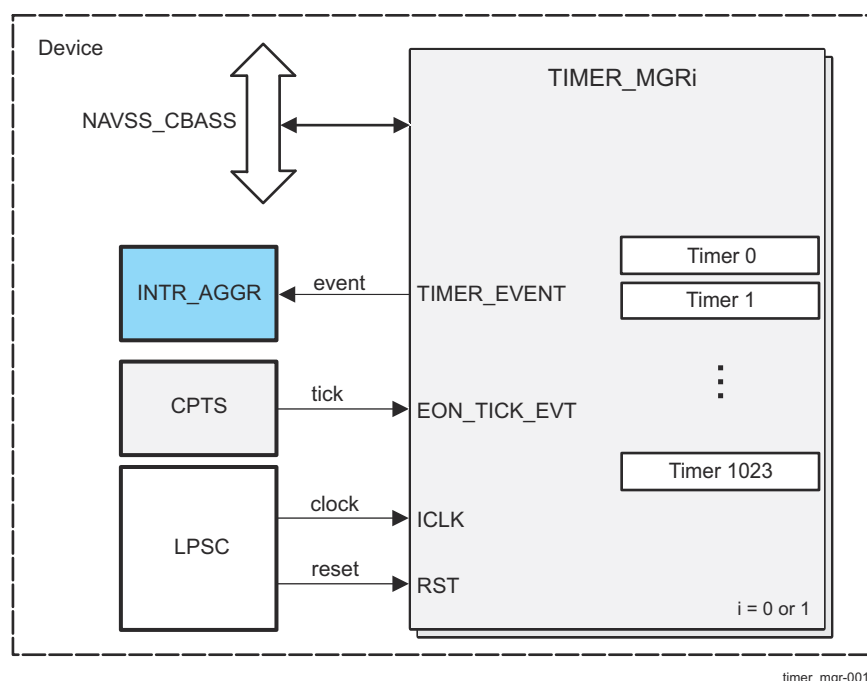
### 11.2.1 Timer Manager Overview

The Timer Manager module provides timers for timing operations for the processes running on multiple processors in the device. NAVSS supports two instances of Timer Manager.

**Table 11-6. TIMER\_MGR Allocation Across Device Domains**

Instance	Domain		
	WKUP	MCU	MAIN
TIMER_MGR0	-	-	✓ (NAVSS)
TIMER_MGR1	-	-	✓ (NAVSS)

Figure 11-5 shows an overview of the Timer Manager module.



**Figure 11-5. Timer Manager Overview**

#### 11.2.1.1 Timer Manager Features

Timer Manager has the following features:

- 1024 × 32-bit RAM-based independent timers (2048 in total for the two Timer Managers)
- Event interface to an interrupt aggregation module in the main NAVSS subsystem
  - Events are triggered when a timer expires or when an expired timer is reset or deactivated
- Host access to determine which timer(s) expired
- 32 registers with individual timeout status (one bit per timer)
- A pair of quick-read registers with the number of timers that have expired and the ids of the first timers to expire
- A bank timeout register with the banks that have expired timers – to avoid having to read all timeout status registers when only a few timers have expired
- Groups of 16 timers separated into pages of 4-K address space
- Timer bits within each page to read expiration status for each timer when software only has access to that page

- 10  $\mu$ s time to cycle through all of the timers
- Host access to reset individual timers
  - Reset values are preprogrammed for each timer
  - Timers may all be programmed before the timer manager is enabled, or programmed after enabled.
- Periodic hardware timers – a timer may be set to automatically reprogram itself without software intervention, so that it expires and reprograms itself endlessly.

#### **11.2.1.2 Timer Manager Not Supported Features**

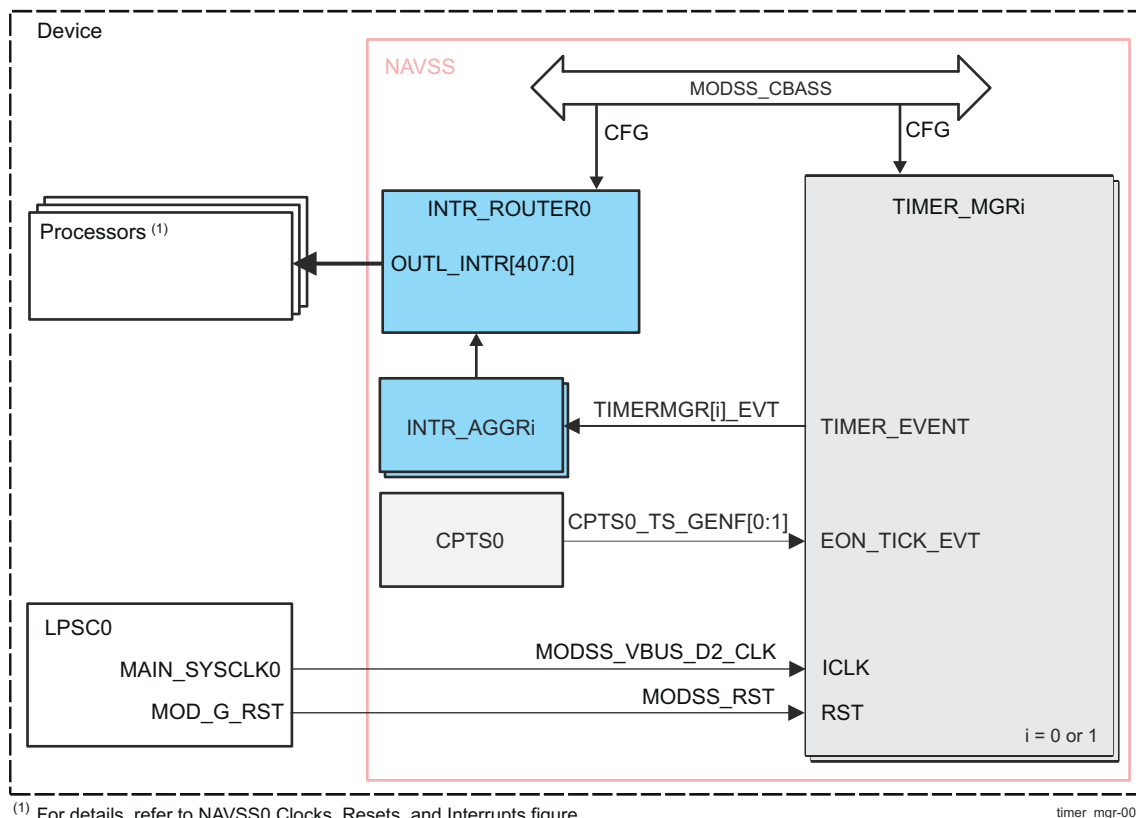
The following features are not supported by the module:

- Tunable GENF CPTS output is an input to this module. The timer bank doesn't manage the divider or the rate/offset compensation. It takes the output of a tuner module as a single input EON\_TICK event.

## 11.2.2 Timer Manager Integration

This section describes module integration in the device, including information about clocks, resets, and hardware requests.

Figure 11-6 shows the Timer Manager integration.



**Figure 11-6. Timer Manager Integration**

Table 11-7 and Table 11-8 summarize the integration of the module in the device.

**Table 11-7. Timer Manager Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
TIMER_MGR0	PSC0	GP	LPSC0	MODSS_CBASS
TIMER_MGR1	PSC0	GP	LPSC0	MODSS_CBASS

**Table 11-8. Timer Manager Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
TIMER_MGR0	TIMER_MGR0_ICLK	MODSS_VBUS_D2_CLK	MAIN_SYSCCLK0	Timer Manager0 interface clock
	EON_TICK_EVT	CPTS0_TS_GENF0	CPTS0	Timer Manager0 asynchronous tick event
TIMER_MGR1	TIMER_MGR1_ICLK	MODSS_VBUS_D2_CLK	MAIN_SYSCCLK0	Timer Manager1 interface clock
	EON_TICK_EVT	CPTS0_TS_GENF1	CPTS0	Timer Manager1 asynchronous tick event
Resets				



**Table 11-8. Timer Manager Clocks and Resets (continued)**

Module Instance	Module Reset Input	Source Reset Signal	Source	Description
TIMER_MGR0	TIMER_MGR0_RST	MODSS_RST	LPSC0	Timer Manager0 hardware reset
TIMER_MGR1	TIMER_MGR1_RST	MODSS_RST	LPSC0	Timer Manager1 hardware reset

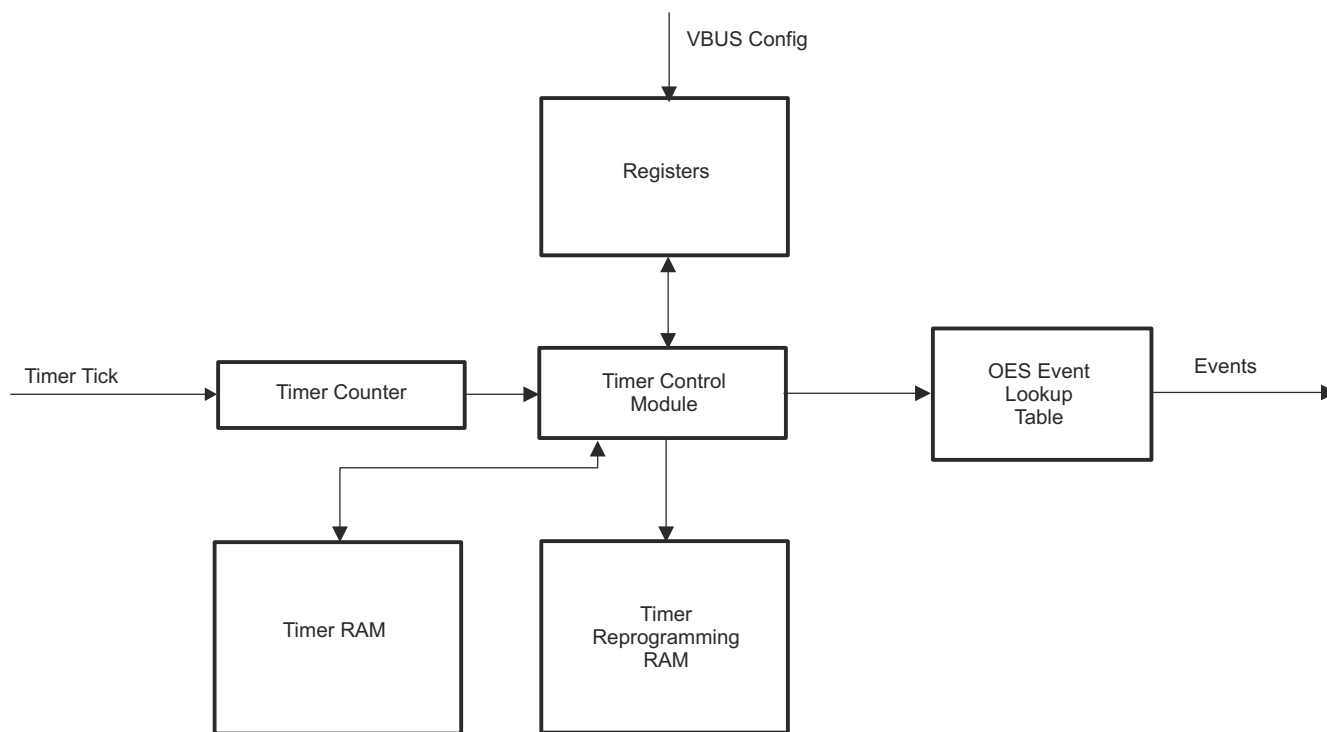
**Table 11-9. Timer Manager Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
TIMER_MGR0	TIMERMGR0_EVT	INTR_AGGR0 SEVI	INTR_AGGR0	Timer expiration events ×1024	ETL
TIMER_MGR1	TIMERMGR1_EVT	INTR_AGGR1 SEVI	INTR_AGGR1	Timer expiration events ×1024	ETL
DMA Events					
Module Instance	Module DMA Event	Destination DMA Event Input	Destination	Description	Type
TIMER_MGR0	-	-	-	No PDMA channels to external DMA engines	-
TIMER_MGR1	-	-	-	No PDMA channels to external DMA engines	-

### 11.2.3 Timer Manager Functional Description

#### 11.2.3.1 Timer Manager Function Overview

Figure 11-7 shows the Timer Manager diagram.



timer\_mgr-004

**Figure 11-7. Timer Manager Block Diagram**

The 1024 timers in the timer manager are organized as the Timer RAM in Figure 11-7. When the timer manager is enabled, the timer counter will increment with the input timer tick, and the control module continuously loops over the Timer RAM, comparing the values against this timer count. When the timer count is greater than a timer value, the timer control module will generate a timer expiration event.

Based on software writes, the timer control module will update the values in the Timer RAM to setup or cancel active timers. Whenever a new value is written to a `TIMERMGR_SETUP_j_k` memory address, the corresponding timer is touched/setup with the current sum of the timer count and the written value; written values are relative to the current timer count. These values are stored in the Timer Reprogramming RAM, so the software may also write to the `TIMERMGR_CONTROL_j_k` registers to flag a timer for reprogramming, using the previously stored reprogramming value. The latter method is preferred when the timer setup value is not changing, as it does not interrupt the RAM state machine.

### 11.2.3.2 Timer Counter

The Timer Manager takes an input timer tick to control the timer rate. All timers share a time base. On each rising edge of this tick event, the timer counter increments. The value of the timer counter may be read by software via the `TIMERMGR_COUNTER` register.

#### 11.2.3.2.1 Timer Counter Rollover

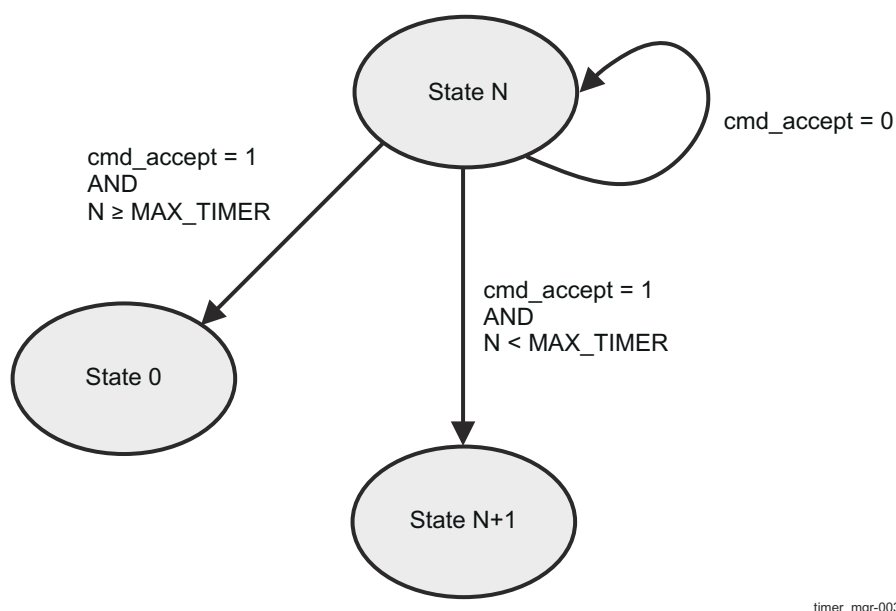
A rollover bit is maintained for each timer, such that when reprogrammed, it is the carry bit from the sum of the current time (counter) and the relative reprogramming value for that timer. This has the effect that any timer whose rollover bit is currently set to a 1 will not expire in the current epoch.

When the counter transitions from `0xFFFF FFFF` to `0x0000 0000`, the timer control module changes the rollover bit for all timers from a 1 to a 0. All timers that previously did not have the rollover bit set, will expire when they are next visited by the state machine, as their timeout value was for the previous epoch.

The overall effect is that the timers continue to expire as expected, regardless of when the counter rollover occurs relative to the state machine.

### 11.2.3.3 Timer Control Module (FSM)

The timer control module controls the RAM accesses for reading and writing. It controls the loops over the timer RAM, checks the values in the RAM to observe expiring timers, flagging timeouts so that events may be triggered and reported. It also takes as input any software-written updates to setup, clear, and touch the timers.



**Figure 11-8. Timer State Machine**

The timer states are sequential, that is, Timer 0 is checked, then Timer 1 is checked, then Timer 2, and so on. The Timer Control FSM will stall in a given state if the `cmd_accept` signal from the `ksdw_spram_ecc` module goes low, and will remain there until the ECC RAM will accept a command.

The timer state always transitions from N to N+1, unless N meets or exceeds the programmed `MAX_TIMER` value, in which case the system will start over at state 0.

#### 11.2.3.4 Timer Reprogramming

There is a reprogramming module that runs in lock step with the FSM, always one state ahead – the FSM exports its `next_state` for use by the reprogramming RAM.

This RAM keeps track of the setup values for the timers as programmed by the `TIMERMGR_SETUP_j_k` memory mapped locations. There is a SET flag for each timer outside of the RAM – if this SET flag is true for the `next_state` timer, the VBUS ready signals are deasserted; if there is a pending timer setup, there cannot be a read or write to the memory, as this would interrupt the Timer FSM state machine and, with enough accesses, potentially cause the system to exceed the 10- $\mu$ s window for timer accuracy.

##### 11.2.3.4.1 Periodic Hardware Timers

By default, timers remain expired until they are either disabled or reset by software. However, by setting the `TIMERMGR_CONTROL_j_k[8]` `AUTORESET` bit for a timer, it will be treated as a periodic hardware timer by the timer manager.

A periodic hardware timer automatically resets itself after expiration. When a periodic hardware timer expires, it is flagged so that on the next pass of the timer control loop, the timer is reprogrammed with the stored `TIMERMGR_SETUP_j_k` value, allowing a timer to repeatedly timeout at intervals equal to the setup value. No software writes are required to clear a periodic hardware timer after it is expired.

There are no *down* events for a periodic hardware timer. A periodic hardware timer pushes an *up* event to the event FIFO when it expires and then resets itself, but does not push any *down* events.

#### 11.2.3.5 Event FIFO

As the subsystem event aggregator requires more than one cycle to process events, there is an internal FIFO to store the generated events. Events may be generated by expiring timers or by activity on the CBA interface, as disabled timers or reset timers must send an event deactivation if the timer was previously expired.

This has an impact on performance, as if there are too many timer expirations in a given window, it potentially violates the 10- $\mu$ s requirement (this is true without the event FIFO as well, given that the system cannot keep up with a timer expiring every cycle, in the worst case).

#### 11.2.3.6 Output Event Lookup (OES RAM)

The expiring timers must be assigned corresponding event numbers for use in the instantiating system. These lookup values correspond to the 16-bit event number that is output on the event interface when a given timer expires.

The lookup table for timer to event number mapping defaults all timers to the invalid event number (0xFFFF), and any invalid event numbers observed upon lookup for an expiring timer will result in no events being generated by the timer manager – invalid (0xFFFF) events are suppressed and will not result in an event on the event interface.

It is expected that the output event assignments will be programmed once per module reset, and that they are static during operation of the timer manager.

## 11.2.4 Timer Manager Programming Guide

### 11.2.4.1 Timer Manager Low-level Programming Models

This section covers the low-level hardware programming sequences for configuration and usage of the module.

#### 11.2.4.1.1 Surrounding Modules Global Initialization

This procedure initializes the surrounding modules when the Timer Manager module is used for the first time after a device reset.

**Table 11-10. Global Initialization of Surrounding Modules**

Surrounding Modules	Comments
LPSC	Timer Manager interface clock is enabled with the NAVSS0 MODSS enabled. For more information, see <i>Clocking</i>
CPTS	CPTS0 configuration must be performed as NAVSS0_CPTS0 is the tick source for Timer manager timers. See <i>CPTS</i>

#### 11.2.4.1.2 Initialization Sequence

Initialization of the Timer Manager must include the following steps:

- Set the number of timers to be used. `TIMERMGR_CNTL[10-1] MAX_TIMER`
  - This controls the loop over the timer RAM and is the highest ID timer that will be used in the operation of the Timer Manager (`MAX_TIMER + 1` is the total number of timers in use)
  - This number must not change during the course of operation; it can only be changed when `TIMERMGR_CNTL[0] ENABLE = 0`. It is assumed that the user will set the `MAX_TIMER` value as required.
  - This step can be skipped if using all timers. `MAX_TIMER` defaults to 1024 (that is, all timers in use).
- Write output event mapping for all timers to the OES output event lookup table. This must be done before enabling timer manager and must not be done again once the timer manager has been enabled.
- Write initial setup values to the `TIMERMGR_SETUP_j_k` registers. All timers that will be used initially must have a setup value before the Timer Manager is enabled.
- Enable individual timers. There are two ways to do this.
  - To enable all timers, use the `TIMERMGR_CNTL[12] MASS_ENABLE` bit. This will enable all timers from 0 to `MAX_TIMER`.
    - Note that `MAX_TIMER` and `MASS_ENABLE` should be set in successive writes, or the previous value of `MAX_TIMER` will be used for `MASS_ENABLE`
    - `MASS_ENABLE` must only be used during initialization.
  - Alternately, individual timers can be enabled or disabled with `TIMERMGR_CONTROL_j_k`
    - It is therefore faster to set the individual `TIMERMGR_CONTROL_j_k` bits if more than half of the in-use timers are to be enabled initially. If more than half of the in-use timers are to be enabled initially, it is faster to use `MASS_ENABLE` and then disable specific timers with the `TIMERMGR_CONTROL_j_k` bits.
- Enable the timer manager. Set `TIMERMGR_CNTL[0] ENABLE` to 1.

#### 11.2.4.1.3 Real-time Operating Requirements

While running, a CPU can take the following actions to enable, disable, or touch the timers:

##### 11.2.4.1.3.1 Timer Touch

When a timer needs to be updated – indicating that an interface or function is still alive and timeout should be delayed – there are two means of touching a timer:

- If the timer's respective `TIMERMGR_SETUP_j_k` has the appropriate value for its function, the timer can be set again by writing a 1 to the SET bit of the `TIMERMGR_CONTROL_j_k` register.
  - As the timer should remain enabled, this effectively means writing a 3 to the `TIMERMGR_CONTROL_j_k` register for the timer to be touched

- If the timer's respective `TIMERMGR_SETUP_j_k` needs a new value, writing this value has the side effect of also setting the timer – there is no need to also write to the SET bit of `TIMERMGR_CONTROL_j_k`. This should be relatively infrequent during operation, as the setup values for all timers must be written during initialization.

The preferred method for touching timers is using the SET bit of `TIMERMGR_CONTROL_j_k`. It is considerably more efficient. The reprogramming RAM is a single port RAM, and repeated writes to the `TIMERMGR_SETUP_j_k` registers will impact overall timer performance and may result in longer periods before a valid timeout is observed.

#### 11.2.4.1.3.2 Timer Disable

When a timer is no longer in use, the timer should be disabled with the respective `TIMERMGR_CONTROL_j_k` enable bit.

A disabled timer does not stop the time for that timer relative to the counter, and its current value is considered invalid once disabled. A timer should not be disabled and then subsequently re-enabled without reprogramming/re-setting, as it may timeout unexpectedly.

#### 11.2.4.1.3.3 Timer Enable

To enable a timer while the Timer Manager is running, the timer must first be setup again.

- If the current `TIMERMGR_SETUP_j_k` is appropriate for the timer's intended use, the timer can be re-enabled by writing a 3 to the `TIMERMGR_CONTROL_j_k` register. (ENABLE = 1, SET = 1)
- If the current `TIMERMGR_SETUP_j_k` needs to be changed, write to the `TIMERMGR_SETUP_j_k` before re-enabling the timer. After this value is changed, the timer can be re-enabled by writing 0x3 to the `TIMERMGR_CONTROL_j_k` register (ENABLE = 1, SET = 1)
  - While `TIMERMGR_SETUP_j_k` implies SET, it must still be set as the timer is enabled.

#### 11.2.4.1.4 Power Up/Power Down Sequence

The timer manager must be disabled (clear `TIMERMGR_CNTL[0]` ENABLE to 0) before the timer manager is powered down. The timer manager will only be idle if it is not enabled and the event FIFO is empty – an enabled Timer Manager means the state machine is running and the timer manager is waiting for expired timers.

To power up the Timer Manager if it has been powered down but not reset, the timer values in the timer RAM must be assumed to be invalid. Any timers must either be reprogrammed (`TIMER_SET`) before the timer is re-enabled. If this is not done, the Timer Manager may report timeouts as soon as it is enabled.

Additionally, all timers that have expired must be cleared before disabling the Timer Manager, otherwise there will be no Event Down generated on the event interface to clear the expired timers from the interrupt aggregator. This effectively means that software must disable all timers before writing a 0 to `TIMERMGR_CNTL[0]` ENABLE.

The event FIFO will continue to drive pending events even if the Timer Manager is disabled, so the remaining Event Down actions will still occur – software only needs to disable the timers to ensure that the Event Down commands are pushed into the event FIFO but does not need to wait for the event FIFO to flush before disabling the timer manager.

## 11.3 Time Sync and Compare Events

### 11.3.1 Time Sync Architecture

This section provides a high-level overview of the SoC time synchronization architecture.

#### 11.3.1.1 Time Sync Architecture Overview

[Table 11-11](#) shows the network time synchronization functions supported by the device.

**Table 11-11. Network Time Synchronization Functions in the SoC**

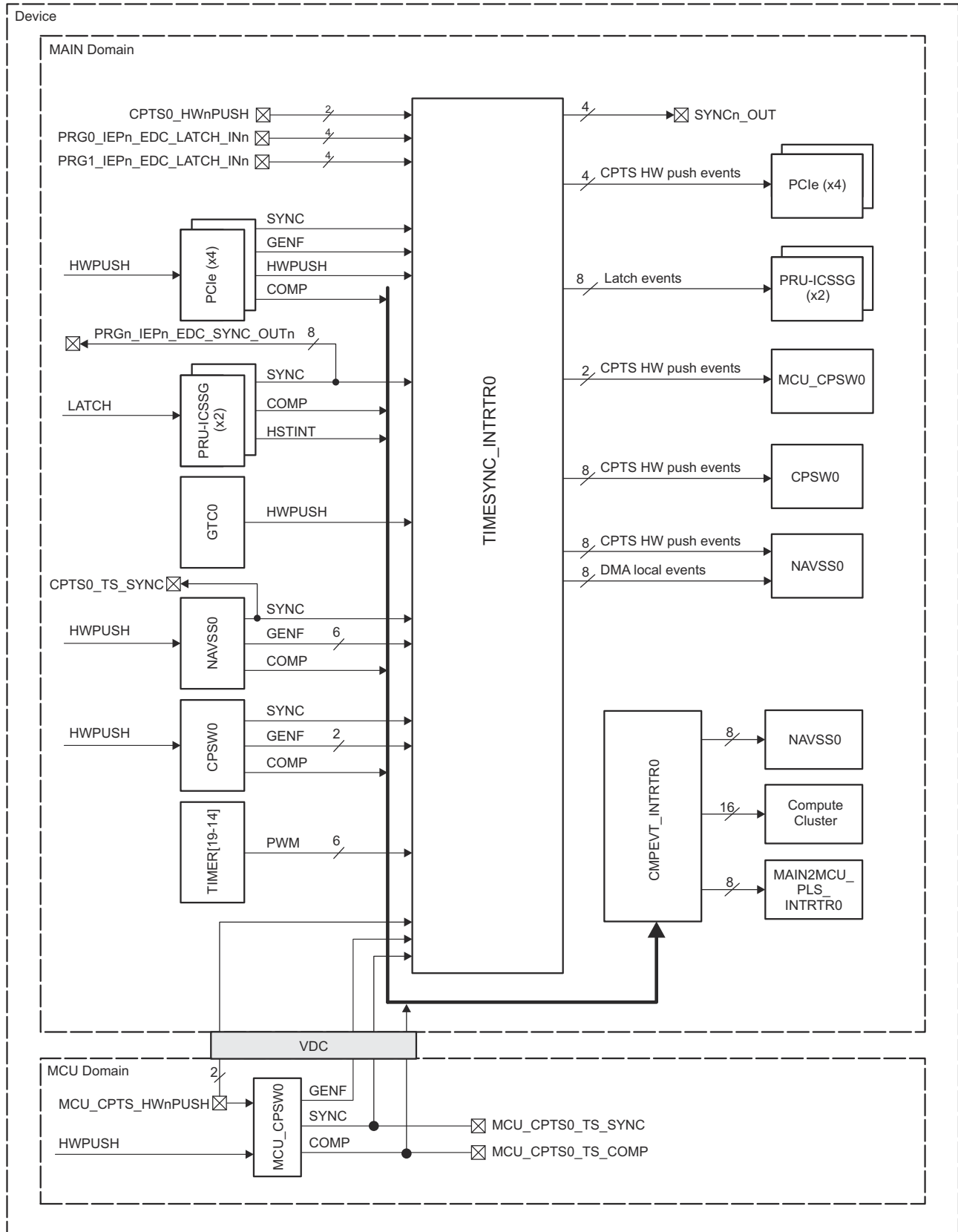
Interface	Time Sync Functions	Supported by
2×PRU-ICSSG	IEEE 1588-2008 (1/2-step), 802.1AS, TSN	PRU-ICSSG firmware
4×PCIE	Precision time measurement (PTM)	PCle controller hardware
2×CPSW	IEEE 1588-2008 (2-step), 802.1AS	CPTS in CPSW
External input reference clock	External time/clock reference	CPTS

#### Note

These time sync functions are described in detail in their respective chapters.

Any of these functions can be a time sync master in the system. A sync router (TIMESYNC\_INTRTR0) provides flexibility for each time domain to choose its synch master independently. In addition, there is also a router (CMPEVT\_INTRTR0) that provides selection of active counter compare events for routing as CPU or DMA events.

[Figure 11-9](#) shows a high-level overview of the SoC time sync architecture.



**Figure 11-9. SoC Time Sync Architecture**

## 11.3.2 Time Sync Routers

### 11.3.2.1 Time Sync Routers Overview

The timesync event router (TIMESYNC\_INTRTR0) and compare event router (CMPEVT\_INTRTR0) are instantiations of the generic interrupt router module in the device.

The TIMESYNC\_INTRTR0 implements a set of multiplexers to provide selection of active CPTS time sync events for routing to CPTS capable modules. There is one register per output that controls the selection (TIMESYNC\_INTRTR0\_MUXCNTL\_y).

The CMPEVT\_INTRTR0 implements a set of multiplexers to provide selection of active CPTS counter compare events for routing as CPU or DMA events. There is one register per output that controls the selection (CMPEVT\_INTRTR0\_MUXCNTL\_y).

[Table 11-12](#) summarizes the configuration details for TIMESYNC\_INTRTR0 and CMPEVT\_INTRTR0.

**Table 11-12. TIMESYNC\_INTRTR0 and CMPEVT\_INTRTR0 Configuration**

Module	Number of Inputs	Number of Outputs	Input Interrupt Type
TIMESYNC_INTRTR0	56	48	Level
CMPEVT_INTRTR0	96	32	Pulse

Refer to [Section 9.3.1, INTRTR Overview](#), for the general programming sequence that is recommended to be followed for all interrupt routers in the device.

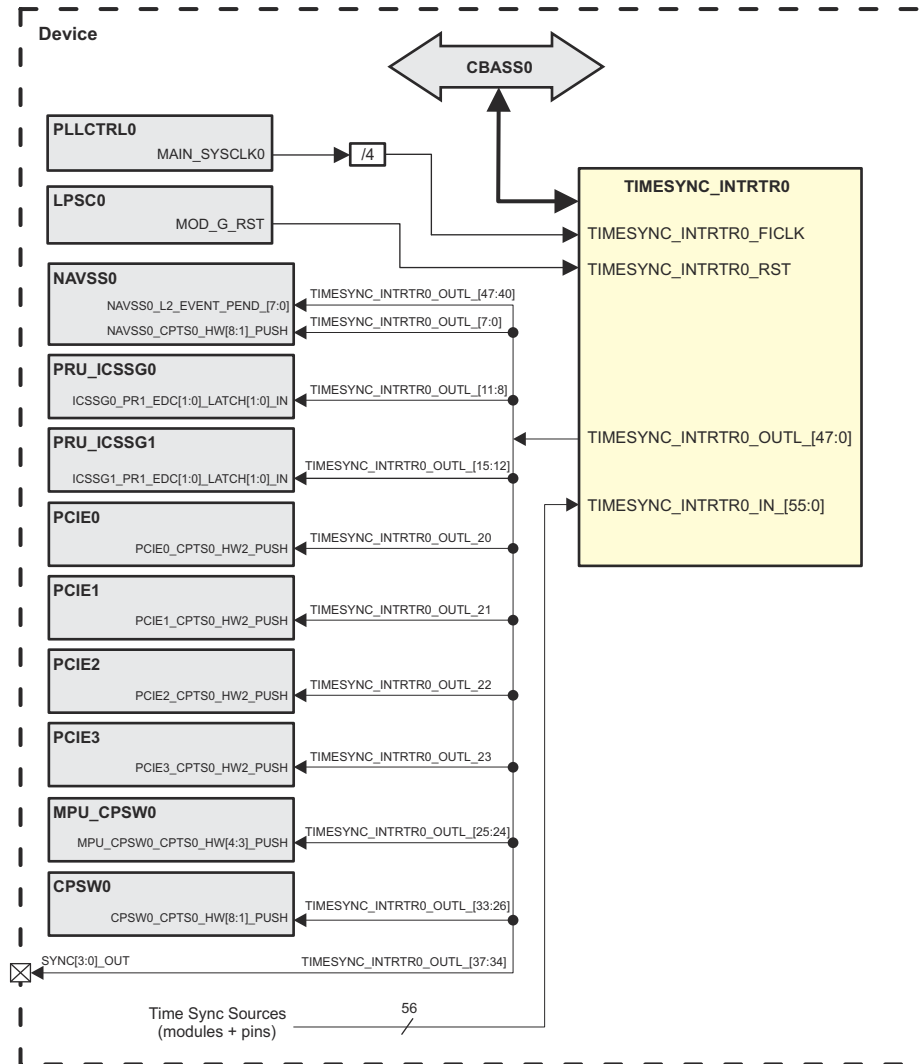


### 11.3.2.2 Time Sync Routers Integration

This section describes the Time Sync Routers integration in the device, including information about clocks, resets, and hardware requests.

### 11.3.2.3 TIMESYNC\_INTRTR0 Integration

Figure 11-10 shows the TIMESYNC\_INTRTR0 integration.



**Figure 11-10. TIMESYNC\_INTRTR0 Integration**

Table 11-13 through Table 11-15 summarize the TIMESYNC\_INTRTR0 integration.

**Table 11-13. TIMESYNC\_INTRTR0 Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
TIMESYNC_INTRTR0	PSC0	PD0	LPSC0	CBASS0

**Table 11-14. TIMESYNC\_INTRTR0 Clocks and Resets**

Clocks
--------

**Table 11-14. TIMESYNC\_INTRTR0 Clocks and Resets (continued)**

Module Instance	Module Clock Input	Source Clock Signal	Source	Description
TIMESYNC_INTRTR0	TIMESYNC_INTRTR0_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	TIMESYNC_INTRTR0 functional and interface clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
TIMESYNC_INTRTR0	TIMESYNC_INTRTR0_RST	MOD_G_RST	LPSC0	TIMESYNC_INTRTR0 hardware reset

**Table 11-15. TIMESYNC\_INTRTR0 Hardware Requests**

Module Time Sync Events (Outputs)					
Module Instance	Module Sync Output	Destination Sync Signal	Destination	Description	Type
TIMESYNC_INTRTR0	TIMESYNC_INTRTR0_OUT_L_0	NAVSS0_CPTS0_HW1_PUSH	NAVSS0	Selectable timesync event 0	Level
	TIMESYNC_INTRTR0_OUT_L_1	NAVSS0_CPTS0_HW2_PUSH	NAVSS0	Selectable timesync event 1	Level
	TIMESYNC_INTRTR0_OUT_L_2	NAVSS0_CPTS0_HW3_PUSH	NAVSS0	Selectable timesync event 2	Level
	TIMESYNC_INTRTR0_OUT_L_3	NAVSS0_CPTS0_HW4_PUSH	NAVSS0	Selectable timesync event 3	Level
	TIMESYNC_INTRTR0_OUT_L_4	NAVSS0_CPTS0_HW5_PUSH	NAVSS0	Selectable timesync event 4	Level
	TIMESYNC_INTRTR0_OUT_L_5	NAVSS0_CPTS0_HW6_PUSH	NAVSS0	Selectable timesync event 5	Level
	TIMESYNC_INTRTR0_OUT_L_6	NAVSS0_CPTS0_HW7_PUSH	NAVSS0	Selectable timesync event 6	Level
	TIMESYNC_INTRTR0_OUT_L_7	NAVSS0_CPTS0_HW8_PUSH	NAVSS0	Selectable timesync event 7	Level
	TIMESYNC_INTRTR0_OUT_L_8	PRU_ICSSG0_PR1_EDC0_LAT_CH0_IN	PRU_ICSSG0	Selectable timesync event 8	Level
	TIMESYNC_INTRTR0_OUT_L_9	PRU_ICSSG0_PR1_EDC0_LAT_CH1_IN	PRU_ICSSG0	Selectable timesync event 9	Level
	TIMESYNC_INTRTR0_OUT_L_10	PRU_ICSSG0_PR1_EDC1_LAT_CH0_IN	PRU_ICSSG0	Selectable timesync event 10	Level
	TIMESYNC_INTRTR0_OUT_L_11	PRU_ICSSG0_PR1_EDC1_LAT_CH1_IN	PRU_ICSSG0	Selectable timesync event 11	Level
	TIMESYNC_INTRTR0_OUT_L_12	PRU_ICSSG1_PR1_EDC0_LAT_CH0_IN	PRU_ICSSG1	Selectable timesync event 12	Level
	TIMESYNC_INTRTR0_OUT_L_13	PRU_ICSSG1_PR1_EDC0_LAT_CH1_IN	PRU_ICSSG1	Selectable timesync event 13	Level
	TIMESYNC_INTRTR0_OUT_L_14	PRU_ICSSG1_PR1_EDC1_LAT_CH0_IN	PRU_ICSSG1	Selectable timesync event 14	Level
	TIMESYNC_INTRTR0_OUT_L_15	PRU_ICSSG1_PR1_EDC1_LAT_CH1_IN	PRU_ICSSG1	Selectable timesync event 15	Level
	TIMESYNC_INTRTR0_OUT_L_20	PCIE0_CPTS0_HW2_PUSH	PCIE0	Selectable timesync event 20	Level
	TIMESYNC_INTRTR0_OUT_L_21	PCIE1_CPTS0_HW2_PUSH	PCIE1	Selectable timesync event 21	Level
	TIMESYNC_INTRTR0_OUT_L_24	MPU_CPSW0_CPTS0_HW3_PUSH	MPU_CPSW0	Selectable timesync event 24	Level
	TIMESYNC_INTRTR0_OUT_L_25	MPU_CPSW0_CPTS0_HW4_PUSH	MPU_CPSW0	Selectable timesync event 25	Level
	TIMESYNC_INTRTR0_OUT_L_26	CPTS0_HW1_PUSH	CPTS0	Selectable timesync event 26	Level

**Table 11-15. TIMESYNC\_INTRTR0 Hardware Requests (continued)**

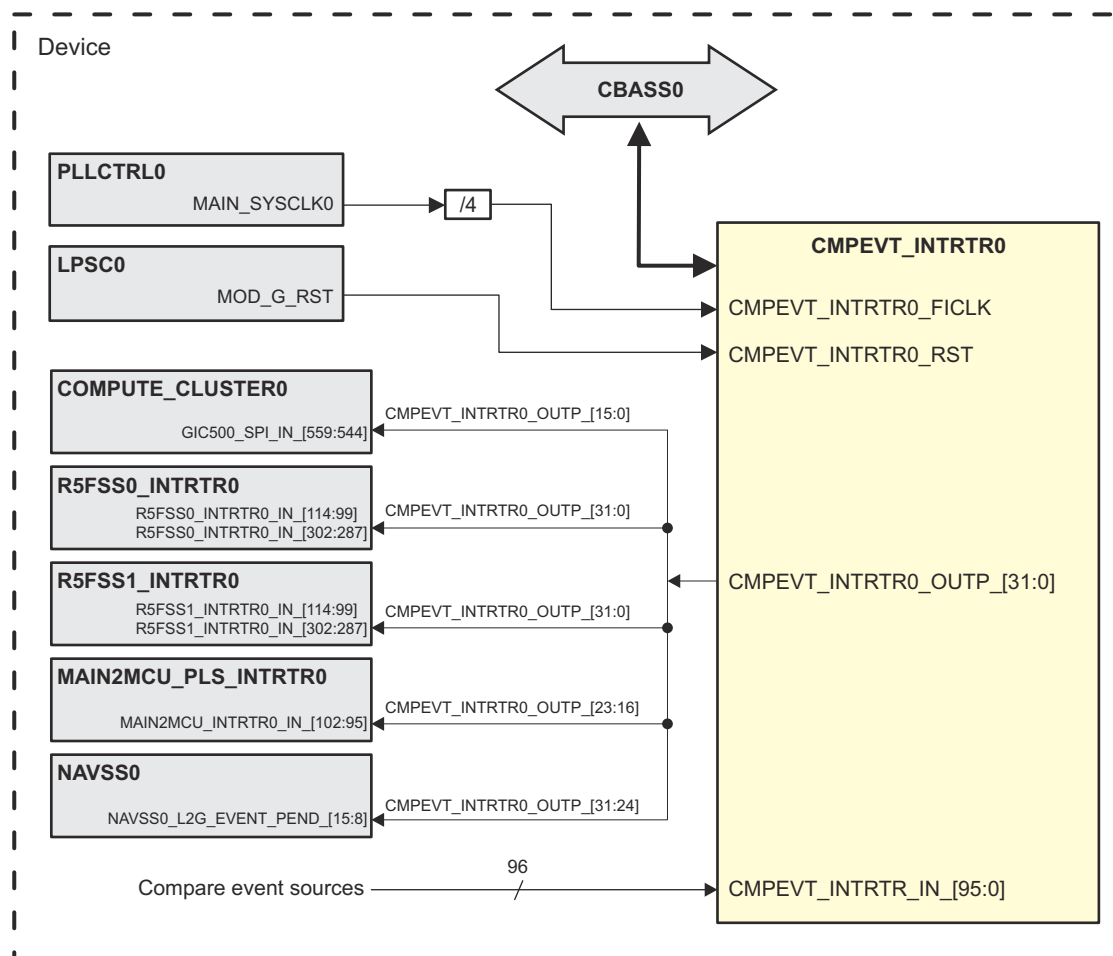
TIMESYNC_INTRTR0_OUT L_27	CPTS0_HW2_PUSH	CPTS0	Selectable timesync event 27	Level
TIMESYNC_INTRTR0_OUT L_28	CPTS0_HW3_PUSH	CPTS0	Selectable timesync event 28	Level
TIMESYNC_INTRTR0_OUT L_29	CPTS0_HW4_PUSH	CPTS0	Selectable timesync event 29	Level
TIMESYNC_INTRTR0_OUT L_30	CPTS0_HW5_PUSH	CPTS0	Selectable timesync event 30	Level
TIMESYNC_INTRTR0_OUT L_31	CPTS0_HW6_PUSH	CPTS0	Selectable timesync event 31	Level
TIMESYNC_INTRTR0_OUT L_32	CPTS0_HW7_PUSH	CPTS0	Selectable timesync event 32	Level
TIMESYNC_INTRTR0_OUT L_33	CPTS0_HW8_PUSH	CPTS0	Selectable timesync event 33	Level
TIMESYNC_INTRTR0_OUT L_34	SYNC0_OUT	Pin	Selectable timesync event 34	Level
TIMESYNC_INTRTR0_OUT L_35	SYNC1_OUT	Pin	Selectable timesync event 35	Level
TIMESYNC_INTRTR0_OUT L_36	SYNC2_OUT	Pin	Selectable timesync event 36	Level
TIMESYNC_INTRTR0_OUT L_37	SYNC3_OUT	Pin	Selectable timesync event 37	Level
TIMESYNC_INTRTR0_OUT L_40	NAVSS0_L2G_EVENT_PEND_0	NAVSS0	Selectable timesync event 40	Level
TIMESYNC_INTRTR0_OUT L_41	NAVSS0_L2G_EVENT_PEND_1	NAVSS0	Selectable timesync event 41	Level
TIMESYNC_INTRTR0_OUT L_42	NAVSS0_L2G_EVENT_PEND_2	NAVSS0	Selectable timesync event 42	Level
TIMESYNC_INTRTR0_OUT L_43	NAVSS0_L2G_EVENT_PEND_3	NAVSS0	Selectable timesync event 43	Level
TIMESYNC_INTRTR0_OUT L_44	NAVSS0_L2G_EVENT_PEND_4	NAVSS0	Selectable timesync event 44	Level
TIMESYNC_INTRTR0_OUT L_45	NAVSS0_L2G_EVENT_PEND_5	NAVSS0	Selectable timesync event 45	Level
TIMESYNC_INTRTR0_OUT L_46	NAVSS0_L2G_EVENT_PEND_6	NAVSS0	Selectable timesync event 46	Level
TIMESYNC_INTRTR0_OUT L_47	NAVSS0_L2G_EVENT_PEND_7	NAVSS0	Selectable timesync event 47	Level

**Module Time Sync Events (Inputs)**

Module Instance	Module Sync Input	Time Sync Event Sources
TIMESYNC_INTRTR0	TIMESYNC_INTRTR0_IN_[55:0]	See <a href="#">Table 11-32</a> for mapping of time sync events to TIMESYNC_INTRTR0 inputs

### 11.3.2.4 CMPEVT\_INTRTR0 Integration

Figure 11-11 shows the CMPEVT\_INTRTR0 integration.



**Figure 11-11. CMPEVT\_INTRTR0 Integration**

Table 11-16 through Table 11-18 summarize the CMPEVT\_INTRTR0 integration.

**Table 11-16. CMPEVT\_INTRTR0 Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
CMPEVT_INTRTR0	PSC0	PD0	LPSC0	CBASS0

**Table 11-17. CMPEVT\_INTRTR0 Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
CMPEVT_INTRTR0	CMPEVT_INTRTR0_FICLK	MAIN_SYSCCLK0/4	PLLCTRL0	CMPEVT_INTRTR0 functional and interface clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
CMPEVT_INTRTR0	CMPEVT_INTRTR0_RST	MOD_G_RST	LPSC0	CMPEVT_INTRTR0 hardware reset

**Table 11-18. CMPEVT\_INTRTR0 Hardware Requests**

Module Compare Events (Outputs)				
---------------------------------	--	--	--	--

**Table 11-18. CMPEVT\_INTRTR0 Hardware Requests (continued)**

Module Instance	Module Compare Event Output	Destination Input	Destination	Description	Type
CMPEVT_INTRTR0	CMPEVENT_INTRTR0_OU TP_0	GIC500_SPI_IN_544 R5FSS0_INTRTR0_IN_287 R5FSS1_INTRTR0_IN_287	COMPUTE_CLUSTER0 R5FSS0_INTRTR0 R5FSS1_INTRTR0	Selectable compare event 0	Pulse
	CMPEVENT_INTRTR0_OU TP_1	GIC500_SPI_IN_545 R5FSS0_INTRTR0_IN_288 R5FSS1_INTRTR0_IN_288	COMPUTE_CLUSTER0 R5FSS0_INTRTR0 R5FSS1_INTRTR0	Selectable compare event 1	Pulse
	CMPEVENT_INTRTR0_OU TP_2	GIC500_SPI_IN_546 R5FSS0_INTRTR0_IN_289 R5FSS1_INTRTR0_IN_289	COMPUTE_CLUSTER0 R5FSS0_INTRTR0 R5FSS1_INTRTR0	Selectable compare event 2	Pulse
	CMPEVENT_INTRTR0_OU TP_3	GIC500_SPI_IN_547 R5FSS0_INTRTR0_IN_290 R5FSS1_INTRTR0_IN_290	COMPUTE_CLUSTER0 R5FSS0_INTRTR0 R5FSS1_INTRTR0	Selectable compare event 3	Pulse
	CMPEVENT_INTRTR0_OU TP_4	GIC500_SPI_IN_548 R5FSS0_INTRTR0_IN_291 R5FSS1_INTRTR0_IN_291	COMPUTE_CLUSTER0 R5FSS0_INTRTR0 R5FSS1_INTRTR0	Selectable compare event 4	Pulse
	CMPEVENT_INTRTR0_OU TP_5	GIC500_SPI_IN_549 R5FSS0_INTRTR0_IN_292 R5FSS1_INTRTR0_IN_292	COMPUTE_CLUSTER0 R5FSS0_INTRTR0 R5FSS1_INTRTR0	Selectable compare event 5	Pulse
	CMPEVENT_INTRTR0_OU TP_6	GIC500_SPI_IN_550 R5FSS0_INTRTR0_IN_293 R5FSS1_INTRTR0_IN_293	COMPUTE_CLUSTER0 R5FSS0_INTRTR0 R5FSS1_INTRTR0	Selectable compare event 6	Pulse
	CMPEVENT_INTRTR0_OU TP_7	GIC500_SPI_IN_551 R5FSS0_INTRTR0_IN_294 R5FSS1_INTRTR0_IN_294	COMPUTE_CLUSTER0 R5FSS0_INTRTR0 R5FSS1_INTRTR0	Selectable compare event 7	Pulse
	CMPEVENT_INTRTR0_OU TP_8	GIC500_SPI_IN_552 R5FSS0_INTRTR0_IN_295 R5FSS1_INTRTR0_IN_295	COMPUTE_CLUSTER0 R5FSS0_INTRTR0 R5FSS1_INTRTR0	Selectable compare event 8	Pulse
	CMPEVENT_INTRTR0_OU TP_9	GIC500_SPI_IN_553 R5FSS0_INTRTR0_IN_296 R5FSS1_INTRTR0_IN_296	COMPUTE_CLUSTER0 R5FSS0_INTRTR0 R5FSS1_INTRTR0	Selectable compare event 9	Pulse
	CMPEVENT_INTRTR0_OU TP_10	GIC500_SPI_IN_554 R5FSS0_INTRTR0_IN_297 R5FSS1_INTRTR0_IN_297	COMPUTE_CLUSTER0 R5FSS0_INTRTR0 R5FSS1_INTRTR0	Selectable compare event 10	Pulse
	CMPEVENT_INTRTR0_OU TP_11	GIC500_SPI_IN_555 R5FSS0_INTRTR0_IN_298 R5FSS1_INTRTR0_IN_298	COMPUTE_CLUSTER0 R5FSS0_INTRTR0 R5FSS1_INTRTR0	Selectable compare event 11	Pulse
	CMPEVENT_INTRTR0_OU TP_12	GIC500_SPI_IN_556 R5FSS0_INTRTR0_IN_299 R5FSS1_INTRTR0_IN_299	COMPUTE_CLUSTER0 R5FSS0_INTRTR0 R5FSS1_INTRTR0	Selectable compare event 12	Pulse
	CMPEVENT_INTRTR0_OU TP_13	GIC500_SPI_IN_557 R5FSS0_INTRTR0_IN_300 R5FSS1_INTRTR0_IN_300	COMPUTE_CLUSTER0 R5FSS0_INTRTR0 R5FSS1_INTRTR0	Selectable compare event 13	Pulse

**Table 11-18. CMPEVT\_INTRTR0 Hardware Requests (continued)**

CMPEVENT_INTRTR0_OU TP_14	GIC500_SPI_IN_558 R5FSS0_INTRTR0_IN_301 R5FSS1_INTRTR0_IN_301	COMPUTE_CLUSTER0 R5FSS0_INTRTR0 R5FSS1_INTRTR0	Selectable compare event 14	Pulse
CMPEVENT_INTRTR0_OU TP_15	GIC500_SPI_IN_559 R5FSS0_INTRTR0_IN_302 R5FSS1_INTRTR0_IN_302	COMPUTE_CLUSTER0 R5FSS0_INTRTR0 R5FSS1_INTRTR0	Selectable compare event 15	Pulse
CMPEVENT_INTRTR0_OU TP_16	R5FSS0_INTRTR0_IN_99 R5FSS1_INTRTR0_IN_99 MAIN2MCU_PLS_INTRTR0_IN_95	R5FSS0_INTRTR0 R5FSS1_INTRTR0 MAIN2MCU_PLS_INTRTR0	Selectable compare event 16	Pulse
CMPEVENT_INTRTR0_OU TP_17	R5FSS0_INTRTR0_IN_100 R5FSS1_INTRTR0_IN_100 MAIN2MCU_PLS_INTRTR0_IN_96	R5FSS0_INTRTR0 R5FSS1_INTRTR0 MAIN2MCU_PLS_INTRTR0	Selectable compare event 17	Pulse
CMPEVENT_INTRTR0_OU TP_18	R5FSS0_INTRTR0_IN_101 R5FSS1_INTRTR0_IN_101 MAIN2MCU_PLS_INTRTR0_IN_97	R5FSS0_INTRTR0 R5FSS1_INTRTR0 MAIN2MCU_PLS_INTRTR0	Selectable compare event 18	Pulse
CMPEVENT_INTRTR0_OU TP_19	R5FSS0_INTRTR0_IN_102 R5FSS1_INTRTR0_IN_102 MAIN2MCU_PLS_INTRTR0_IN_98	R5FSS0_INTRTR0 R5FSS1_INTRTR0 MAIN2MCU_PLS_INTRTR0	Selectable compare event 19	Pulse
CMPEVENT_INTRTR0_OU TP_20	R5FSS0_INTRTR0_IN_103 R5FSS1_INTRTR0_IN_103 MAIN2MCU_PLS_INTRTR0_IN_99	R5FSS0_INTRTR0 R5FSS1_INTRTR0 MAIN2MCU_PLS_INTRTR0	Selectable compare event 20	Pulse
CMPEVENT_INTRTR0_OU TP_21	R5FSS0_INTRTR0_IN_104 R5FSS1_INTRTR0_IN_104 MAIN2MCU_PLS_INTRTR0_IN_100	R5FSS0_INTRTR0 R5FSS1_INTRTR0 MAIN2MCU_PLS_INTRTR0	Selectable compare event 21	Pulse
CMPEVENT_INTRTR0_OU TP_22	R5FSS0_INTRTR0_IN_105 R5FSS1_INTRTR0_IN_105 MAIN2MCU_PLS_INTRTR0_IN_101	R5FSS0_INTRTR0 R5FSS1_INTRTR0 MAIN2MCU_PLS_INTRTR0	Selectable compare event 22	Pulse
CMPEVENT_INTRTR0_OU TP_23	R5FSS0_INTRTR0_IN_106 R5FSS1_INTRTR0_IN_106 MAIN2MCU_PLS_INTRTR0_IN_102	R5FSS0_INTRTR0 R5FSS1_INTRTR0 MAIN2MCU_PLS_INTRTR0	Selectable compare event 23	Pulse
CMPEVENT_INTRTR0_OU TP_24	R5FSS0_INTRTR0_IN_107 R5FSS1_INTRTR0_IN_107 NAVSS0_L2G_EVENT_PEN_D_8	R5FSS0_INTRTR0 R5FSS1_INTRTR0 NAVSS0	Selectable compare event 24	Pulse
CMPEVENT_INTRTR0_OU TP_25	R5FSS0_INTRTR0_IN_108 R5FSS1_INTRTR0_IN_108 NAVSS0_L2G_EVENT_PEN_D_9	R5FSS0_INTRTR0 R5FSS1_INTRTR0 NAVSS0	Selectable compare event 25	Pulse
CMPEVENT_INTRTR0_OU TP_26	R5FSS0_INTRTR0_IN_109 R5FSS1_INTRTR0_IN_109 NAVSS0_L2G_EVENT_PEN_D_10	R5FSS0_INTRTR0 R5FSS1_INTRTR0 NAVSS0	Selectable compare event 26	Pulse
CMPEVENT_INTRTR0_OU TP_27	R5FSS0_INTRTR0_IN_110 R5FSS1_INTRTR0_IN_110 NAVSS0_L2G_EVENT_PEN_D_11	R5FSS0_INTRTR0 R5FSS1_INTRTR0 NAVSS0	Selectable compare event 27	Pulse
CMPEVENT_INTRTR0_OU TP_28	R5FSS0_INTRTR0_IN_111 R5FSS1_INTRTR0_IN_111 NAVSS0_L2G_EVENT_PEN_D_12	R5FSS0_INTRTR0 R5FSS1_INTRTR0 NAVSS0	Selectable compare event 28	Pulse

**Table 11-18. CMPEVT\_INTRTR0 Hardware Requests (continued)**

CMPEVENT_INTRTR0_OUTP_29	R5FSS0_INTRTR0_IN_112 R5FSS1_INTRTR0_IN_112 NAVSS0_L2G_EVENT_PEN D_13	R5FSS0_INTRTR0 R5FSS1_INTRTR0 NAVSS0	Selectable compare event 29	Pulse
CMPEVENT_INTRTR0_OUTP_30	R5FSS0_INTRTR0_IN_113 R5FSS1_INTRTR0_IN_113 NAVSS0_L2G_EVENT_PEN D_14	R5FSS0_INTRTR0 R5FSS1_INTRTR0 NAVSS0	Selectable compare event 30	Pulse
CMPEVENT_INTRTR0_OUTP_31	R5FSS0_INTRTR0_IN_114 R5FSS1_INTRTR0_IN_114 NAVSS0_L2G_EVENT_PEN D_15	R5FSS0_INTRTR0 R5FSS1_INTRTR0 NAVSS0	Selectable compare event 31	Pulse

Module Compare Events (Inputs)		
Module Instance	Module Compare Event Input	Compare Event Sources
CMPEVT_INTRTR0	CMPEVENT_INTRTR0_IN_[95:0]	See <a href="#">Table 11-31</a> for mapping of compare events to CMPEVT_INTRTR0 inputs

### 11.3.2.5 Time Sync Routers Registers

### 11.3.2.6 TIMESYNC\_INTRTR0 Registers

[Table 11-20](#) lists the memory-mapped registers for the TIMESYNC\_INTRTR0. All register offset addresses not listed in [Table 11-20](#) should be considered as reserved locations and the register contents should not be modified.

**Table 11-19. TIMESYNC\_INTRTR0 Instances**

Instance	Base Address
TIMESYNC_INTRTR0_INTR_ROUTER_CFG	00A4 0000h

**Table 11-20. TIMESYNC\_INTRTR0 Registers**

Offset	Acronym	Register Name	TIMESYNC_INTRTR0_INTR_ROUTER_CFG Physical Address
0h	<a href="#">TIMESYNC_INTRTR0_PID</a>	Peripheral identification register	00A4 0000h
4h + formula	<a href="#">TIMESYNC_INTRTR0_MUXCNTL_y</a>	Event mux control register	00A4 0004h + formula



### 11.3.2.7 TIMESYNC\_INTRTR0\_PID Register (Offset = 0h) [reset = 66947900h]

TIMESYNC\_INTRTR0\_PID is shown in [Figure 11-12](#) and described in [Table 11-22](#).

Description: Peripheral identification register. Uniquely identifies the module and its specific revision.

Return to [Summary Table](#).

**Table 11-21. TIMESYNC\_INTRTR0\_PID Instances**

Instance	Physical Address
TIMESYNC_INTRTR0_INTR_ROUTER_CFG	00A4 0000h

**Figure 11-12. TIMESYNC\_INTRTR0\_PID Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REV																															
R-66947900h																															

LEGEND: R = Read Only; -n = value after reset

**Table 11-22. TIMESYNC\_INTRTR0\_PID Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	REV	R	66947900h	TI internal data. Identifies revision of peripheral.

### 11.3.2.8 TIMESYNC\_INTRTR0\_MUXCNTL\_y Register (Offset = 4h + formula) [reset = X]

TIMESYNC\_INTRTR0\_MUXCNTL\_y is shown in [Figure 11-13](#) and described in [Table 11-24](#).

Description: Event mux control register.

Offset = 4h + (y \* 4h); where y = 0h to 27h.

Return to [Summary Table](#).

**Table 11-23. TIMESYNC\_INTRTR0\_MUXCNTL\_y  
Instances**

Instance	Physical Address
TIMESYNC_INTRTR0_INTR_ROUTER_CFG	00A4 0004h + formula

**Figure 11-13. TIMESYNC\_INTRTR0\_MUXCNTL\_y Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							INT_ENABLE
R-0h							R/W-0h
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				ENABLE			
R-0h				R/W-0h			

LEGEND: R/W = Read/Write; -n = value after reset

**Table 11-24. TIMESYNC\_INTRTR0\_MUXCNTL\_y Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reserved
16	INT_ENABLE	R/W	0h	Enable for event output N 0h = Disabled 1h = Enabled
15-7	RESERVED	R	0h	Reserved
6-0	ENABLE	R/W	0h	Mux control for event output N 0h = Select event input 0 (IN0) 1h = Select event input 1 (IN1) ... and so on. Note: Avoid programming this register when input events are active. This could lead to spurious asynchronous output toggles which may lead to unpredictable behavior.

### 11.3.2.9 CMPEVT\_INTRTR0 Registers

Table 11-26 lists the memory-mapped registers for the CMPEVT\_INTRTR0. All register offset addresses not listed in Table 11-26 should be considered as reserved locations and the register contents should not be modified.

**Table 11-25. CMPEVT\_INTRTR0 Instances**

Instance	Base Address
CMPEVENT_INTRTR0_INTR_ROUTER_CFG	00A3 0000h

**Table 11-26. CMPEVT\_INTRTR0 Registers**

Offset	Acronym	Register Name	CMPEVENT_INTRTR0_INTR_ROUTER_CFG Physical Address
0h	<a href="#">CMPEVT_INTRTR0_PID</a>	Peripheral identification register	00A3 0000h
4h + formula	<a href="#">CMPEVT_INTRTR0_MUXCNTL_y</a>	Event mux control register	00A3 0004h + formula

### 11.3.2.10 CMPEVT\_INTRTR0\_PID Register (Offset = 0h) [reset = 66947900h]

CMPEVT\_INTRTR0\_PID is shown in [Figure 11-14](#) and described in [Table 11-28](#).

Description: Peripheral identification register. Uniquely identifies the module and its specific revision.

Return to [Summary Table](#).

**Table 11-27. CMPEVT\_INTRTR0\_PID Instances**

Instance	Physical Address
CMPEVENT_INTRTR0_INTR_ROUTER_CFG	00A3 0000h

**Figure 11-14. CMPEVT\_INTRTR0\_PID Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REV																															
R-66947900h																															

LEGEND: R = Read Only; -n = value after reset

**Table 11-28. CMPEVT\_INTRTR0\_PID Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	REV	R	66947900h	TI internal data. Identifies revision of peripheral.

### 11.3.2.11 CMPEVT\_INTRTR0\_MUXCNTL\_y Register (Offset = 4h + formula) [reset = X]

CMPEVT\_INTRTR0\_MUXCNTL\_y is shown in [Figure 11-15](#) and described in [Table 11-30](#).

Description: Event mux control register.

Offset = 4h + (y \* 4h); where y = 0h to 1Fh.

Return to [Summary Table](#).

**Table 11-29. CMPEVT\_INTRTR0\_MUXCNTL\_y  
Instances**

Instance	Physical Address
CMPEVENT_INTRTR0_INTR_ROUTER_CFG	00A3 0004h + formula

**Figure 11-15. CMPEVT\_INTRTR0\_MUXCNTL\_y Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							INT_ENABLE
R-0h							R/W-0h
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				ENABLE			
R-0h				R/W-0h			

LEGEND: R/W = Read/Write; -n = value after reset

**Table 11-30. CMPEVT\_INTRTR0\_MUXCNTL\_y Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reserved
16	INT_ENABLE	R/W	0h	Enable for event output N 0h = Disabled 1h = Enabled
15-7	RESERVED	R	0h	Reserved
6-0	ENABLE	R/W	0h	Mux control for event output N 0h = Select event input 0 (IN0) 1h = Select event input 1 (IN1) ... and so on. Note: Avoid programming this register when input events are active. This could lead to spurious asynchronous output toggles which may lead to unpredictable behavior.

### 11.3.3 Time Sync Event Sources

#### 11.3.3.1 CMPEVT\_INTRTR0 Event Map

Table 11-31 shows the mapping of timesync event sources to CMPEVT\_INTRTR0 event inputs.

**Table 11-31. CMPEVT\_INTRTR0 Event Map**

Module Event Input	Event ID	Event Source
CMPEVENT_INTRTR0_IN_4	4	PCIE0_PCIE_CPTS_COMP_0
CMPEVENT_INTRTR0_IN_5	5	PCIE1_PCIE_CPTS_COMP_0
CMPEVENT_INTRTR0_IN_6	6	PCIE2_PCIE_CPTS_COMP_0
CMPEVENT_INTRTR0_IN_7	7	PCIE3_PCIE_CPTS_COMP_0
CMPEVENT_INTRTR0_IN_8	8	NAVSS0_CPTS0_COMP_0
CMPEVENT_INTRTR0_IN_9	9	CPSW0_CPTS_COMP_0
CMPEVENT_INTRTR0_IN_10	10	MCU_CPSW0_CPTS_COMP_0
CMPEVENT_INTRTR0_IN_16	16	PRU_ICSSG0_PR1_HOST_INTR_REQ_0
CMPEVENT_INTRTR0_IN_17	17	PRU_ICSSG0_PR1_HOST_INTR_REQ_1
CMPEVENT_INTRTR0_IN_18	18	PRU_ICSSG0_PR1_HOST_INTR_REQ_2
CMPEVENT_INTRTR0_IN_19	19	PRU_ICSSG0_PR1_HOST_INTR_REQ_3
CMPEVENT_INTRTR0_IN_20	20	PRU_ICSSG0_PR1_HOST_INTR_REQ_4
CMPEVENT_INTRTR0_IN_21	21	PRU_ICSSG0_PR1_HOST_INTR_REQ_5
CMPEVENT_INTRTR0_IN_22	22	PRU_ICSSG0_PR1_HOST_INTR_REQ_6
CMPEVENT_INTRTR0_IN_23	23	PRU_ICSSG0_PR1_HOST_INTR_REQ_7
CMPEVENT_INTRTR0_IN_24	24	PRU_ICSSG1_PR1_HOST_INTR_REQ_0
CMPEVENT_INTRTR0_IN_25	25	PRU_ICSSG1_PR1_HOST_INTR_REQ_1
CMPEVENT_INTRTR0_IN_26	26	PRU_ICSSG1_PR1_HOST_INTR_REQ_2
CMPEVENT_INTRTR0_IN_27	27	PRU_ICSSG1_PR1_HOST_INTR_REQ_3
CMPEVENT_INTRTR0_IN_28	28	PRU_ICSSG1_PR1_HOST_INTR_REQ_4
CMPEVENT_INTRTR0_IN_29	29	PRU_ICSSG1_PR1_HOST_INTR_REQ_5
CMPEVENT_INTRTR0_IN_30	30	PRU_ICSSG1_PR1_HOST_INTR_REQ_6
CMPEVENT_INTRTR0_IN_31	31	PRU_ICSSG1_PR1_HOST_INTR_REQ_7
CMPEVENT_INTRTR0_IN_32	32	PRU_ICSSG0_PR1_IEP0_CMP_INTR_REQ_0
CMPEVENT_INTRTR0_IN_33	33	PRU_ICSSG0_PR1_IEP0_CMP_INTR_REQ_1
CMPEVENT_INTRTR0_IN_34	34	PRU_ICSSG0_PR1_IEP0_CMP_INTR_REQ_2
CMPEVENT_INTRTR0_IN_35	35	PRU_ICSSG0_PR1_IEP0_CMP_INTR_REQ_3
CMPEVENT_INTRTR0_IN_36	36	PRU_ICSSG0_PR1_IEP0_CMP_INTR_REQ_4
CMPEVENT_INTRTR0_IN_37	37	PRU_ICSSG0_PR1_IEP0_CMP_INTR_REQ_5
CMPEVENT_INTRTR0_IN_38	38	PRU_ICSSG0_PR1_IEP0_CMP_INTR_REQ_6
CMPEVENT_INTRTR0_IN_39	39	PRU_ICSSG0_PR1_IEP0_CMP_INTR_REQ_7
CMPEVENT_INTRTR0_IN_40	40	PRU_ICSSG0_PR1_IEP0_CMP_INTR_REQ_8
CMPEVENT_INTRTR0_IN_41	41	PRU_ICSSG0_PR1_IEP0_CMP_INTR_REQ_9
CMPEVENT_INTRTR0_IN_42	42	PRU_ICSSG0_PR1_IEP0_CMP_INTR_REQ_10
CMPEVENT_INTRTR0_IN_43	43	PRU_ICSSG0_PR1_IEP0_CMP_INTR_REQ_11
CMPEVENT_INTRTR0_IN_44	44	PRU_ICSSG0_PR1_IEP0_CMP_INTR_REQ_12
CMPEVENT_INTRTR0_IN_45	45	PRU_ICSSG0_PR1_IEP0_CMP_INTR_REQ_13
CMPEVENT_INTRTR0_IN_46	46	PRU_ICSSG0_PR1_IEP0_CMP_INTR_REQ_14
CMPEVENT_INTRTR0_IN_47	47	PRU_ICSSG0_PR1_IEP0_CMP_INTR_REQ_15
CMPEVENT_INTRTR0_IN_48	48	PRU_ICSSG0_PR1_IEP1_CMP_INTR_REQ_0
CMPEVENT_INTRTR0_IN_49	49	PRU_ICSSG0_PR1_IEP1_CMP_INTR_REQ_1

**Table 11-31. CMPEVT\_INTRTR0 Event Map (continued)**

Module Event Input	Event ID	Event Source
CMPEVENT_INTRTR0_IN_50	50	PRU_ICSSG0_PR1_IEP1_CMP_INTR_REQ_2
CMPEVENT_INTRTR0_IN_51	51	PRU_ICSSG0_PR1_IEP1_CMP_INTR_REQ_3
CMPEVENT_INTRTR0_IN_52	52	PRU_ICSSG0_PR1_IEP1_CMP_INTR_REQ_4
CMPEVENT_INTRTR0_IN_53	53	PRU_ICSSG0_PR1_IEP1_CMP_INTR_REQ_5
CMPEVENT_INTRTR0_IN_54	54	PRU_ICSSG0_PR1_IEP1_CMP_INTR_REQ_6
CMPEVENT_INTRTR0_IN_55	55	PRU_ICSSG0_PR1_IEP1_CMP_INTR_REQ_7
CMPEVENT_INTRTR0_IN_56	56	PRU_ICSSG0_PR1_IEP1_CMP_INTR_REQ_8
CMPEVENT_INTRTR0_IN_57	57	PRU_ICSSG0_PR1_IEP1_CMP_INTR_REQ_9
CMPEVENT_INTRTR0_IN_58	58	PRU_ICSSG0_PR1_IEP1_CMP_INTR_REQ_10
CMPEVENT_INTRTR0_IN_59	59	PRU_ICSSG0_PR1_IEP1_CMP_INTR_REQ_11
CMPEVENT_INTRTR0_IN_60	60	PRU_ICSSG0_PR1_IEP1_CMP_INTR_REQ_12
CMPEVENT_INTRTR0_IN_61	61	PRU_ICSSG0_PR1_IEP1_CMP_INTR_REQ_13
CMPEVENT_INTRTR0_IN_62	62	PRU_ICSSG0_PR1_IEP1_CMP_INTR_REQ_14
CMPEVENT_INTRTR0_IN_63	63	PRU_ICSSG0_PR1_IEP1_CMP_INTR_REQ_15
CMPEVENT_INTRTR0_IN_64	64	PRU_ICSSG1_PR1_IEP0_CMP_INTR_REQ_0
CMPEVENT_INTRTR0_IN_65	65	PRU_ICSSG1_PR1_IEP0_CMP_INTR_REQ_1
CMPEVENT_INTRTR0_IN_66	66	PRU_ICSSG1_PR1_IEP0_CMP_INTR_REQ_2
CMPEVENT_INTRTR0_IN_67	67	PRU_ICSSG1_PR1_IEP0_CMP_INTR_REQ_3
CMPEVENT_INTRTR0_IN_68	68	PRU_ICSSG1_PR1_IEP0_CMP_INTR_REQ_4
CMPEVENT_INTRTR0_IN_69	69	PRU_ICSSG1_PR1_IEP0_CMP_INTR_REQ_5
CMPEVENT_INTRTR0_IN_70	70	PRU_ICSSG1_PR1_IEP0_CMP_INTR_REQ_6
CMPEVENT_INTRTR0_IN_71	71	PRU_ICSSG1_PR1_IEP0_CMP_INTR_REQ_7
CMPEVENT_INTRTR0_IN_72	72	PRU_ICSSG1_PR1_IEP0_CMP_INTR_REQ_8
CMPEVENT_INTRTR0_IN_73	73	PRU_ICSSG1_PR1_IEP0_CMP_INTR_REQ_9
CMPEVENT_INTRTR0_IN_74	74	PRU_ICSSG1_PR1_IEP0_CMP_INTR_REQ_10
CMPEVENT_INTRTR0_IN_75	75	PRU_ICSSG1_PR1_IEP0_CMP_INTR_REQ_11
CMPEVENT_INTRTR0_IN_76	76	PRU_ICSSG1_PR1_IEP0_CMP_INTR_REQ_12
CMPEVENT_INTRTR0_IN_77	77	PRU_ICSSG1_PR1_IEP0_CMP_INTR_REQ_13
CMPEVENT_INTRTR0_IN_78	78	PRU_ICSSG1_PR1_IEP0_CMP_INTR_REQ_14
CMPEVENT_INTRTR0_IN_79	79	PRU_ICSSG1_PR1_IEP0_CMP_INTR_REQ_15
CMPEVENT_INTRTR0_IN_80	80	PRU_ICSSG1_PR1_IEP1_CMP_INTR_REQ_0
CMPEVENT_INTRTR0_IN_81	81	PRU_ICSSG1_PR1_IEP1_CMP_INTR_REQ_1
CMPEVENT_INTRTR0_IN_82	82	PRU_ICSSG1_PR1_IEP1_CMP_INTR_REQ_2
CMPEVENT_INTRTR0_IN_83	83	PRU_ICSSG1_PR1_IEP1_CMP_INTR_REQ_3
CMPEVENT_INTRTR0_IN_84	84	PRU_ICSSG1_PR1_IEP1_CMP_INTR_REQ_4
CMPEVENT_INTRTR0_IN_85	85	PRU_ICSSG1_PR1_IEP1_CMP_INTR_REQ_5
CMPEVENT_INTRTR0_IN_86	86	PRU_ICSSG1_PR1_IEP1_CMP_INTR_REQ_6
CMPEVENT_INTRTR0_IN_87	87	PRU_ICSSG1_PR1_IEP1_CMP_INTR_REQ_7
CMPEVENT_INTRTR0_IN_88	88	PRU_ICSSG1_PR1_IEP1_CMP_INTR_REQ_8
CMPEVENT_INTRTR0_IN_89	89	PRU_ICSSG1_PR1_IEP1_CMP_INTR_REQ_9
CMPEVENT_INTRTR0_IN_90	90	PRU_ICSSG1_PR1_IEP1_CMP_INTR_REQ_10
CMPEVENT_INTRTR0_IN_91	91	PRU_ICSSG1_PR1_IEP1_CMP_INTR_REQ_11
CMPEVENT_INTRTR0_IN_92	92	PRU_ICSSG1_PR1_IEP1_CMP_INTR_REQ_12
CMPEVENT_INTRTR0_IN_93	93	PRU_ICSSG1_PR1_IEP1_CMP_INTR_REQ_13
CMPEVENT_INTRTR0_IN_94	94	PRU_ICSSG1_PR1_IEP1_CMP_INTR_REQ_14

**Table 11-31. CMPEVT\_INTRTR0 Event Map (continued)**

Module Event Input	Event ID	Event Source
CMPEVENT_INTRTR0_IN_95	95	PRU_ICSSG1_PR1_IEP1_CMP_INTR_REQ_15



### 11.3.3.2 TIMESYNC\_INTRTR0 Event Map

Table 11-32 shows the mapping of timesync event sources to TIMESYNC\_INTRTR0 event inputs.

**Table 11-32. TIMESYNC\_INTRTR0 Event Map**

Module Event Input	Event ID	Event Source
TIMESYNC_INTRTR0_IN_1	1	GTC0_GTC_PUSH_EVENT_0
TIMESYNC_INTRTR0_IN_2	2	TIMER14_TIMER_PWM_0
TIMESYNC_INTRTR0_IN_3	3	TIMER15_TIMER_PWM_0
TIMESYNC_INTRTR0_IN_4	4	NAVSS0_CPTS0_GENF0_0
TIMESYNC_INTRTR0_IN_5	5	NAVSS0_CPTS0_GENF1_0
TIMESYNC_INTRTR0_IN_6	6	NAVSS0_CPTS0_GENF2_0
TIMESYNC_INTRTR0_IN_7	7	NAVSS0_CPTS0_GENF3_0
TIMESYNC_INTRTR0_IN_8	8	NAVSS0_CPTS0_GENF4_0
TIMESYNC_INTRTR0_IN_9	9	NAVSS0_CPTS0_GENF5_0
TIMESYNC_INTRTR0_IN_10	10	PCIE0_PCIE_CPTS_GENF0_0
TIMESYNC_INTRTR0_IN_11	11	PCIE1_PCIE_CPTS_GENF0_0
TIMESYNC_INTRTR0_IN_12	12	PCIE2_PCIE_CPTS_GENF0_0
TIMESYNC_INTRTR0_IN_13	13	PCIE3_PCIE_CPTS_GENF0_0
TIMESYNC_INTRTR0_IN_14	14	CPSW0_CPTS_GENF0_0
TIMESYNC_INTRTR0_IN_15	15	CPSW0_CPTS_GENF1_0
TIMESYNC_INTRTR0_IN_16	16	MCU_CPSW0_CPTS_GENF0_0
TIMESYNC_INTRTR0_IN_17	17	MCU_CPSW0_CPTS_GENF1_0
TIMESYNC_INTRTR0_IN_20	20	PCIE0_PCIE_CPTS_HW1_PUSH_0
TIMESYNC_INTRTR0_IN_21	21	PCIE1_PCIE_CPTS_HW1_PUSH_0
TIMESYNC_INTRTR0_IN_22	22	PCIE2_PCIE_CPTS_HW1_PUSH_0
TIMESYNC_INTRTR0_IN_23	23	PCIE3_PCIE_CPTS_HW1_PUSH_0
TIMESYNC_INTRTR0_IN_24	24	PRU_ICSSG0_PR1_EDC0_SYNC0_OUT_0
TIMESYNC_INTRTR0_IN_25	25	PRU_ICSSG0_PR1_EDC0_SYNC1_OUT_0
TIMESYNC_INTRTR0_IN_26	26	PRU_ICSSG0_PR1_EDC1_SYNC0_OUT_0
TIMESYNC_INTRTR0_IN_27	27	PRU_ICSSG0_PR1_EDC1_SYNC1_OUT_0
TIMESYNC_INTRTR0_IN_28	28	PRU_ICSSG1_PR1_EDC0_SYNC0_OUT_0
TIMESYNC_INTRTR0_IN_29	29	PRU_ICSSG1_PR1_EDC0_SYNC1_OUT_0
TIMESYNC_INTRTR0_IN_30	30	PRU_ICSSG1_PR1_EDC1_SYNC0_OUT_0
TIMESYNC_INTRTR0_IN_31	31	PRU_ICSSG1_PR1_EDC1_SYNC1_OUT_0
TIMESYNC_INTRTR0_IN_32	32	PCIE0_PCIE_CPTS_SYNC_0
TIMESYNC_INTRTR0_IN_33	33	PCIE1_PCIE_CPTS_SYNC_0
TIMESYNC_INTRTR0_IN_34	34	PCIE2_PCIE_CPTS_SYNC_0
TIMESYNC_INTRTR0_IN_35	35	PCIE3_PCIE_CPTS_SYNC_0
TIMESYNC_INTRTR0_IN_36	36	NAVSS0_CPTS0_SYNC_0
TIMESYNC_INTRTR0_IN_37	37	CPSW0_CPTS_SYNC_0
TIMESYNC_INTRTR0_IN_38	38	MCU_CPSW0_CPTS_SYNC_0
TIMESYNC_INTRTR0_IN_40	40	TIMER16_TIMER_PWM_0
TIMESYNC_INTRTR0_IN_41	41	TIMER17_TIMER_PWM_0
TIMESYNC_INTRTR0_IN_42	42	TIMER18_TIMER_PWM_0
TIMESYNC_INTRTR0_IN_43	43	TIMER19_TIMER_PWM_0
TIMESYNC_INTRTR0_IN_44	44	PINFUNCTION_CPTS0_HW1TSPUSHIN_CPTS0_HW1TSPUSH_0
TIMESYNC_INTRTR0_IN_45	45	PINFUNCTION_CPTS0_HW2TSPUSHIN_CPTS0_HW2TSPUSH_0

**Table 11-32. TIMESYNC\_INTRTR0 Event Map (continued)**

Module Event Input	Event ID	Event Source
TIMESYNC_INTRTR0_IN_46	46	PINFUNCTION_MCU_CPTS0_HW1TSPUSHIN_MCU_CPTS0_HW1TSPUSH_0
TIMESYNC_INTRTR0_IN_47	47	PINFUNCTION_MCU_CPTS0_HW2TSPUSHIN_MCU_CPTS0_HW2TSPUSH_0
TIMESYNC_INTRTR0_IN_48	48	PINFUNCTION_PRG0_IEP0_EDC_LATCH_IN0IN_PRG0_IEP0_EDC_LATCH_IN0_0
TIMESYNC_INTRTR0_IN_49	49	PINFUNCTION_PRG0_IEP0_EDC_LATCH_IN1IN_PRG0_IEP0_EDC_LATCH_IN1_0
TIMESYNC_INTRTR0_IN_50	50	PINFUNCTION_PRG0_IEP1_EDC_LATCH_IN0IN_PRG0_IEP1_EDC_LATCH_IN0_0
TIMESYNC_INTRTR0_IN_51	51	PINFUNCTION_PRG0_IEP1_EDC_LATCH_IN1IN_PRG0_IEP1_EDC_LATCH_IN1_0
TIMESYNC_INTRTR0_IN_52	52	PINFUNCTION_PRG1_IEP0_EDC_LATCH_IN0IN_PRG1_IEP0_EDC_LATCH_IN0_0
TIMESYNC_INTRTR0_IN_53	53	PINFUNCTION_PRG1_IEP0_EDC_LATCH_IN1IN_PRG1_IEP0_EDC_LATCH_IN1_0
TIMESYNC_INTRTR0_IN_54	54	PINFUNCTION_PRG1_IEP1_EDC_LATCH_IN0IN_PRG1_IEP1_EDC_LATCH_IN0_0
TIMESYNC_INTRTR0_IN_55	55	PINFUNCTION_PRG1_IEP1_EDC_LATCH_IN1IN_PRG1_IEP1_EDC_LATCH_IN1_0

### 11.3.3.3 PRU\_ICSSG0 Sync Event Map

Table 11-33 shows the mapping of timesync event sources to PRU\_ICSSG0 latch inputs.

**Table 11-33. PRU\_ICSSG0 Sync Event Map**

Module Event Input	Event Source	Description	Type
PRU_ICSSG0_PR1_EDC0_LATCH0_IN	TIMESYNC_INTRTR0_OUTL_8	TIMESYNC_INTRTR0 selectable timesync event 8	Level
PRU_ICSSG0_PR1_EDC0_LATCH1_IN	TIMESYNC_INTRTR0_OUTL_9	TIMESYNC_INTRTR0 selectable timesync event 9	Level
PRU_ICSSG0_PR1_EDC1_LATCH0_IN	TIMESYNC_INTRTR0_OUTL_10	TIMESYNC_INTRTR0 selectable timesync event 10	Level
PRU_ICSSG0_PR1_EDC1_LATCH1_IN	TIMESYNC_INTRTR0_OUTL_11	TIMESYNC_INTRTR0 selectable timesync event 11	Level

### 11.3.3.4 PRU\_ICSSG1 Sync Event Map

Table 11-34 shows the mapping of timesync event sources to PRU\_ICSSG1 latch inputs.

**Table 11-34. PRU\_ICSSG1 Sync Event Map**

Module Event Input	Event Source	Description	Type
PRU_ICSSG1_PR1_EDC0_LATCH0_IN	TIMESYNC_INTRTR0_OUTL_12	TIMESYNC_INTRTR0 selectable timesync event 12	Level
PRU_ICSSG1_PR1_EDC0_LATCH1_IN	TIMESYNC_INTRTR0_OUTL_13	TIMESYNC_INTRTR0 selectable timesync event 13	Level
PRU_ICSSG1_PR1_EDC1_LATCH0_IN	TIMESYNC_INTRTR0_OUTL_14	TIMESYNC_INTRTR0 selectable timesync event 14	Level
PRU_ICSSG1_PR1_EDC1_LATCH1_IN	TIMESYNC_INTRTR0_OUTL_15	TIMESYNC_INTRTR0 selectable timesync event 15	Level

### 11.3.3.5 NAVSS0 Sync Event Map

Table 11-35 shows the mapping of timesync and compare event sources to NAVSS0\_CPTS0 hardware push and local-to-global event inputs.

**Table 11-35. NAVSS0 Sync Event Map**

Module Event Input	Event Source	Description	Type
NAVSS0_CPTS0_HW1_PUSH	TIMESYNC_INTRTR0_OUTL_0	TIMESYNC_INTRTR0 selectable timesync event 0	Level
NAVSS0_CPTS0_HW2_PUSH	TIMESYNC_INTRTR0_OUTL_1	TIMESYNC_INTRTR0 selectable timesync event 1	Level
NAVSS0_CPTS0_HW3_PUSH	TIMESYNC_INTRTR0_OUTL_2	TIMESYNC_INTRTR0 selectable timesync event 2	Level
NAVSS0_CPTS0_HW4_PUSH	TIMESYNC_INTRTR0_OUTL_3	TIMESYNC_INTRTR0 selectable timesync event 3	Level
NAVSS0_CPTS0_HW5_PUSH	TIMESYNC_INTRTR0_OUTL_4	TIMESYNC_INTRTR0 selectable timesync event 4	Level
NAVSS0_CPTS0_HW6_PUSH	TIMESYNC_INTRTR0_OUTL_5	TIMESYNC_INTRTR0 selectable timesync event 5	Level
NAVSS0_CPTS0_HW7_PUSH	TIMESYNC_INTRTR0_OUTL_6	TIMESYNC_INTRTR0 selectable timesync event 6	Level
NAVSS0_CPTS0_HW8_PUSH	TIMESYNC_INTRTR0_OUTL_7	TIMESYNC_INTRTR0 selectable timesync event 7	Level
NAVSS0_L2G_EVENT_PEND_0	TIMESYNC_INTRTR0_OUTL_40	TIMESYNC_INTRTR0 selectable timesync event 40	Level
NAVSS0_L2G_EVENT_PEND_1	TIMESYNC_INTRTR0_OUTL_41	TIMESYNC_INTRTR0 selectable timesync event 41	Level
NAVSS0_L2G_EVENT_PEND_2	TIMESYNC_INTRTR0_OUTL_42	TIMESYNC_INTRTR0 selectable timesync event 42	Level
NAVSS0_L2G_EVENT_PEND_3	TIMESYNC_INTRTR0_OUTL_43	TIMESYNC_INTRTR0 selectable timesync event 43	Level
NAVSS0_L2G_EVENT_PEND_4	TIMESYNC_INTRTR0_OUTL_44	TIMESYNC_INTRTR0 selectable timesync event 44	Level
NAVSS0_L2G_EVENT_PEND_5	TIMESYNC_INTRTR0_OUTL_45	TIMESYNC_INTRTR0 selectable timesync event 45	Level
NAVSS0_L2G_EVENT_PEND_6	TIMESYNC_INTRTR0_OUTL_46	TIMESYNC_INTRTR0 selectable timesync event 46	Level
NAVSS0_L2G_EVENT_PEND_7	TIMESYNC_INTRTR0_OUTL_47	TIMESYNC_INTRTR0 selectable timesync event 47	Level
NAVSS0_L2G_EVENT_PEND_8	CMPEVT_INTRTR0_OUTP_24	CMPEVT_INTRTR0 selectable compare event 24	Pulse
NAVSS0_L2G_EVENT_PEND_9	CMPEVT_INTRTR0_OUTP_25	CMPEVT_INTRTR0 selectable compare event 25	Pulse
NAVSS0_L2G_EVENT_PEND_10	CMPEVT_INTRTR0_OUTP_26	CMPEVT_INTRTR0 selectable compare event 26	Pulse
NAVSS0_L2G_EVENT_PEND_11	CMPEVT_INTRTR0_OUTP_27	CMPEVT_INTRTR0 selectable compare event 27	Pulse
NAVSS0_L2G_EVENT_PEND_12	CMPEVT_INTRTR0_OUTP_28	CMPEVT_INTRTR0 selectable compare event 28	Pulse
NAVSS0_L2G_EVENT_PEND_13	CMPEVT_INTRTR0_OUTP_29	CMPEVT_INTRTR0 selectable compare event 29	Pulse
NAVSS0_L2G_EVENT_PEND_14	CMPEVT_INTRTR0_OUTP_30	CMPEVT_INTRTR0 selectable compare event 30	Pulse
NAVSS0_L2G_EVENT_PEND_15	CMPEVT_INTRTR0_OUTP_31	CMPEVT_INTRTR0 selectable compare event 31	Pulse

### 11.3.3.6 PCIE0 Sync Event Map

Table 11-36 shows the mapping of timesync event sources to PCIE0\_CPTS0 hardware push inputs.

**Table 11-36. PCIE0 Sync Event Map**

Module Event Input	Event Source	Description	Type
PCIE0_CPTS0_HW1_PUSH	PCIE0_PTM0_CORR_TIME_EVT_0	PCIE0_PTM0 corrected time event	–
PCIE0_CPTS0_HW2_PUSH	TIMESYNC_INTRTR0_OUTL_20	TIMESYNC_INTRTR0 selectable timesync event 20	Level

### 11.3.3.7 PCIE1 Sync Event Map

Table 11-37 shows the mapping of timesync event sources to PCIE1\_CPTS0 hardware push inputs.

**Table 11-37. PCIE1 Sync Event Map**

Module Event Input	Event Source	Description	Type
PCIE1_CPTS0_HW1_PUSH	PCIE1_PTM0_CORR_TIME_EVT_0	PCIE1_PTM0 corrected time event	–
PCIE1_CPTS0_HW2_PUSH	TIMESYNC_INTRTR0_OUTL_21	TIMESYNC_INTRTR0 selectable timesync event 21	Level

### 11.3.3.8 PCIE2 Sync Event Map

Table 11-38 shows the mapping of timesync event sources to PCIE2\_CPTS0 hardware push inputs.

**Table 11-38. PCIE2 Sync Event Map**

Module Event Input	Event Source	Description	Type
PCIE2_CPTS0_HW1_PUSH	PCIE2_PTM0_CORR_TIME_EVT_0	PCIE2_PTM0 corrected time event	–
PCIE2_CPTS0_HW2_PUSH	TIMESYNC_INTRTR0_OUTL_22	TIMESYNC_INTRTR0 selectable timesync event 21	Level

### 11.3.3.9 PCIE3 Sync Event Map

Table 11-39 shows the mapping of timesync event sources to PCIE3\_CPTS0 hardware push inputs.

**Table 11-39. PCIE3 Sync Event Map**

Module Event Input	Event Source	Description	Type
PCIE3_CPTS0_HW1_PUSH	PCIE3_PTM0_CORR_TIME_EVT_0	PCIE3_PTM0 corrected time event	–
PCIE3_CPTS0_HW2_PUSH	TIMESYNC_INTRTR0_OUTL_23	TIMESYNC_INTRTR0 selectable timesync event 21	Level

### 11.3.3.10 MCU\_CPSW0 Sync Event Map

Table 11-40 shows the mapping of timesync event sources to MCU\_CPSW0\_CPTS0 hardware push inputs.

**Table 11-40. MCU\_CPSW0 Sync Event Map**

Module Event Input	Event Source	Description	Type
MCU_CPSW0_CPTS0_HW1_PUSH	MCU_CPTS0_HW1TSPUSH	Timesync event from I/O pin	–
MCU_CPSW0_CPTS0_HW2_PUSH	MCU_CPTS0_HW2TSPUSH	Timesync event from I/O pin	–
MCU_CPSW0_CPTS0_HW3_PUSH	TIMESYNC_INTRTR0_OUTL_24	TIMESYNC_INTRTR0 selectable timesync event 24	Level
MCU_CPSW0_CPTS0_HW4_PUSH	TIMESYNC_INTRTR0_OUTL_25	TIMESYNC_INTRTR0 selectable timesync event 25	Level

### 11.3.3.11 CPSW0 Sync Event Map

Table 11-41 shows the mapping of timesync event sources to CPSW0\_CPTS0 hardware push inputs.

**Table 11-41. CPSW0 Sync Event Map**

Module Event Input	Event Source	Description	Type
CPTS0_HW1_PUSH	TIMESYNC_INTRTR0_OUTL_26	TIMESYNC_INTRTR0 selectable timesync event 26	Level
CPTS0_HW2_PUSH	TIMESYNC_INTRTR0_OUTL_27	TIMESYNC_INTRTR0 selectable timesync event 27	Level
CPTS0_HW3_PUSH	TIMESYNC_INTRTR0_OUTL_28	TIMESYNC_INTRTR0 selectable timesync event 28	Level
CPTS0_HW4_PUSH	TIMESYNC_INTRTR0_OUTL_29	TIMESYNC_INTRTR0 selectable timesync event 29	Level
CPTS0_HW5_PUSH	TIMESYNC_INTRTR0_OUTL_30	TIMESYNC_INTRTR0 selectable timesync event 30	Level
CPTS0_HW6_PUSH	TIMESYNC_INTRTR0_OUTL_31	TIMESYNC_INTRTR0 selectable timesync event 31	Level
CPTS0_HW7_PUSH	TIMESYNC_INTRTR0_OUTL_32	TIMESYNC_INTRTR0 selectable timesync event 32	Level
CPTS0_HW8_PUSH	TIMESYNC_INTRTR0_OUTL_33	TIMESYNC_INTRTR0 selectable timesync event 33	Level

### 11.3.3.12 I/O Sync Event Map

[Table 11-42](#) shows the mapping of timesync event sources to SYNC[3:0] device output pins.

**Table 11-42. I/O Sync Event Map**

Pin Event	Event Source	Description	Type
SYNC0_OUT	TIMESYNC_INTRTR0_OUTL_34	TIMESYNC_INTRTR0 selectable timesync event 34	Level
SYNC1_OUT	TIMESYNC_INTRTR0_OUTL_35	TIMESYNC_INTRTR0 selectable timesync event 35	Level
SYNC2_OUT	TIMESYNC_INTRTR0_OUTL_36	TIMESYNC_INTRTR0 selectable timesync event 36	Level
SYNC3_OUT	TIMESYNC_INTRTR0_OUTL_37	TIMESYNC_INTRTR0 selectable timesync event 37	Level



<b>12.1 General Connectivity Peripherals</b>	<b>1367</b>
<b>12.2 High-speed Serial Interfaces</b>	<b>1614</b>
<b>12.3 Memory Interfaces</b>	<b>1920</b>
<b>12.4 Industrial and Control Interfaces</b>	<b>2212</b>
<b>12.5 Audio Interfaces</b>	<b>2435</b>
<b>12.6 Display Subsystem (DSS) and Peripherals</b>	<b>2513</b>
<b>12.7 Camera Subsystem</b>	<b>3254</b>
<b>12.8 Shared MIPI D-PHY Transmitter (DPHY_TX)</b>	<b>3317</b>
<b>12.9 Video Processing Front End (VPFE)</b>	<b>3324</b>
<b>12.10 Timer Modules</b>	<b>3354</b>
<b>12.11 Internal Diagnostics Modules</b>	<b>3430</b>



## 12.1 General Connectivity Peripherals

This section describes the general connectivity peripherals in the device.

### 12.1.1 Analog-to-Digital Converter (ADC)

This chapter describes the Analog-to-Digital Converter (ADC) in the device.

#### 12.1.1.1 ADC Overview

The Analog-to-Digital Converter (ADC) module contains a single 12-bit ADC which can be multiplexed to any 1 of 8 analog inputs (channels).

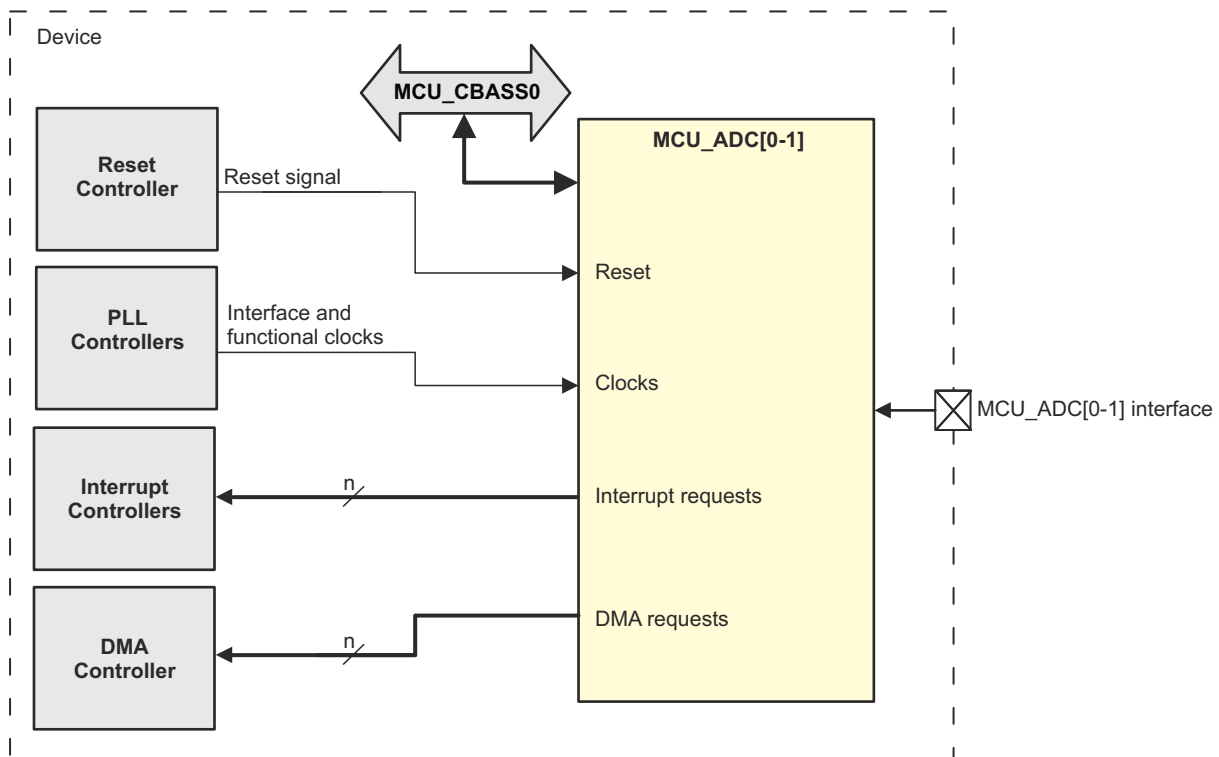
There are two ADC modules in the device.

Table 12-1 shows ADC modules allocation across device domains.

**Table 12-1. ADC Allocation Across Device Domains**

Instance	Domain		
	WKUP	MCU	MAIN
MCU_ADC0	-	✓	-
MCU_ADC1	-	✓	-

Figure 12-1 presents an overview of the ADC modules.



adc-001

**Figure 12-1. ADC Modules Overview**

#### 12.1.1.1.1 ADC Features

Each ADC module has the following features:

- 4 MSPS rate with a 60 MHz SMPL\_CLK

- Functional Internal Diagnostic Debug Mode
- Single-ended or differential input options
- Each ADC module can be configured and transformed into a digital test inputs
- Programmable Finite State Machine (FSM) sequencer that supports the following:
  - Software initiated start of conversion
  - Optional hardware start of conversion (SOC), synchronized to external hardware event
  - Single conversion (one-shot mode)
  - Continuous conversions (continuous mode)
  - Sequence through all enabled steps based on a mask
  - Programmable open delay before executing each step
  - Programmable sampling delay for each step
  - Programmable averaging (16, 8, 4, 2, or 1) of input samples for each step
  - Store data in either of two FIFOs – 256-word × 16-bit
  - Option to encode input (channel) number with data
  - Support for servicing FIFOs via DMA or processor
  - Programmable DMA request event (for each FIFO)
- Support for the following interrupts and status, with masking:
  - Interrupt if AFE fails to return end of conversion (EOC)
  - Interrupt after a sequence of conversions (all non-masked steps)
  - Interrupt for FIFO threshold levels
  - Interrupt if sampled data is out of a programmable range
  - Interrupt for FIFO overflow and underflow conditions
  - Status bit to indicate if ADC is busy converting

#### **12.1.1.1.2 ADC Not Supported Features**

- No packing of 16-bit FIFO data onto 32-bit DMA bus
- Big endian
- Support only 32-bit aligned read/write accesses on the MCU\_CBASS0/DMA ports
- Simultaneous sampling

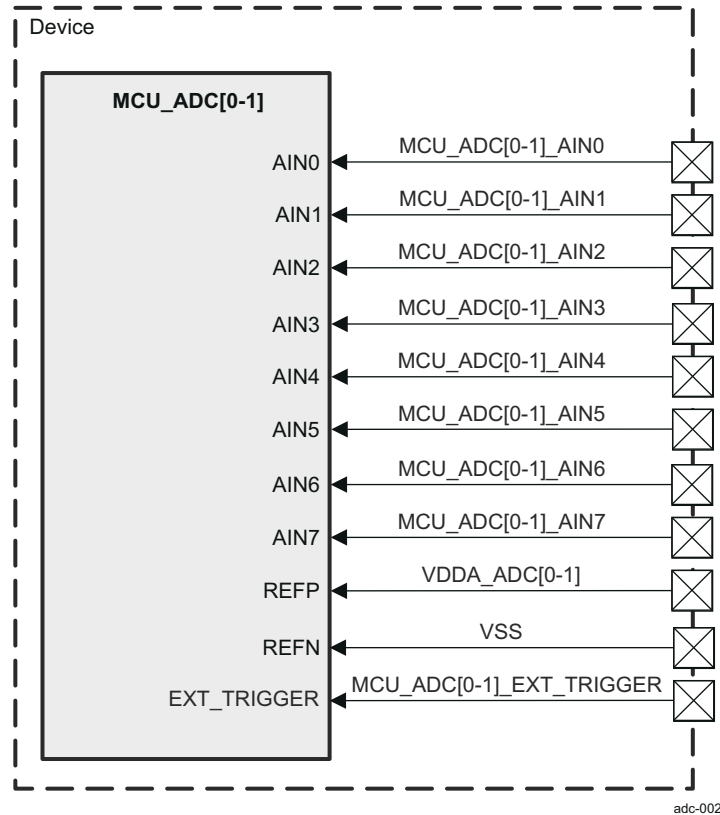
### 12.1.1.2 ADC Environment

The MCU\_ADC[0-1] modules are hereinafter referred to as ADC module.

This section describes the ADC external connections (environment).

#### 12.1.1.2.1 ADC Interface Signals

Figure 12-2 shows the ADC signals.



**Figure 12-2. ADC Signals**

Table 12-2 describes the ADC I/O signals.

**Table 12-2. ADC I/O Signals**

Module Pin	Device Level Signal	I/O <sup>(1)</sup>	Description	Module Pin Reset Value <sup>(2)</sup>
<b>MCU_ADC[0-1]</b>				
AIN0	MCU_ADC[0-1]_AIN0	A/I	Analog input 0	HiZ
AIN1	MCU_ADC[0-1]_AIN1	A/I	Analog input 1	HiZ
AIN2	MCU_ADC[0-1]_AIN2	A/I	Analog input 2	HiZ
AIN3	MCU_ADC[0-1]_AIN3	A/I	Analog input 3	HiZ
AIN4	MCU_ADC[0-1]_AIN4	A/I	Analog input 4	HiZ
AIN5	MCU_ADC[0-1]_AIN5	A/I	Analog input 5	HiZ
AIN6	MCU_ADC[0-1]_AIN6	A/I	Analog input 6	HiZ
AIN7	MCU_ADC[0-1]_AIN7	A/I	Analog input 7	HiZ
REFP	VDDA_ADC[0-1]	PWR	Reference voltage input positive	
REFN	VSS	GND	Reference voltage input negative	

**Table 12-2. ADC I/O Signals (continued)**

Module Pin	Device Level Signal	I/O <sup>(1)</sup>	Description	Module Pin Reset Value <sup>(2)</sup>
<b>MCU_ADC[0-1]</b>				
EXT_TRIGGER	MCU_ADC[0-1]_EXT_TRIGGER	I	External trigger for ADC	HiZ

(1) I = Input; A = Analog

(2) HiZ = High Impedance

### Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

### 12.1.1.3 ADC Integration

This section describes ADC integration in the device, including information about clocks, resets, and hardware requests.

#### 12.1.1.3.1 ADC Integration in MCU Domain

There are two ADC modules integrated in the device MCU domain - MCU\_ADC0 and MCU\_ADC1. [Figure 12-3](#) shows the integration of MCU\_ADC0 and MCU\_ADC1.

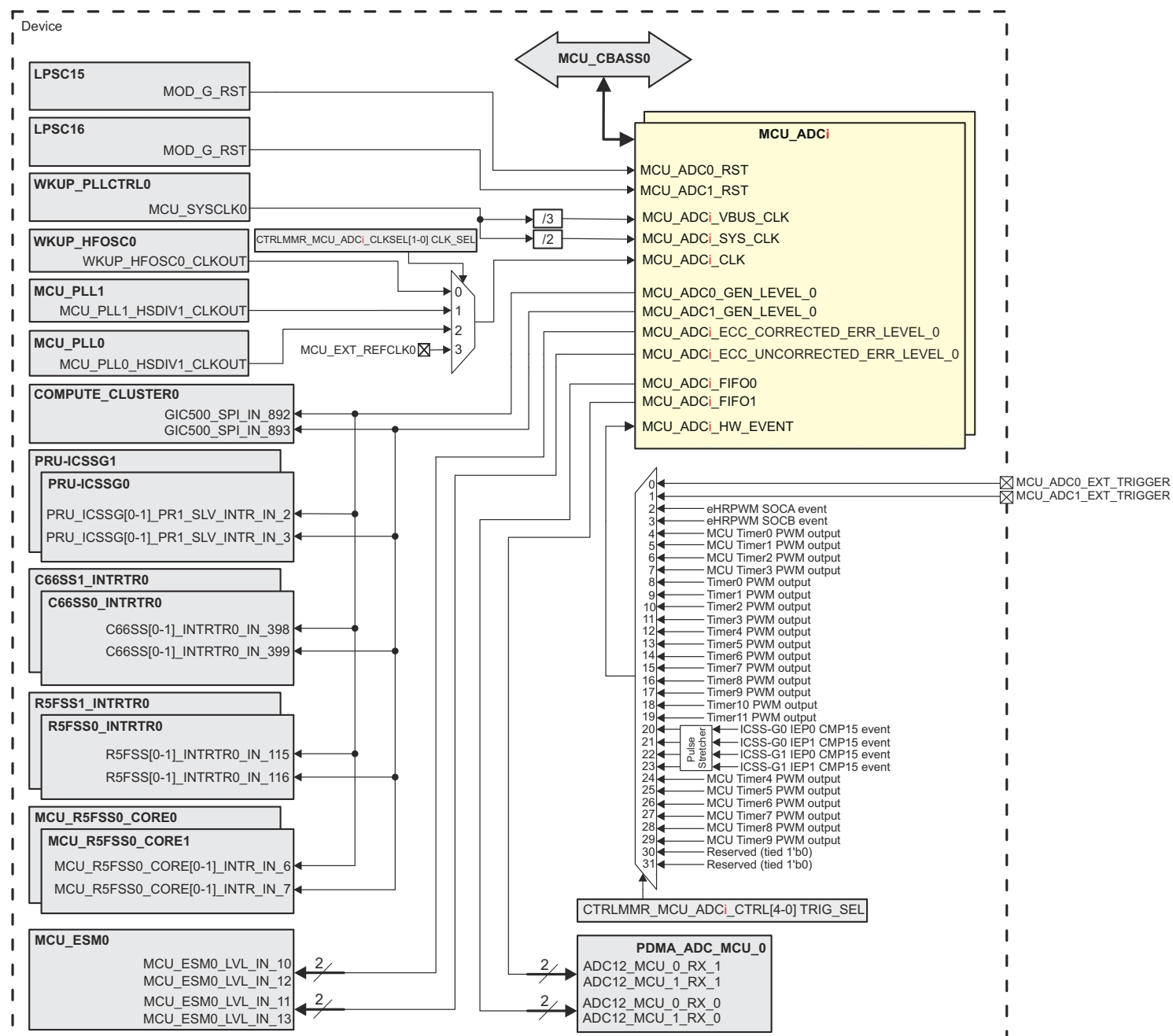


Table 12-3 through Table 12-5 summarize the integration of MCU\_ADC0 and MCU\_ADC1 in device MCU domain.

**Table 12-3. MCU\_ADC[0-1] Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
MCU_ADC0	WKUP_PSC0	PD0	LPSC15	MCU_CBASS0
MCU_ADC1	WKUP_PSC0	PD0	LPSC16	MCU_CBASS0

**Table 12-4. MCU\_ADC[0-1] Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
MCU_ADC0	MCU_ADC0_VBUS_CLK	MCU_SYSCLK0/3	WKUP_PLLCTRL0	MCU_ADC0 interface clock
	MCU_ADC0_SYS_CLK	MCU_SYSCLK0/2	WKUP_PLLCTRL0	MCU_ADC0 system clock
	MCU_ADC0_CLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	MCU_ADC0 clock. Output of multiplexer, see <a href="#">Figure 12-3, ADC Integration</a> . Multiplexer control is provided via CTRLMMR_MCU_ADC0_CLKSEL[1-0] CLK_SEL bit field. Default state is HFOSC0_CLKOUT.
		MCU_PLL1_HSDIV1_CLKOUT	MCU_PLL1	
		MCU_PLL0_HSDIV1_CLKOUT	MCU_PLL0	
		MCU_EXT_REFCLK0	I/O pin	
MCU_ADC1	MCU_ADC1_VBUS_CLK	MCU_SYSCLK0/3	WKUP_PLLCTRL0	MCU_ADC1 interface clock
	MCU_ADC1_SYS_CLK	MCU_SYSCLK0/2	WKUP_PLLCTRL0	MCU_ADC1 system clock
	MCU_ADC1_CLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	MCU_ADC1 sampling clock. Output of multiplexer, see <a href="#">Figure 12-3, ADC Integration</a> . Multiplexer control is provided via CTRLMMR_MCU_ADC1_CLKSEL[1-0] CLK_SEL bit field. Default state is HFOSC0_CLKOUT.
		MCU_PLL1_HSDIV1_CLKOUT	MCU_PLL1	
		MCU_PLL0_HSDIV1_CLKOUT	MCU_PLL0	
		MCU_EXT_REFCLK0	I/O pin	
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
MCU_ADC0	MCU_ADC0_RST	MOD_G_RST	LPSC15	MCU_ADC0 reset
MCU_ADC1	MCU_ADC1_RST	MOD_G_RST	LPSC16	MCU_ADC1 reset

**Table 12-5. MCU\_ADC[0-1] Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCU_ADC0	MCU_ADC0_GEN_LEVEL_0	GIC500_SPI_IN_892	COMPUTE_CLUSTER0	MCU_ADC0 interrupt request	Level
		PRU_ICSSG0_PR1_SLV_INTR_IN_2	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INTR_IN_2	PRU-ICSSG1		
		C66SS0_INTRTR0_IN_398	C66SS0_INTRTR0		



**Table 12-5. MCU\_ADC[0-1] Hardware Requests (continued)**

		C66SS1_INTRTR0_IN_398	C66SS1_INTRTR0		
		R5FSS0_INTRTR0_IN_115	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_115	R5FSS1_INTRTR0		
		MCU_R5FSS0_CORE0_INTR_IN_6	MCU_R5FSS0_CORE0		
		MCU_R5FSS0_CORE1_INTR_IN_6	MCU_R5FSS0_CORE1		
MCU_ADC1	MCU_ADC0_ECC_CORRECTED_ERR_LEVEL_0	MCU_ESM0_LVL_IN_10	MCU_ESM0	MCU_ADC0 ECC corrected error interrupt request	Level
	MCU_ADC0_ECC_UNCORRECTED_ERR_LEVEL_0	MCU_ESM0_LVL_IN_11	MCU_ESM0	MCU_ADC0 ECC uncorrected error interrupt request	Level
	MCU_ADC1_GEN_LEVEL_0	GIC500_SPI_IN_893	COMPUTE_CLUSTER0	MCU_ADC1 interrupt request	Level
		PRU_ICSSG0_PR1_SLV_INTR_IN_3	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INTR_IN_3	PRU-ICSSG1		
		C66SS0_INTRTR0_IN_399	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_399	C66SS1_INTRTR0		
		R5FSS0_INTRTR0_IN_116	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_116	R5FSS1_INTRTR0		
		MCU_R5FSS0_CORE0_INTR_IN_7	MCU_R5FSS0_CORE0		
		MCU_R5FSS0_CORE1_INTR_IN_7	MCU_R5FSS0_CORE1		
	MCU_ADC1_ECC_CORRECTED_ERR_LEVEL_0	MCU_ESM0_LVL_IN_12	MCU_ESM0	MCU_ADC1 ECC corrected error interrupt request	Level
	MCU_ADC1_ECC_UNCORRECTED_ERR_LEVEL_0	MCU_ESM0_LVL_IN_13	MCU_ESM0	MCU_ADC1 ECC uncorrected error interrupt request	Level
DMA Events					
Module Instance	Module DMA Event	Destination DMA Event Input	Destination	Description	Type
MCU_ADC0	MCU_ADC0_FIFO0	ADC12_MCU_0_RX_0	PDMA_ADC_MCU_0	MCU_ADC0 receive request line	Pulse
	MCU_ADC0_FIFO1	ADC12_MCU_0_RX_1	PDMA_ADC_MCU_0	MCU_ADC0 receive request line	Pulse
MCU_ADC1	MCU_ADC1_FIFO0	ADC12_MCU_1_RX_0	PDMA_ADC_MCU_0	MCU_ADC1 receive request line	Pulse
	MCU_ADC1_FIFO1	ADC12_MCU_1_RX_1	PDMA_ADC_MCU_0	MCU_ADC1 receive request line	Pulse

### 12.1.1.4 ADC Functional Description

#### 12.1.1.4.1 ADC FSM Sequencer Functional Description

The FSM sequencer supports up to 16 steps that can be configured to perform analog-to-digital conversions. Each of the 16 steps can individually be enabled and configured to operate as software enabled steps or hardware enabled steps. Each step has its own programmable step configuration (ADC\_CONFIG\_j) and step delay (ADC\_DELAY\_j) registers that allow users to configure the ADC operation on a per-step basis.

The FSM sequencer enters Idle immediately after the ADC is enabled and remains in Idle while waiting for any of the 16 programmable steps to be enabled. When one or more step is enabled, the FSM sequencer will begin executing or skipping each step starting with the lowest and incrementing up through each step until the highest step has been executed or skipped. The FSM sequencer will execute enabled steps and skip disabled steps. Once all of the steps have been executed or skipped, the FSM sequencer will return to Idle or immediately begin repeating steps configured for continuous mode.

An ENDOSEQUENCE interrupt can be generated after the last enabled step has been executed. If any software enabled steps are configured for continuous operation and they are still enabled when the FSM sequencer has incremented through all the steps, the FSM sequencer will start the sequence again with the first enabled step. Hardware enabled steps are mapped to a hardware event and will need the hardware event to occur before being scheduled.

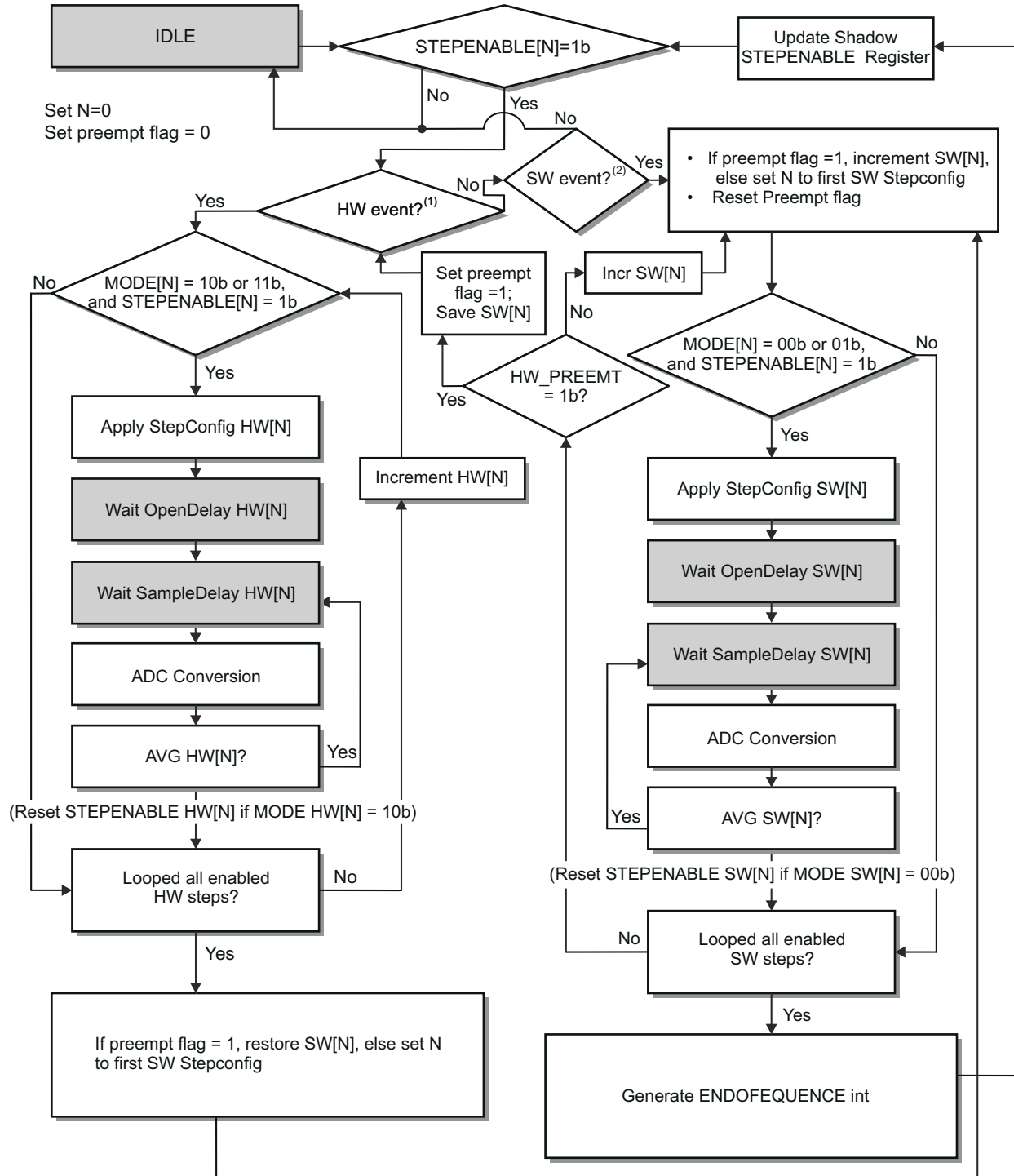
Figure 12-4 illustrates how the FSM sequencer works. Each shaded box represents a FSM state.

---

#### Note

Figure 12-4 does not represent the number of SMPL\_CLK cycles required by each state since some states implement user programmable delays that may require more than one SMPL\_CLK cycle to complete.

---



(1) MODE[N] = 10b or 11b, STEPENABLE[N] = 1b, and HW Event = 1

(2) MODE[N] = 00b or 01b and STEPENABLE[N] = 1b

adc-005

**Figure 12-4. Sequencer FSM**

#### 12.1.1.4.1.1 Step Enable

A step is enabled or disabled via the ADC\_STEPENABLE register. The system software should only write to the ADC\_STEPENABLE register when the ADC is disabled or the FSM sequencer is idle. Therefore, it may

be necessary to disable the ADC before updating the ADC\_STEPPABLE register if the FSM sequencer is executing back-to-back steps where it never enters idle.

#### **12.1.1.4.1.2 Step Configuration**

##### **12.1.1.4.1.2.1 One-Shot (Single) or Continuous Mode**

Each step can be configured to operate in one-shot or continuous mode via the least significant bit of the MODE bit-field of the respective ADC\_CONFIG\_j register. Steps configured for one-shot mode will perform a single conversion before being disabled by the FSM sequencer. Steps configured for continuous mode will automatically be re-enabled by the FSM sequencer after each conversion until disabled by software.

##### **12.1.1.4.1.2.2 Software- or Hardware-Enabled Steps**

Each step can be configured to operate in SW Enabled mode or HW Enabled mode independently via the most significant bit of the MODE bit-field in the respective ADC\_CONFIG\_j register. However, the hardware event source is applied globally to all HW Enabled steps and configured via the ADC\_CONTROL register.

The user can configure each step to begin immediately after the step is enabled by software (SW Enabled), or wait for a hardware event to occur (HW Enabled).

The EXT\_TRIGGER input signal is used by HW Enabled steps. The trigger input has the option of being sourced from a device input pin or several on-device peripherals via a multiplexer controlled by TRIG\_SEL of the respective CTRLMMR\_MCU\_ADC0\_CTRL or CTRLMMR\_MCU\_ADC1\_CTRL register. When this option is selected, the hardware event will be triggered after the EXT\_TRIGGER input signal transitions from low to high and is held high for a period greater than two cycles of SYS\_CLK. This hardware event will only schedule one complete sequence, even for continuous mode. If HW Enabled steps need to be executed again, a new hardware event must be generated after the previously triggered sequence has completed.

SW Enabled steps can be preempted by HW Enabled steps when the HW\_PREEMPT bit in the ADC\_CONTROL register is set.

##### **12.1.1.4.1.2.3 Averaging of Samples**

Each step can be configured, via the respective ADC\_CONFIG\_j register, to sample an input 1, 2, 4, 8, or 16 times and provide an average data value. If a step is configured to sample more than once, the additional samples are taken back-to-back immediately after the first sample. However, open delay is only applied to the first sample while sample delay is applied to all samples. Once the last sample has been taken the average data value will be stored in the FIFO.

##### **12.1.1.4.1.2.4 Analog Multiplexer Input Select**

The AFE contains two 9-to-1 analog multiplexers which are used to select the source connected to the positive input (INP) and negative input (INM) of the ADC. Each step can be configured, via the respective ADC\_CONFIG\_j register, to select the appropriate input sources. For more information, refer to [Figure 12-6](#) and the ADC\_CONFIG\_j register field descriptions.

##### **12.1.1.4.1.2.5 Differential Control**

The ADC supports differential inputs and each step can be configured, via the respective ADC\_CONFIG\_j register, to select single-ended input mode or differential input mode.

---

#### **Note**

The ADC INM input must be sourced from REFN for any steps configured to operate in single-ended mode.

---

##### **12.1.1.4.1.2.6 FIFO Select**

The ADC contains two FIFOs and each step can be configured, via the respective ADC\_CONFIG\_j register, to store its ADC data into either FIFO.

#### 12.1.1.4.1.2.7 Range Check Interrupt Enable

Each step can be configured, via the respective ADC\_CONFIG\_j register, to compare the value of ADC data to high and low limits programmed in the ADC\_RANGE register. The out-of-range interrupt is generated if the value of ADC data is greater than the value of HIRANGE or less than the value of LOWRANGE. Refer to the ADC\_RANGE register field descriptions for information related to setting the high and low limits that trigger an out-of-range interrupt.

#### 12.1.1.4.1.3 Open Delay and Sample Delay

The FSM sequencer provides two programmable delays for each step. The value of OPENDELAY is used to control when the acquisition begins after the step starts and the value of SAMPLEDELAY is used to control the acquisition period. Delays for each of the 16 steps can be configured independently via the respective ADC\_DELAY\_j register.

##### 12.1.1.4.1.3.1 Open Delay

OPENDELAY defaults to a value of zero which causes the acquisition period to begin as soon as the step starts. The start of the acquisition period can be delayed one SMPL\_CLK clock period for each incremental value of OPENDELAY.

##### 12.1.1.4.1.3.2 Sample Delay

SAMPLEDELAY defaults to a value of zero which causes the acquisition period to be equal to two SMPL\_CLK clock periods. The acquisition period can be extended one SMPL\_CLK clock for each incremental value of SAMPLEDELAY.

The value of SAMPLEDELAY should be configured to provide enough time for the respective external voltage source to completely charge the AFE input capacitance during the acquisition period.

#### 12.1.1.4.1.4 Interrupts

The ADC has 9 internal events that can trigger an interrupt to the processor. These events are logically or'ed together to produce a single interrupt to the processor. The user can individually enable or disable each interrupt source via the ADC\_ENABLE\_SET and ADC\_ENABLE\_CLR registers. Once an interrupt is generated, the ADC\_STATUS register can be read to determine which interrupt source(s) interrupted the processor. The interrupt source(s) can be cleared by writing a '1' to the corresponding bit of the ADC\_STATUS register.

The following interrupts are supported by the device ADC:

- An ENDOSEQUENCE interrupt is generated after the FSM sequencer completes the last enabled step.
- FIFO0UNFL/FIFO1UNFL or FIFO0OVFL/FIFO1OVFL interrupts. Each FIFO also has the ability to generate overrun and underflow interrupts. The ADC does not recover from overrun or underflow conditions automatically. Therefore, software will need to disable and re-enable the ADC after this occurs. The FSM sequencer must be Idle before turning the ADC module back on. This can be confirmed by reading the ADC\_SEQUENCER\_STAT register, when FSM\_BUSY = 0x0 and STEP\_IDLE = 0x10000.
- FIFO0THRS and FIFO1THRS interrupts. Each FIFO has the ability to generate an interrupt when the FIFO word count has reached a programmable threshold value. The FIFO0THRS and FIFO1THRS interrupt is generated when the value of NUMWDS in the respective ADC\_FIFO0WC or ADC\_FIFO1WC register equals the word count threshold value programmed in THRESHOLD of the respective ADC\_FIFO0THRESHOLD or ADC\_FIFO1THRESHOLD register.
- OUTOFRANGE interrupt. This interrupt can be enabled or disabled on each step. If enabled, the ADC data value is compared to the low and high thresholds programmed in the global ADC\_RANGE register. The OUTOFRANGE interrupt is generated if the ADC data value is greater than the value programmed in HIRANGE or less than the value programmed in LOWRANGE.
- AFE\_EOC\_MISSING interrupt. This interrupt is generated when the FSM sequencer does not receive an expected EOC from the AFE.

#### 12.1.1.4.1.5 Power Management

The software also has the option to turn off the AFE power by writing to the ADC\_CONTROL[4] PD bit. By default, after the initial power on the AFE is powered down and software is responsible for turning it on. Software

must wait minimum 4  $\mu$ s after a AFE power up and before starting a conversion (must be followed every time the AFE is powered up). It is the software's responsibility to empty the FIFO before requesting the ADC to enter sleep state.

#### 12.1.1.4.1.6 DMA Requests

Each FIFO can be serviced by the processor or DMA. The user can individually enable or disable each FIFO DMA request via the ADC\_DMAENABLE\_SET and ADC\_DMAENABLE\_CLR registers.

#### 12.1.1.4.2 ADC AFE Functional Description

The AFE contains a single 12-bit successive approximation ADC which can be connected to one of eight user selectable analog inputs for each step executed by the FSM sequencer. There is also an option to configure the ADC to operate in differential mode where it samples the differential voltage applied to two of eight user selectable analog inputs.

The ADC sits idle until the FSM sequencer sends a Start of Conversion (SOC) pulse. Once the FSM sequencer sends a SOC pulse to the ADC, it begins sampling the analog input signal on the rising edge of SOC and continues sampling the analog input signal one cycle of SMPL\_CLK after the falling edge of SOC. The SOC pulse width will be the value of SAMPLEDELAY plus one SMPL\_CLK periods, so the total ADC acquisition time will be the value of SAMPLEDELAY plus two SMPL\_CLK periods. The ADC captures the analog input signal at the end of the acquisition period and starts conversion, which requires thirteen SMPL\_CLK periods to digitize the sampled input. An EOC signal is returned to the FSM sequencer to indicate the digital data is ready.

The ADC output is positive binary weighted data ranging from 0 to  $(2^{12} - 1)$ . When configured for single-ended mode, ADC output data values 0 to  $(2^{12} - 1)$  represents a corresponding input voltage that ranges from the ADC negative reference (REFN) voltage to the ADC positive reference (REFP) voltage. When configured for differential mode, ADC output data values 0 to  $((2^{12}/2) - 1)$  represents a corresponding negative differential input voltage that ranges from the REFP voltage to the REFN voltage and output data values  $(2^{12}/2)$  to  $(2^{12} - 1)$  represents a corresponding positive differential input voltage that ranges from the REFN voltage to the REFP voltage.

Using the minimum values of OPENDELAY and SAMPLEDELAY, the ADC can sample an analog input every 15 SMPL\_CLK periods.

[Figure 12-5](#) illustrates the operation of the FSM sequencer and ADC AFE, with indicators showing when the hardware configuration defined by ADC\_CONFIG\_j and ADC\_DELAY\_j registers are applied.

#### Note

[Figure 12-5](#) assumes both steps shown are software enabled with averaging turned off, an OPENDELAY value of 1, and SAMPLEDELAY value of 0. The analog input is sampled for the duration of the SOC pulse plus one cycle of the SMPL\_CLK. If the value of OPENDELAY were 0, it would appear as if the time associated with the Open portion of the FSM waveform would be removed from all the waveforms.

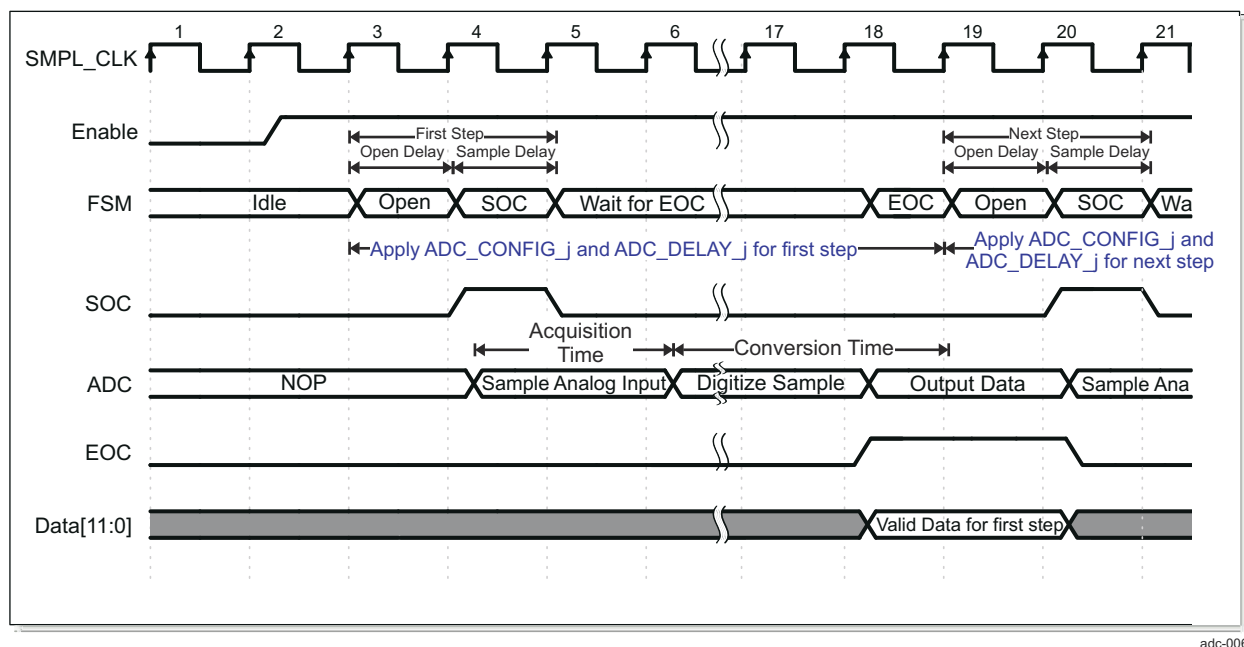


Figure 12-5. Example Timing Diagram for Sequencer

#### 12.1.1.4.2.1 AFE Functional Block Diagram

The AFE has 8 analog inputs which can be configured as either singled-ended or differential inputs to the ADC. The user may configure a step to use any 1 of the 8 inputs when operating in single-ended mode or any 2 of the 8 inputs when operating in differential mode. For more information, refer to [Figure 12-6](#) and the ADC\_CONFIG\_j register field descriptions.

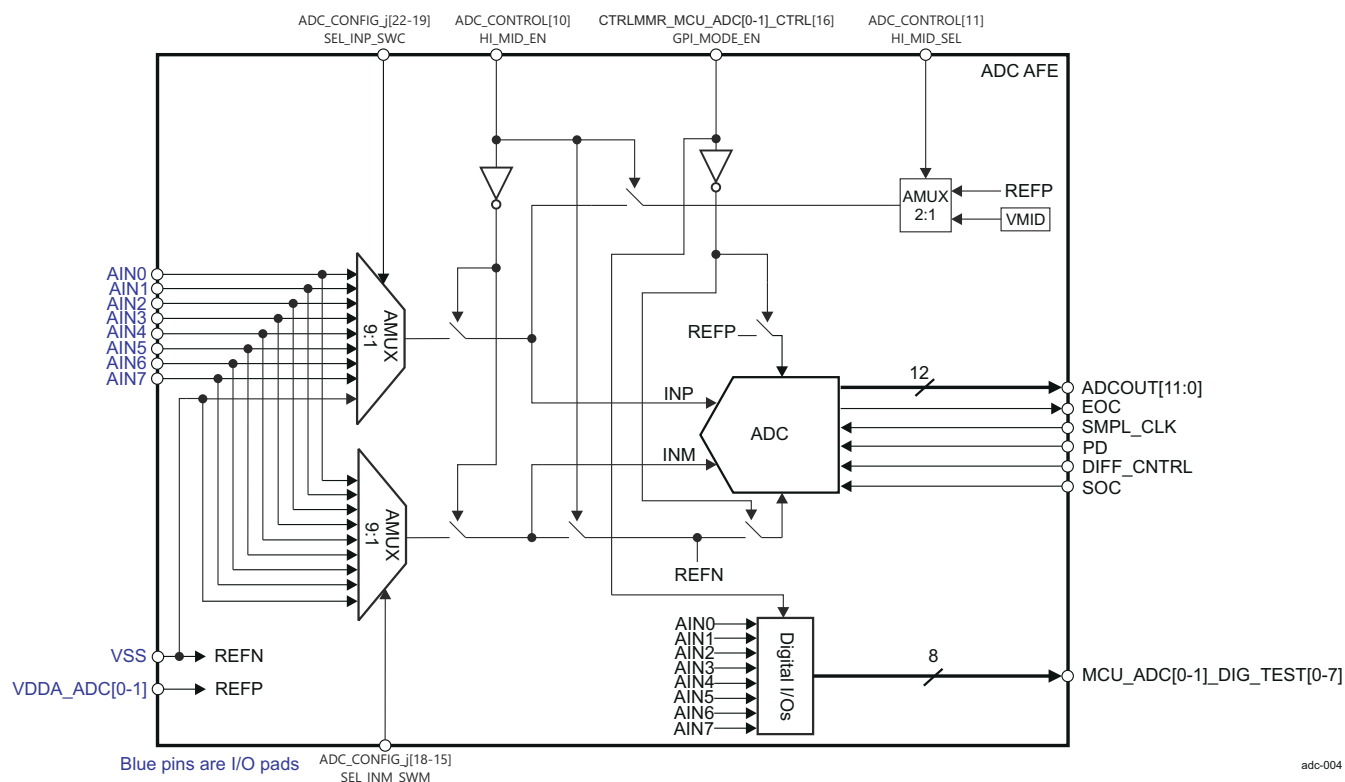


Figure 12-6. Functional Block Diagram



#### 12.1.1.4.2.2 ADC GPI Integration

The ADC module can be configured to be used as a general digital test block which can generate 8 VDDA domain digital signals MCU\_ADC[0-1]\_DIG\_TEST[0-7]. These digital outputs are made available to the device PinMux, where are used as general purpose digital inputs with some restrictions. The CTRLMMR\_MCU\_ADC[0-1]\_CTRL[16] GPI\_MODE\_EN is the enable bit.

When GPI\_MODE\_EN bit is '0', the ADC is in normal operation.

When GPI\_MODE\_EN bit is '1', the ADC multiplexes the 8 analog inputs to 8 VDDA domain digital I/O block and outputs the signals to DIG\_TEST[7-0] pins.

Restrictions on this usage are:

- The GPI\_MODE\_EN bit is not switched dynamically.
- ADC supports only 1.8V logic level.
- All ADC inputs on each ADC function either can be configured as digital or analog inputs.
- These are GPI only, not GPIO (no output buffer).

#### 12.1.1.4.3 ADC FIFOs and DMA

##### 12.1.1.4.3.1 FIFOs

There are two sample FIFOs (ADC\_FIFO0 and ADC\_FIFO1), which store sample data from the AFE. Can be configured, via the respective ADC\_CONFIG\_j register.

The processor can read ADC data directly from the FIFO by using the respective ADC\_FIFO0DATA or ADC\_FIFO1DATA register. The internal logic will pop the next data from the FIFO and increment the FIFO read pointers.

The FIFO data will no longer be accessible after the ADC is disabled because the FIFO pointers are reset. Therefore, it is important to read all FIFO data before disabling the ADC.

##### 12.1.1.4.3.2 DMA

The ADC has a dedicated slave port for the DMA which allows it to perform continuous burst reads when accessing the FIFOs.

Each FIFO has its own DMA request which can be enabled via the ADC\_DMAENABLE\_SET register and disabled via the ADC\_DMAENABLE\_CLR register.

The first DMA request is generated after the FIFO leaves the EMPTY state and fills to the level programmed in DMAREQLEVEL of the respective ADC\_FIFO0DMAREQ or ADC\_FIFO1DMAREQ register.

Subsequently, a new DMA request is automatically generated on the next clock cycle after the current DMA access completes if the previous DMA access did not empty the FIFO.

#### 12.1.1.4.4 ADC Error Correcting Code (ECC)

The Error Correcting Code (ECC) is a mechanism for providing increased system reliability (via reduction of memory soft errors) by allowing single bit errors to be detected and corrected and double bit errors to be detected.

The ECC Aggregator provides a single EOI-handshake based interrupt to the host (for both single and double error detections).

The ECC Aggregator will write 0's to all of memory after reset is released. The ADC\_SEQUENCER\_STAT[6] MEM\_INIT\_DONE bit will be 0 during this period and will be set to 1 upon completion.

For more information about the ECC Aggregator, see *ECC Aggregator* and *ADC ECC Registers*.

See also *Testing ECC Error Injection* and *ADC ECC Registers*.

##### 12.1.1.4.4.1 Testing ECC Error Injection

The read enable (REN) input to the RAMs instantiated in this module are constantly tied to 1'b1. As a result, read is continuously in progress.



In order to test ECC single error injection on a particular RAM row, the following conditions have to be maintained before setting the ADC\_ECC\_CTRL[3] FORCE\_SEC bit:

- ADC\_ECC\_CTRL[5] FORCE\_N\_ROW bit has to be clear. This is required in order to configure required row in
- ADC\_ECC\_CTRL[6] ERROR\_ONCE bit has to be clear. If this bit is set, the ECC error occurs immediately after setting ADC\_ECC\_CTRL[3] FORCE\_SEC bit since REN is always set
- ADC\_ECC\_ERR\_CTRL1[31:0] ECC\_ROW bit field has to be point to the required row address

#### **12.1.1.4.5 ADC Functional Internal Diagnostic Debug Mode**

The ADC function can be validated by measuring known reference voltage sources. This feature provides register control that connects the AFE to known reference voltage sources. However, it does not automatically configure the FSM to perform conversions to measure these sources. Software will need to configure the FSM to perform a measurement after selecting this mode of operation.

ADC\_CONTROL[10] HI\_MID\_EN is used to disconnect the AFE from all analog inputs and connect it to one of two known reference voltage sources. ADC\_CONTROL[11] HI\_MID\_SEL is used to select REFP or VMID as the known reference voltage source.

The conversion results is expected to produce a full scale value when the AFE is connected to REFP and approximate half scale value when the AFE is connected to VMID.

The steps are:

- Disable the ADC if previously enabled and wait until the FSM is not busy.
- Put the AFE in a functional internal diagnostic debug mode by setting the ADC\_CONTROL[10] HI\_MID\_EN bit to 1.
- Select one of the two reference voltage sources by setting the ADC\_CONTROL[11] HI\_MID\_SEL bit.
  - 0 selects VMID
  - 1 selects REFP
- Configure the FSM to measure the selected reference voltage source.
- Enable the ADC and evaluate the results.
- When debug is complete, disable the ADC and wait until the FSM is not busy, re-program the FSM for functional operation, then re-enable the ADC.

### 12.1.1.5 ADC Programming Guide

#### 12.1.1.5.1 ADC Low-Level Programming Models

##### 12.1.1.5.1.1 Global Initialization

##### 12.1.1.5.1.1.1 Surrounding Modules Global Initialization

This section identifies the requirements for initializing the surrounding modules when the module is to be used for the first time after a device reset.

**Table 12-6. Global Initialization of Surrounding Modules**

Surrounding Modules	Comments
LPSC15	Module reset must be enabled. For more information about the module configuration, see <i>Reset</i> .
LPSC16	Module reset must be enabled. For more information about the module configuration, see <i>Reset</i> .
WKUP_PLLCTRL0	WKUP_PLLCTRL0 configuration must be done to enable the clocks to the ADC modules, see <i>Clocking</i> .
WKUP_HFOSC0	WKUP_HFOSC0 configuration must be done to enable the clocks to the ADC modules, see <i>Clocking</i> .
MCU_PLL0	MCU_PLL0 configuration must be done to enable the clocks to the ADC modules, see <i>Clocking</i> .
MCU_PLL1	MCU_PLL1 configuration must be done to enable the clocks to the ADC modules, see <i>Clocking</i> .
COMPUTE_CLUSTER0	Device INTCs must be configured to enable the interrupt request generation. For information about enabling COMPUTE_CLUSTER0 interrupts, see <i>Interrupts</i> .
PRU-ICSSG0/1	Device INTCs must be configured to enable the interrupt request generation. For information about enabling PRU-ICSSG0/1 interrupts, see <i>Interrupts</i> .
C66SS0/1_INTRTR0	Device INTCs must be configured to enable the interrupt request generation. For information about enabling C66SS0/1_INTRTR0 interrupts, see <i>Interrupts</i> .
R5FSS0/1_INTRTR0	Device INTCs must be configured to enable the interrupt request generation. For information about enabling R5FSS0/1_INTRTR0 interrupts, see <i>Interrupts</i> .
MCU_R5FSS0_CORE0/1	Device INTCs must be configured to enable the interrupt request generation. For information about enabling MCU_R5FSS0_CORE0/1 interrupts, see <i>Interrupts</i> .
MCU_ESM0	Device INTCs must be configured to enable the interrupt request generation. For information about enabling MCU_ESM0 interrupts, see <i>Interrupts</i> .
PDMA_ADC_MCU_0	PDMA_ADC_MCU_0 controllers configuration must be done to enable the module PDMA_ADC_MCU_0 input request, see <i>Data Movement Architecture (DMA)</i> .
Interconnect	For information about the MCU_CBASS0 interconnect configuration, see <i>System Interconnect</i> .

##### 12.1.1.5.1.1.2 General Programming Model

**Table 12-7. General Programming Model**

Step	Register/Bit Field/Programming Model	Value
Power up the AFE	ADC_CONTROL[4] PD	0
If using DMA, set the DMA threshold and enable the appropriate DMA request(s); or set the processor threshold and enable the appropriate threshold interrupt(s) as part of the next step.	ADC_DMAENABLE_SET/ADC_FIFO0DMAREQ/ ADC_FIFO1DMAREQ/ ADC_FIFO0THRESHOLD/ ADC_FIFO1THRESHOLD	0x-
Setup or mask the appropriate interrupts.	ADC_ENABLE_SET	0x-
Configure each step being used to perform an ADC conversion.	ADC_CONFIG_j	0x-
Configure the delays for each step being used to perform an ADC conversion.	ADC_DELAY_j	0x-
Enable each step being used to perform an ADC conversion.	ADC_STEPENABLE	0x-
Wait minimum 4 $\mu$ s before starting a conversion to provide time for the AFE to power-up. (referenced from POWER_DOWN write to 0)		
Enable the ADC module	ADC_CONTROL[0] MODULE_ENABLE	1

### 12.1.1.5.1.2 During Operation

**Table 12-8. Turn-Off**

Step	Register/Bit Field/Programming Model	Value
Turn off ADC module enable bit when finished. This will stop the ADC after the current step is complete.	ADC_CONTROL[0] MODULE_ENABLE	0

**Table 12-9. Turn-On After Turn-Off**

Step	Register/Bit Field/Programming Model	Value
Enable the ADC module enable bit	ADC_CONTROL[0] MODULE_ENABLE	1

## 12.1.2 General-Purpose Interface (GPIO)

This chapter describes the General-Purpose Input/Output (GPIO) for the device.

### 12.1.2.1 GPIO Overview

The General-Purpose Input/Output (GPIO) peripheral provides dedicated general-purpose pins that can be configured as either inputs or outputs. When configured as an output, the user can write to an internal register to control the state driven on the output pin. When configured as an input, user can obtain the state of the input by reading the state of an internal register.

In addition, the GPIO peripheral can produce host CPU interrupts and DMA synchronization events in different interrupt/event generation modes.

The device has ten instances of GPIO modules. The GPIO modules are integrated in three groups.

- Group one: WKUP\_GPIO0 and WKUP\_GPIO1. On I/O pins this group is named WKUP\_GPIO0.
- Group two: GPIO0, GPIO2, GPIO4, and GPIO6. On I/O pins this group is named GPIO0.
- Group three: GPIO1, GPIO3, GPIO5, and GPIO7. On I/O pins this group is named GPIO1.

The corresponding groups I/O pins are multiplexed within each group modules.

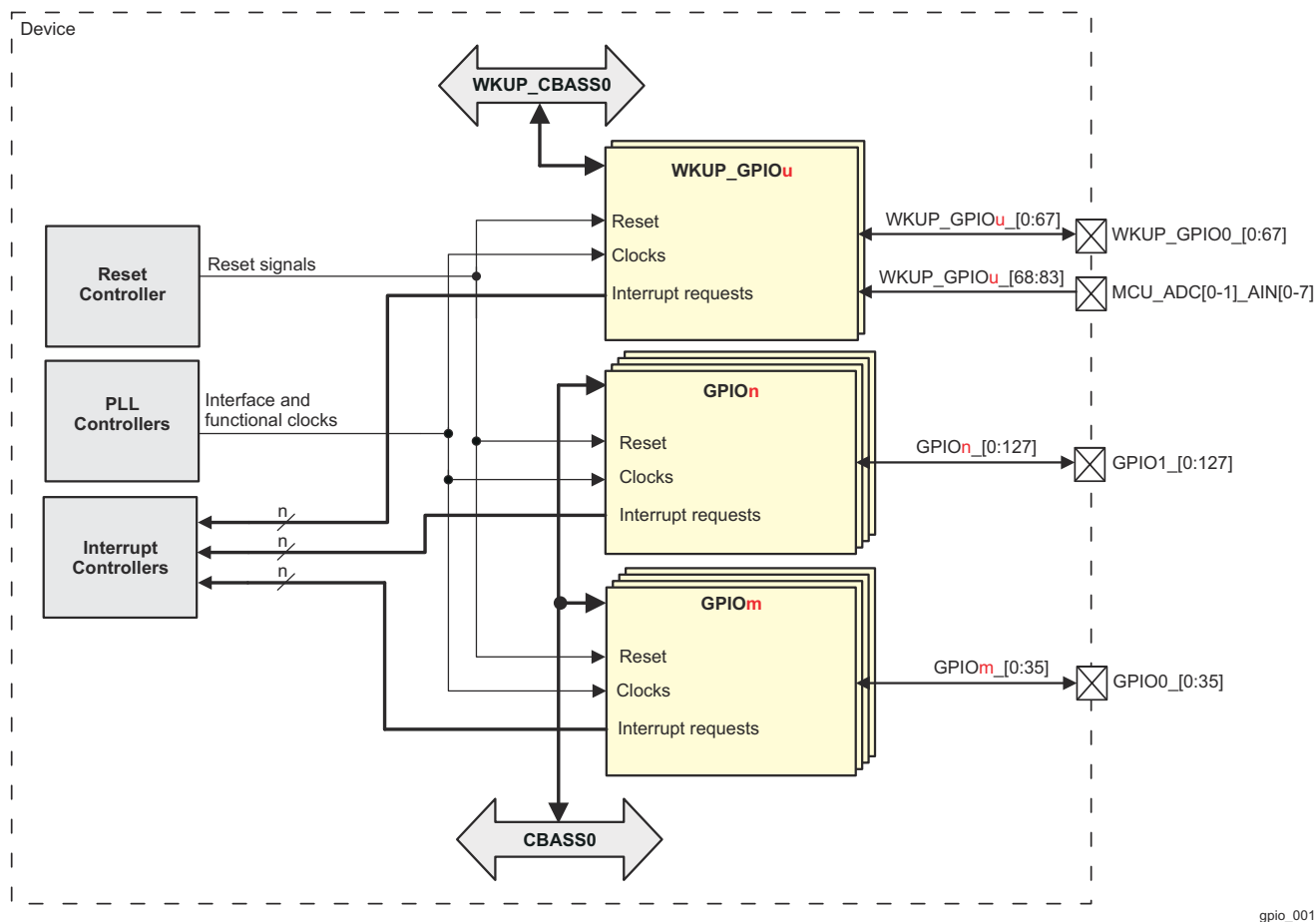
The GPIO pins are grouped into banks (16 pins per bank and 9 banks per module), which means that each GPIO module provides up to 144 dedicated general-purpose pins with input and output capabilities; thus, the general-purpose interface supports up to 432 (3 group instances  $\times$  (9 banks  $\times$  16 pins)) I/O pins. Since WKUP\_GPIOu\_[84:143] ( $u = 0, 1$ ), GPIOu\_[128:143] ( $n = 0, 2, 4, 6$ ), and GPIOm\_[36:143] ( $m = 1, 3, 5, 7$ ) are reserved in this device, general purpose interface supports up to 248 I/O pins.

Table 12-10 shows GPIO modules allocation across device domains.

**Table 12-10. GPIO Allocation Across Device Domains**

Instance	Domain		
	WKUP	MCU	MAIN
WKUP_GPIO0	✓	-	-
WKUP_GPIO1	✓	-	-
GPIO0	-	-	✓
GPIO1	-	-	✓
GPIO2	-	-	✓
GPIO3	-	-	✓
GPIO4	-	-	✓
GPIO5	-	-	✓
GPIO6	-	-	✓
GPIO7	-	-	✓

Figure 12-7 presents the GPIO modules overview.



gpio\_001

- A.  $u = 0, 1$   
 B.  $n = 0, 2, 4, 6$   
 C.  $m = 1, 3, 5, 7$

**Figure 12-7. GPIO Modules Overview**

#### 12.1.2.1.1 GPIO Features

Each channel in the GPIO modules has the following features:

- Supports 9 banks of 16 GPIO signals
- Supports up to 9 banks of interrupt capable GPIOs
- Interrupts:
  - Can enable interrupts for each bank of 16 GPIO signals
  - Interrupts can be triggered by rising and/or falling edge, specified for each interrupt capable GPIO signal
- Set/clear functionality:
  - Firmware writes 1 to corresponding bit position(s) to set or to clear GPIO signal(s). This allows multiple firmware processes to toggle GPIO output signals without critical section protection (disable interrupts, program GPIO, re-enable interrupts, to prevent context switching to another process during GPIO programming).
- Separate Input/Output registers:
  - If preferred by firmware, some GPIO output signals can be toggled by direct write to the output register(s) in addition to set/clear.
  - Output register, when read in, reflects output drive status. This, in addition to the input register reflecting pin status this allows wired logic to be implemented.

**12.1.2.1.2 GPIO Not Supported Features**

- The following apply to WKUP\_GPIOu (u = 0, 1):
  - WKUP\_GPIOu\_[84:143] are not pinned out.
  - Interrupts [84:143] are not pinned out.
  - Bank Interrupts [8:6] are not pinned out.
  - From WKUP\_GPO[68] to WKUP\_GPO[83] buffer is output only.
- The following apply to GPIO<sub>n</sub> (n = 0, 2, 4, 6):
  - GPIO<sub>n</sub>\_[128:143] are not pinned out.
  - Interrupts [128:143] are not pinned out.
  - Bank Interrupt 8 is not pinned out.
- The following apply to GPIO<sub>m</sub> (m = 1, 3, 5, 7):
  - GPIO<sub>m</sub>\_[36:143] are not pinned out.
  - Interrupts [36:143] are not pinned out.
  - Bank Interrupts [8:2] are not pinned out.

### 12.1.2.2 GPIO Environment

The WKUP\_GPIOu (u = 0, 1), GPIO<sub>n</sub> (n = 0, 2, 4, 6), and GPIO<sub>m</sub> (m = 1, 3, 5, 7) modules are hereinafter referred to as GPIO module.

This section describes the GPIO external connections (environment).

The general-purpose interface combines ten GPIO modules in three groups for a flexible, user-programmable, general-purpose input/output (I/O) controller, and to allow protection between two different processor virtual worlds. The general-purpose interface implements functions that are not implemented with the dedicated controllers in the device and require simple input and/or output software-controlled signals. The GPIO allows a variety of custom connections and expands the I/O capabilities of the system to the real world.

The general-purpose interface can physically connect the device to a keyboard matrix and peripheral integrated circuits (ICs).

Figure 12-8 shows a typical application using the general-purpose interface configured to operate with WKUP\_GPIO0, GPIO0, and GPIO1 on I/O pins.

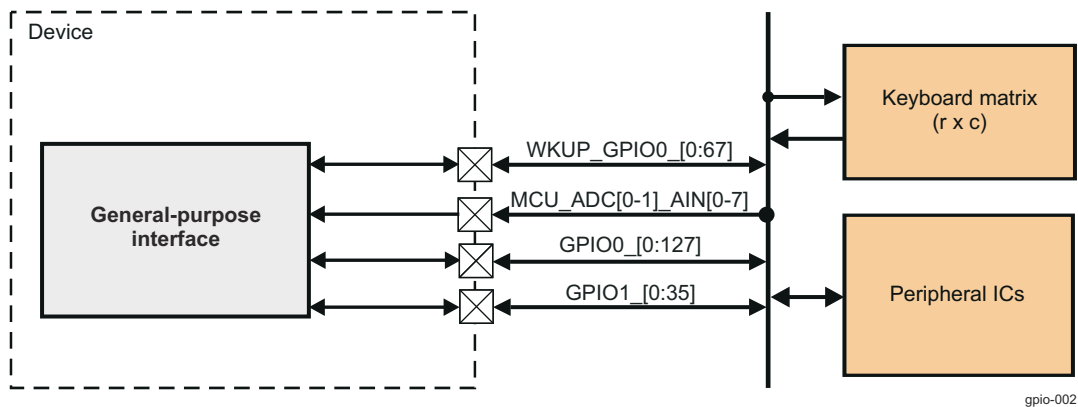
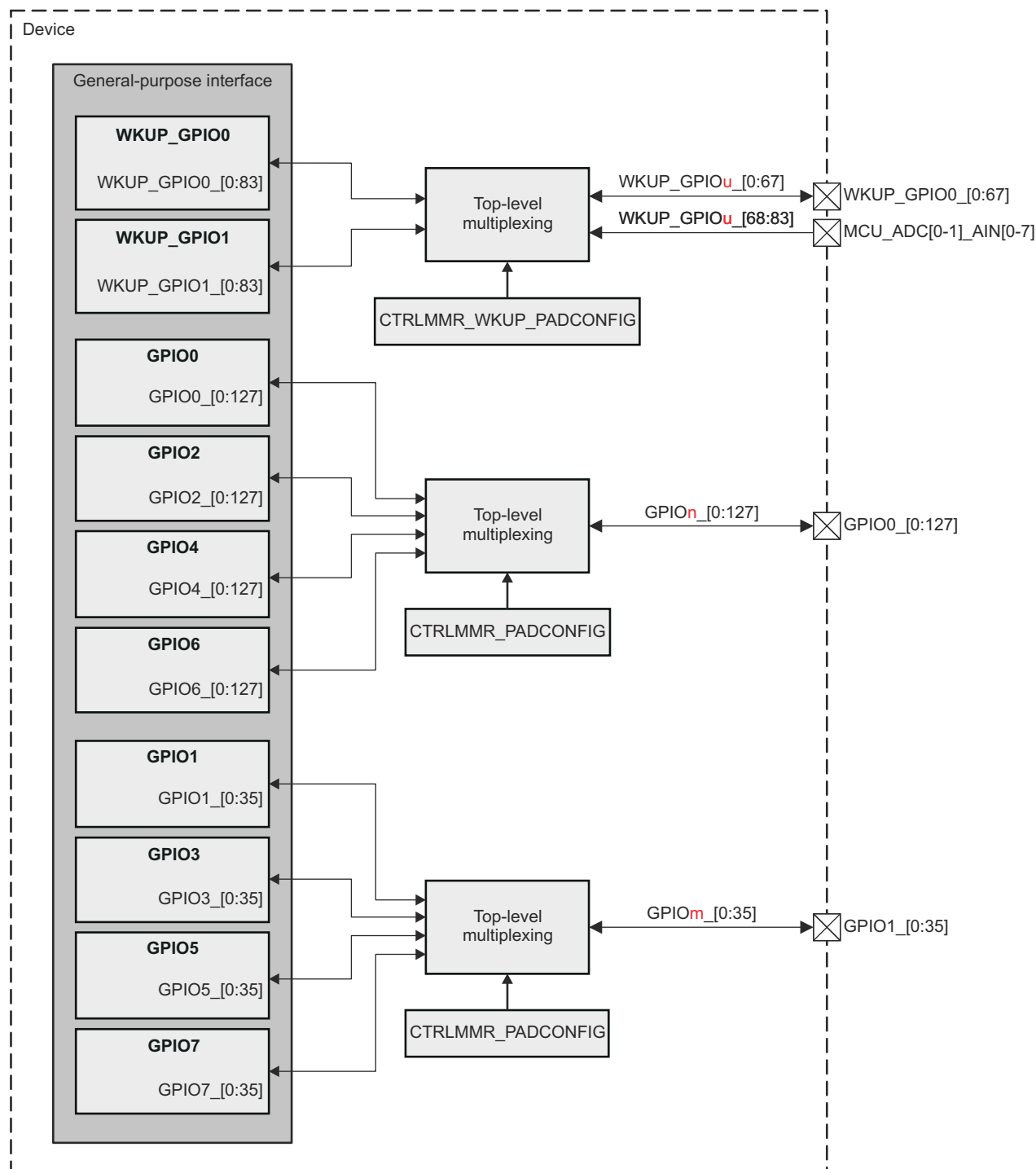


Figure 12-8. GPIO Typical Application

#### 12.1.2.2.1 GPIO Interface Signals

Figure 12-9 shows all of the GPIO interface signals.



gpio-003

- A.  $u = 0, 1$
- B.  $n = 0, 2, 4, 6$
- C.  $m = 1, 3, 5, 7$

**Figure 12-9. GPIO Interface Signals**

The active GPIO module can be selected using CTRLMMR\_WKUP\_PADCONFIG or CTRLMMR\_PADCONFIG registers. Doing so will also allow the individual per channel interrupts to be multiplexed based on the same



muxmode. At any time, only one GPIO module may control a given pin/ interrupt channel, as shown in Table [Table 12-11](#), Figure [Figure 12-9](#), and Figure [Section 12.1.2.4.4](#).

**Table 12-11. GPIO Instance Multiplexing**

PADCONFIG[3:0] MUXMODE	PADCONFIG[5:4] VGPI0_SEL	Selected Pin/Interrupt on WKUP_GPIO0	Selected Pin/Interrupt on GPIO0	Selected Pin/Interrupt on GPIO1
7h	0h	GPIO0	GPIO0	GPIO1
7h	1h	GPIO1	GPIO2	GPIO3
7h	2h	RESERVED	GPIO4	GPIO5
7h	3h	RESERVED	GPIO6	GPIO7

- 1. PADCONFIG[5:4] VGPI0\_SEL must be left at 0h for all non 7h muxmodes

Table 12-12 describes the GPIO I/O signals.

**Table 12-12. GPIO I/O Signals**

Module <sup>(5) (6) (8)</sup>	Module Pin	Device Pin <sup>(2)</sup>	I/O <sup>(1)</sup>	Description	Module Pin Reset Value <sup>(3)</sup>
WKUP_GPIOu	GPIO_BANK0_[0-15]	WKUP_GPIOu_[0:15]	I/O	General-purpose input/output. <sup>(4)</sup>	HiZ
	GPIO_BANK1_[0-15]	WKUP_GPIOu_[16:31]	I/O	General-purpose input/output. <sup>(4)</sup>	HiZ
	GPIO_BANK2_[0-15]	WKUP_GPIOu_[32:47]	I/O	General-purpose input/output. <sup>(4)</sup>	HiZ
	GPIO_BANK3_[0-15]	WKUP_GPIOu_[48:63]	I/O	General-purpose input/output. <sup>(4)</sup>	HiZ
	GPIO_BANK4_[0-3]	WKUP_GPIOu_[64:67]	I/O	General-purpose input/output. <sup>(4)</sup>	HiZ
	GPIO_BANK4_4	WKUP_GPIOu_68	I	General-purpose input. <sup>(4)</sup> Signal from MCU_ADC[0-1] module MCU_ADC0_DIG_TEST0. <sup>(7) (8)</sup>	HiZ
	GPIO_BANK4_5	WKUP_GPIOu_69	I	General-purpose input. <sup>(4)</sup> Signal from MCU_ADC[0-1] module MCU_ADC0_DIG_TEST1. <sup>(7) (8)</sup>	HiZ
	GPIO_BANK4_6	WKUP_GPIOu_70	I	General-purpose input. <sup>(4)</sup> Signal from MCU_ADC[0-1] module MCU_ADC0_DIG_TEST2. <sup>(7) (8)</sup>	HiZ
	GPIO_BANK4_7	WKUP_GPIOu_71	I	General-purpose input. <sup>(4)</sup> Signal from MCU_ADC[0-1] module MCU_ADC0_DIG_TEST3. <sup>(7) (8)</sup>	HiZ
	GPIO_BANK4_8	WKUP_GPIOu_72	I	General-purpose input. <sup>(4)</sup> Signal from MCU_ADC[0-1] module MCU_ADC0_DIG_TEST4. <sup>(7) (8)</sup>	HiZ
	GPIO_BANK4_9	WKUP_GPIOu_73	I	General-purpose input. <sup>(4)</sup> Signal from MCU_ADC[0-1] module MCU_ADC0_DIG_TEST5. <sup>(7) (8)</sup>	HiZ
	GPIO_BANK4_10	WKUP_GPIOu_74	I	General-purpose input. <sup>(4)</sup> Signal from MCU_ADC[0-1] module MCU_ADC0_DIG_TEST6. <sup>(7) (8)</sup>	HiZ
	GPIO_BANK4_11	WKUP_GPIOu_75	I	General-purpose input. <sup>(4)</sup> Signal from MCU_ADC[0-1] module MCU_ADC0_DIG_TEST7. <sup>(7) (8)</sup>	HiZ
	GPIO_BANK4_12	WKUP_GPIOu_76	I	General-purpose input. <sup>(4)</sup> Signal from MCU_ADC[0-1] module MCU_ADC1_DIG_TEST0. <sup>(7) (8)</sup>	HiZ
	GPIO_BANK4_13	WKUP_GPIOu_77	I	General-purpose input. <sup>(4)</sup> Signal from MCU_ADC[0-1] module MCU_ADC1_DIG_TEST1. <sup>(7) (8)</sup>	HiZ
	GPIO_BANK4_14	WKUP_GPIOu_78	I	General-purpose input. <sup>(4)</sup> Signal from MCU_ADC[0-1] module MCU_ADC1_DIG_TEST2. <sup>(7) (8)</sup>	HiZ
	GPIO_BANK4_15	WKUP_GPIOu_79	I	General-purpose input. <sup>(4)</sup> Signal from MCU_ADC[0-1] module MCU_ADC1_DIG_TEST3. <sup>(7) (8)</sup>	HiZ
	GPIO_BANK5_0	WKUP_GPIOu_80	I	General-purpose input. <sup>(4)</sup> Signal from MCU_ADC[0-1] module MCU_ADC1_DIG_TEST4. <sup>(7) (8)</sup>	HiZ
	GPIO_BANK5_1	WKUP_GPIOu_81	I	General-purpose input. <sup>(4)</sup> Signal from MCU_ADC[0-1] module MCU_ADC1_DIG_TEST5. <sup>(7) (8)</sup>	HiZ
	GPIO_BANK5_2	WKUP_GPIOu_82	I	General-purpose input. <sup>(4)</sup> Signal from MCU_ADC[0-1] module MCU_ADC1_DIG_TEST6. <sup>(7) (8)</sup>	HiZ
	GPIO_BANK5_3	WKUP_GPIOu_83	I	General-purpose input. <sup>(4)</sup> Signal from MCU_ADC[0-1] module MCU_ADC1_DIG_TEST7. <sup>(7) (8)</sup>	HiZ

**Table 12-12. GPIO I/O Signals (continued)**

Module <sup>(5) (6) (8)</sup>	Module Pin	Device Pin <sup>(2)</sup>	I/O <sup>(1)</sup>	Description	Module Pin Reset Value <sup>(3)</sup>
GPIO <sub>n</sub>	GPIO_BANK[6-8]_[0-15]	NC			
	GPIO_BANK0_[0-15]	GPIO0_[0:15]	I/O	General-purpose input/output. <sup>(4)</sup>	HiZ
	GPIO_BANK1_[0-15]	GPIO0_[16:31]	I/O	General-purpose input/output. <sup>(4)</sup>	HiZ
	GPIO_BANK2_[0-15]	GPIO0_[32:47]	I/O	General-purpose input/output. <sup>(4)</sup>	HiZ
	GPIO_BANK3_[0-15]	GPIO0_[48:63]	I/O	General-purpose input/output. <sup>(4)</sup>	HiZ
	GPIO_BANK4_[0-15]	GPIO0_[64:79]	I/O	General-purpose input/output. <sup>(4)</sup>	HiZ
	GPIO_BANK5_[0-15]	GPIO0_[80:95]	I/O	General-purpose input/output. <sup>(4)</sup>	HiZ
	GPIO_BANK6_[0-15]	GPIO0_[96:111]	I/O	General-purpose input/output. <sup>(4)</sup>	HiZ
	GPIO_BANK7_[0-15]	GPIO0_[112:127]	I/O	General-purpose input/output. <sup>(4)</sup>	HiZ
GPIO <sub>m</sub>	GPIO_BANK8_[0-15]	NC			
	GPIO_BANK0_[0-15]	GPIO0_[0:15]	I/O	General-purpose input/output. <sup>(4)</sup>	HiZ
	GPIO_BANK1_[0-15]	GPIO0_[16:31]	I/O	General-purpose input/output. <sup>(4)</sup>	HiZ
	GPIO_BANK2_[0-3]	GPIO0_[32:35]	I/O	General-purpose input/output. <sup>(4)</sup>	HiZ
	GPIO_BANK2_[4-15]	NC			
	GPIO_BANK[3-8]_[0-15]	NC			

(1) I = Input; O = Output; I/O = Bidirectional

(2) NC = Not Connected

(3) HiZ = High Impedance

(4) Only GPIOs enabled through from the CTRLMMR\_WKUP\_PADCONFIG0 to CTRLMMR\_WKUP\_PADCONFIG93 or from the CTRLMMR\_PADCONFIG0 to CTRLMMR\_PADCONFIG172 registers for a particular instance can operate on I/O pins. For more information see *Pad Configuration Registers*.

(5) Only GPIOs enabled through from the CTRLMMR\_WKUP\_PADCONFIG0 to CTRLMMR\_WKUP\_PADCONFIG100 or from the CTRLMMR\_PADCONFIG0 to CTRLMMR\_PADCONFIG73 registers for a particular instance can operate on I/O pins. For more information see *Pad Configuration Registers*.

(6) u = 0, 1

(7) n = 0, 2, 4, 6

(8) m = 1, 3, 5, 7

### Note

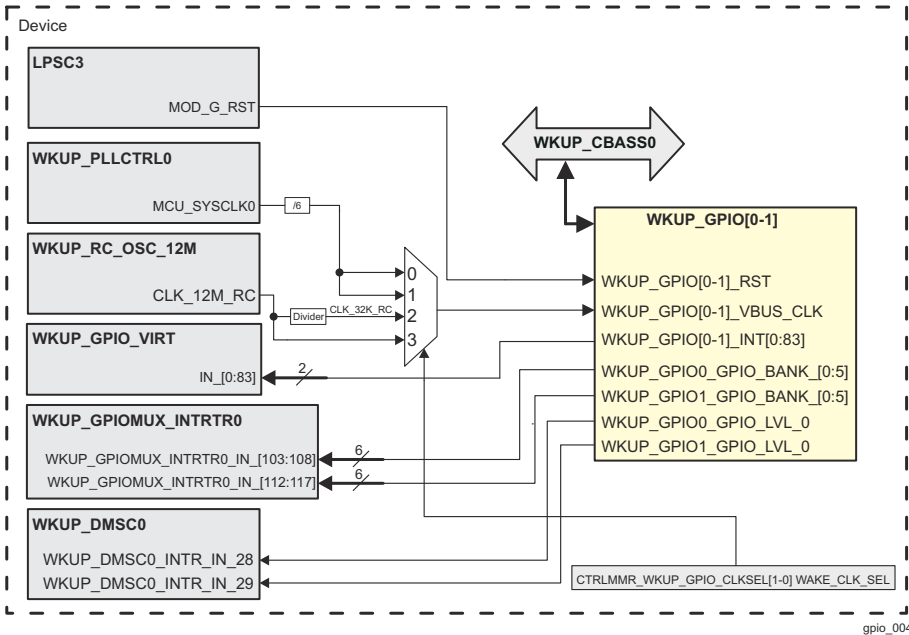
For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

### 12.1.2.3 GPIO Integration

This section describes modules integration in the device, including information about clocks, resets, and hardware requests.

#### 12.1.2.3.1 GPIO Integration in WKUP Domain

There are two WKUP\_GPIO modules integrated in the device WKUP domain - WKUP\_GPIO0 and WKUP\_GPIO1. [Figure 12-10](#) shows the integration of WKUP\_GPIO[0-1].



**Figure 12-10. WKUP\_GPIO[0-1] Integration**

[Table 12-13](#) through [Table 12-15](#) summarize the integration of WKUP\_GPIO[0-1] in the device WKUP domain.

**Table 12-13. WKUP\_GPIO[0-1] Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
WKUP_GPIO0	WKUP_PSC0	PD0	LPSC3	WKUP_CBASS0
WKUP_GPIO1	WKUP_PSC0	PD0	LPSC3	WKUP_CBASS0

**Table 12-14. WKUP\_GPIO[0-1] Clocks and Resets**

Clocks
--------

**Table 12-14. WKUP\_GPIO[0-1] Clocks and Resets (continued)**

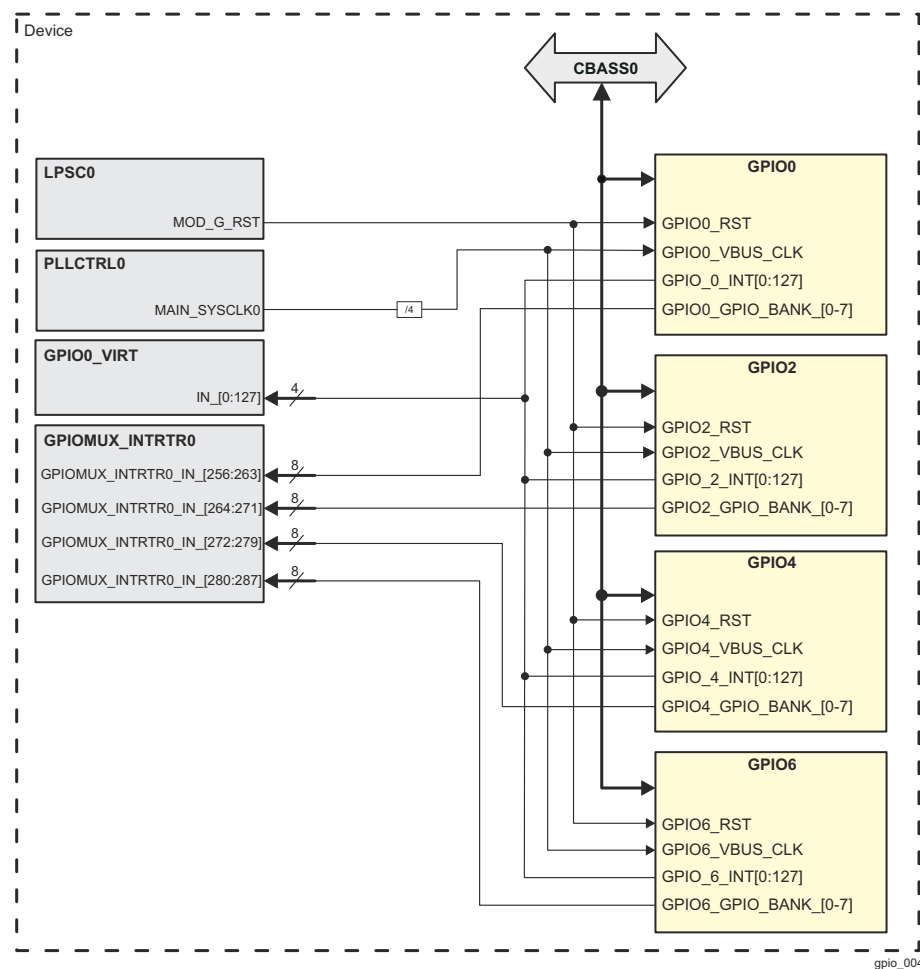
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
WKUP_GPIO0	WKUP_GPIO0_VBUS_CLK	MCU_SYSCLK0/6	WKUP_PLLCTRL0	WKUP_GPIO0 Functional and Interface clock. Output of multiplexer, see <a href="#">Figure 12-10</a> , WKUP_GPIO0 Integration. Multiplexers control is provided via <a href="#">Figure 12-10</a> [1-0] WAKE_CLK_SEL bit field in <i>Control Module (CTRL_MMR)</i> .
		CLK_32K_RC	WKUP_RC_OSC_12M	
		CLK_12M_RC	WKUP_RC_OSC_12M	
WKUP_GPIO1	WKUP_GPIO1_VBUS_CLK	MCU_SYSCLK0/6	WKUP_PLLCTRL0	WKUP_GPIO1 Functional and Interface clock. Output of multiplexer, see <a href="#">Figure 12-10</a> , WKUP_GPIO0 Integration. Multiplexers control is provided via <a href="#">Figure 12-10</a> [1-0] WAKE_CLK_SEL bit field in <i>Control Module (CTRL_MMR)</i> .
		CLK_32K_RC	WKUP_RC_OSC_12M	
		CLK_12M_RC	WKUP_RC_OSC_12M	
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
WKUP_GPIO0	WKUP_GPIO0_RST	MOD_G_RST	LPSC3	WKUP_GPIO0 reset
WKUP_GPIO1	WKUP_GPIO1_RST	MOD_G_RST	LPSC3	WKUP_GPIO1 reset

**Table 12-15. WKUP\_GPIO[0-1] Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
WKUP_GPIO0	WKUP_GPIO0_GPIO_LVL_0	WKUP_DMSC0_INTR_IN_28	WKUP_DMSC0	WKUP_GPIO0 interrupt request	Level
	WKUP_GPIO0_INT[0:83]	IN_[0:83]	WKUP_GPIO_VIRT	WKUP_GPIO0 pins[0:83] interrupt request	Pulse
	WKUP_GPIO0_GPIO_BANK_0	WKUP_GPIOMUX_INTRTR0_IN_103	WKUP_GPIOMUX_INTRTR0	WKUP_GPIO0 bank0 interrupt request	Pulse
	WKUP_GPIO0_GPIO_BANK_1	WKUP_GPIOMUX_INTRTR0_IN_104	WKUP_GPIOMUX_INTRTR0	WKUP_GPIO0 bank1 interrupt request	Pulse
	WKUP_GPIO0_GPIO_BANK_2	WKUP_GPIOMUX_INTRTR0_IN_105	WKUP_GPIOMUX_INTRTR0	WKUP_GPIO0 bank2 interrupt request	Pulse
	WKUP_GPIO0_GPIO_BANK_3	WKUP_GPIOMUX_INTRTR0_IN_106	WKUP_GPIOMUX_INTRTR0	WKUP_GPIO0 bank3 interrupt request	Pulse
	WKUP_GPIO0_GPIO_BANK_4	WKUP_GPIOMUX_INTRTR0_IN_107	WKUP_GPIOMUX_INTRTR0	WKUP_GPIO0 bank4 interrupt request	Pulse
	WKUP_GPIO0_GPIO_BANK_5	WKUP_GPIOMUX_INTRTR0_IN_108	WKUP_GPIOMUX_INTRTR0	WKUP_GPIO0 bank5 interrupt request	Pulse
WKUP_GPIO1	WKUP_GPIO1_GPIO_LVL_0	WKUP_DMSC0_INTR_IN_29	WKUP_DMSC0	WKUP_GPIO1 interrupt request	Level
	WKUP_GPIO1_INT[0:83]	IN_[0:83]	WKUP_GPIO_VIRT	WKUP_GPIO1 pins[0:83] interrupt request	Pulse
	WKUP_GPIO1_GPIO_BANK_0	WKUP_GPIOMUX_INTRTR0_IN_112	WKUP_GPIOMUX_INTRTR0	WKUP_GPIO1 bank0 interrupt request	Pulse
	WKUP_GPIO1_GPIO_BANK_1	WKUP_GPIOMUX_INTRTR0_IN_113	WKUP_GPIOMUX_INTRTR0	WKUP_GPIO1 bank1 interrupt request	Pulse
	WKUP_GPIO1_GPIO_BANK_2	WKUP_GPIOMUX_INTRTR0_IN_114	WKUP_GPIOMUX_INTRTR0	WKUP_GPIO1 bank2 interrupt request	Pulse
	WKUP_GPIO1_GPIO_BANK_3	WKUP_GPIOMUX_INTRTR0_IN_115	WKUP_GPIOMUX_INTRTR0	WKUP_GPIO1 bank3 interrupt request	Pulse
	WKUP_GPIO1_GPIO_BANK_4	WKUP_GPIOMUX_INTRTR0_IN_116	WKUP_GPIOMUX_INTRTR0	WKUP_GPIO1 bank4 interrupt request	Pulse
	WKUP_GPIO1_GPIO_BANK_5	WKUP_GPIOMUX_INTRTR0_IN_117	WKUP_GPIOMUX_INTRTR0	WKUP_GPIO1 bank5 interrupt request	Pulse

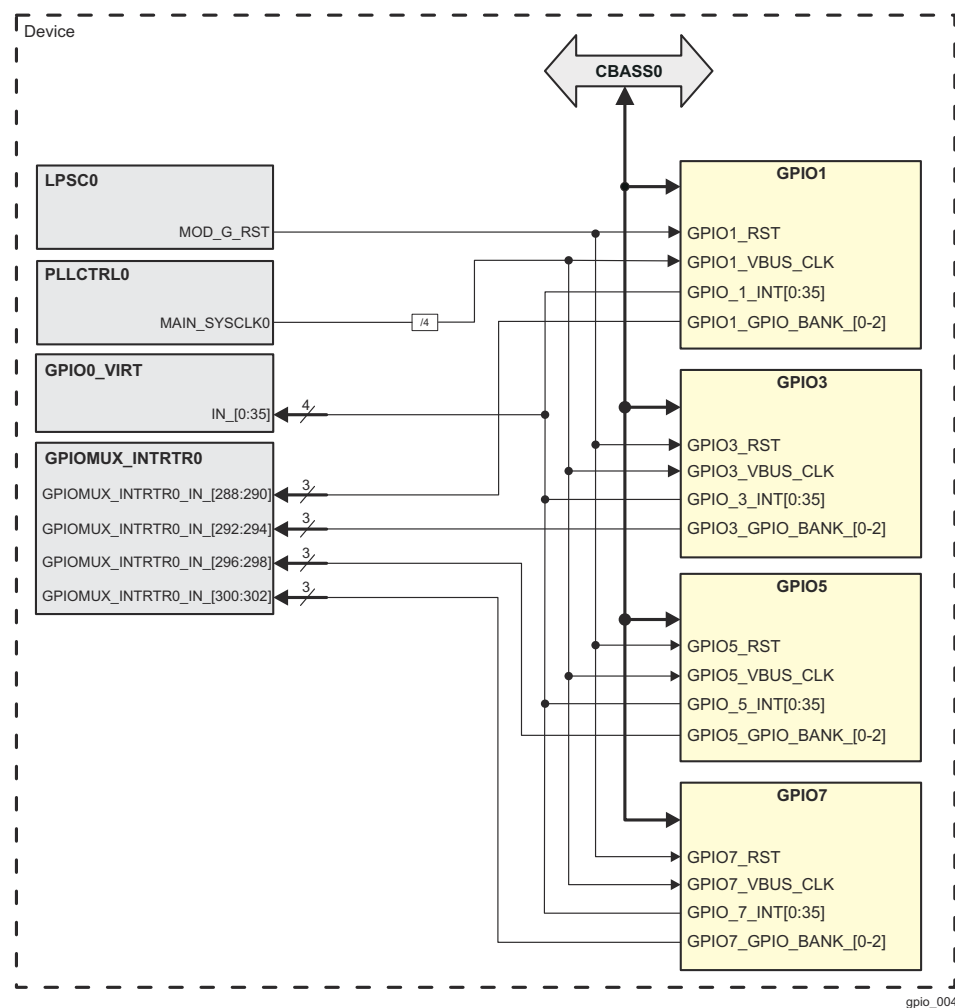
### 12.1.2.3.2 GPIO Integration in MAIN Domain

There are eight GPIO modules integrated in the device MAIN domain - GPIO0, GPIO1, GPIO2, GPIO3, GPIO4, GPIO5, GPIO6, GPIO7. [Figure 12-11](#) and [Figure 12-12](#) shows the integration of GPIO[0-7].



A. n = 0, 2, 4, 6

**Figure 12-11. GPIO Integration**



gpio\_004

A.  $m = 1, 3, 5, 7$

**Figure 12-12. GPIOm Integration**

Table 12-16 through summarize the integration of GPIO[0-7] in the device MAIN domain.

**Table 12-16. GPIO[0-7] Integration Attributes <sup>(1)</sup>**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect

**Table 12-16. GPIO[0-7] Integration Attributes <sup>(1)</sup> (continued)**

GPIO0	PSC0	PD0	LPSC0	CBASS0
GPIO1	PSC0	PD0	LPSC0	CBASS0
GPIO2	PSC0	PD0	LPSC0	CBASS0
GPIO3	PSC0	PD0	LPSC0	CBASS0
GPIO4	PSC0	PD0	LPSC0	CBASS0
GPIO5	PSC0	PD0	LPSC0	CBASS0
GPIO6	PSC0	PD0	LPSC0	CBASS0
GPIO7	PSC0	PD0	LPSC0	CBASS0

(1) n = 0, 2, 4, 6

**Table 12-17. GPIO[0-7] Clocks and Resets <sup>(1)</sup>**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
GPIO0	GPIO0_VBUS_CLK	SYSCLK0/4	PLLCTRL0	GPIO0 functional and interface clock
GPIO1	GPIO1_VBUS_CLK	SYSCLK0/4	PLLCTRL0	GPIO1 functional and interface clock
GPIO2	GPIO2_VBUS_CLK	SYSCLK0/4	PLLCTRL0	GPIO2 functional and interface clock
GPIO3	GPIO3_VBUS_CLK	SYSCLK0/4	PLLCTRL0	GPIO3 functional and interface clock
GPIO4	GPIO4_VBUS_CLK	SYSCLK0/4	PLLCTRL0	GPIO4 functional and interface clock
GPIO5	GPIO5_VBUS_CLK	SYSCLK0/4	PLLCTRL0	GPIO5 functional and interface clock
GPIO6	GPIO6_VBUS_CLK	SYSCLK0/4	PLLCTRL0	GPIO6 functional and interface clock
GPIO7	GPIO7_VBUS_CLK	SYSCLK0/4	PLLCTRL0	GPIO7 functional and interface clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
GPIO0	GPIO0_RST	MOD_G_RST	LPSC0	GPIO0 reset
GPIO1	GPIO1_RST	MOD_G_RST	LPSC0	GPIO1 reset
GPIO2	GPIO2_RST	MOD_G_RST	LPSC0	GPIO2 reset
GPIO3	GPIO3_RST	MOD_G_RST	LPSC0	GPIO3 reset
GPIO4	GPIO4_RST	MOD_G_RST	LPSC0	GPIO4 reset
GPIO5	GPIO5_RST	MOD_G_RST	LPSC0	GPIO5 reset
GPIO6	GPIO6_RST	MOD_G_RST	LPSC0	GPIO6 reset
GPIO7	GPIO7_RST	MOD_G_RST	LPSC0	GPIO7 reset

(1) n = 0, 2, 4, 6



**Table 12-18. GPIO[0-7] Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
GPIO0	GPIO_0_INT[0:127]	IN_[0:127]	GPIO0_VIRT	GPIO0 pins[0:127] interrupt request	Pulse
	GPIO0_GPIO_BANK_0	GPIOMUX_INTRTR0_IN_256	GPIOMUX_INTRTR0	GPIO0 bank0 interrupt request	Pulse
	GPIO0_GPIO_BANK_1	GPIOMUX_INTRTR0_IN_257	GPIOMUX_INTRTR0	GPIO0 bank1 interrupt request	Pulse
	GPIO0_GPIO_BANK_2	GPIOMUX_INTRTR0_IN_258	GPIOMUX_INTRTR0	GPIO0 bank2 interrupt request	Pulse
	GPIO0_GPIO_BANK_3	GPIOMUX_INTRTR0_IN_259	GPIOMUX_INTRTR0	GPIO0 bank3 interrupt request	Pulse
	GPIO0_GPIO_BANK_4	GPIOMUX_INTRTR0_IN_260	GPIOMUX_INTRTR0	GPIO0 bank4 interrupt request	Pulse
	GPIO0_GPIO_BANK_5	GPIOMUX_INTRTR0_IN_261	GPIOMUX_INTRTR0	GPIO0 bank5 interrupt request	Pulse
	GPIO0_GPIO_BANK_6	GPIOMUX_INTRTR0_IN_262	GPIOMUX_INTRTR0	GPIO0 bank6 interrupt request	Pulse
	GPIO0_GPIO_BANK_7	GPIOMUX_INTRTR0_IN_263	GPIOMUX_INTRTR0	GPIO0 bank7 interrupt request	Pulse
GPIO1	GPIO_1_INT[0:35]	IN_[0:35]	GPIO1_VIRT	GPIO1 pins[0:35] interrupt request	Pulse
	GPIO1_GPIO_BANK_0	GPIOMUX_INTRTR0_IN_288	GPIOMUX_INTRTR0	GPIO1 bank0 interrupt request	Pulse
	GPIO1_GPIO_BANK_1	GPIOMUX_INTRTR0_IN_289	GPIOMUX_INTRTR0	GPIO1 bank1 interrupt request	Pulse
	GPIO1_GPIO_BANK_2	GPIOMUX_INTRTR0_IN_290	GPIOMUX_INTRTR0	GPIO1 bank2 interrupt request	Pulse
GPIO2	GPIO_2_INT[0:127]	IN_[0:127]	GPIO0_VIRT	GPIO2 pins[0:127] interrupt request	Pulse
	GPIO2_GPIO_BANK_0	GPIOMUX_INTRTR0_IN_264	GPIOMUX_INTRTR0	GPIO2 bank0 interrupt request	Pulse
	GPIO2_GPIO_BANK_1	GPIOMUX_INTRTR0_IN_265	GPIOMUX_INTRTR0	GPIO2 bank1 interrupt request	Pulse
	GPIO2_GPIO_BANK_2	GPIOMUX_INTRTR0_IN_266	GPIOMUX_INTRTR0	GPIO2 bank2 interrupt request	Pulse
	GPIO2_GPIO_BANK_3	GPIOMUX_INTRTR0_IN_267	GPIOMUX_INTRTR0	GPIO2 bank3 interrupt request	Pulse
	GPIO2_GPIO_BANK_4	GPIOMUX_INTRTR0_IN_268	GPIOMUX_INTRTR0	GPIO2 bank4 interrupt request	Pulse
	GPIO2_GPIO_BANK_5	GPIOMUX_INTRTR0_IN_269	GPIOMUX_INTRTR0	GPIO2 bank5 interrupt request	Pulse
	GPIO2_GPIO_BANK_6	GPIOMUX_INTRTR0_IN_270	GPIOMUX_INTRTR0	GPIO2 bank6 interrupt request	Pulse
	GPIO2_GPIO_BANK_7	GPIOMUX_INTRTR0_IN_271	GPIOMUX_INTRTR0	GPIO2 bank7 interrupt request	Pulse
GPIO3	GPIO_3_INT[0:35]	IN_[0:35]	GPIO1_VIRT	GPIO3 pins[0:35] interrupt request	Pulse
	GPIO3_GPIO_BANK_0	GPIOMUX_INTRTR0_IN_292	GPIOMUX_INTRTR0	GPIO3 bank0 interrupt request	Pulse
	GPIO3_GPIO_BANK_1	GPIOMUX_INTRTR0_IN_293	GPIOMUX_INTRTR0	GPIO3 bank1 interrupt request	Pulse
	GPIO3_GPIO_BANK_2	GPIOMUX_INTRTR0_IN_294	GPIOMUX_INTRTR0	GPIO3 bank2 interrupt request	Pulse
GPIO4	GPIO_4_INT[0:127]	IN_[0:127]	GPIO0_VIRT	GPIO4 pins[0:127] interrupt request	Pulse
	GPIO4_GPIO_BANK_0	GPIOMUX_INTRTR0_IN_272	GPIOMUX_INTRTR0	GPIO4 bank0 interrupt request	Pulse
	GPIO4_GPIO_BANK_1	GPIOMUX_INTRTR0_IN_273	GPIOMUX_INTRTR0	GPIO4 bank1 interrupt request	Pulse
	GPIO4_GPIO_BANK_2	GPIOMUX_INTRTR0_IN_274	GPIOMUX_INTRTR0	GPIO4 bank2 interrupt request	Pulse

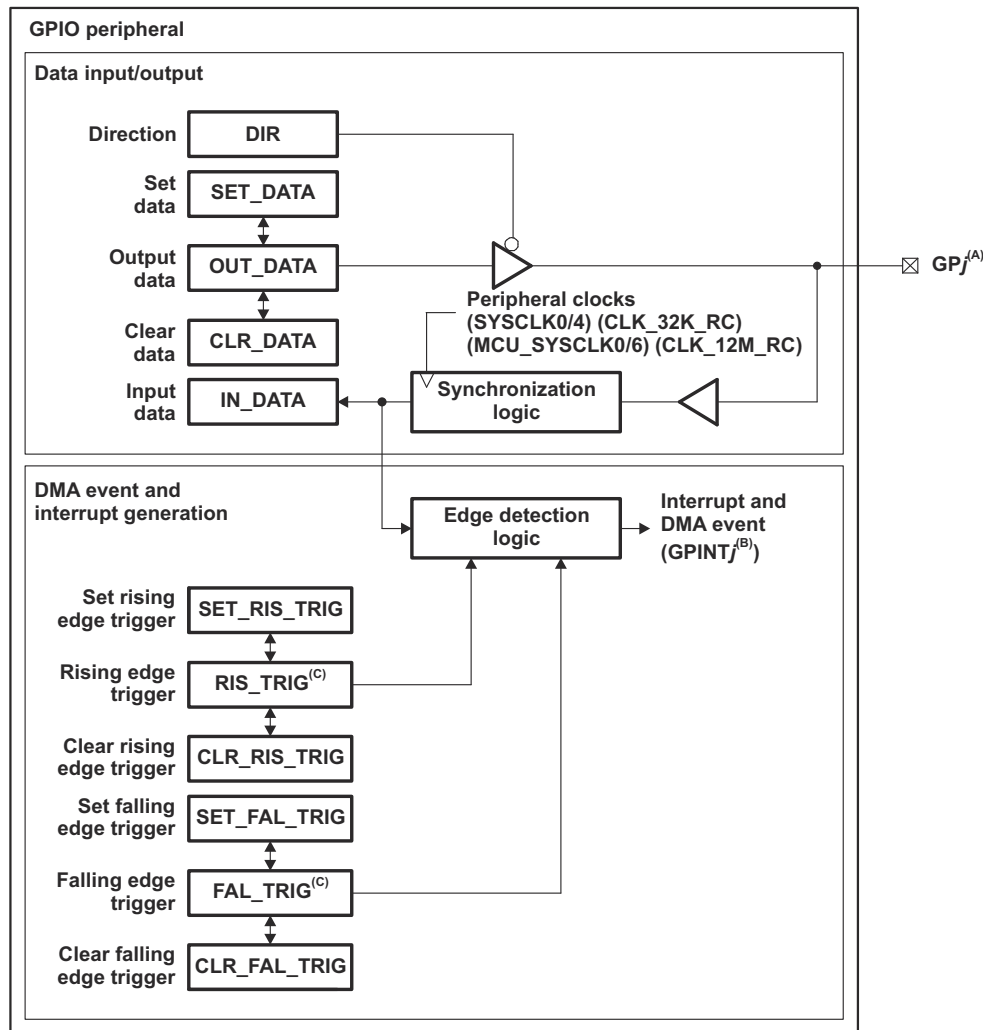
**Table 12-18. GPIO[0-7] Hardware Requests (continued)**

	GPIO4_GPIO_BANK_3	GPIOMUX_INTRTR0_IN_275	GPIOMUX_INTRTR0	GPIO4 bank3 interrupt request	Pulse
	GPIO4_GPIO_BANK_4	GPIOMUX_INTRTR0_IN_276	GPIOMUX_INTRTR0	GPIO4 bank4 interrupt request	Pulse
	GPIO4_GPIO_BANK_5	GPIOMUX_INTRTR0_IN_277	GPIOMUX_INTRTR0	GPIO4 bank5 interrupt request	Pulse
	GPIO4_GPIO_BANK_6	GPIOMUX_INTRTR0_IN_278	GPIOMUX_INTRTR0	GPIO4 bank6 interrupt request	Pulse
	GPIO4_GPIO_BANK_7	GPIOMUX_INTRTR0_IN_279	GPIOMUX_INTRTR0	GPIO4 bank7 interrupt request	Pulse
GPIO5	GPIO_5_INT[0:35]	IN_[0:35]	GPIO1_VIRT	GPIO5 pins[0:35] interrupt request	Pulse
	GPIO5_GPIO_BANK_0	GPIOMUX_INTRTR0_IN_296	GPIOMUX_INTRTR0	GPIO5 bank0 interrupt request	Pulse
	GPIO5_GPIO_BANK_1	GPIOMUX_INTRTR0_IN_297	GPIOMUX_INTRTR0	GPIO5 bank1 interrupt request	Pulse
	GPIO5_GPIO_BANK_2	GPIOMUX_INTRTR0_IN_298	GPIOMUX_INTRTR0	GPIO5 bank2 interrupt request	Pulse
GPIO6	GPIO_6_INT[0:127]	IN_[0:127]	GPIO0_VIRT	GPIO6 pins[0:127] interrupt request	Pulse
	GPIO6_GPIO_BANK_0	GPIOMUX_INTRTR0_IN_280	GPIOMUX_INTRTR0	GPIO6 bank0 interrupt request	Pulse
	GPIO6_GPIO_BANK_1	GPIOMUX_INTRTR0_IN_281	GPIOMUX_INTRTR0	GPIO6 bank1 interrupt request	Pulse
	GPIO6_GPIO_BANK_2	GPIOMUX_INTRTR0_IN_282	GPIOMUX_INTRTR0	GPIO6 bank2 interrupt request	Pulse
	GPIO6_GPIO_BANK_3	GPIOMUX_INTRTR0_IN_283	GPIOMUX_INTRTR0	GPIO6 bank3 interrupt request	Pulse
	GPIO6_GPIO_BANK_4	GPIOMUX_INTRTR0_IN_284	GPIOMUX_INTRTR0	GPIO6 bank4 interrupt request	Pulse
	GPIO6_GPIO_BANK_5	GPIOMUX_INTRTR0_IN_285	GPIOMUX_INTRTR0	GPIO6 bank5 interrupt request	Pulse
	GPIO6_GPIO_BANK_6	GPIOMUX_INTRTR0_IN_286	GPIOMUX_INTRTR0	GPIO6 bank6 interrupt request	Pulse
	GPIO6_GPIO_BANK_7	GPIOMUX_INTRTR0_IN_287	GPIOMUX_INTRTR0	GPIO6 bank7 interrupt request	Pulse
GPIO7	GPIO_7_INT[0:35]	IN_[0:35]	GPIO1_VIRT	GPIO7 pins[0:35] interrupt request	Pulse
	GPIO7_GPIO_BANK_0	GPIOMUX_INTRTR0_IN_300	GPIOMUX_INTRTR0	GPIO7 bank0 interrupt request	Pulse
	GPIO7_GPIO_BANK_1	GPIOMUX_INTRTR0_IN_301	GPIOMUX_INTRTR0	GPIO7 bank1 interrupt request	Pulse
	GPIO7_GPIO_BANK_2	GPIOMUX_INTRTR0_IN_302	GPIOMUX_INTRTR0	GPIO7 bank2 interrupt request	Pulse

## 12.1.2.4 GPIO Functional Description

### 12.1.2.4.1 GPIO Block Diagram

Figure 12-13 shows the general-purpose interface block diagram.



gpio\_005

- A. Some of the GPj pins are muxed with other device signals. For details, see the device-specific Datasheet.
- B. All GPINTj can be used as host CPU interrupts and synchronization events to the DMA.
- C. The RIS\_TRIG and FAL\_TRIG registers are internal to the GPIO module and are not visible to the host CPU.

#### Note

The synchronization logic and tristate buffer are present in the SoC pinmux logic.

**Figure 12-13. GPIO Block Diagram**

### 12.1.2.4.2 GPIO Function

Each GPIO pin (GPj) can be independently configured as either an input or an output using the GPIO direction registers. The GPIO direction register (DIR) specifies the direction of each GPIO signal. Logic 0 indicates the GPIO pin is configured as output, and logic 1 indicates input.

When configured as output, writing a 0x1 to a bit in the set data register drives the corresponding GPj to a logic-high state. Writing a 0x1 to a bit in the clear data register drives the corresponding GPj to a logic-low state. The output state of each GPj can also be directly controlled by writing to the output data register.

For example, to set GP8 to a logic-high state, the software can perform one of the following:

- Write 100h to the GPIO\_SET\_DATA01 register.
- Write 0h to the GPIO\_DIR01 register to configure as output pin.
- Read in GPIO\_OUT\_DATA01 register, change bit 8 to 0x1, and write the new value back to GPIO\_OUT\_DATA01.

To set GP8 to a logic-low state, the software can perform one of the following:

- Write 100h to the GPIO\_CLR\_DATA01 register.
- Write 0h to the GPIO\_DIR01 register to configure as output pin.
- Read in GPIO\_OUT\_DATA01 register, change bit 8 to 0x0, and write the new value back to GPIO\_OUT\_DATA01.

Note that writing a 0x0 to bits in the set data and clear data registers does not affect the GPIO pin state.

Also, for GPIO pins configured as input, writing to the set data, clear data, or output data registers does not affect the pin state.

For a GPIO pin configured as input, reading the input data register (IN\_DATA) will return the pin state. Reading the SET\_DATA register or the CLR\_DATA data register will return the value in OUT\_DATA, not the actual pin state. The pin state is available by reading the input data register. Note that when the direction is configured as input, the output state is determined by software's programming set/clear/output registers, and may not agree with the pin state, which is driven by an external device.

#### **12.1.2.4.3 GPIO Interrupt and Event Generation**

Each GPIO pin (GPj) can be configured to generate a host CPU interrupt (GPINTj) or a synchronization event to the DMA (GPINTj). Configuration is on per-bank basis. Each bit of the BINTEN parameter dictates YES/NO option for each bank. Bit 0 controls bank 0, bit 1 controls bank 1, and so on.

The interrupt can be generated on the rising-edge, falling-edge, or on both edges of the GPIO signal. The edge detection logic is synchronized to the GPIO peripheral clock.

The direction of the GPIO pin does not need to be input when using the pin to generate the interrupt or DMA event. When the GPIO pin is configured as input, transitions on the pin trigger interrupts or DMA events. When the GPIO pin is configured as output, software can toggle the GPIO output register to change the pin state and in turn trigger the interrupt or DMA event.

Note that the direction of the pin need not be input for interrupt generation to work. When the GPIO pin is configured as input, transitions on the pin trigger interrupts. When the GPIO pin is configured as output, firmware can toggle the GPIO output register to change the pin state, and in turn trigger interrupts.

Each interrupt output of GPIO signal are available at the module boundary. Each group of 16 GPIO\_INTR\_INTj signals also has their masked interrupt outputs ORed together to generate a per bank interrupt, available at the module boundary. The idea is to either connect individual interrupts or per bank interrupts to the system interrupt controller.

##### **12.1.2.4.3.1 Interrupt Enable (per Bank)**

The GPIO\_BINTEN register provides interrupt enable/disable feature for each bank of 16 GPINT signals.

##### **12.1.2.4.3.2 Trigger Configuration (per Bit)**

Two internal registers, RIS\_TRIG and FAL\_TRIG, specify which edge of the GPj signal generates an interrupt or DMA event. Each bit in these two registers corresponds to a GPj pin. [Table 12-19](#) describes the host CPU interrupt and DMA event generation of GPj pin based on the bit settings of the RIS\_TRIG and FAL\_TRIG registers.

**Table 12-19. GPIO Interrupt and DMA Event Configuration Options**

RIS_TRIG Bit n	FAL_TRIG Bit n	Host CPU Interrupt and DMA Event Generation
0	0	GPINTj interrupt and DMA event is disabled
0	1	GPINTj interrupt and DMA event is triggered on falling edge of GPj signal
1	0	GPINTj interrupt and DMA event is triggered on rising edge of GPj signal
1	1	GPINTj interrupt and DMA event is triggered on both rising and falling edge of GPj signal

The RIS\_TRIG and FAL\_TRIG registers are not directly accessible or visible to the host CPU. These registers are accessed indirectly through four registers: SET\_RIS\_TRIG, CLR\_RIS\_TRIG, SET\_FAL\_TRIG, and CLR\_FAL\_TRIG. Writing 1 to a bit on the SET\_RIS\_TRIG register sets the corresponding bit on the RIS\_TRIG register. Writing 1 to a bit of the CLR\_RIS\_TRIG register clears the corresponding bit on the RIS\_TRIG register. Writing to the SET\_FAL\_TRIG and CLR\_FAL\_TRIG registers works the same way on the FAL\_TRIG register.

Reading the SET\_RIS\_TRIG or CLR\_RIS\_TRIG register returns the value of the RIS\_TRIG register. Reading from the SET\_FAL\_TRIG or CLR\_FAL\_TRIG register returns the value of the FAL\_TRIG register.

To use the GPIO pins as sources for host CPU interrupts and DMA events, the associated bank interrupt enable register bit in GPIO\_BINTEN must also be set to 1. For example, to enable GPIO0\_19 (which is in bank 1), GPIO\_BINTEN[1] = 1 should be set to enable interrupts for bank 1.

#### 12.1.2.4.3.3 Interrupt Status and Clear (per Bit)

The INTSTAT registers provide interrupt status upon reading, and interrupt clear feature upon writing 1 to the corresponding bit position(s). Upon receiving an interrupt, the ISR can examine the interrupt status and clear the processed interrupts.

#### Note

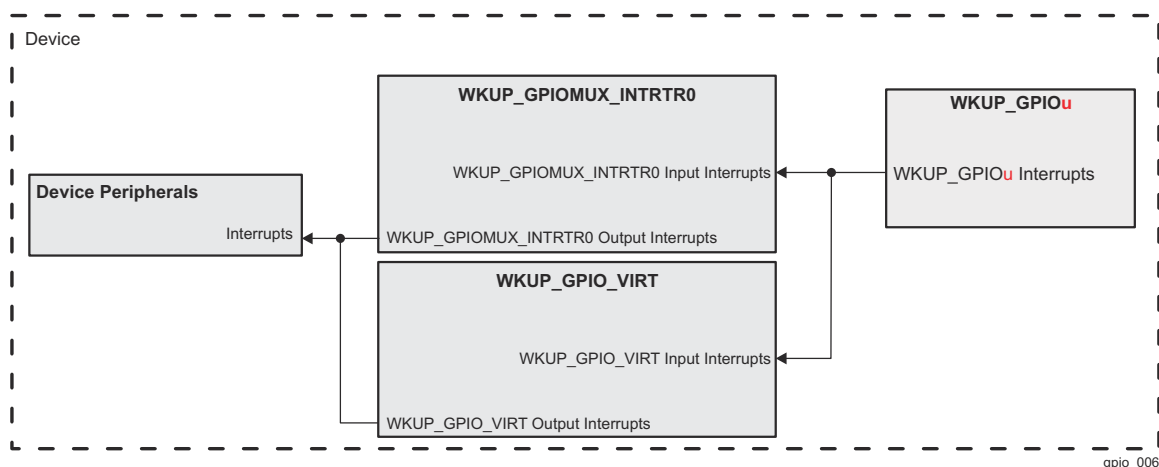
The GPIO module generates an interrupt pulse on the individual GPINT interrupt in response to each occurrence of the specified edge condition. Therefore, for GPINT signals having their interrupts routed directly to the interrupt controller, it is not necessary to clear the status bits in this module. The interrupt status and clear register is a facility for the per-bank interrupt connection.

#### 12.1.2.4.4 GPIO Interrupt Connectivity

Because this device muxes GPIO signals with other functional signals, the availability of any particular GPIO and hence the usability of its associated interrupt will change based on the use case pin muxing. The large number of possible GPIO interrupt sources makes it impractical to route all interrupt events to each processing element. Since most applications do not typically require a large number of GPIO interrupts, the interrupt uncertainty is resolved by mapping all GPIO interrupts to a series of event muxes implemented using Interrupt Router (IntRouter) modules. These muxes allow any one of the available GPIO interrupts to be selected and passed on as an event to the various processor interrupt controllers and DMA controllers. Event selection is controlled through associated registers within each IntRouter.

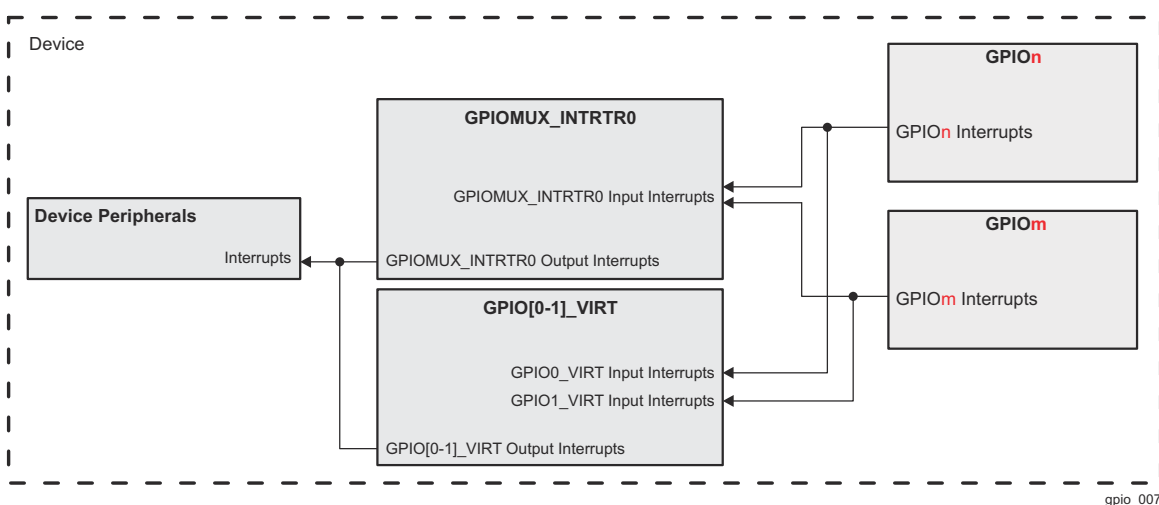
The GPIO bank interrupts already represent a consolidation of the 16 GPIO interrupts associated with each bank and are routed directly to various interrupt controllers rather than through the GPIO IntRouters.

[Figure 12-14](#) and [Figure 12-15](#) shows the GPIO Interrupt Routers connectivity. Refer to *Interrupt Routers*, for more details on the GPIO Interrupt Routers connectivity.



A.  $u = 0, 1$

**Figure 12-14. WKUP\_GPIO Interrupt Router Connectivity**



A.  $n = 0, 2, 4, 6$

B.  $m = 1, 3, 5, 7$

**Figure 12-15. GPIO Interrupt Router Connectivity**

#### 12.1.2.4.5 GPIO DeepSleep Mode

During DeepSleep power saving mode, the GPIO functional clock is powered down. This would the WKUP\_GPIOu ( $u = 0, 1$ ) modules from detecting transitions on GPIO pins to be used as wakeup from DeepSleep events. In order to correct this issue, special clocking and controls are implemented as part of the WKUP\_GPIOu ( $u = 0, 1$ ) integration to allow GPIO transition to remain detectable.

A clock MUX is provided to allow the GPIO clock to be switched to an on-chip clock source prior to gating of the standard clock source (MCU\_SYSCLOCK0/6) and power-down of the off-chip WKUP\_HFOSC0 oscillator. This clock MUX is controlled by the WKUP\_CTRL\_MMR0 register bits.

Because there is no asynchronous bridge between the WKUP\_GPIOu ( $u = 0, 1$ ) modules and the WKUP\_CBASS0, the module register may only be accessed when it is clocked using the synchronous MCU\_SYSCLOCK0/6 clock source. Prior to switching the clock source to prepare for DeepSleep, all clock accesses to the WKUP\_GPIOu ( $u = 0, 1$ ) modules must be blocked through the dedicated LPSC5 using a clock stop request. Note that when wakeup functionality for the WKUP\_GPIOu ( $u = 0, 1$ ) is enabled, (through a WKUP\_CTRL\_MMR0 registers), this clock stop request is not actually propagate to the WKUP\_GPIOu ( $u = 0,$

1) module (or stop its clock). Instead, it will be fed back to the LPSC3 as a clock stop acknowledged. This will cause the associated WKUP\_CBASS0 to route all future WKUP\_GPIO register accesses to a null endpoint. The WKUP\_GPIOu (u = 0, 1) LPSC3 must be maintained in clock stop mode until MCU\_SYSClk0/6 is fully restored upon wakeup from DeepSleep and the WKUP\_GPIOu (u = 0, 1) clock MUX has finished switching back the normal clock source.

The following are the steps required (expected to be performed by the DMSC) to enable WKUP\_GPIOu (u = 0, 1) wakeup events prior to DeepSleep:

- Determine which events are to be used for DeepSleep wakeup and enable the interrupts through the corresponding GPIO\_SET\_RIS\_TRIG and GPIO\_SET\_FALL\_TRIG registers. Disable non-wakeup interrupts through the corresponding GPIO\_CLR\_RIS\_TRIG and GPIO\_CLR\_FALL\_TRIG registers
- Set the CTRLMMR\_WKUP\_GPIO\_CTRL[0] WAKEN bit to enable DeepSleep LPSC3 operation.
- Disable further access to the WKUP\_GPIOu (u = 0, 1) through the LPSC3 clock stop request.
- Change the WKUP\_GPIOu (u = 0, 1) clock source to CLK\_32K\_RC or CLK\_12M\_RC as desired by setting the CTRLMMR\_WKUP\_GPIO\_CLKSEL[1-0] WAKE\_CLK\_SEL bit field in the WKUP\_CTRL\_MMR0. (Wait at least one clock cycle of the new clock to ensure switch has occurred.)
- Perform normal DeepSleep entry routine (enable DMSC wakeup events, clock gate MCU clocks, powerdown oscillators, etc.)

Once in DeepSleep mode, any GPIO transition (low to high or high to low) intended to cause a wakeup must be maintained at its new value for at least 0x2 of the selected functional clock (CLK\_32K\_RC or CLK\_12M\_RC) maximum periods to ensure proper detection. (Note that process variance of the RC clocks must be taken into account). The detected event will be latched in the GPIO\_INTSTAT register and the WKUP\_GPIOMUX\_INTRTR0 wakeup event sent to the DMSC to trigger the wakeup FSM.

After wakeup, the DMSC must restore the MCU\_SYSClk0/6 operation to allow the wakeup source event to be latched within the DMSC. Once this is done the DMSC may restore access to the WKUP\_GPIOu (u = 0, 1) module by reversing the steps above:

- Change the WKUP\_GPIOu (u = 0, 1) clock source back to MCU\_SYSClk0/6 using the CTRLMMR\_WKUP\_GPIO\_CLKSEL[1-0] WAKE\_CLK\_SEL bit field in the WKUP\_CTRL\_MMR0. (Wait at least 1 of the previously selected clock cycles to ensure glitch-free switch is complete.)
- Re-enable access to the WKUP\_GPIOu (u = 0, 1) through the LPSC3 clock stop request.

The specific GPIO source of the wakeup event may now be determined by reading the GPIO\_INTSTAT register and cleared by writing 0x1 to the set bits of the same register.

#### **12.1.2.4.6 GPIO Emulation Halt Operation**

The GPIO peripheral is not affected by emulation halts.



## 12.1.2.5 GPIO Programming Guide

### 12.1.2.5.1 GPIO Low-Level Programming Models

#### 12.1.2.5.1.1 Global Initialization

##### 12.1.2.5.1.1.1 Surrounding Modules Global Initialization

This section identifies the requirements for initializing the surrounding modules when the general-purpose interface module is to be used for the first time after a device reset. This initialization of surrounding modules is based on the environment and integration of the general-purpose interface. For more information, see *GPIO Environment*, and *GPIO Integration*.

**Table 12-20. Global Initialization of Surrounding Modules**

Surrounding Modules	Comments
LPSC0	Module reset must be enabled. For more information about the module configuration, see <i>Reset</i> .
LPSC3	Module reset must be enabled. For more information about the module configuration, see <i>Reset</i> .
PLLCTRL0	Interface and functional clocks must be enabled. For more information about the module configuration, see <i>Clocking</i> .
WKUP_PLLCTRL0	Interface and functional clocks must be enabled. For more information about the module configuration, see <i>Clocking</i> .
WKUP_RC_OSC_12M	Interface and functional clocks must be enabled. For more information about the module configuration, see <i>Clocking</i> .
GIOMUX_INTRTR0	Device INTCs must be configured to enable the interrupt request generation. For information about enabling GIOMUX_INTRTR0 interrupts, see <i>Interrupts</i> .
WKUP_GIOMUX_INTRTR0	Device INTCs must be configured to enable the interrupt request generation. For information about enabling WKUP_GIOMUX_INTRTR0 interrupts, see <i>Interrupts</i> .
WKUP_GPIO_VIRT	Device INTCs must be configured to enable the interrupt request generation. For information about enabling WKUP_GPIO_VIRT interrupts, see <i>Interrupts</i> .
GPIO0_VIRT	Device INTCs must be configured to enable the interrupt request generation. For information about enabling GPIO0_VIRT interrupts, see <i>Interrupts</i> .
GPIO1_VIRT	Device INTCs must be configured to enable the interrupt request generation. For information about enabling GPIO1_VIRT interrupts, see <i>Interrupts</i> .
DMSC_WKUP_0	Device INTCs must be configured to enable the interrupt request generation. For information about enabling DMSC_WKUP_0 interrupts, see <i>Interrupts</i> .
Interconnects	For information about the WKUP_CBASS0, and CBASS0 interconnects configuration, see <i>System Interconnect</i> .

##### 12.1.2.5.1.1.2 GPIO Module Global Initialization

This procedure initializes the general-purpose Interface module after a power-on reset (POR) or software reset.

**Table 12-21. GPIO Global Initialization**

Step	Register/Bit Field/Programming Model	Value
Configure GPIO channels as input or output of the corresponding bank	DIR	-h
<b>Interrupt requests configuration</b>		
Configure detection events	SET_RIS_TRIG and/or SET_FAL_TRIG	-h
Clear interrupt status	INTSTAT	FFFFh
Enable interrupts for desired banks [0:8]	GPIO_BINTEN[8-0]	-h



### 12.1.2.5.1.2 GPIO Operational Modes Configuration

#### 12.1.2.5.1.2.1 GPIO Read Input Register

**Table 12-22. GPIO Read Input Register**

Step	Register/Bit Field/Programming Model	Value
Read interrupt status of the corresponding bank	INTSTAT	-h
Read input register value	IN_DATA	-h
Clear interrupt status	INTSTAT	FFFF FFFFh

#### 12.1.2.5.1.2.2 GPIO Set Bit Function

**Table 12-23. GPIO Set Bit Function**

Step	Register/Bit Field/Programming Model	Value
Write 1h to set desired bit(s) in SET_DATA register.	SET_DATA	-h

#### 12.1.2.5.1.2.3 GPIO Clear Bit Function

**Table 12-24. GPIO Clear Bit Function**

Step	Register/Bit Field/Programming Model	Value
Write 1h to clear desired bit(s) in CLR_DATA register.	CLR_DATA	-h

### 12.1.3 Inter-Integrated Circuit (I2C) Interface

This section describes the Inter-Integrated Circuit (I2C) module in the device.

#### 12.1.3.1 I2C Overview

The device contains ten multimaster Inter-Integrated Circuit (I2C) controllers each of which provides an interface between a local host (LH), such as an Arm or a Digital Signal Processor (DSP), and any I<sup>2</sup>C-bus-compatible device that connects via the I<sup>2</sup>C serial bus. External components attached to the I<sup>2</sup>C bus can serially transmit and receive up to 8 bits of data to and from the LH device through the 2-wire I<sup>2</sup>C interface.

Each multimaster I2C module can be configured to act like a slave or master I<sup>2</sup>C-compatible device.

The WKUP\_I2C0, MCU\_I2C0, I2C0, and I2C1 controllers have dedicated I<sup>2</sup>C compliant open drain buffers and support high speed mode (up to 3.4 Mbps in 1.8 V mode and up to 400 Kbps in 3.3 V mode). The MCU\_I2C1, I2C2, I2C3, I2C4, I2C5, and I2C6 controllers are multiplexed with standard LVCMOS I/O, connected to emulate open drain, and support fast mode (up to 400 Kbps in 1.8 V/3.3 V mode). The I<sup>2</sup>C emulation is achieved by configuring the LVCMOS buffers to output Hi-Z instead of driving high when transmitting logic 1.

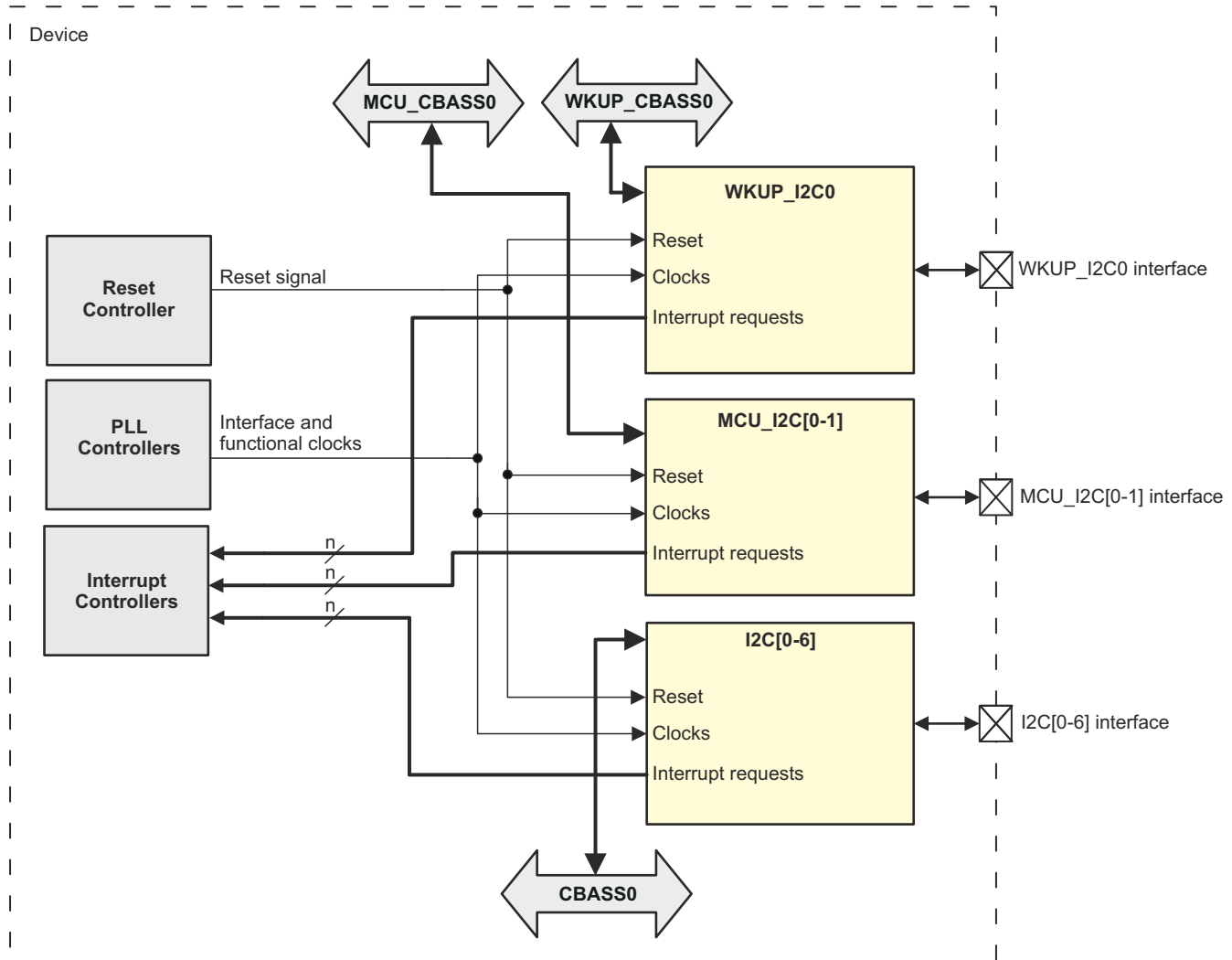
For the specific I/O timing characteristics of the different I2C instances, see the device-specific Datasheet.

[Table 12-25](#) shows I2C modules allocation across device domains.

**Table 12-25. I2C Allocation Across Device Domains**

Instance	Domain		
	WKUP	MCU	MAIN
WKUP_I2C0	✓	-	-
MCU_I2C0	-	✓	-
MCU_I2C1	-	✓	-
I2C0	-	-	✓
I2C1	-	-	✓
I2C2	-	-	✓
I2C3	-	-	✓
I2C4	-	-	✓
I2C5	-	-	✓
I2C6	-	-	✓

[Figure 12-16](#) shows the I2C modules overview.



i2c\_001

**Figure 12-16. I2C Modules Overview**

#### 12.1.3.1.1 I2C Features

Each multicontroller I2C module has the following features:

- Compliant with Philips I<sup>2</sup>C-bus specification version 2.1
- Supports a standard mode (up to 100 Kbps) and fast mode (up to 400 Kbps)
- Supports HS mode (up to 3.4 Mbps) on some instances. See the the device-specific Data Manual for details.
- 7-bit and 10-bit device addressing modes
- General call
- Start/Restart/Stop
- Multicontroller transmitter/target receiver mode
- Multicontroller receiver/target transmitter mode
- Combined controller transmit/receive and receive/transmit mode
- Built-in FIFO for buffered read
- Module enable/disable capability
- Programmable multitarget channel (responds to four separate addresses)
- Programmable clock generation
- 8-bit-wide data access
- Low power consumption

- Support Auto Idle mechanism
- Support Idle Request/Idle Acknowledge handshake mechanism
- Support for asynchronous wakeup mechanism
- Wide interrupt capability

**12.1.3.1.2 I2C Not Supported Features**

- Serial Camera Control Bus (SCCB) Protocol
- DMA Mode
- Full I<sup>2</sup>C electrical compliance (only for MAIN and MCU domain I2C modules)
- Local power management of clock activity
- Debug suspend mode
- Clockstop wakeup interrupt (only for MCU domain I2C modules)

### 12.1.3.2 I2C Environment

The WKUP\_I2C0, MCU\_I2C[0-1], and I2C[0-6] modules are hereinafter referred to as I2C module.

This section describes the I2C external connections (environment).

#### 12.1.3.2.1 I2C Typical Application

Figure 12-17 shows the multicontroller I2C controller and their related connections with I<sup>2</sup>C-compliant devices.

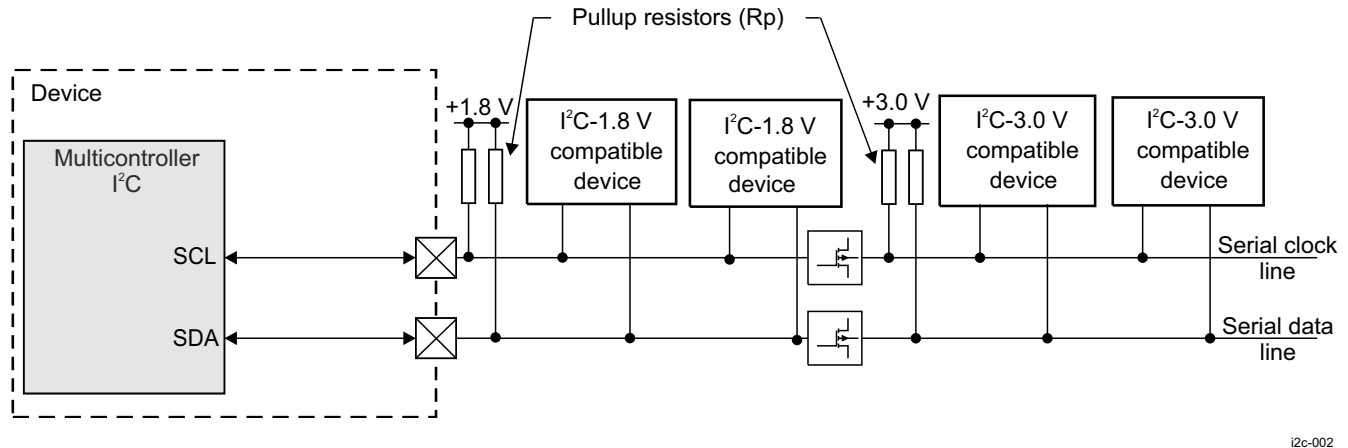


Figure 12-17. I2C and Typical Connections to I2C Devices

#### 12.1.3.2.1.1 I2C Pins for Typical Connections in I2C Mode

Figure 12-18 shows the multicontroller I2C controller pins used for typical connections with I<sup>2</sup>C devices.

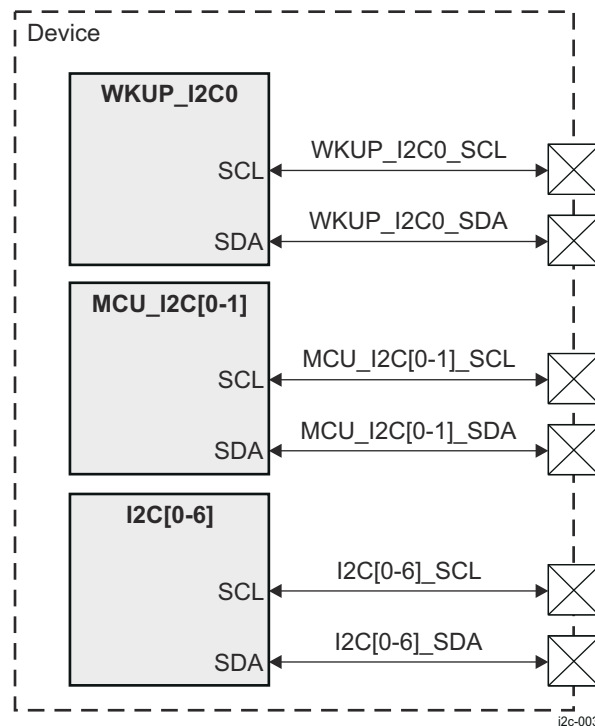


Figure 12-18. I2C Interface Signals

#### 12.1.3.2.1.2 I2C Interface Typical Connections

Table 12-26 describes the I2C I/O signals.

**Table 12-26. I2C I/O Signals**

Module Pin	Device Level Signal	I/O <sup>(1)</sup>	Description	Module Pin Reset Value
<b>WKUP_I2C0</b>				
SCL	WKUP_I2C0_SCL	I/O	I <sup>2</sup> C serial clock line. Open-drain output buffer.	1
SDA	WKUP_I2C0_SDA	I/O	I <sup>2</sup> C serial data line. Open-drain output buffer.	1
<b>MCU_I2C[0-1]</b>				
SCL	MCU_I2C0_SCL	I/O	I <sup>2</sup> C serial clock line. Open-drain output buffer.	1
SDA	MCU_I2C0_SDA	I/O	I <sup>2</sup> C serial data line. Open-drain output buffer.	1
SCL	MCU_I2C1_SCL	I/O	I <sup>2</sup> C serial clock line. Emulated open-drain output buffer.	1
SDA	MCU_I2C1_SDA	I/O	I <sup>2</sup> C serial data line. Emulated open-drain output buffer.	1
<b>I2C[0-6]</b>				
SCL	I2C[0-1]_SCL	I/O	I <sup>2</sup> C serial clock line. Open-drain output buffer.	1
SDA	I2C[0-1]_SDA	I/O	I <sup>2</sup> C serial data line. Open-drain output buffer.	1
SCL	I2C[2-6]_SCL	I/O	I <sup>2</sup> C serial clock line. Emulated open-drain output buffer.	1
SDA	I2C[2-6]_SDA	I/O	I <sup>2</sup> C serial data line. Emulated open-drain output buffer.	1

(1) I = Input; O = Output; I/O = Bidirectional

### 12.1.3.2.1.3

#### Note

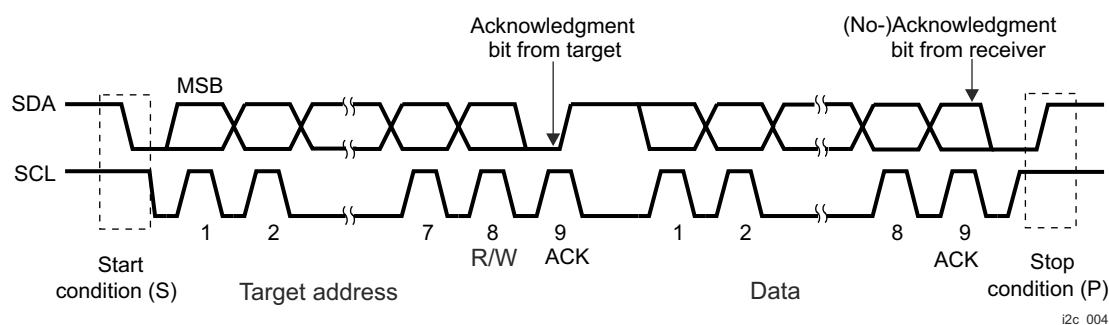
For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

### 12.1.3.2.2 I2C Typical Connection Protocol and Data Format

#### 12.1.3.2.2.1 I2C Serial Data Format

The I2C controller operates in 8-bit word data format (byte write access supported for the last access). Each byte transmitted or received on the serial data line is 8 bits long. The number of bytes that can be transmitted or received is not restricted. The data is transferred with the most-significant bit (MSB) first. In receiver mode, each byte is followed by an acknowledge bit from the I2C module.

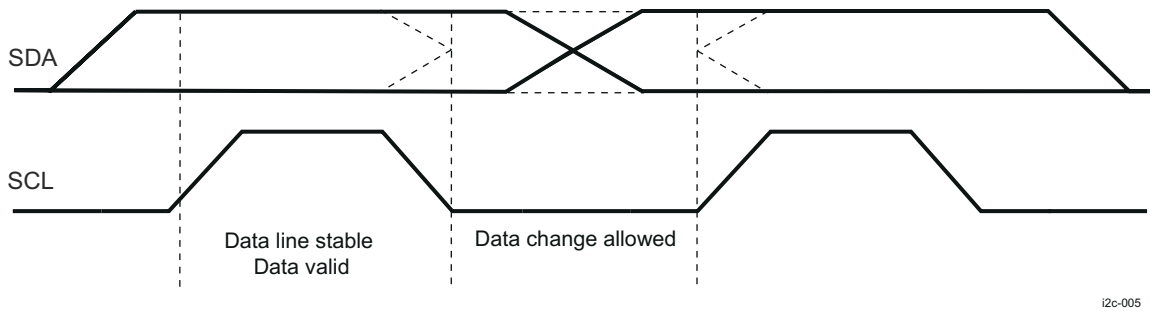
Figure 12-19 shows a typical I<sup>2</sup>C communication format.


**Figure 12-19. I2C Data Transfer**

#### 12.1.3.2.2.2 I2C Data Validity

The data on the serial data line (SDA) must be stable during the high period of the serial clock line. The high and low states of the data line can change only when the clock signal on the serial clock line (SCL) is low.

Figure 12-20 is an example of data validity requirements.



**Figure 12-20. I2C Bit Transfer on the I2C Bus**

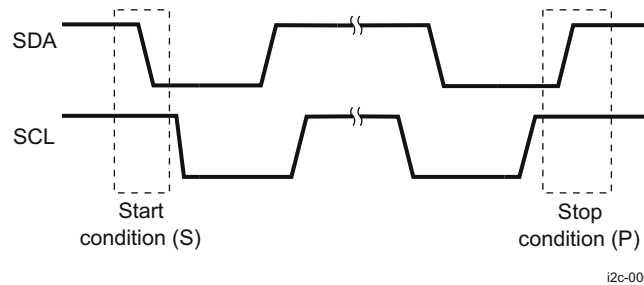
#### 12.1.3.2.2.3 I2C Start and Stop Conditions

The I2C module generates start (S) and stop (P) conditions when it is configured as a controller.

- An S condition is a high-to-low transition on the serial data line while serial clock line is high.
- A P condition is a low-to-high transition on the serial data line while serial clock line is high.

The bus is considered busy after the S condition (the I2C\_IRQSTATUS\_RAW [12] BB bit is 1 to indicate that the bus is busy) and free after the P condition (the I2C\_IRQSTATUS\_RAW [12] BB bit is 0 to indicate that the bus is free).

Figure 12-21 shows the waveforms that occur during an S and a P condition.



**Figure 12-21. I2C S and P Condition Events**

#### Note

I2C controller does not support messages non-compliant with I<sup>2</sup>C standard. Void messages are non-standard I<sup>2</sup>C messages and will lockup the controller. A void message is a START condition followed by a STOP condition, in other words, while the bus is free the SDA line is pulled low (START) and then released (STOP). This would result in a timeout (software) of the next controller transfer which would never complete. A soft reset of the controller is recommended for recovery.

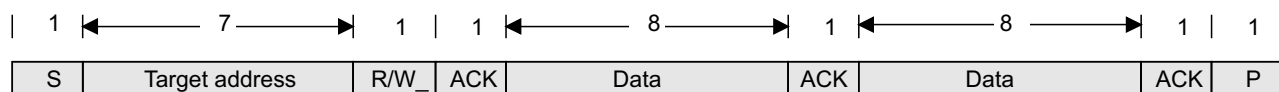
#### 12.1.3.2.2.4 I2C Addressing

The I2C module supports two data formats in fast/standard (F/S) mode:

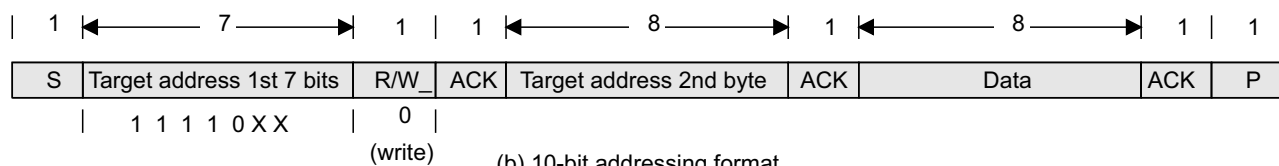
- 7-bit/10-bit addressing format
- 7-bit/10-bit addressing format with repeated start (Sr) condition

##### 12.1.3.2.2.4.1 Data Transfer Formats in F/S Mode

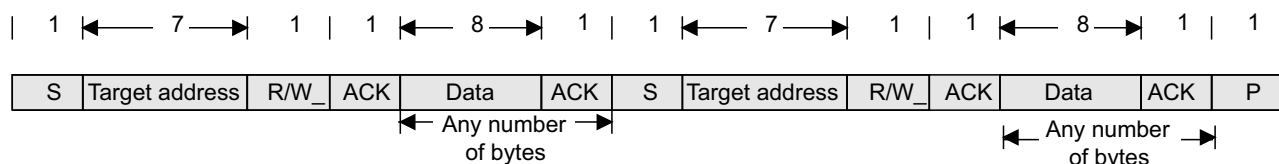
Figure 12-22 shows the I2C data transfer formats in F/S mode.



(a) 7-bit addressing format



(b) 10-bit addressing format



(c) Addressing format with repeated start condition

i2c-007

**Figure 12-22. I2C Data Transfer Formats in F/S Mode**

The first word after an S condition consists of 8 bits. In acknowledge mode, an extra dedicated acknowledgment bit is inserted after each byte.

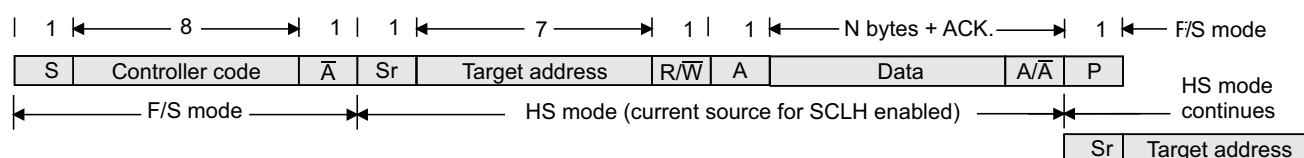
In addressing formats with 7-bit addresses, the first byte is composed of 7 MSB target address bits and 1 least-significant bit (LSB) R/W\_ bit.

The LSB R/W\_ bit of the address byte indicates the transmission direction of the data bytes that follow it. If R/W\_ is 0, the controller writes data to the selected target; if it is 1, the controller reads data from the target.

In addressing formats with 10-bit addresses, the structure of the first byte is 11110XXY, where XX is the two MSBs of the 10-bit addresses, and Y is the R/W\_ bit. If the R/W\_ bit is 0, the next byte contains the last 8 bits of the target address. If the R/W\_ bit is 1, the next byte contains data transmitted from the target to the controller.

#### 12.1.3.2.2.4.2 Data Transfer Format in HS Mode

Figure 12-23 shows the I2C data transfer format in HS mode.



i2c-008

**Figure 12-23. HS I2C Data Transfer in HS Mode**

Each multicontroller I2C controller can also operate in HS mode. In this case, after the S condition, the module, which is in F/S mode, writes the controller code address (0x00001XXX, where XXX is the variable portion of the controller code) on the bus. No device connected on the same bus acknowledges this address. The module switches the clock to the HS clock and after an Sr condition, and sends the target address and the data, as shown in Figure 12-23.



#### 12.1.3.2.2.5 I2C Controller Transmitter

In controller transmitter mode, data assembled in one of the previously described data formats is shifted out on the serial data line SDA in sync with the self-generated clock pulses on the serial clock line SCL. The clock pulses are inhibited and SCL is held low when the intervention of the processor is required (XUDF) after a byte is transmitted.

#### 12.1.3.2.2.6 I2C Controller Receiver

Controller receiver mode can be entered only from controller transmitter mode. With any of the address formats (a), (b), or (c) (see Figure 12-22), if R/W\_ is high, the module enters controller receiver mode after the target address byte and bit R/W\_ are transmitted. Serial data bits received on bus line SDA are shifted in synchronization with the self-generated clock pulses on SCL.

#### 12.1.3.2.2.7 I2C Target Transmitter

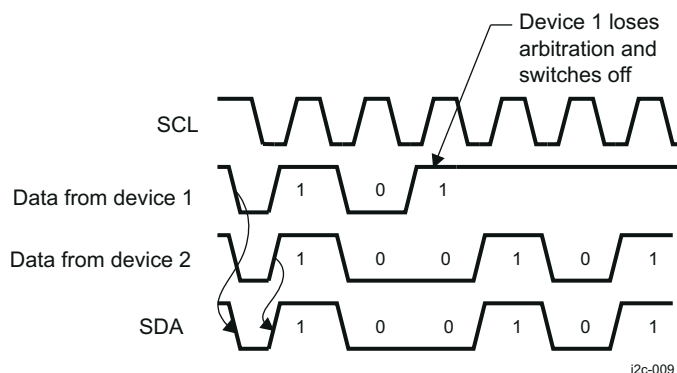
Target transmitter mode can be entered only from target receiver mode. With any of the address formats (a), (b), or (c) (see Figure 12-22), the target transmitter is entered if the target address byte is the same as its own address and bit R/W\_ is transmitted, if R/W\_ is high. The target transmitter shifts the serial data out on the data line SDA in sync with the clock pulses that are generated by the controller device. It does not generate the clock but it can hold clock line SCL low while intervention of the LH is required (XUDF).

#### 12.1.3.2.2.8 I2C Target Receiver

In this mode, serial data bits received on the bus line SDA are shifted-in in sync with the clock pulses on SCL that are generated by the controller device. It does not generate the clock but it can hold clock line SCL low while intervention of the LH is required (ROVR) after a byte is received.

#### 12.1.3.2.2.9 I2C Bus Arbitration

If two or more controller transmitters start a transmission on the same bus almost simultaneously, an arbitration procedure is invoked. The arbitration procedure uses the data presented on the serial bus by the competing transmitters. When a transmitter senses that a high signal it has presented on the bus has been overruled by a low signal, it switches to the target receiver mode, sets the arbitration lost (I2C\_IRQSTATUS\_RAW[0] AL) flag, and generates the arbitration lost interrupt. Figure 12-24 shows the arbitration procedure between two devices. The arbitration procedure gives priority to the device that transmits the serial data stream with the lowest binary value. If two or more devices send identical first bytes, arbitration continues on the subsequent bytes.



**Figure 12-24. I2C Arbitration Between Controller Transmitters**

#### 12.1.3.2.2.10 I2C Clock Generation and Synchronization

Under normal conditions, only one controller device generates the clock signal - SCL. However, there are two or more controller devices during the arbitration procedure, and the clock must be synchronized so that the data output can be compared. The wired-AND property of the clock line means that a device that first generates a low period of the clock line overrules the other devices. At this high/low transition, the clock generators of the other devices are forced to start generation of their own low period. The clock line is then held low by the device with the longest low period, while the other devices that finish their low periods must wait for the clock line to be

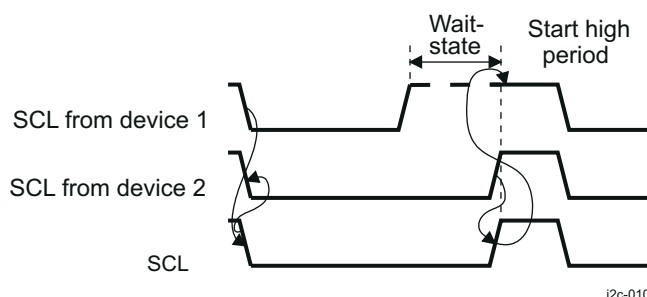
released before starting their high periods. A synchronized signal on the clock line is thus obtained, where the slowest device determines the length of the low period and the fastest device determines the length of the high period. If a device pulls down the clock line for a longer time, the result is that all clock generators must enter the WAIT-state. In this way a target can slow down a fast controller and the slow device can create enough time to store a received byte or prepare a byte to be transmitted (clock stretching).

### Note

In case the SCL or SDA lines are stuck low, a bus clearing operation is supported:

- If the clock line (SCL) is stuck low, the preferential procedure is to reset the bus using the hardware reset signal if the I<sup>2</sup>C devices have hardware reset inputs. If the I<sup>2</sup>C devices do not have hardware reset inputs, cycle power to the devices to activate the mandatory internal power-on reset (POR) circuit.
- If the data line (SDA) is stuck low, the controller should send nine clock pulses. The device that held the bus low should release it sometime within those nine clocks. If not, then use the hardware reset or cycle power to clear the bus.

Figure 12-25 shows clock synchronization.



**Figure 12-25. I2C Clock Generators Synchronization**

### 12.1.3.3 I2C Integration

This section describes I2C integration in the device, including information about clocks, resets, and hardware requests.

#### 12.1.3.3.1 I2C Integration in WKUP Domain

A single WKUP\_I2C0 module is integrated in the device WKUP domain. Figure 12-26 shows the integration of WKUP\_I2C0.

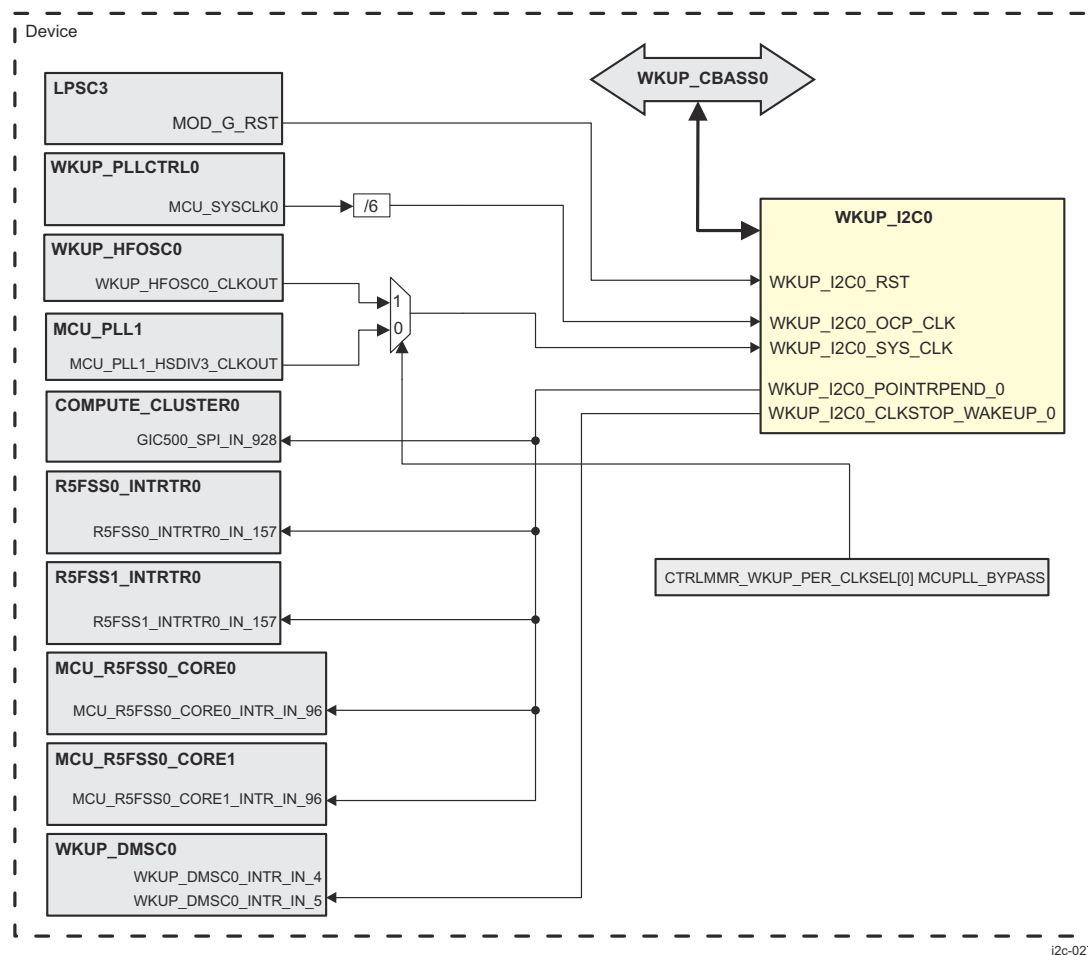


Figure 12-26. WKUP\_I2C0 Integration

Table 12-27 through Table 12-29 summarize the integration of WKUP\_I2C0 in device WKUP domain.

**Table 12-27. WKUP\_I2C0 Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
WKUP_I2C0	WKUP_PSC0	PD0	LPSC3	WKUP_CBASS0

**Table 12-28. WKUP\_I2C0 Clocks and Resets <sup>(1)</sup>**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
WKUP_I2C0	WKUP_I2C0_OCP_CLK	MCU_SYSCLK0/6	WKUP_PLLCTRL0	WKUP_I2C0 interface clock.
	WKUP_I2C0_SYS_CLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	WKUP_I2C0 functional clock. Output of multiplexer, see <a href="#">Figure 12-26</a> , WKUP_I2C0 Integration. Multiplexers control is provided via CTRLMMR_WKUP_PER_CLKSEL[0] MCUPLL_BYPASS bit field in <i>Control Module (CTRL_MMR)</i> .
		MCU_PLL1_HSDIV3_CLKOUT	MCU_PLL1	
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
WKUP_I2C0	WKUP_I2C0_RST	MOD_G_RST	LPSC3	WKUP_I2C0 reset

(1) When 3.4Mbps mode is needed, WKUP\_I2C0\_SYS\_CLK must be set to 96MHz. In this case UART IrDA mode cannot be supported since it requires 48MHz clock.

**Table 12-29. WKUP\_I2C0 Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
WKUP_I2C0	WKUP_I2C0_CLKSTOP_WAKEUP_0	WKUP_DMSC0_INTR_IN_5	WKUP_DMSC0	WKUP_I2C0 wakeup interrupt	Pulse
	WKUP_I2C0_POINTRPEND_0	WKUP_DMSC0_INTR_IN_4	WKUP_DMSC0	WKUP_I2C0 interrupt request	Level
		GIC500_SPI_IN_928	COMPUTE_CLUSTER0		
		R5FSS0_INTRTR0_IN_157	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_157	R5FSS1_INTRTR0		
		MCU_R5FSS0_CORE0_INTR_IN_96	MCU_R5FSS0_CORE0		
		MCU_R5FSS0_CORE1_INTR_IN_96	MCU_R5FSS0_CORE1		

### Note

I2C interrupts are further described in [Section 12.1.3.4.5, I2C Interrupt Requests](#).

### 12.1.3.3.2 I2C Integration in MCU Domain

There are two I2C modules integrated in the device MCU domain - MCU\_I2C0 and MCU\_I2C1. [Figure 12-27](#) shows the integration of MCU\_I2C0 and MCU\_I2C1.

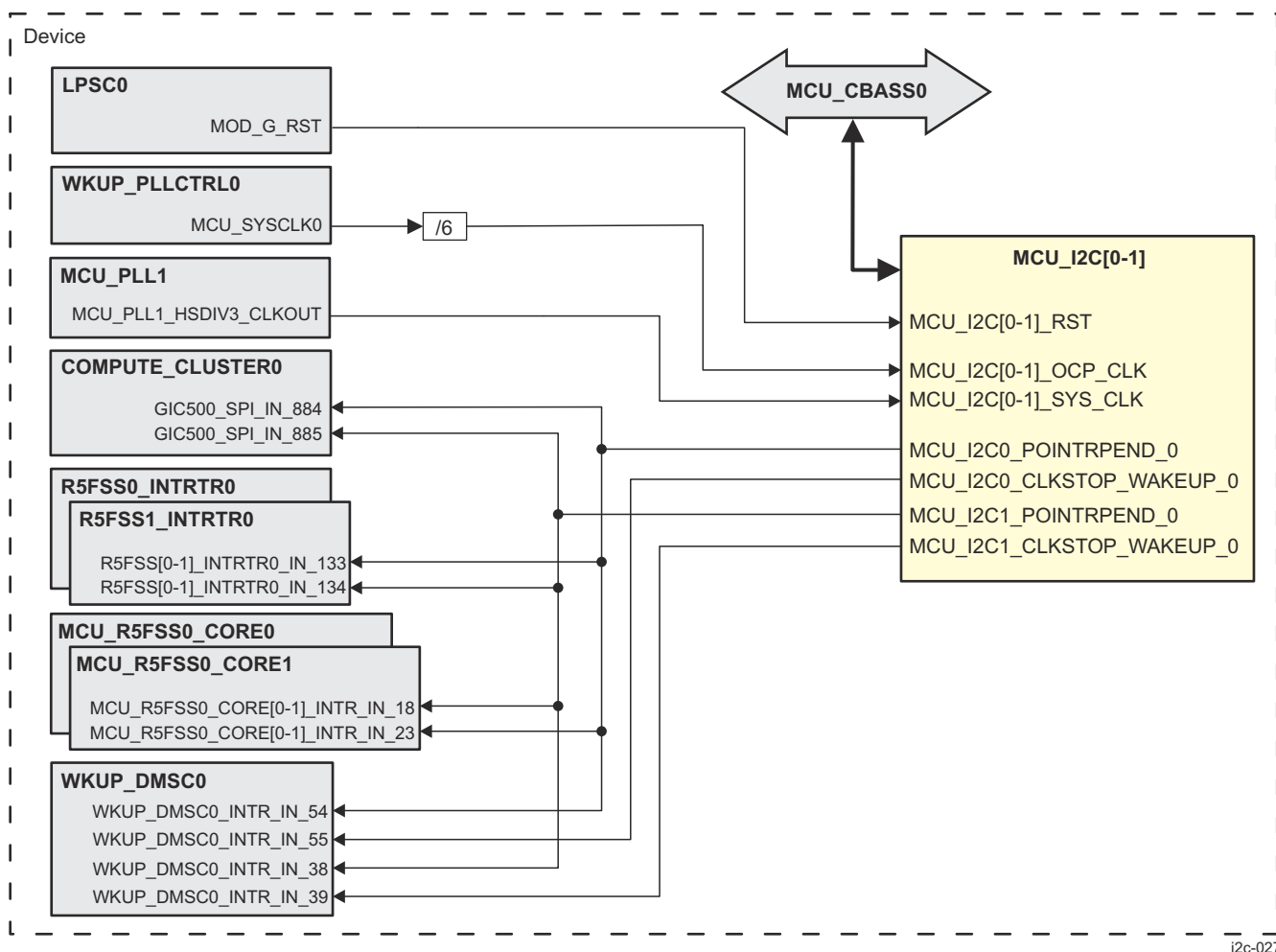


Figure 12-27. MCU\_I2C[0-1] Integration

Table 12-30 through Table 12-32 summarize the integration of MCU\_I2C[0-1] in device MCU domain.

Table 12-30. MCU\_I2C[0-1] Integration Attributes

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
MCU_I2C0	WKUP_PSC0	PD0	LPSC0	MCU_CBASS0

**Table 12-30. MCU\_I2C[0-1] Integration Attributes (continued)**

MCU_I2C1	WKUP_PSC0	PD0	LPSC0	MCU_CBASS0
----------	-----------	-----	-------	------------

**Table 12-31. MCU\_I2C[0-1] Clocks and Resets <sup>(1)</sup>**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
MCU_I2C0	MCU_I2C0_OCP_CLK	MCU_SYSCLK0/6	WKUP_PLLCTRL0	MCU_I2C0 interface clock
	MCU_I2C0_SYS_CLK	MCU_PLL1_HSDIV3_CLKOUT	MCU_PLL1	MCU_I2C0 functional clock
MCU_I2C1	MCU_I2C1_OCP_CLK	MCU_SYSCLK0/6	WKUP_PLLCTRL0	MCU_I2C1 interface clock
	MCU_I2C1_SYS_CLK	MCU_PLL1_HSDIV3_CLKOUT	MCU_PLL1	MCU_I2C1 functional clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
MCU_I2C0	MCU_I2C0_RST	MOD_G_RST	LPSC0	MCU_I2C0 reset
MCU_I2C1	MCU_I2C1_RST	MOD_G_RST	LPSC0	MCU_I2C1 reset

(1) When 3.4Mbps mode is needed, MCU\_I2C0\_SYS\_CLK or MCU\_I2C1\_SYS\_CLK must be set to 96MHz. In this case UART IrDA mode cannot be supported since it requires 48MHz clock.

**Table 12-32. MCU\_I2C[0-1] Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCU_I2C0	MCU_I2C0_CLKSTOP_WAKEUP_0	WKUP_DMSC0_INTR_IN_55	WKUP_DMSC0	MCU_I2C0 wakeup interrupt. The interrupt is connected to DMSC0_WKUP, but instance is in an always on domain. See <i>I2C Not Supported Features</i> .	Pulse
		WKUP_DMSC0_INTR_IN_54	WKUP_DMSC0	MCU_I2C0 interrupt request	Level
		GIC500_SPI_IN_884	COMPUTE_CLUSTER0		
		R5FSS0_INTRTR0_IN_133	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_133	R5FSS1_INTRTR0		
		MCU_R5FSS0_CORE0_INTR_IN_23	MCU_R5FSS0_CORE0		
MCU_I2C1	MCU_I2C1_CLKSTOP_WAKEUP_0	MCU_R5FSS0_CORE1_INTR_IN_23	MCU_R5FSS0_CORE1		
		WKUP_DMSC0_INTR_IN_39	WKUP_DMSC0	MCU_I2C1 wakeup interrupt. The interrupt is connected to DMSC0_WKUP, but instance is in an always on domain. See <i>I2C Not Supported Features</i> .	Pulse
		WKUP_DMSC0_INTR_IN_38	WKUP_DMSC0	MCU_I2C1 interrupt request	Level
		GIC500_SPI_IN_885	COMPUTE_CLUSTER0		
		R5FSS0_INTRTR0_IN_134	R5FSS0_INTRTR0		

**Table 12-32. MCU\_I2C[0-1] Hardware Requests (continued)**

R5FSS1_INTRTR0_IN_134	R5FSS1_INTRTR0
MCU_R5FSS0_CORE0_INTR_IN_18	MCU_R5FSS0_CORE0
MCU_R5FSS0_CORE1_INTR_IN_18	MCU_R5FSS0_CORE1

#### 12.1.3.3.3 I2C Integration in MAIN Domain

There are seven I2C modules integrated in the device MAIN domain - I2C0, I2C1, I2C2, I2C3, I2C4, I2C5, and I2C6. [Figure 12-28](#) shows the integration of I2C0, I2C1, I2C2, I2C3, I2C4, I2C5, and I2C6.

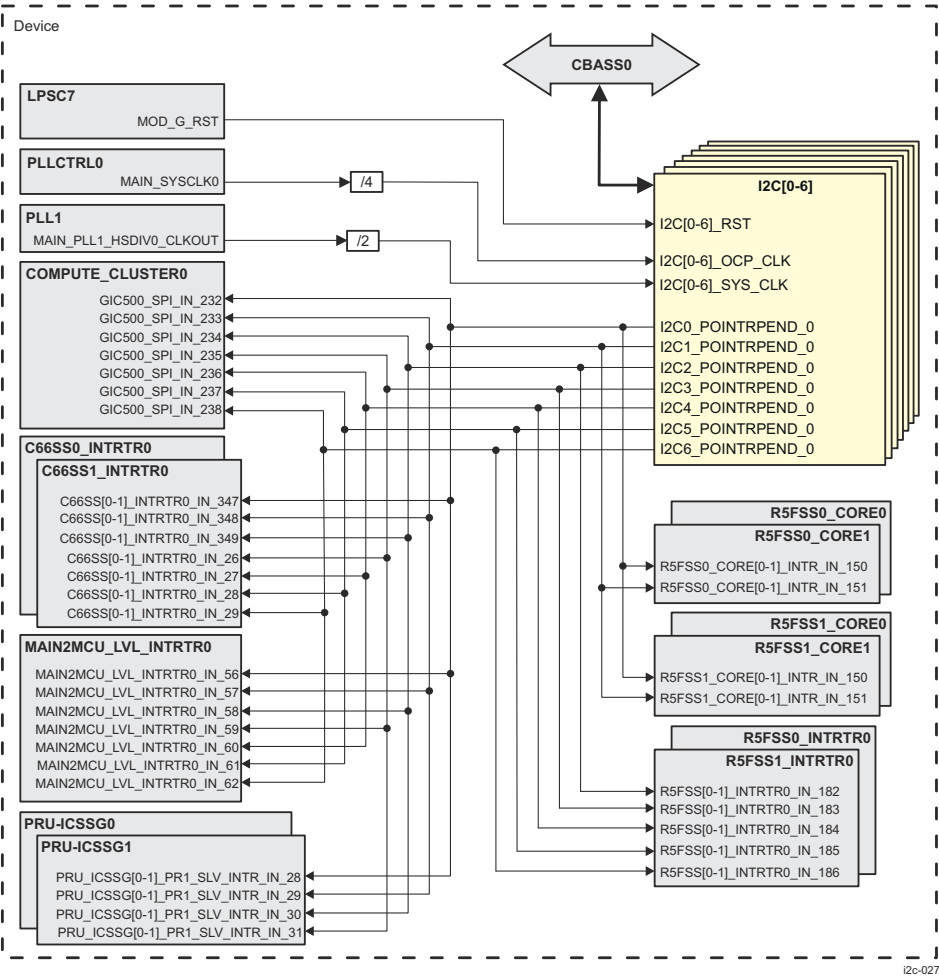


Figure 12-28. I2C[0-6] Integration

Table 12-33 through Table 12-35 summarize the integration of I2C[0-6] in the device MAIN domain.

Table 12-33. I2C[0-6] Integration Attributes

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
I2C0	PSC0	PD0	LPSC7	CBASS0



**Table 12-33. I2C[0-6] Integration Attributes (continued)**

I2C1	PSC0	PD0	LPSC7	CBASS0
I2C2	PSC0	PD0	LPSC7	CBASS0
I2C3	PSC0	PD0	LPSC7	CBASS0
I2C4	PSC0	PD0	LPSC7	CBASS0
I2C5	PSC0	PD0	LPSC7	CBASS0
I2C6	PSC0	PD0	LPSC7	CBASS0

**Table 12-34. I2C[0-6] Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
I2C0	I2C0_OCP_CLK	MAIN_SYSCLK0/4	PLLCTRL0	I2C0 interface clock
	I2C0_SYS_CLK	MAIN_PLL1_HSDIV0_CLKOUT/2	PLL1	I2C0 functional clock
I2C1	I2C1_OCP_CLK	MAIN_SYSCLK0/4	PLLCTRL0	I2C1 interface clock
	I2C1_SYS_CLK	MAIN_PLL1_HSDIV0_CLKOUT/2	PLL1	I2C1 functional clock
I2C2	I2C2_OCP_CLK	MAIN_SYSCLK0/4	PLLCTRL0	I2C2 interface clock
	I2C2_SYS_CLK	MAIN_PLL1_HSDIV0_CLKOUT/2	PLL1	I2C2 functional clock
I2C3	I2C3_OCP_CLK	MAIN_SYSCLK0/4	PLLCTRL0	I2C3 interface clock
	I2C3_SYS_CLK	MAIN_PLL1_HSDIV0_CLKOUT/2	PLL1	I2C3 functional clock
I2C4	I2C4_OCP_CLK	MAIN_SYSCLK0/4	PLLCTRL0	I2C4 interface clock
	I2C4_SYS_CLK	MAIN_PLL1_HSDIV0_CLKOUT/2	PLL1	I2C4 functional clock
I2C5	I2C5_OCP_CLK	MAIN_SYSCLK0/4	PLLCTRL0	I2C5 interface clock
	I2C5_SYS_CLK	MAIN_PLL1_HSDIV0_CLKOUT/2	PLL1	I2C5 functional clock
I2C6	I2C6_OCP_CLK	MAIN_SYSCLK0/4	PLLCTRL0	I2C6 interface clock
	I2C6_SYS_CLK	MAIN_PLL1_HSDIV0_CLKOUT/2	PLL1	I2C6 functional clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
I2C0	I2C0_RST	MOD_G_RST	LPSC7	I2C0 reset
I2C1	I2C1_RST	MOD_G_RST	LPSC7	I2C1 reset
I2C2	I2C2_RST	MOD_G_RST	LPSC7	I2C2 reset
I2C3	I2C3_RST	MOD_G_RST	LPSC7	I2C3 reset
I2C4	I2C4_RST	MOD_G_RST	LPSC7	I2C4 reset
I2C5	I2C5_RST	MOD_G_RST	LPSC7	I2C5 reset
I2C6	I2C6_RST	MOD_G_RST	LPSC7	I2C6 reset

**Table 12-35. I2C[0-6] Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
I2C0	I2C0_POINTRPEND_0	GIC500_SPI_IN_232	COMPUTE_CLUSTER0	I2C0 Interrupt request	Level
		PRU_ICSSG0_PR1_SLV_INTR_IN_28	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INTR_IN_28	PRU-ICSSG1		
		C66SS0_INTRTR0_IN_347	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_347	C66SS1_INTRTR0		
		R5FSS1_CORE0_INTR_IN_150	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_150	R5FSS1_CORE1		
		R5FSS0_CORE0_INTR_IN_150	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_150	R5FSS0_CORE1		
		MAIN2MCU_LVL_INTRTR0_IN_56	MAIN2MCU_LVL_INTRTR0		
I2C1	I2C1_POINTRPEND_0	GIC500_SPI_IN_233	COMPUTE_CLUSTER0	I2C1 Interrupt request	Level
		PRU_ICSSG0_PR1_SLV_INTR_IN_29	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INTR_IN_29	PRU-ICSSG1		
		C66SS0_INTRTR0_IN_348	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_348	C66SS1_INTRTR0		
		R5FSS1_CORE0_INTR_IN_151	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_151	R5FSS1_CORE1		
		R5FSS0_CORE0_INTR_IN_151	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_151	R5FSS0_CORE1		
		MAIN2MCU_LVL_INTRTR0_IN_57	MAIN2MCU_LVL_INTRTR0		
I2C2	I2C2_POINTRPEND_0	GIC500_SPI_IN_234	COMPUTE_CLUSTER0	I2C2 Interrupt request	Level
		PRU_ICSSG0_PR1_SLV_INTR_IN_30	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INTR_IN_30	PRU-ICSSG1		
		C66SS0_INTRTR0_IN_349	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_349	C66SS1_INTRTR0		
		R5FSS0_INTRTR0_IN_182	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_182	R5FSS1_INTRTR0		
		MAIN2MCU_LVL_INTRTR0_IN_58	MAIN2MCU_LVL_INTRTR0		
I2C3	I2C3_POINTRPEND_0	GIC500_SPI_IN_235	COMPUTE_CLUSTER0	I2C3 Interrupt request	Level
		PRU_ICSSG0_PR1_SLV_INTR_IN_31	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INTR_IN_31	PRU-ICSSG1		
		C66SS0_INTRTR0_IN_26	C66SS0_INTRTR0		

**Table 12-35. I2C[0-6] Hardware Requests (continued)**

		C66SS1_INTRTR0_IN_26	C66SS1_INTRTR0		
		R5FSS0_INTRTR0_IN_183	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_183	R5FSS1_INTRTR0		
		MAIN2MCU_LVL_INTRTR0_IN_59	MAIN2MCU_LVL_INTRTR0		
I2C4	I2C4_POINTRPEND_0	GIC500_SPI_IN_236	COMPUTE_CLUSTER0	I2C4 Interrupt request	Level
		C66SS0_INTRTR0_IN_27	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_27	C66SS1_INTRTR0		
		R5FSS0_INTRTR0_IN_184	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_184	R5FSS1_INTRTR0		
		MAIN2MCU_LVL_INTRTR0_IN_60	MAIN2MCU_LVL_INTRTR0		
I2C5	I2C5_POINTRPEND_0	GIC500_SPI_IN_237	COMPUTE_CLUSTER0	I2C5 Interrupt request	Level
		C66SS0_INTRTR0_IN_28	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_28	C66SS1_INTRTR0		
		R5FSS0_INTRTR0_IN_185	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_185	R5FSS1_INTRTR0		
		MAIN2MCU_LVL_INTRTR0_IN_61	MAIN2MCU_LVL_INTRTR0		
I2C6	I2C6_POINTRPEND_0	GIC500_SPI_IN_238	COMPUTE_CLUSTER0	I2C6 Interrupt request	Level
		C66SS0_INTRTR0_IN_29	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_29	C66SS1_INTRTR0		
		R5FSS0_INTRTR0_IN_186	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_186	R5FSS1_INTRTR0		
		MAIN2MCU_LVL_INTRTR0_IN_62	MAIN2MCU_LVL_INTRTR0		

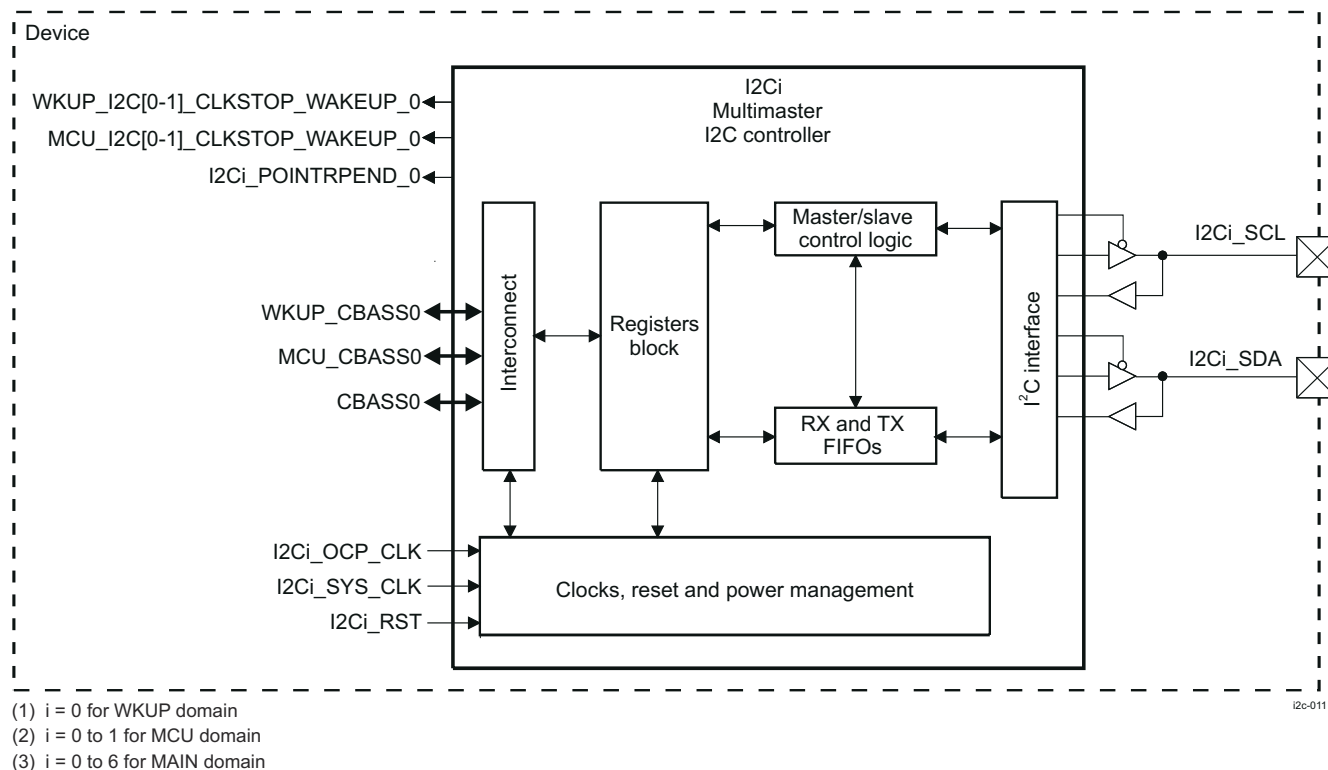
### 12.1.3.4 I2C Functional Description

#### 12.1.3.4.1 I2C Block Diagram

#### Note

DMA mode and SCCB Protocol are not supported on this family of devices.

Figure 12-29 presents the multimaster I2C controller block diagram.



**Figure 12-29. I2C Block Diagram**

The ten multimaster I2C controllers can be configured in F/S I2C mode or HS I2C mode. The operation mode is selected by configuring the I2C\_CON[13-12] OPMODE bit field.

Table 12-36 lists the available operation modes.

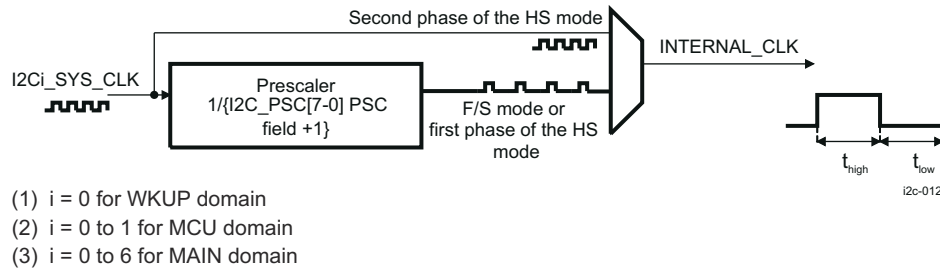
**Table 12-36. I2C Operation Mode Selection**

Operation Mode	Value of I2C_CON[13-12] OPMODE
F/S I2C	0x0
HS I2C	0x1
SCCB	0x2
Reserved (not used)	0x3

#### 12.1.3.4.2 I2C Clocks

##### 12.1.3.4.2.1 I2C Clocking

Figure 12-30 shows the I2C clock generation of the I2C controllers.



**Figure 12-30. I2C Clock Generation**

Each multimaster I2C controller uses the SYS\_CLK functional clock in the Device Configuration section. The internal sampling clock INTERNAL\_CLK is generated by dividing the functional clock by the I2C\_PSC[7-0] PSC bit field value + 1 in F/S mode, or in the first phase of HS mode; or by directly using the functional clock in the second phase of HS mode (prescaler is bypassed).

The low time of the SCLL signal is determined by the I2C\_SCLL[7-0] SCLL bit field in F/S mode and in the first phase of HS mode; or by the I2C\_SCLL[15-8] HSSCLL bit field in the second phase of HS mode.

The high time of the SCLH signal is determined by the I2C\_SCLH[7-0] SCLH bit field in F/S mode and in the first phase of HS mode; or by the I2C\_SCLH[15-8] HSSCLH bit field in the second phase of HS mode.

Table 12-37 lists the  $t_{LOW}$  and  $t_{high}$  values in master mode only (in slave mode, the I2C controller does not generate the I2C clock).

**Table 12-37. I2C  $t_{LOW}$  and  $t_{high}$  Values of the I2C Clock**

Mode	Clock	$t_{LOW}$	$t_{high}$
F/S or HS first phase	$INTERNAL\_CLK = SYS\_CLK / (I2C\_PSC[7-0] \text{ PSC bit field} + 1)$	$(I2C\_SCLL[7-0] \text{ SCLL bit field value} + 7) \times INTERNAL\_CLK \text{ period}$	$(I2C\_SCLH[7-0] \text{ SCLH bit field value} + 5) \times I2Ci\_INTERNAL\_CLK \text{ period}$
HS second phase	SYS_CLK	$(I2C\_SCLL[15-8] \text{ HSSCLL bit field value} + 7) \times SYS\_CLK \text{ period}$	$(I2C\_SCLH[15-8] \text{ HSSCLH bit field value} + 5) \times I2Ci\_SYS\_CLK \text{ period}$

**Note**

For HS mode, the I2C\_SCLL[15-8] HSSCLL and I2C\_SCLL[7-0] SCLL bit fields must be programmed (the first phase of an HS transaction is performed at F/S speed).

For HS mode, the I2C\_SCLH[15-8] HSSCLH and I2C\_SCLH[7-0] SCLH bit fields must be programmed (the first phase of an HS transaction is performed at F/S speed).

**Note**

The equations in Table 12-37 give the SCLL timing values for SCLL/SCLH/HSSCLL/HSSCLH at I2C controller outputs. Actual  $t_{low}$  and  $t_{high}$  periods may vary depending on the board (the load capacitance on the SCLL signal). If necessary, any adjustments to the SCLL/SCLH/HSSCLL/HSSCLH values must be determined by measurements of actual SCL signal on the board.

**CAUTION**

During active mode (the I2C\_CON[15] I2C\_EN bit is set to 1), make no changes to the I2C\_SCLL and I2C\_SCLH registers. Changes may result in unpredictable behavior.

Table 12-38 lists the register values for obtaining the maximum I<sup>2</sup>C bit rates and the maximum period of the filtered spikes in F/S mode and HS mode.

**Table 12-38. I2C Register Values for Maximum I2C Bit Rates in I2C F/S, I2C HS Modes**

	I <sup>2</sup> C Mode for			Description
	Standard Mode	Fast Mode	High-Speed Mode	
SYS_CLK frequency (MHz)	96			
OCP_CLK frequency (MHz)	133			
<b>I2C_PSC[7-0] PSC</b>	<b>23</b>	<b>9</b>	<b>1</b>	Prescaler value for F/S and HS modes
INTERNAL_CLK frequency (MHz)	4	9.6	96	
<b>I2C_SCLL[7-0] SCLL</b>	<b>13</b>	<b>7</b>	<b>115</b>	Value for F/S mode and first phase of HS mode
<b>I2C_SCLH[7-0] SCLH</b>	<b>15</b>	<b>5</b>	<b>113</b>	Value for F/S mode and first phase of HS mode
Maximum bit rate (Mbps)	0.1	0.4	0.4	F/S mode and first phase in HS mode maximum bit rate
Maximum filter period (ns)	250	104.2	10	
<b>I2C_SCLL[15-8] HSSCLL</b>			<b>12</b>	Values for second phase of HS mode
<b>I2C_SCLH[15-8] HSSCLH</b>			<b>5</b>	Values for second phase of HS mode
HS mode maximum bit rate (Mbps)			3.31	HS mode maximum bit rate
Maximum filter period (ns)			10	

#### Note

This table presents informative values only for the configuration parameters and the I<sup>2</sup>C bus performance obtained according to these values. The delays added by the analog pads are not considered in these figures.

#### Note

For WKUP\_I2C0

For MCU\_I2C[0-1]

For I2C[0-6]

$I2Ci\_INTERNAL\_CLK \text{ freq} = I2Ci\_SYS\_CLK / (PSC + 1)$

$F/S \text{ filter period} = 1 / I2Ci\_INTERNAL\_CLK$

$HS \text{ filter period} = 1 / I2Ci\_SYS\_CLK \text{ freq}$

$HS \text{ bit rate} = I2Ci\_SYS\_CLK \text{ freq} / (HSSCLL + 7 + HSSCLH + 5)$

$FS \text{ bit rate} = I2Ci\_INTERNAL\_CLK / (SCLL + 7 + SCLH + 5)$

#### 12.1.3.4.2.2 I2C Automatic Blocking of the I2C Clock Feature

This feature offers the possibility for the LH to command the blocking of the I<sup>2</sup>C clock after the target addressing phase, when the I2C controller is addressed by an external controller device using a certain Own Address.

The release of the I<sup>2</sup>C clock can be performed independently for each Own Address (I2C\_OA, and I2C\_OAx registers, where x = 1, 2, 3) by deasserting the corresponding bit in the I2C\_SBLOCK register.

#### 12.1.3.4.3 I2C Software Reset

Each multicontroller I2C supports the software reset by accessing the I2C\_SYSC[1] SRST bit (1: reset; 0: normal mode).

The software reset status can be checked by accessing the I2C\_SYSS[0] RDONE bit (1: reset is done; 0: reset is ongoing).

To do a software reset, the following steps must be done:

1. Ensure that the module is disabled (clear the I2C\_CON[15] I2C\_EN bit to 0).
2. Set the I2C\_SYSC[1] SRST bit to 1.
3. Enable the module by setting I2C\_CON[15] I2C\_EN bit to 1.
4. Check the I2C\_SYSS[0] RDONE bit until it is set to 1 to indicate the software reset is complete.

#### Note

The I2C\_CON[15] I2C\_EN bit can hold the functional clock domain of the multicontroller I2C in reset after the device reset has been released. When the system bus reset is removed, this bit remains cleared. The functional part of the I2C controller is held in reset state while this bit is 0, and all configuration registers can be accessed.

The I2C\_CON[15] I2C\_EN bit must be set to 1 to enable the functional part of the I2C controller.

The I2C\_SYSS[0] RDONE bit is asserted only after the module is enabled by setting the I2C\_CON[15] I2C\_EN bit to 1.

#### 12.1.3.4.4 I2C Power Management

[Table 12-39](#) describes power-management features available for the multimaster I2C controllers.

#### Note

For information about source clock gating, see *Power*, in the *Device Configuration*.

#### Note

Some of the I2C features described in this section may not be supported on this family of devices. For more information, see *I2C Not Supported Features*.

**Table 12-39. I2C Local Power-Management Features**

Feature	Registers	Description
Clock auto gating	I2C_SYSC[0] AUTOIDLE	This bit allows a local power optimization inside the module.
Slave idle modes	I2C_SYSC[4-3] IDLEMODE	Force-idle, no-idle, smart-idle, and smart-idle wakeup-capable modes are available.
Clock activity	I2C_SYSC[9-8] CLOCKACTIVITY	For configuration details, see <a href="#">Table 12-40</a> .
Global wake-up enable	I2C_SYSC[2] ENAWAKEUP	This bit enables the wake-up feature at module level.

**Table 12-40. I2C Clock Activity Settings**

I2C_SYSC[9-8] CLOCKACTIVITY	Clock State When Module is in IDLE State		Features Available/Unavailable When Module is in IDLE State
	OCP_CLK	SYS_CLK	
00	OFF	OFF	Both clocks are disabled.
10	OFF	ON	Interface clock is disabled; Functional clock is enabled
01	ON	OFF	Functional clock is disabled; Interface clock is enabled
11	ON	ON	Both clocks are enabled.

### 12.1.3.4.5 I2C Interrupt Requests

The I2C controller must allocate a minimum of five registers for interrupts.

- Interrupt Raw Status (I2C\_IRQSTATUS\_RAW)
- Interrupt Enabled Status (I2C\_IRQSTATUS)
- Interrupt Enable Set (I2C\_IRQENABLE\_SET)
- Interrupt enable Clear (I2C\_IRQENABLE\_CLR)
- End of interrupt (I2C\_EOI)

End of Interrupt (I2C\_EOI) is used by software to trigger an interrupt service completion.

Table 12-41 lists the event flags, and their mask, that can cause module interrupts.

**Table 12-41. I2C Events**

Event Flag	Event Unmask	Event Mask	Description
I2C_IRQSTATUS[0] AL	I2C_IRQENABLE_SET[0] AL_IE	I2C_IRQENABLE_CLR[0] AL_IE	Arbitration lost. This bit is automatically set by the hardware when it loses the arbitration in controller transmit mode, an interrupt is signaled to the host.
I2C_IRQSTATUS[1] NACK	I2C_IRQENABLE_SET[1] NACK_IE	I2C_IRQENABLE_CLR[1] NACK_IE	No acknowledgement. Bit is set when No Acknowledge is received, an interrupt is signaled to the host.
I2C_IRQSTATUS[2] ARDY	I2C_IRQENABLE_SET[2] ARDY_IE	I2C_IRQENABLE_CLR[2] ARDY_IE	Register access ready. When set to 1 it indicates that previous access has been performed and registers are ready to be accessed again. An interrupt is signaled to the host.
I2C_IRQSTATUS[3] RRDY	I2C_IRQENABLE_SET[3] RRDY_IE	I2C_IRQENABLE_CLR[3] RRDY_IE	Receive data ready. Set to 1 by core when in receiver mode, a new data can be read. An interrupt is signaled to the host.
I2C_IRQSTATUS[4] XRDY	I2C_IRQENABLE_SET[4] XRDY_IE	I2C_IRQENABLE_CLR[4] XRDY_IE	Transmit data ready. Set to 1 by core when transmitter is ready for new data. An interrupt is signaled to the host.
I2C_IRQSTATUS[5] GC	I2C_IRQENABLE_SET[5] GC_IE	I2C_IRQENABLE_CLR[5] GC_IE	General call. Set to 1 by core when General Call address was detected. An interrupt is signaled to the host.
I2C_IRQSTATUS[6] STC	I2C_IRQENABLE_SET[6] STC_IE	I2C_IRQENABLE_CLR[6] STC_IE	Start condition detected. An interrupt is signaled to the host.
I2C_IRQSTATUS[7] AERR	I2C_IRQENABLE_SET[7] AERR_IE	I2C_IRQENABLE_CLR[7] AERR_IE	Bus Access Error. An interrupt is signaled to the host.
I2C_IRQSTATUS[8] BF	I2C_IRQENABLE_SET[8] BF_IE	I2C_IRQENABLE_CLR[8] BF_IE	Bus free. An interrupt is signaled to the host.
I2C_IRQSTATUS[9] AAS	I2C_IRQENABLE_SET[9] AAS_IE	I2C_IRQENABLE_CLR[9] AAS_IE	Address recognized as target. An interrupt is signaled to the host.
I2C_IRQSTATUS[10] XUDF	I2C_IRQENABLE_SET [10] XUDF_IE	I2C_IRQENABLE_CLR[10] XUDF_IE	Transmit underflow. An interrupt is signaled to the host.
I2C_IRQSTATUS[11] ROVR	I2C_IRQENABLE_SET [11] ROVR_IE	I2C_IRQENABLE_CLR[11] ROVR_IE	Receive overrun. An interrupt is signaled to the host.
I2C_IRQSTATUS[12] BB	N/A	N/A	Bus busy indicator
I2C_IRQSTATUS[13] RDR	I2C_IRQENABLE_SET [13] RDR_IE	I2C_IRQENABLE_CLR[13] RDR_IE	Receive draining. An interrupt is signaled to the host.
I2C_IRQSTATUS[14] XDR	I2C_IRQENABLE_SET [14] XDR_IE	I2C_IRQENABLE_CLR[14] XDR_IE	Transmit draining. An interrupt is signaled to the host.



#### 12.1.3.4.6 I2C Programmable Multitarget Channel Feature

This feature allows each multicontroller I2C to be addressed using four separate Own Addresses configured in the I2C\_OA and I2C\_OAx registers (where x = 1, 2, 3). An additional register (I2C\_ACTOA) is used to indicate to the LH which address is used by the external controller to communicate with the I2C controller.

Each Own Address can be independently configured in 7-bit or 10-bit mode by setting the corresponding bit (I2C\_CON[7] XOA0, I2C\_CON[6] XOA1, I2C\_CON[5] XOA2, or I2C\_CON[4] XOA3).

#### 12.1.3.4.7 I2C FIFO Management

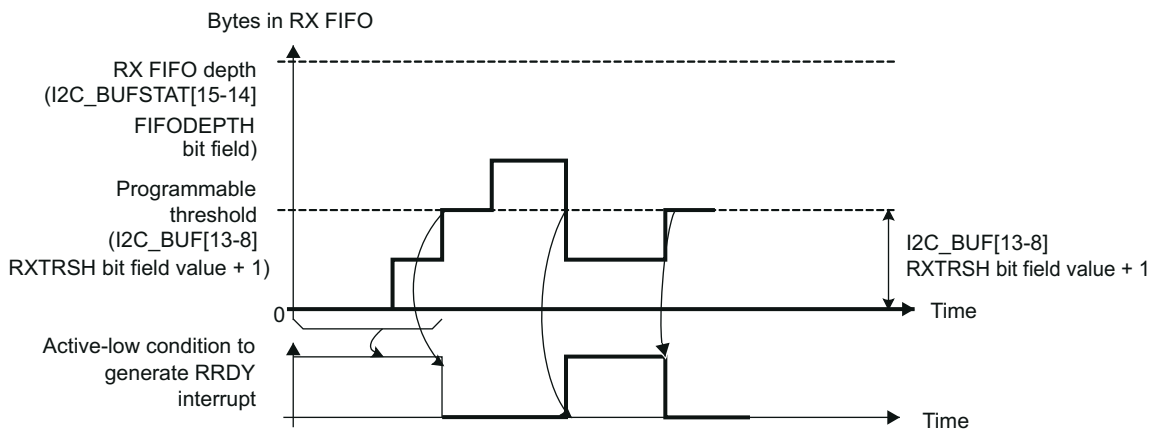
Each multicontroller I2C implements two internal 8-bit FIFOs, the RX and TX FIFOs.

The depth of the RX and TX FIFOs can be checked by reading the I2C\_BUFSTAT[15-14] FIFODEPTH bit field (0x0: 8 bytes, 0x1: 16 bytes, 0x2: 32 bytes, and 0x3: 64 bytes).

##### 12.1.3.4.7.1 I2C FIFO Interrupt Mode

In FIFO interrupt mode (relevant interrupts enabled by the I2C\_IRQENABLE\_SET register), an interrupt signal informs the processor of the receiver and transmitter status. These interrupts are raised when the RX/TX FIFO thresholds (defined by the I2C\_BUF[13-8] RXTRSH bit field value + 1 for the RX FIFO or the I2C\_BUF[5-0] TXTRSH bit field value + 1 for the TX FIFO) are reached; the interrupt signals instruct the LH to transfer data to the destination (from the I2C controller in receive mode and/or from any source to the I2C controller FIFO in transmit mode).

Figure 12-31 and Figure 12-32 show receive and transmit operations, respectively, from a FIFO management point of view.

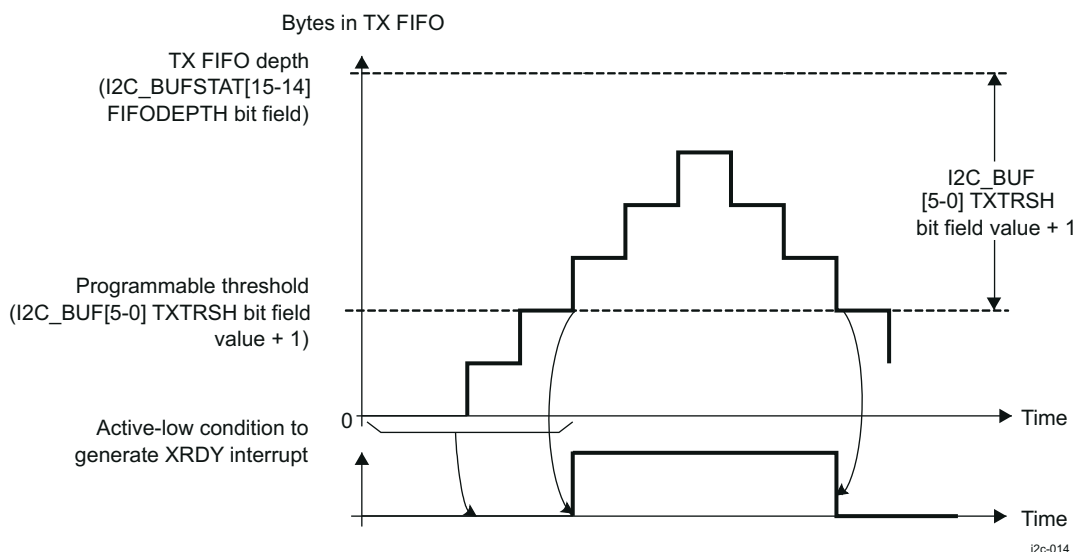


i2c-013

**Figure 12-31. I2C Receive FIFO Interrupt Request Generation**

In Figure 12-31, the RRDY interrupt condition shows that the condition for generating an RRDY interrupt is achieved. The interrupt request is generated when this signal is active, and it can be cleared only by the LH by writing 1 in the corresponding bit. If the condition is still present after clearing the previous interrupt, another interrupt request is generated.

In receive mode, an RRDY interrupt is generated as soon as the FIFO reaches its receive threshold (I2C\_BUF[13-8] RXTRSH bit field value + 1). The interrupt can be deasserted only when the LH has handled enough bytes to make the number of bytes in the RX FIFO lower than the programmed threshold. For each interrupt, the LH can be configured to read a number of bytes equal to the value of the RX FIFO threshold.



**Figure 12-32. I2C Transmit FIFO Interrupt Request Generation**

In [Figure 12-32](#), the XRDY interrupt condition shows that the condition for generating an XRDY interrupt is achieved. The interrupt request is generated when TX FIFO is empty or when the TX FIFO threshold is not reached, and the LH can clear the XRDY status bit by setting the I2C\_IRQENABLE\_CLR [4] XRDY\_IE bit to 1 after transmitting the configured number of bytes. If the condition is still present after clearing the previous interrupt, another interrupt request is generated.

In interrupt mode, the module offers two options for the LH application to handle the interrupts:

- When detecting an interrupt request (XRDY or RRDY type), the LH can write/read 1 data byte to/from the TX/RX FIFO and then clear the interrupt. The module reasserts the interrupt until the interrupt condition is not met.
- When detecting an interrupt request (XRDY or RRDY type), the LH can be programmed to write/read the amount of data bytes specified by the corresponding FIFO threshold (I2C\_BUF[5-0] TXTRSH + 1 or I2C\_BUF[5-0] RXTRSH + 1). In this case, the interrupt condition is cleared and the next interrupt is asserted again when the XRDY or RRDY condition is met again.

If the second-interrupt-serving approach is used, an additional mechanism (draining feature) is implemented for cases where the transfer length is not a multiple of the FIFO threshold value (see [Section 12.1.3.4.7.3, Draining Feature \[I2C Mode Only\]](#)).

#### Note

In target transmit mode (the I2C\_CON[10] MST bit is cleared and the I2C\_CON[9] TRX bit is set to 1), the draining feature must not be used, because the transfer length is not known at configuration time, and the external controller can end the transfer at any point by not acknowledging 1 data byte. If the draining feature is used in target transmit mode, data can remain in the TX FIFO without being transmitted over the I<sup>2</sup>C bus. In this case, the TX FIFO must be cleared by setting the I2C\_BUF[6] TXFIFO\_CLR bit.

#### 12.1.3.4.7.2 I2C FIFO Polling Mode

In FIFO polling mode (the I2C\_IRQENABLE\_SET[4] XRDY\_IE and I2C\_IRQENABLE\_SET[3] RRDY\_IE bits are disabled), the status of the module (receiver or transmitter) can be checked by polling the I2C\_IRQSTATUS\_RAW[4] XRDY and the I2C\_IRQSTATUS\_RAW[3] RRDY bits (the I2C\_IRQSTATUS\_RAW[13] RDR and I2C\_IRQSTATUS\_RAW[14] XDR bits can also be polled if the draining feature is enabled). The I2C\_IRQSTATUS\_RAW[4] XRDY and I2C\_IRQSTATUS\_RAW[3] RRDY bits accurately reflect the interrupt conditions described in the discussion of FIFO interrupt mode.

### 12.1.3.4.7.3 I2C Draining Feature

#### Note

DMA mode and SCCB Protocol are not supported on this family of devices.

The draining feature is implemented to handle the end of a transfer whose length is not a multiple of the FIFO threshold values (the I2C\_BUF[13-8] RXTRSH bit field value + 1 for the RX threshold and the I2C\_BUF[5-0] TXTRSH field value + 1 for the TX threshold). It can also transfer the remaining number of bytes (because the threshold is not reached).

This feature prevents the LH or the DMA controller from trying more FIFO accesses than necessary (for example, to generate at the end of a transfer a DMA RX request having fewer bytes in the FIFO than the configured DMA transfer length). Otherwise, an AERR interrupt is generated by the I2C\_IRQSTATUS\_RAW[7] AERR bit.

The draining mechanism generates an interrupt using the I2C\_IRQSTATUS\_RAW[13] RDR or I2C\_IRQSTATUS\_RAW[14] XDR bit at the end of the transfer, informing the LH that it must check the amount of data left to be transferred (the I2C\_BUFSTAT[13-8] RXSTAT or I2C\_BUFSTAT[5-0] TXSTAT bit fields) and enable the draining feature of the DMA controller by reconfiguring the DMA transfer length according to this value (when the DMA mode is enabled) or perform only the required number of data accesses (when the DMA mode is disabled).

In receive mode (controller or target), if the RX FIFO threshold (the I2C\_BUF[13-8] RXTRSH bit field value + 1) is not reached, but the transfer ends on the I<sup>2</sup>C bus and data remains in the RX FIFO (less than the threshold), the receive draining interrupt (the I2C\_IRQSTATUS\_RAW[13] RDR bit) is asserted to inform the LH that it can read the amount of data in the RX FIFO (the I2C\_BUFSTAT[13-8] RXSTAT bit field). The LH performs a number of data read accesses equal to the I2C\_BUFSTAT[13-8] RXSTAT bit field (interrupt or polling mode), or reconfigures the DMA controller with the required value to drain the FIFO.

In controller transmit mode, if the TX FIFO threshold (the I2C\_BUF[5-0] TXTRSH bit field value + 1) is not reached, but the amount of data remaining to be written in the TX FIFO is less than the threshold, the transmit draining interrupt (the I2C\_IRQSTATUS\_RAW[14] XDR bit) is asserted to inform the LH that it can read the amount of data remaining to be written in the TX FIFO (the I2C\_BUFSTAT[5-0] TXSTAT bit field). The LH must write the required number of data bytes specified by the I2C\_BUFSTAT[5-0] TXSTAT bit field value or reconfigure the DMA controller with the value required to transfer the last bytes to the FIFO.

In controller mode, the LH can alternately not check the values of the I2C\_BUFSTAT[5-0] TXSTAT and I2C\_BUFSTAT[13-8] RXSTAT bit fields, because it can obtain this information internally (by computing the I2C\_CNT[15-0] DATACOUNT bit field value modulo I2C\_BUF[13-8] RXTRSH or I2C\_BUF[5-0] TXTRSH).

By default, the draining feature is disabled; it can be enabled using the I2C\_IRQENABLE\_SET[14] XDR\_IE or I2C\_IRQENABLE\_SET[13] RDR\_IE bits (default disabled) only for transfers with lengths not equal to the threshold values (I2C\_BUF[5-0] TXTRSH bit field value + 1 for the TX threshold or the I2C\_BUF[13-8] RXTRSH bit field value + 1 for the RX threshold).

### 12.1.3.4.8 I2C Noise Filter

The noise filter is used to suppress any noise that is 50 ns or less in case of F/S operation modes, and any noise that is 10 ns or less in case of HS mode operation. The noise filter is always one period of the INTERNAL\_CLK clock. This way, for HS mode operation (prescaler bypassed), the filter suppresses spikes of less than 10.4 ns.

For standard mode (for example, the I2C\_PSC[7-0] PSC bit field = 4), the maximum width of suppressed spikes is 46.1 ns.

To ensure correct filtering, the prescaler must be programmed accordingly by the I2Ci.I2C\_PSC[7-0] PSC bit field.

### 12.1.3.4.9 I2C System Test Mode

A system test mode is available for multicontroller I2C module testing. This mode is enabled by setting the I2C\_SYSTEST[15] ST\_EN bit to 1. When this bit is cleared to 0, the I2C controller is configured in normal operation mode.

In system test mode, the I2C\_SYSTEST[13-12] TMODE bit field selects the type of test. [Table 12-42](#) lists the tests available for the multicontroller HS I2C controllers.

**Table 12-42. I2C List of Tests**

I2C_SYSTEST[13-12] TMODE	Test	Description
00	Functional mode	Normal operation mode
01	Reserved (not used)	
10	Test of SCL serial clock line	The SCL line is driven with a permanent clock as if controlled with the parameters set in the I2C_PSC, I2C_SCLL, and I2C_SCLH registers.
11	Loop-back mode + SCL/SDA I/O	In controller transmit mode only, data transmitted out of the I2C_DATA register (write action) is received in the same I2C_DATA register through an internal path through the FIFO buffers. The interrupt request is normally generated if it is enabled. Moreover, the SCL and SDA are controlled with the I2C_SYSTEST[3-0] bits.

#### Note

When the I2C\_SYSTEST[13-12] TMODE bit field is set to 11, the I2C controller must be configured in I<sup>2</sup>C F/S (I2C\_CON[13-12] OPMODE set to 00) or I<sup>2</sup>C HS mode (I2C\_CON[13-12] OPMODE set to 01).

#### Note

In normal operation mode (the I2C\_SYSTEST[15] ST\_EN bit cleared to 0), the I2C\_SYSTEST[3-0] bits that control the SCL, SDA lines in system test mode are read-only bits.

In system test mode (the I2C\_SYSTEST[15] ST\_EN bit set to 1), the I2C\_IRQSTATUS\_RAW[4] XRDY, I2C\_IRQSTATUS\_RAW[3] RRDY, I2C\_IRQSTATUS\_RAW[10] XUDF, I2C\_IRQSTATUS\_RAW[11] ROVR, I2C\_IRQSTATUS\_RAW[2] ARDY and I2C\_IRQSTATUS\_RAW[1] NACK status bits can be set to 1 when the I2C\_SYSTEST[11] SSB bit is set to 1. Clearing the I2C\_SYSTEST[11] SSB bit to 0 does not clear the I2C\_IRQSTATUS\_RAW bits to 0. The I2C\_IRQSTATUS\_RAW bit field can be cleared to 0 only by writing 1 in the corresponding bits.

### 12.1.3.5 I2C Programming Guide

#### 12.1.3.5.1 I2C Low-Level Programming Models

##### 12.1.3.5.1.1 I2C Programming Model

This section describes the programming model of the multimaster I2C controllers configured in I<sup>2</sup>C mode.

##### 12.1.3.5.1.1.1 Main Program

###### 12.1.3.5.1.1.1.1 Configure the Module Before Enabling the I2C Controller

Before enabling the I2C controller, perform the following steps:

1. Enable the functional and interface clocks (see *WKUP I2C Clocks and Resets*, *MCU\_I2C Clocks and Resets*, and *I2C Clocks and Resets*).
2. Program the prescaler to obtain an approximately 12-MHz internal sampling clock by programming the corresponding value in the I2C\_PSC[7-0] PSC bit field. This value depends on the frequency of the functional clock (SYS\_CLK).
3. Program the I2C\_SCLL[7-0] SCLL and I2C\_SCLH[7-0] SCLH bit fields to obtain a bit rate of 100 Kbps or 400 Kbps. These values depend on the internal sampling clock frequency (see [Table 12-37](#)).
4. (Optional) Program the I2C\_SCLL[15-8] HSSCLL and I2C\_SCLH[15-8] HSSCLH bit fields to obtain a bit rate of 400 Kbps or 3.4 Mbps (for the second phase of HS mode). These values depend on the internal sampling clock frequency (see [Table 12-37](#)).
5. Configure the Own Address of the I2C controller by storing it in the I2C\_OA register. Up to four Own Addresses can be programmed in the I2C\_OA and I2C\_OAx registers (where x = 1, 2, 3) for each I2C controller.

---

#### Note

For a 10-bit address, set the corresponding expand Own Address bit in the I2C\_CON register.

6. Set the TX threshold (in transmitter mode) and the RX threshold (in receiver mode) by setting the I2C\_BUF[5-0] TXTRSH bit field to (TX threshold – 1) and the I2C\_BUF[13-8] RXTRSH bit field to (RX threshold – 1), where the TX and RX thresholds are greater than or equal to 1.
7. Take the I2C controller out of reset by setting the I2C\_CON[15] I2C\_EN bit to 1.

##### 12.1.3.5.1.1.1.2 Initialize the I2C Controller

To initialize the I2C controller, perform the following steps:

1. Configure the I2C\_CON register:
  - For controller or target mode, set the I2C\_CON[10] MST bit (0: target; 1: controller).
  - For transmitter or receiver mode, set the I2C\_CON[9] TRX bit (0: receiver; 1: transmitter).
2. If using an interrupt to transmit and receive data, set the corresponding bit in the I2C\_IRQENABLE\_SET register to 1 (the I2C\_IRQENABLE\_SET [4] XRDY\_IE bit for the transmit interrupt, the I2C\_IRQENABLE\_SET [3] RRDY bit for the receive interrupt).

##### 12.1.3.5.1.1.1.3 Configure Target Address and the Data Control Register

In controller mode, configure the target address register by programming the I2C\_SA[9-0] SA bit field and the number of data bytes (I<sup>2</sup>C data payload) associated with the transfer by programming the I2C\_CNT[15-0] DCOUNT bit field.

---

#### Note

For a 10-bit address, set the I2C\_CON[8] XSA bit to 1.

##### 12.1.3.5.1.1.1.4 Initiate a Transfer

Poll the I2C\_IRQSTATUS\_RAW [12] BB bit. If it is cleared to 0 (bus not busy), configure the I2C\_CON[0] STT and I2C\_CON[1] STP bits. To initiate a transfer, the I2C\_CON[0] STT bit must be set to 1, and it is not mandatory to set the I2C\_CON[1] STP bit to 1.

#### 12.1.3.5.1.1.5 Receive Data

Poll the I2C\_IRQSTATUS\_RAW [3] RRDY bit, or use the RRDY interrupt (the I2C\_IRQENABLE\_SET [3] RRDY\_IE bit must be set to 1) to read the receive data in the I2C\_DATA register.

If the transfer length does not equal the RX FIFO threshold (the I2C\_BUF[13-8] RTRSH bit field + 1), use the draining feature (enable the RDR interrupt by setting the I2C\_IRQENABLE\_SET [13] RDR\_IE bit to 1).

#### Note

In receive mode only, the I2C\_IRQSTATUS\_RAW [11] ROVR (receive overrun) bit indicates whether the receiver has experienced overrun. An overrun condition occurs when the shift register and the RX FIFO are full. An overrun condition does not result in data loss; the I2C controller simply holds SCL to low to prevent other bytes from being received.

The I2C\_IRQSTATUS\_RAW[7] AERR bit is set to 1 when a read access is performed in the I2C\_DATA register while the RX FIFO is empty. The corresponding interrupt can be enabled by setting the I2C\_IRQENABLE\_SET [7] AERR\_IE bit to 1.

#### 12.1.3.5.1.1.6 Transmit Data

Poll the I2C\_IRQSTATUS\_RAW [4] XRDY bit, or use the XRDY interrupt (the I2C\_IRQENABLE\_SET [4] XRDY\_IE bit must be set to 1) to write data to the I2C\_DATA register.

If the transfer length does not equal the TX FIFO threshold (the I2C\_BUF[5-0] TXTRSH bit field + 1), use the draining feature (enable the XDR interrupt by setting the I2C\_IRQENABLE\_SET [14] XDR\_IE bit to 1).

#### Note

In transmit mode only, the I2C\_IRQSTATUS\_RAW [10] XUDF bit indicates whether the transmitter has experienced underflow.

In controller transmit mode, underflow occurs when the shift register and the TX FIFO are empty and there are still some bytes to transmit (the value of the I2C\_CNT[15-0] DCOUNT bit field is not 0).

In target transmit mode, underflow occurs when the shift register and the TX FIFO are empty and the external I<sup>2</sup>C controller device still requests data bytes to be read.

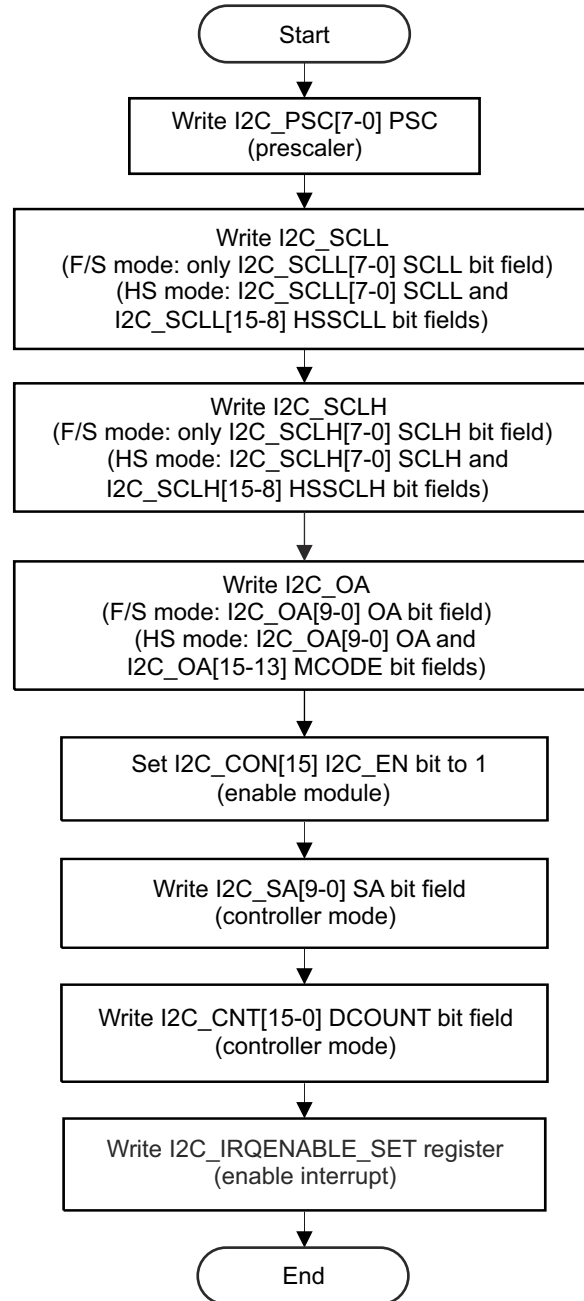
The I2C\_IRQSTATUS\_RAW [7] AERR bit is set to 1 when a write access is performed in the I2C\_DATA register while the TX FIFO is full. The corresponding interrupt can be enabled by setting the I2C\_IRQENABLE\_SET [7] AERR\_IE bit to 1.

#### 12.1.3.5.1.1.2 Interrupt Subroutine Sequence

1. Test for arbitration lost (the I2C\_IRQSTATUS\_RAW [0] AL bit) and resolve accordingly.
2. Test for no acknowledgment (the I2C\_IRQSTATUS\_RAW [1] NACK bit) and resolve accordingly.
3. Test for register access ready (the I2C\_IRQSTATUS\_RAW [2] ARDY bit) and resolve accordingly.
4. Test for receive data ready (the I2C\_IRQSTATUS\_RAW [3] RRDY bit) and resolve accordingly.
5. Test for transmit data ready (the I2C\_IRQSTATUS\_RAW [4] XRDY bit) and resolve accordingly.
6. Test for general call (the I2C\_IRQSTATUS\_RAW [5] GC bit) and resolve accordingly.
7. Test for start (S) condition (the I2C\_IRQSTATUS\_RAW [6] STC bit) and resolve accordingly. For this test, the functional clock must be inactive.
8. Test for access error (the I2C\_IRQSTATUS\_RAW [7] AERR bit) and resolve accordingly.
9. Test for bus free (the I2C\_IRQSTATUS\_RAW [8] BF bit) and resolve accordingly.

#### 12.1.3.5.1.1.3 Programming Flow-Diagrams

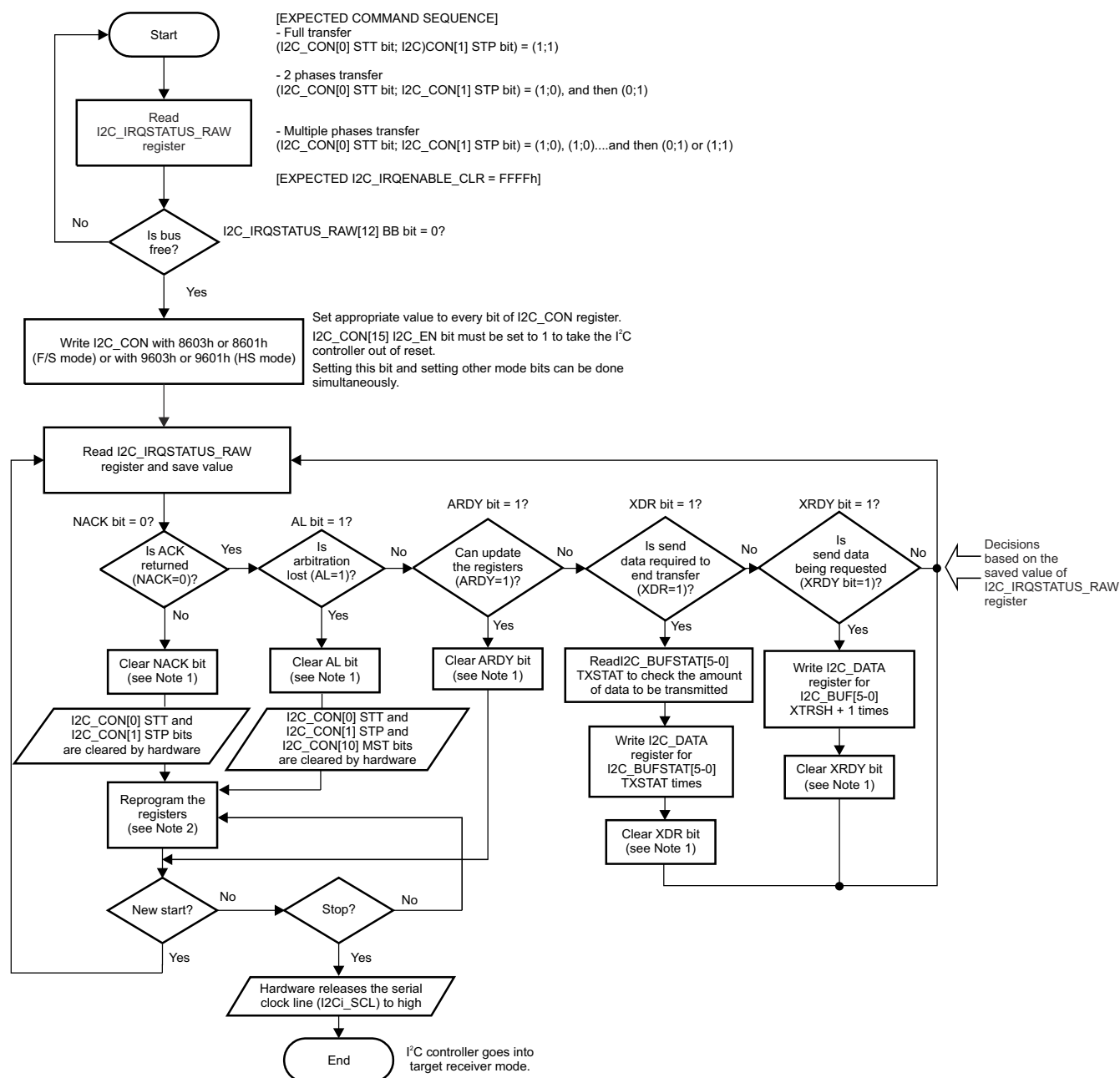
Figure 12-33 through Figure 12-39 are procedure flow charts for programming the F/S and HS I<sup>2</sup>C modes.



i2c-018

**Figure 12-33. I2C Setup Procedure**





I2C-019

- The NACK, AL, ARDY, XDR, and XRDY bits are cleared by writing 1 to each corresponding bit in the I2C\_IRQSTATUS register.
- Reprogram the registers means: I2C\_CON[11] STB and/or I2C\_CON[10] MST bit and/or I2C\_SA[9-0] SA register and/or I2C\_CNT[15-0] DCOUNT register and/or I2C\_CON[0] STT bit and/or I2C\_CON[1] STP bit.

**Figure 12-34. I2C Controller Transmitter Mode, Polling Method, in F/S and HS Modes**

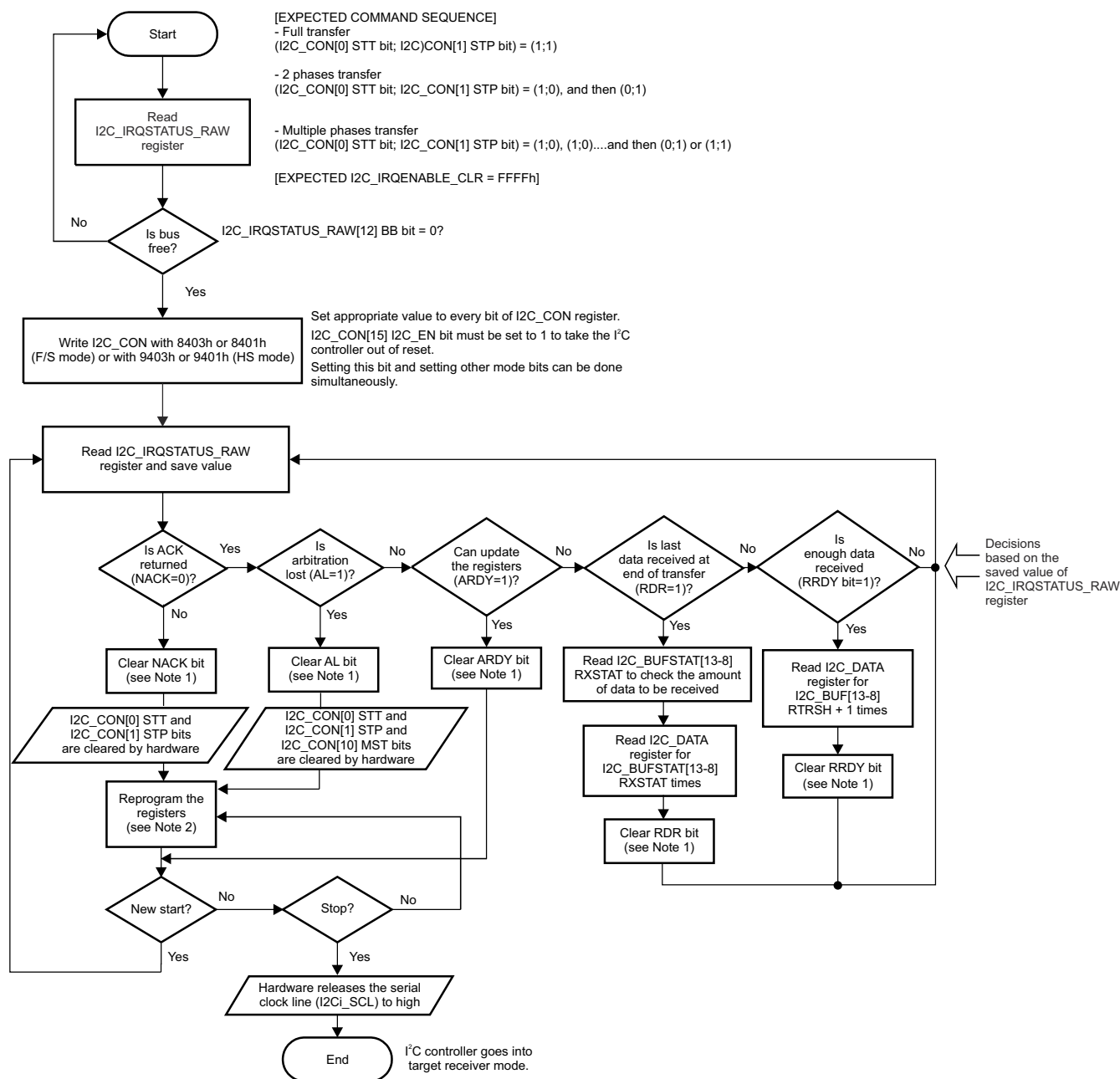
### Note

The FIFO clearing can be made when the module is configured as transmitter, the receiver send a NACK in the middle of the transfer, and there is still data in the FIFO.



## Note

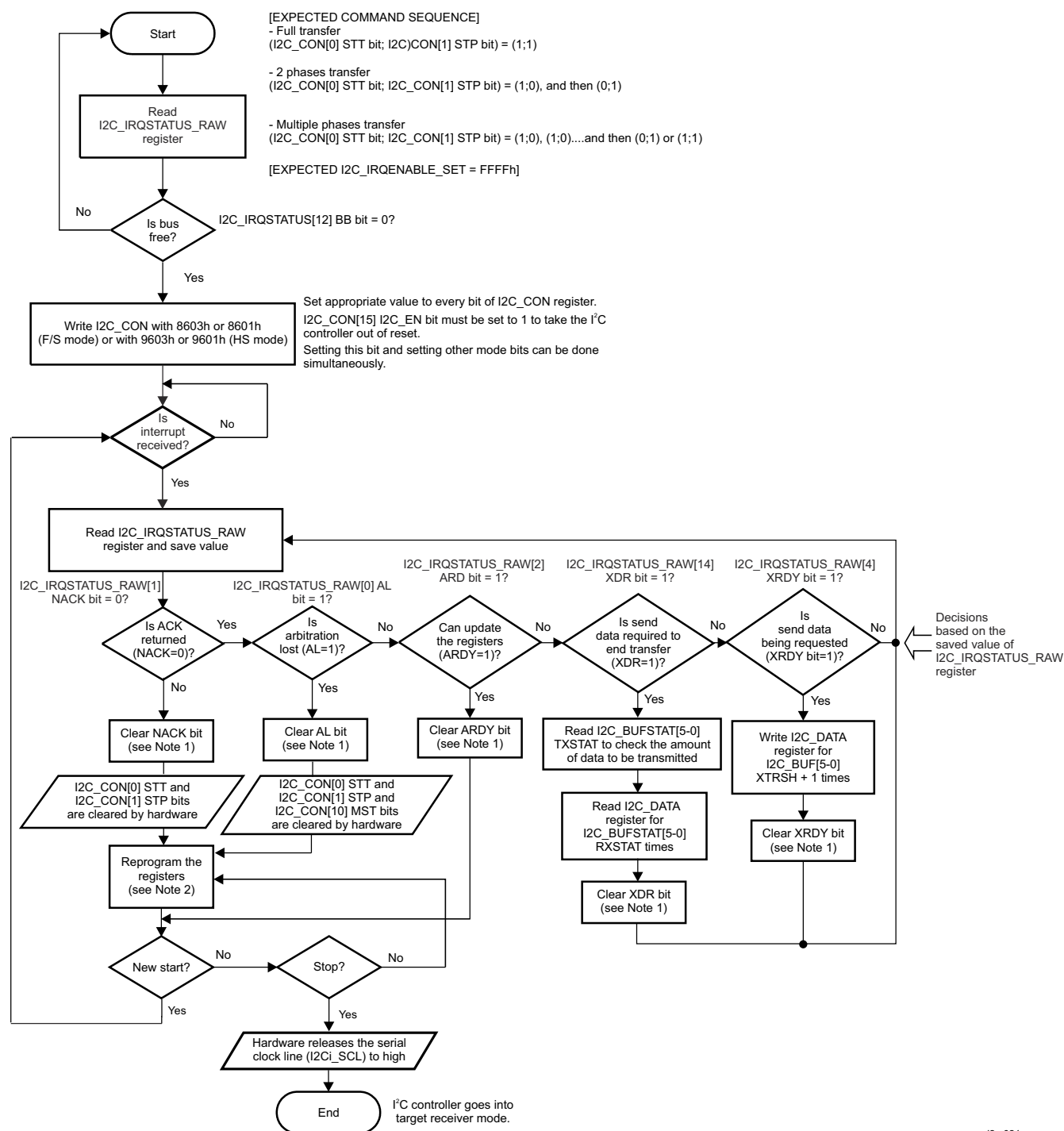
In HS mode, the Sr condition and clock frequency switching are automatically generated by the multicontroller I2C.



i2c-020

- The NACK, AL, ARDY, RDR, and RRDY bits are cleared by writing 1 to each corresponding bit in the I2C\_IRQSTATUS register.
- Reprogram registers means: I2C\_CON[11] STB and/or I2C\_CON[10] MST bit and/or I2C\_SA[9-0] SA register and/or I2C\_CNT[15-0] DCOUNT register and/or I2C\_CON[0] STT bit and/or I2C\_CON[1] STP bit.

**Figure 12-35. I2C Controller Receiver Mode, Polling Method, in F/S and HS Modes**



I2C-021

- The NACK, AL, ARDY, XDR, and XRDY bits are cleared by writing 1 to each corresponding bit in the I2C\_IRQSTATUS register.
- Reprogram registers means: I2C\_CON[11] STB and/or I2C\_CON[10] MST bit and/or I2C\_SA[9-0] SA register and/or I2C\_CNT[15-0] DCOUNT register and/or I2C\_CON[0] STT bit and/or I2C\_CON[1] STP bit.

**Figure 12-36. I2C Controller Transmitter Mode, Interrupt Method, in F/S and HS Modes**

---

**Note**

The FIFO clearing can be made when the module is configured as transmitter, the receiver send a NACK in the middle of the transfer, and there is still data in the FIFO.

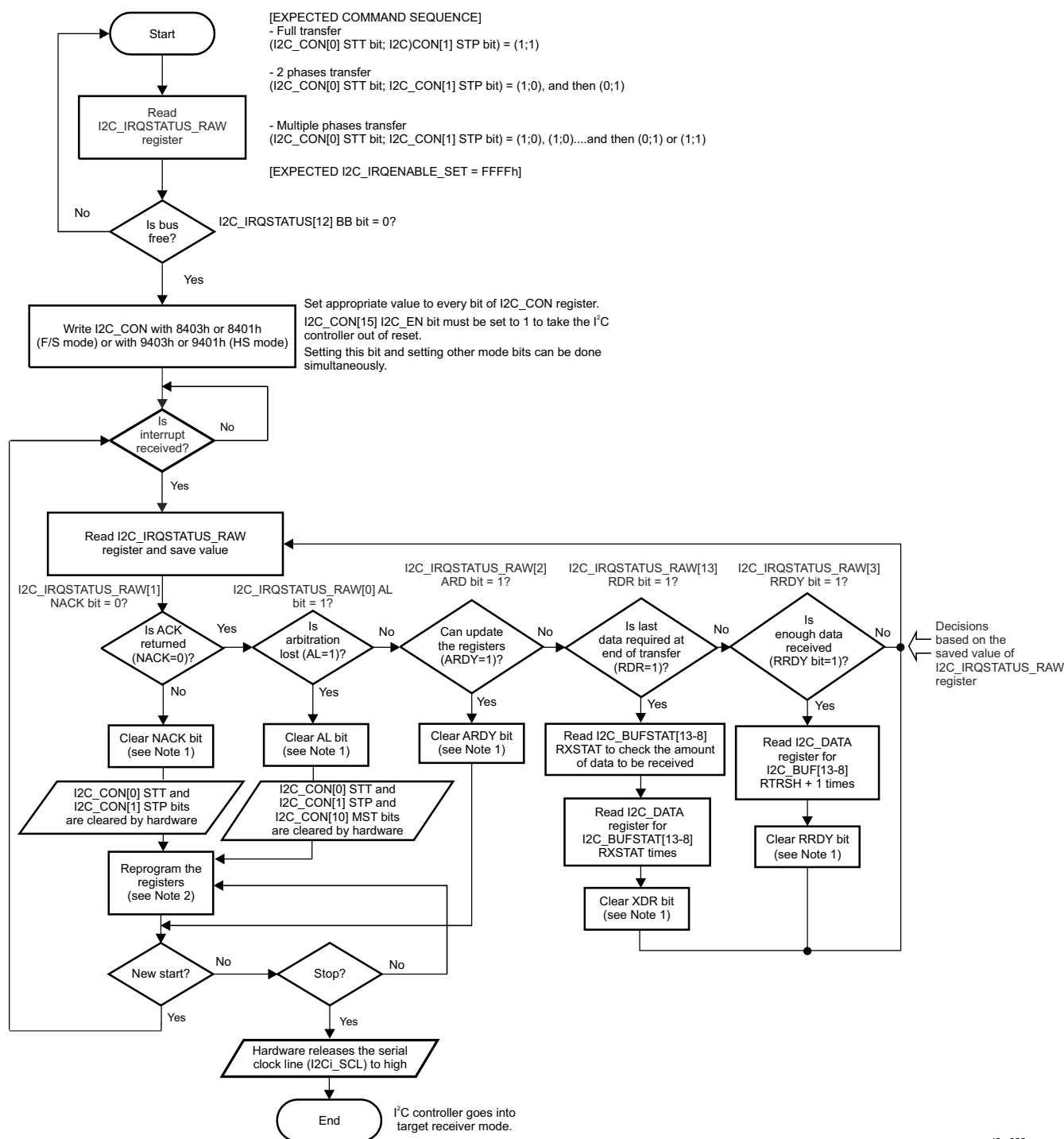
---

---

**Note**

In HS mode, the Sr condition and clock frequency switching are automatically generated by the multicontroller I2C.

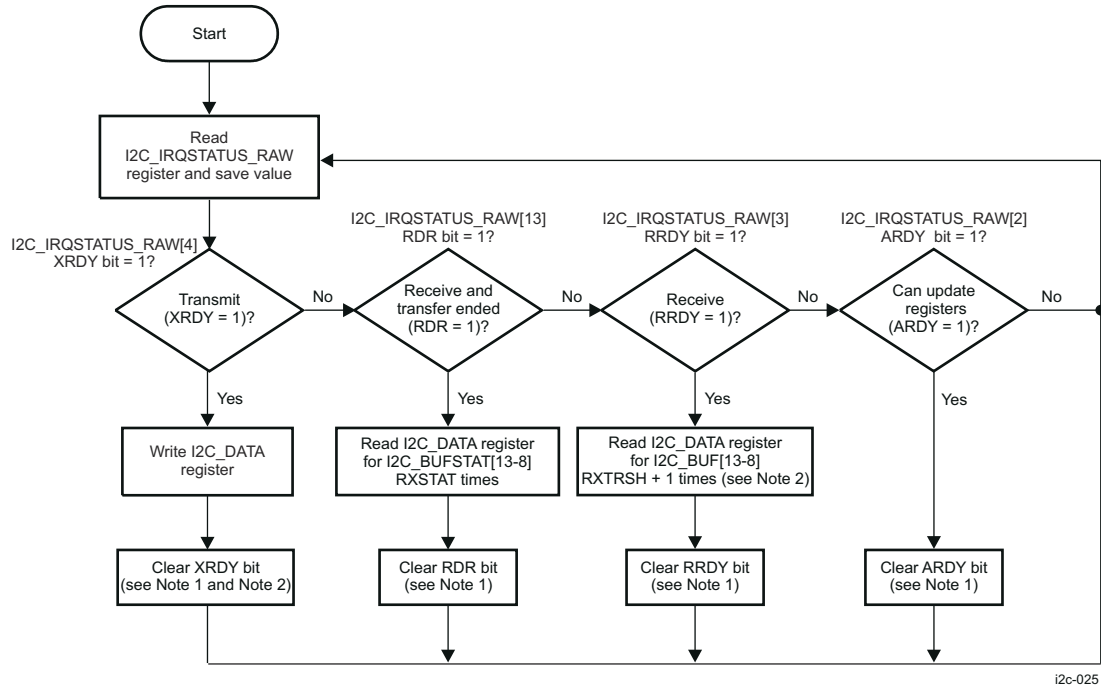
---



i2c-022

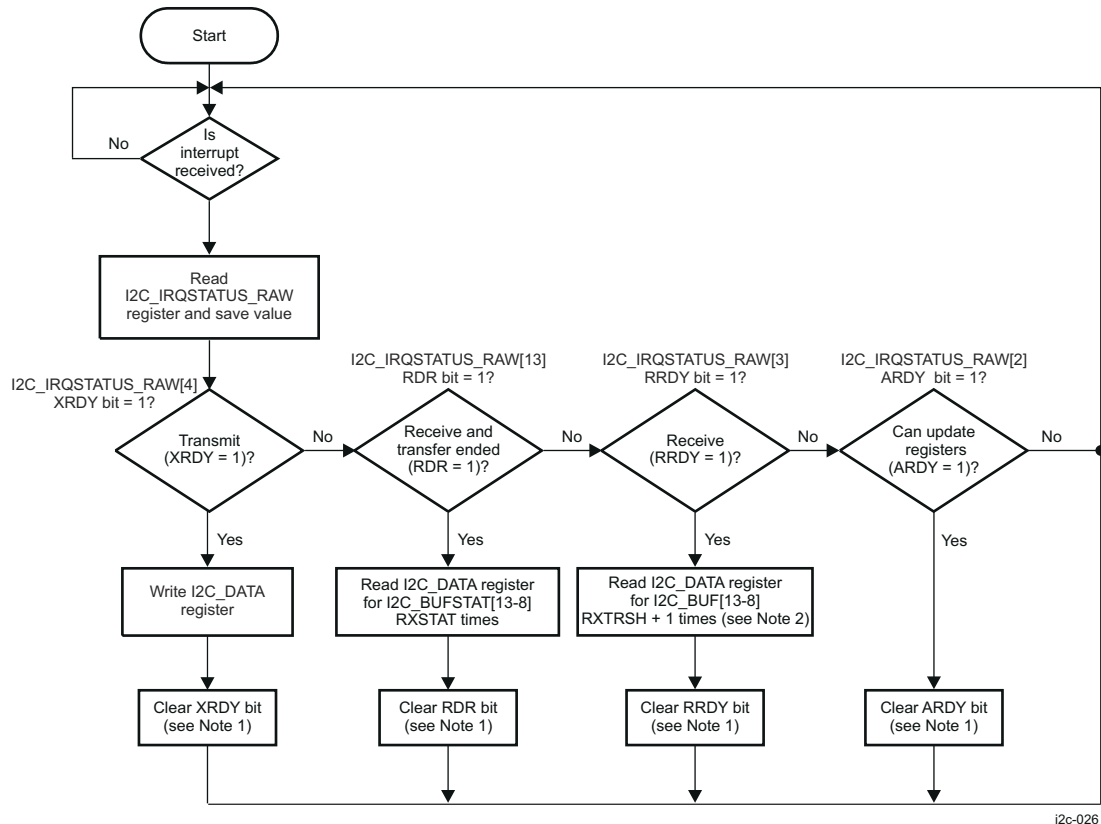
- The NACK, AL, ARDY, RDR, and RRDY bits are cleared by writing 1 to each corresponding bit in the I2C\_IRQSTATUS register.
- Reprogram registers means: I2C\_CON[11] STB and/or I2C\_CON[10] MST bit and/or I2C\_SA[9-0] SA register and/or I2C\_CNT[15-0] DCOUNT register and/or I2C\_CON[0] STT bit and/or I2C\_CON[1] STP bit.

**Figure 12-37. I2C Controller Receiver Mode, Interrupt Method, in F/S and HS Modes**



- A. The XRDY, RDR, RRDY, and ARDY bits are cleared by writing 1 to each corresponding bit in the I2C\_IRQSTATUS register.
- B. In target transmitter mode, the amount of data requested by the external controller I<sup>2</sup>C device is unknown; thus, the I2C\_BUF[5-0] XTRSH bit field must be configured to 0x0 (TX threshold = 1).

**Figure 12-38. I2C Target Transmitter/Receiver Mode, Polling**



- A. The XRDY, RDR, RRDY, and ARDY bits are cleared by writing 1 to each corresponding bit in the I2C\_IRQSTATUS register.

- B. In target transmitter mode, the amount of data requested by the external controller I<sup>2</sup>C device is unknown; thus, the I2C\_BUF[5-0] XTRSH bit field must be configured to 0x0 (TX threshold = 1).

**Figure 12-39. I2C Target Transmitter/Receiver Mode, Interrupt**

## 12.1.4 Improved Inter-Integrated Circuit (I3C) Interface

This section describes the Improved Inter-Integrated Circuit (I3C) module in the device.

### 12.1.4.1 I3C Overview

The device contains three Improved Inter-Integrated Circuit (I3C) controllers each of which provides an interface between a local host (LH), such as an Arm, and any I3C-bus-compatible device that connects via the I3C serial bus.

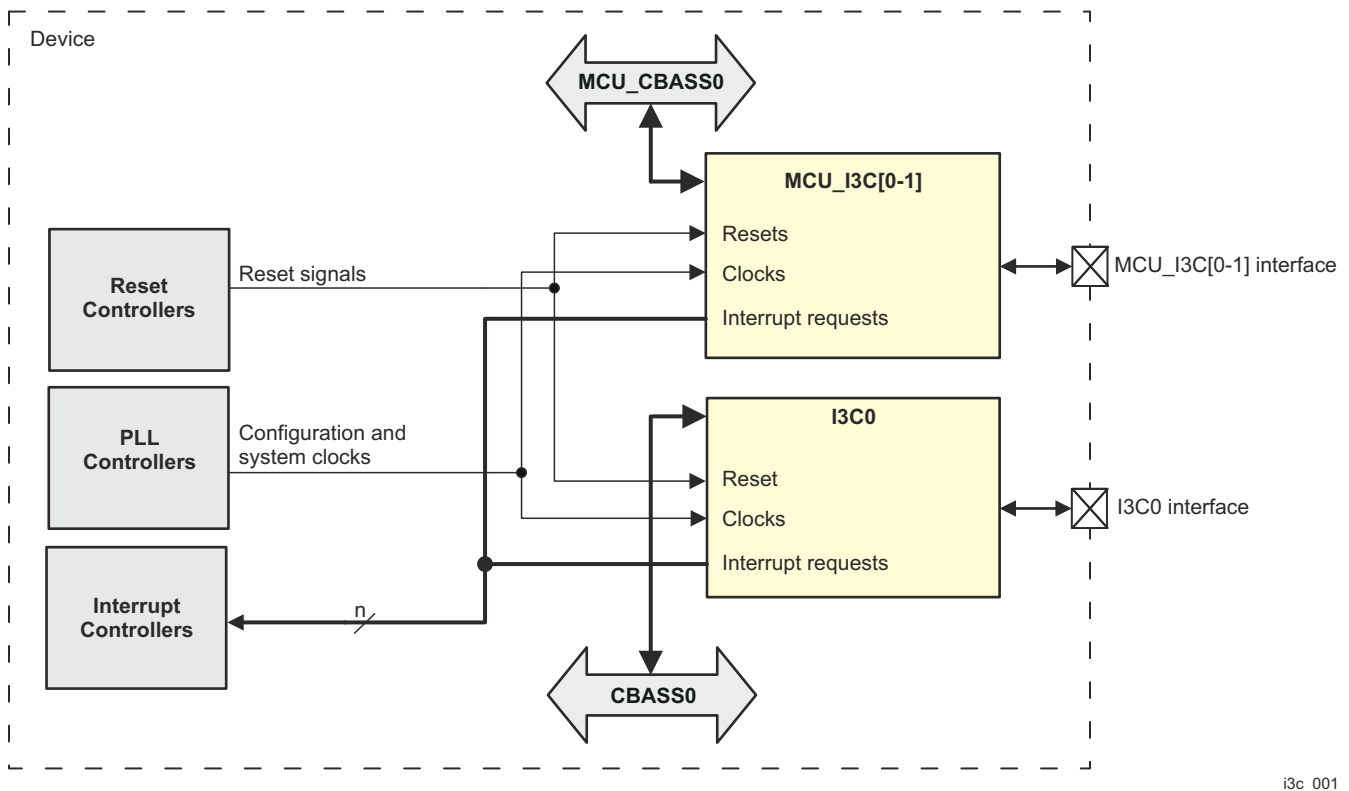
For the specific I/O timing characteristics of the I3C instances, see the device-specific Datasheet.

Table 12-43 shows I3C modules allocation across device domains.

**Table 12-43. I3C Allocation Across Device Domains**

Instance	Domain		
	WKUP	MCU	MAIN
MCU_I3C0	-	✓	-
MCU_I3C1	-	✓	-
I3C0	-	-	✓

Figure 12-40 shows the I3C modules overview.



i3c\_001

**Figure 12-40. I3C Modules Overview**

#### 12.1.4.1.1 I3C Features

Each I3C module has the following features:

- Communication Modes:
  - Single Data Rate (SDR) mode
  - High Data Rate – Dual Data Rate (HD-DDR) mode

- Bus Modes:
  - Pure Bus Mode (supports only I3C devices). Maximum supported frequency is 10.375 MHz
- Master and Multi-Master modes
- Common Command Codes (CCC)
- Bus Enumeration and Discovery:
  - Slaves implement standardized characteristics registers that can be queried by the master after power up to enumerate the devices attached to the bus.
- Hot-Join capability:
  - A slave device can be powered on and join the bus after initial enumeration.
- In Band Interrupts (IBI):
  - Slave devices request interrupts through the I3C bus rather than with separate side-band signals. This reduces the total IO count of the SoC as well as PCB routing.
- Dynamic Address Assignment (DAA):
  - The master can assign each slave device an address during configuration. This supports IBI because interrupt priority among slaves is arbitrated with the same address arbitration scheme that is used for all communication.
- Static Addressing (SA)
- FIFO Buffers:
  - CMD Queueing out going commands:
    - CMD0\_FIFO stores the least significant half-word of the command
    - CMD1\_FIFO stores the most significant half-word of the command
  - TXFIFO stores data transmitted with each command
  - RXFIFO stores data received as command response
  - IBI\_DATA\_FIFO stores data received as part of the in-band-interrupt transfer (IBI can transfer 0, 1, or N bytes of data payload depending on the slave)
- Immediate high priority command queue
- Registers to store the parameters for the response to an IBI interrupt from a number of slaves:
  - Each register holds the slave address and appropriate response type and payload length information for two different slaves, so that the controller can process in band interrupts without CPU intervention.

#### 12.1.4.1.2 I3C Not Supported Features

- High Data Rate – Ternary Symbol Legacy (HDR-TSL) mode
- High Data Rate – Ternary Symbol Pure (HDR-TSP) mode
- Mixed Bus (I3C devices and Legacy I2C devices with open drain or emulated open drain I/O)
- Secondary master modes
- Slave mode
- Master takeovers
- Secondary Master Hot-Join
- DMA service of I3C FIFOs
- Stamping interface (for DMA FIFO levels)
- GPIO Interface
- Optional CCCs:
  - GETMXDS



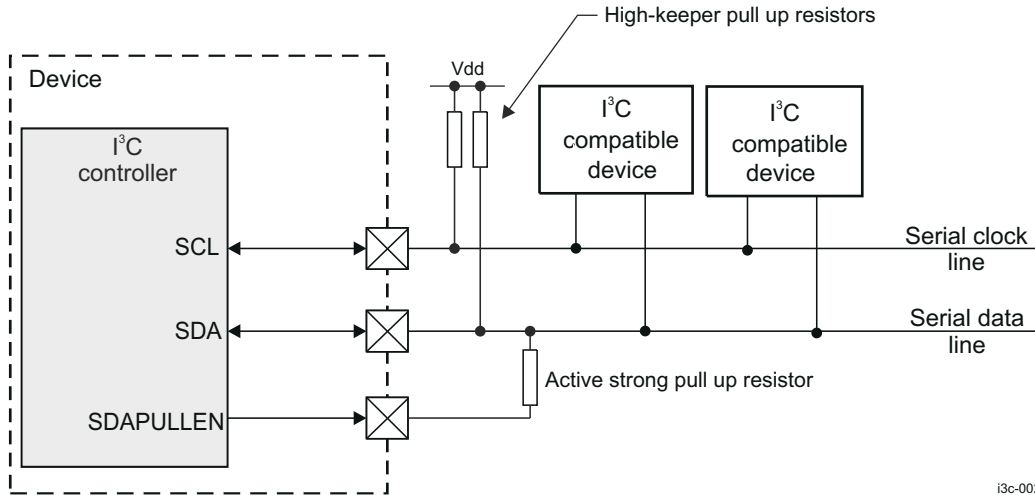
### 12.1.4.2 I3C Environment

The MCU\_I3C0, MCU\_I3C1 and I3C0 modules are hereinafter referred to as I3C module.

This section describes the I3C external connections (environment).

#### 12.1.4.2.1 I3C Typical Application

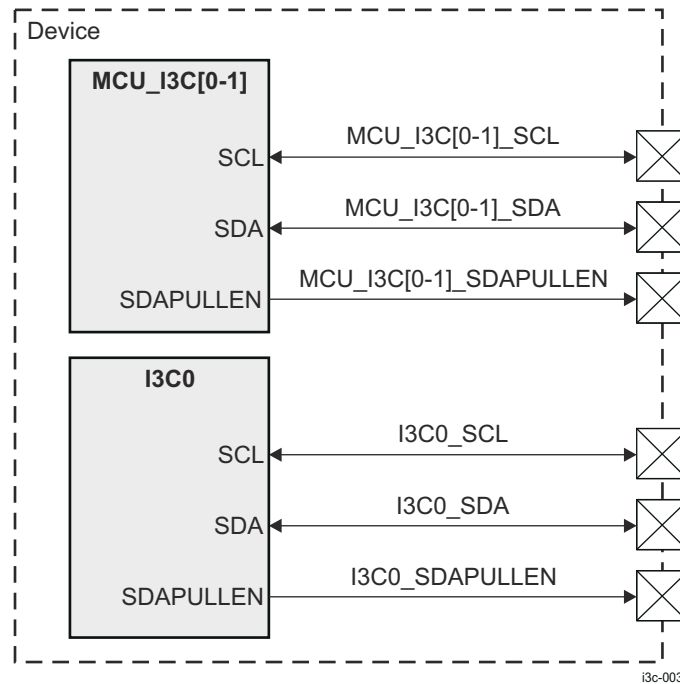
Figure 12-41 shows the I3C controllers and their related connections with I<sup>3</sup>C-compliant devices.



**Figure 12-41. I3C and Typical Connections to I3C Devices**

#### 12.1.4.2.1.1 I3C Pins for Typical Connections

Figure 12-42 shows the I3C controller pins used for typical connections with I<sup>3</sup>C devices.



**Figure 12-42. I3C Interface Signals**

#### 12.1.4.2.1.2 I3C Interface Typical Connections

Table 12-44 describes the I3C I/O signals.

**Table 12-44. I3C I/O Signals**

Module Pin	Device Level Signal	I/O <sup>(1)</sup>	Description	Module Pin Reset Value
<b>MCU_I3C[0-1]</b>				
SCL	MCU_I3C[0-1]_SCL	I/O	I <sup>3</sup> C serial clock line. Emulated open-drain output buffer.	
SDA	MCU_I3C[0-1]_SDA	I/O	I <sup>3</sup> C serial data line. Emulated open-drain output buffer.	
SDAPULLEN	MCU_I3C[0-1]_SDAPULLEN	O	I <sup>3</sup> C data pull enable. <sup>(2)</sup>	
<b>I3C0</b>				
SCL	I3C0_SCL	I/O	I <sup>3</sup> C serial clock line. Emulated open-drain output buffer.	
SDA	I3C0_SDA	I/O	I <sup>3</sup> C serial data line. Emulated open-drain output buffer.	
SDAPULLEN	I3C0_SDAPULLEN	O	I <sup>3</sup> C data pull enable. <sup>(2)</sup>	

(1) I = Input; O = Output; I/O = Bidirectional

(2) Enable buffer/switch for open drain class pull-up, enabled during open drain mode.

### 12.1.4.2.1.3

#### Note

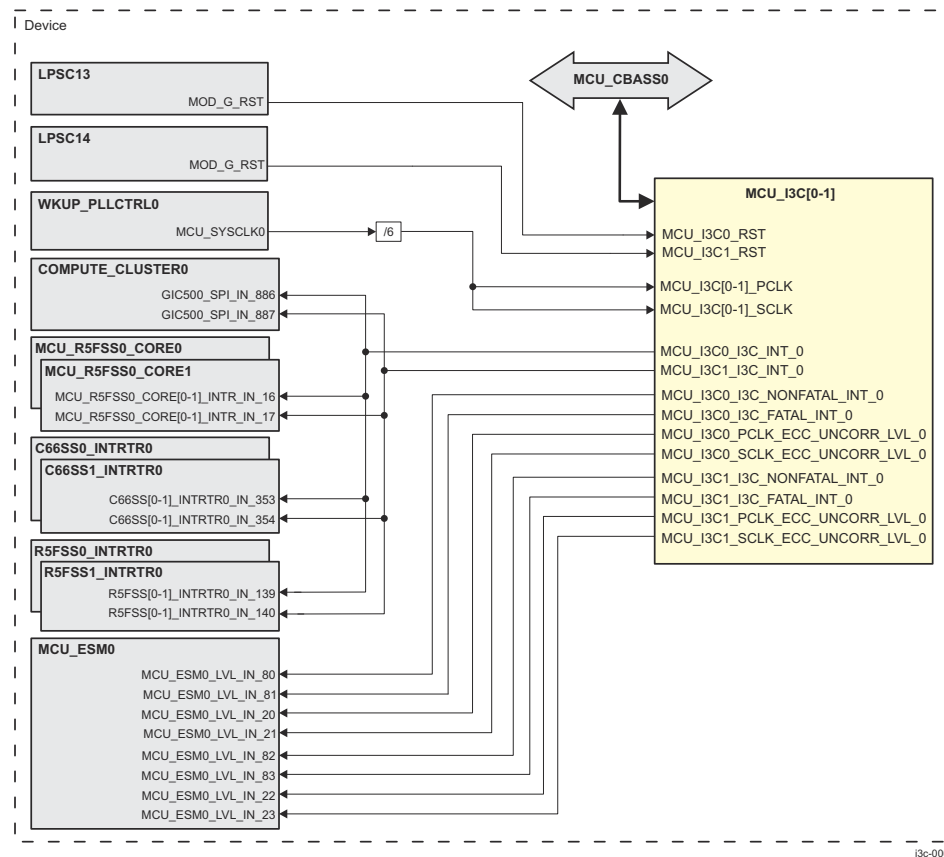
For more information about device level signals (buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

### 12.1.4.3 I3C Integration

This section describes I3C integration in the device, including information about clocks, resets, and hardware requests.

#### 12.1.4.3.1 I3C Integration in MCU Domain

There are two I3C modules integrated in the device MCU domain - MCU\_I3C0, MCU\_I3C1. [Figure 12-43](#) shows the integration of MCU\_I3C0, MCU\_I3C1.



**Figure 12-43. MCU\_I3C[0-1] Integration**

[Table 12-45](#) through [Table 12-47](#) summarize the integration of MCU\_I3C[0-1] in device MCU domain.

**Table 12-45. MCU\_I3C[0-1] Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
MCU_I3C0	WKUP_PSC0	PD0	LPSC13	MCU_CBASS0
MCU_I3C1	WKUP_PSC0	PD0	LPSC14	MCU_CBASS0

**Table 12-46. MCU\_I3C[0-1] Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
MCU_I3C0	MCU_I3C0_PCLK	MCU_SYSCLK0/6	WKUP_PLLCTRL0	MCU_I3C0 configuration clock
	MCU_I3C0_SCLK			MCU_I3C0 system clock
MCU_I3C1	MCU_I3C1_PCLK	MCU_SYSCLK0/6	WKUP_PLLCTRL0	MCU_I3C1 configuration clock
	MCU_I3C1_SCLK			MCU_I3C1 system clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
MCU_I3C0	MCU_I3C0_RST	MOD_G_RST	LPSC13	MCU_I3C0 reset
MCU_I3C1	MCU_I3C1_RST	MOD_G_RST	LPSC14	MCU_I3C1 reset

**Table 12-47. MCU\_I3C[0-1] Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCU_I3C0	MCU_I3C0_I3C_INT_0	GIC500_SPI_IN_886	COMPUTE_CLUSTER0	MCU_I3C0 interrupt request	Level
		C66SS0_INTRTR0_IN_353	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_353	C66SS1_INTRTR0		
		R5FSS0_INTRTR0_IN_139	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_139	R5FSS1_INTRTR0		
		MCU_R5FSS0_CORE0_INTR_IN_16	MCU_R5FSS0_CORE0		
		MCU_R5FSS0_CORE1_INTR_IN_16	MCU_R5FSS0_CORE1		
	MCU_I3C0_I3C_NONFATAL_INT_0	MCU_ESM0_LVL_IN_80	MCU_ESM0	MCU_I3C0 non fatal interrupt request	Level
	MCU_I3C0_I3C_FATAL_INT_0	MCU_ESM0_LVL_IN_81		MCU_I3C0 fatal interrupt request	Level
	MCU_I3C0_PCLK_ECC_UNCORR_LVL_0	MCU_ESM0_LVL_IN_20		MCU_I3C0 PCLK ECC interrupt request	Level
MCU_I3C1	MCU_I3C1_I3C_INT_0	MCU_ESM0_LVL_IN_21		MCU_I3C0 SCLK ECC interrupt request	Level
		GIC500_SPI_IN_887	COMPUTE_CLUSTER0	MCU_I3C1 interrupt request	Level
		C66SS0_INTRTR0_IN_354	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_354	C66SS1_INTRTR0		

**Table 12-47. MCU\_I3C[0-1] Hardware Requests (continued)**

	R5FSS0_INTRTR0_IN_140	R5FSS0_INTRTR0		
	R5FSS1_INTRTR0_IN_140	R5FSS1_INTRTR0		
	MCU_R5FSS0_CORE0_INTR_IN_17	MCU_R5FSS0_CORE0		
	MCU_R5FSS0_CORE1_INTR_IN_17	MCU_R5FSS0_CORE1		
MCU_I3C1_I3C_NONFATAL_INT_0	MCU_ESM0_LVL_IN_82	MCU_ESM0	MCU_I3C1 non fatal interrupt request	Level
MCU_I3C1_I3C_FATAL_INT_0	MCU_ESM0_LVL_IN_83		MCU_I3C1 fatal interrupt request	Level
MCU_I3C1_PCLK_ECC_UNCORR_LVL_0	MCU_ESM0_LVL_IN_22		MCU_I3C1 PCLK ECC interrupt request	Level
MCU_I3C1_SCLK_ECC_UNCORR_LVL_0	MCU_ESM0_LVL_IN_23		MCU_I3C1 SCLK ECC interrupt request	Level

#### 12.1.4.3.2 I3C Integration in MAIN Domain

A single I3C0 module is integrated in the device MAIN domain. [Figure 12-44](#) shows the integration of I3C0.

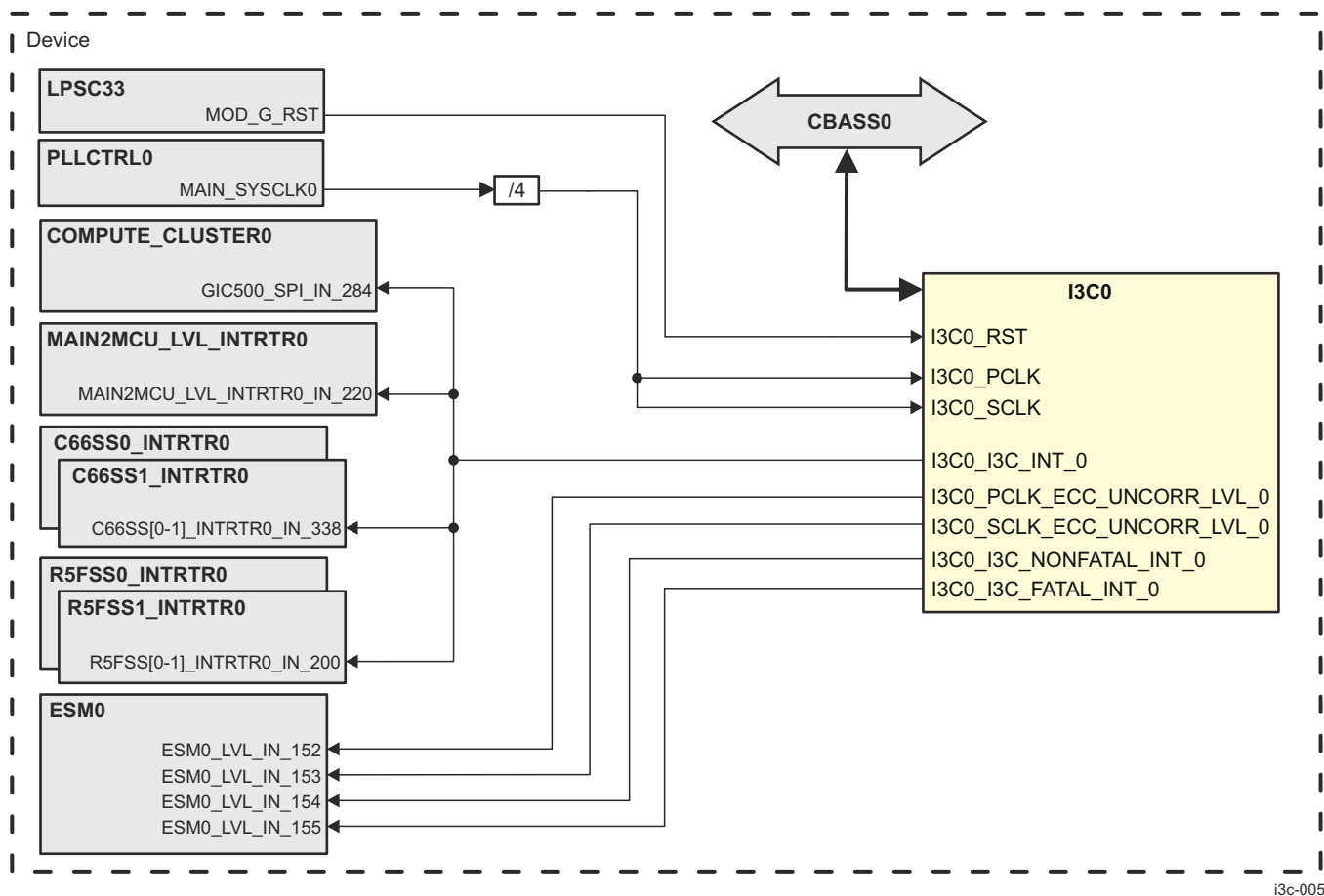


Figure 12-44. I3C Integration

Table 12-48 through Table 12-50 summarize the integration of I3C0 in the device MAIN domain.

Table 12-48. I3C0 Integration Attributes

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
I3C0	PSC0	PD0	LPSC33	CBASS0

Table 12-49. I3C0 Clocks and Resets

Clocks
--------

**Table 12-49. I3C0 Clocks and Resets (continued)**

Module Instance	Module Clock Input	Source Clock Signal	Source	Description
I3C0	I3C0_PCLK	MAIN_SYCLK0/4	PLLCTRL0	I3C0 configuration clock
	I3C0_SCLK			I3C0 system clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
I3C0	I3C0_RST	MOD_G_RST	LPSC33	I3C0 reset

**Table 12-50. I3C0 Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
I3C0	I3C0_I3C_INT_0	GIC500_SPI_IN_284	COMPUTE_CLUSTER0	I3C0 interrupt request	Level
		C66SS0_INTRTR0_IN_338	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_338	C66SS1_INTRTR0		
		R5FSS0_INTRTR0_IN_200	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_200	R5FSS1_INTRTR0		
		MAIN2MCU_LVL_INTRTR0_IN_220	MAIN2MCU_LVL_INTRTR0		
	I3C0_PCLK_ECC_UNCORR_LVL_0	ESM0_LVL_IN_152	ESM0	I3C0 PCLK ECC interrupt request	Level
	I3C0_SCLK_ECC_UNCORR_LVL_0	ESM0_LVL_IN_153		I3C0 SCLK ECC interrupt request	Level
	I3C0_I3C_NONFATAL_INT_0	ESM0_LVL_IN_154		I3C0 non fatal interrupt request	Level
	I3C0_I3C_FATAL_INT_0	ESM0_LVL_IN_155		I3C0 fatal interrupt request	Level





$$I3C\_PRESCL\_CTRL0[9-0] \text{ I3C} = \frac{I3C0\_SCL / MCU\_I3C[0-1]\_SCL}{(I3C\_FREQ \times 4)} - 1$$

i3c\_009

**Figure 12-46. I3C Base Frequency**

Programming the prescalers must be performed prior to enabling I3C Master controller with I3C\_CTRL[31] DEV\_EN bit in the control register.

In all timing schemes, the base frequency resulting from prescalers corresponds directly to the duration of the high level of the SCL line which takes always 50% of the base period. Depending on various conditions, the duration of low level is extended beyond the base period as necessary.

There is a specific relationship between SCL frequency and system clock frequency, which requires system clock to have a multiple of four cycles per each SCL period. Thus the selection of system clock frequency must be done with caution, since only specific frequencies will allow configuring SCL to fit into desired bandwidth.

Table 12-52 presents achievable SCL frequencies for exemplary set of system clock frequency values.

**Table 12-52. Achievable SCL Frequencies**

I3C0_SCLK / MCU_I3C[0-1]_SCLK frequency	I3C_PRESCL_CTRL0[9-0] I3C	Actual frequencies
100 MHz	0x2	8.33 MHz
133 MHz	0x3	8.313 MHz
166 MHz	0x3	10.375 MHz
200 MHz	0x4	10 MHz

#### 12.1.4.4.2.2 Asymmetric Push-Pull SCL Timing

In case of I<sup>3</sup>C messaging, asymmetric generation pattern for driving SCL can be enabled. The main purpose for this feature is supporting I3C slaves that are not capable to use maximal SCL frequency during private read/write data transfers. Host can recognize limited speed device by examining value of I3C\_DEV\_IDn\_RR2[15-8] bit-field for n = 0 to 11. Maximum frequency supported by slave can be retrieved by GETMSCL command. The low period of SCL is extended by the value programmed in I3C\_PRESCL\_CTRL1[15-8] PP\_LOW bit-field, which can be calculated using the following equation:

$$I3C\_PRESCL\_CTRL1[15-8] \text{ PP\_LOW} = \frac{I3C0\_SCL / MCU\_I3C[0-1]_SCL}{SCL\_FREQ \times (I3C\_PRESCL\_CTRL0[9-0] \text{ I3C} + 1)} - 4$$

i3c\_013

Where:

SCL\_FREQ: Maximum frequency supported by slave.

**Figure 12-47. SCL Asymmetric Push-Pull**

The I3C\_PRESCL\_CTRL1[15-8] PP\_LOW resolution is set to ¼ of SCL period. The high period of the SCL remains unchanged and is equal to half period of I<sup>3</sup>C frequency.

The low period used in push-pull mode for slaves with limited SCL capability results from following equation:

$$SCL\_LOW = I3C\_PRESCL\_CTRL1[15-8] \text{ PP\_LOW} \times \frac{T}{4}$$

i3c\_014

Where:

SCL\_LOW: SCL low period.

T: base SCL period.

**Figure 12-48. SCL Asymmetric Push-Pull Actual Low Period**

#### 12.1.4.4.2.3 Open-Drain SCL Timing

In open-drain mode, due to protocol requirement, low period cannot be shorter than 160ns to support slowly rising SDA line. To fulfill this constrain in I<sup>3</sup>C SDR mode, the low period must be always extended beyond the value resulting from the base I<sup>3</sup>C frequency. The low period of SCL is extended by the value programmed in I3C\_PRESCAL\_CTRL1[7-0] OD\_LOW bit-field, which can be calculated using the following equation:

$$\text{I3C\_PRESCAL\_CTRL1[7-0] OD\_LOW} = \frac{T\_LOW \times \text{I3C0\_SCL} / \text{MCU\_I3C[0-1]\_SCL}}{\text{I3C\_PRESCAL\_CTRL0[9-0] I3C} + 1} - 2$$

i3c\_010

Where:

T\_LOW: Required SCL low period.

#### Figure 12-49. SCL Open-Drain Low Period

The I3C\_PRESCAL\_CTRL1[7-0] OD\_LOW bit-field resolution is set to ¼ of SCL period. The high period remains unchanged and is equal to half-period of I<sup>3</sup>C frequency.

The actual SCL low period used in open drain mode results from the following equation:

$$\text{SCL\_LOW} = \text{I3C\_PRESCAL\_CTRL1[7-0] OD\_LOW} \times \frac{T}{4}$$

i3c\_015

Where:

SCL\_LOW: SCL low period.

T: base SCL period.

#### Figure 12-50. SCL Open-Drain Actual Low Period

#### 12.1.4.4.2.4 Changing Programmed Frequencies

Since SCL clock is derived, in order to change the SCL timing for any mode or to keep current I<sup>3</sup>C bus timing, the following procedure should be used:

1. Disable the I3C controller by setting the I3C\_CTRL[31] DEV\_EN bit to 0.
2. Adjust I3C0\_SCLK / MCU\_I3C[0-1]\_SCLK to the new frequency if necessary, avoiding any glitches during change.
3. Reprogram the prescaler registers to achieve desired I<sup>3</sup>C timing.
4. Enable the core by setting the I3C\_CTRL[31] DEV\_EN bit to 1.

#### 12.1.4.4.3 I3C Interrupt Requests

The I3C controller has two sets of five interrupt registers:

- Interrupt Enable Register (I3C\_MST\_IER)
- Interrupt Disable Register (I3C\_MST\_IDR)
- Interrupt Mask Register (I3C\_MST\_IMR)
- Interrupt Clear Register (I3C\_MST\_ICR)
- Interrupt Status Register (I3C\_MST\_ISR)

After reset, status (I3C\_MST\_ISR) and mask (I3C\_MST\_IMR) registers both read zeroes, so all interrupts are disabled. CPU handles interrupts by several typical operations:

- To activate an interrupt, interrupt enable register (I3C\_MST\_IER) should be programmed with 0x1 on bit positions corresponding to interrupts sources which should be signaled to CPU. This interrupt mask is written to I3C\_MST\_IMR and defines which sources will be propagated.
- After receiving interrupt, I3C\_MST\_ISR should be read to identify the interrupt source. Since I3C\_MST\_ISR is always updated by all interrupt sources regardless of I3C\_MST\_IMR, the I3C\_MST\_ISR value should be compared to the programmed mask (stored in software or read from I3C\_MST\_IMR) as it can contain other flags which were not propagated. Interrupt handler should clear flag which triggered interrupt in I3C\_MST\_ISR by writing 0x1 to corresponding bits of I3C\_MST\_ICR. It is crucial for correct interrupt source identification to read status, apply mask and clear current interrupt flag before another (from the same or

different source) interrupt can occur, so these actions should be executed at the beginning of the interrupt handler routine.

- If during operation additional interrupt sources need to be enabled, clearing their flags prior to enabling is necessary to avoid false signaling of events which occurred earlier.
- To disable interrupt source anytime during operation, corresponding bit in interrupt disable register (I3C\_MST\_IDR) should be programmed with 0x1, which in turn clears corresponding bit in I3C\_MST\_IMR, disabling interrupt source.

#### 12.1.4.4.4 I3C Power Configuration

Before I3C controller is put into operation, several settings need to be adjusted after cycling the power on. The general settings collected in I3C\_CTRL register. For most common usage scheme, two fields are the most important:

1. I3C\_CTRL[31] DEV\_EN bit allows enabling and disabling the I3C bus controller. It must be enabled before any I3C bus message is initiated, however following settings need to be programmed prior to enabling controller:
  - a. Bus Mode (Only Pure Bus Mode is supported)
  - b. SCL Divider prescalers (See *I3C Clock Configuration*)
  - c. Devices Retaining Registers (See *I3C Retaining Registers Space*)

During normal operation the controller should stay enabled. Before disabling, the I3C\_MST\_STATUS0[18] IDLE should be examined to confirm that no bus operation is pending.

2. I3C\_CTRL[1-0] BUS\_MODE bit-field is set to Pure Bus Mode by default. The I3C controller supports only I3C compatible devices on I3C bus and if changed bit-field has no effect. For more information see *I3C Not Supported Features*.

#### 12.1.4.4.5 I3C Dynamic Address Management

Enumeration of I3C devices involves Dynamic Address Assignment procedure. At power up of the I3C bus, the I3C Master should provide dynamic address for all I3C devices under the following conditions:

- If I3C Slave devices with Static Address are available (not supported for this device), firmware may choose to assign Dynamic Address using SETDASA CCC to shorten bus initialization time; such devices are called Static I3C Devices.
- For the devices that are enumerated using SETDASA CCC command the Provisional ID, BCR and DCR information should be manually acquired, using GETPID / GETBCR / GETDCR CCC commands. This step is not mandatory if the values are known prior to bus initialization and programmed into I3C Master by host application.
- Devices with no Static Address and these with Static Address not enumerated receive Dynamic Address using the DAA procedure initiated by issuing the ENTDAAC CCC command.
- For the devices that are enumerated using ENTDAAC CCC command the Provisional ID, BCR and DCR information are automatically received during DAA procedure.

During I3C bus operation, current master can disable one or all I3C devices by sending Direct or Broadcast RSTDAA CCC command. In such case, corresponding \*\_CLR bits in I3C\_DEVS\_CTRL should be set by firmware to clean retaining registers. In order to re-enable these devices, DAA procedure must be performed (initiated by master or by Hot-Join requests sent by slave devices).

Depending on I3C slave implementation, the slave that received RSTDAA CCC command may restore its static address. In order to perform private transfers to such slave or assign dynamic address using SETDASA CCC command rather than DAA procedure, the corresponding I3C\_DEV\_IDn\_RR0[7-0] DEV\_ADDR bit field (n = 0 to 11) should be updated with static address. Clearing Retaining Registers by setting corresponding I3C\_DEVS\_CTRL register is not mandatory in this scenario.

Current master can also assign new Dynamic Address to device which already has one by sending SETNEWDA command. After sending this command with new Dynamic Address in the data field, the I3C\_DEV\_IDn\_RR0[7-0] DEV\_ADDR bit field (n = 0 to 11) must be also updated with this address in order to perform subsequent transaction to this device.

#### 12.1.4.4.6 I3C Retaining Registers Space

The information on configuration, Provisional ID, BCR and DCR of each device connected to the I<sup>3</sup>C bus is stored by the I<sup>3</sup>C master in a set of retaining registers divided into Device ID sections associated with each device. Application host take any action to guarantee maintaining this information anytime the bus topology changes.

The retaining register space is divided to up to 12 sections (depending on selected configuration), from I3C\_DEV\_ID0 to I3C\_DEV\_ID11. Each containing three 32-bit registers to hold whole necessary information on I3C devices connected to the bus (e.g. hot-join device is attached, some of DA are reset to new value, etc.).

The AMBA APB interface address space for retaining registers starts at 0x080 and is divided into 12 quadruplets.

The map organization is presented in [Table 12-53](#) below.

**Table 12-53. Retaining Registers Map Organization**

Register	Address	Contents
I3C_DEV_ID0_RR0	0x080	Configuration register which stores settings for device that needs to be programmed by application host.
I3C_DEV_ID0_RR1	0x084	Register stores MSB of Provisional ID <sup>(1)</sup>
I3C_DEV_ID0_RR2	0x088	Register stores LSB of Provisional ID, DCR, BCR <sup>(1)</sup>
I3C_DEV_ID1_RR0	0x090	Configuration register which stores settings for device that needs to be programmed by application host.
I3C_DEV_ID1_RR1	0x094	Register stores MSB of Provisional ID <sup>(1)</sup>
I3C_DEV_ID1_RR2	0x098	Register stores LSB of Provisional ID, DCR, BCR <sup>(1)</sup>
I3C_DEV_ID2_RR0	0x0A0	Configuration register which stores settings for device that needs to be programmed by application host.
I3C_DEV_ID2_RR1	0x0A4	Register stores MSB of Provisional ID <sup>(1)</sup>
I3C_DEV_ID2_RR2	0x0A8	Register stores LSB of Provisional ID, DCR, BCR <sup>(1)</sup>
...	...	...
I3C_DEV_ID11_RR0	0x130	Configuration register which stores settings for device that needs to be programmed by application host.
I3C_DEV_ID11_RR1	0x134	Register stores MSB of Provisional ID <sup>(1)</sup>
I3C_DEV_ID11_RR2	0x138	Register stores LSB of Provisional ID, DCR, BCR <sup>(1)</sup>

(1) The information is relevant only for I<sup>3</sup>C devices and is automatically filled by hardware during DAA procedure. Application host needs to fill the information for all devices with Dynamic Addresses Assignment with SETDASA CCC command.

A special device control register (I3C\_DEVS\_CTRL) is provided to store information on devices currently connected to the bus. Each DeviceID section contains configuration register and two registers holding Provisional ID, BCR and DCR of the corresponding device. The active state bit is set automatically by hardware after Dynamic Address is assigned to given device. Firmware shall set it for devices enumerated with SETDASA CCC command after their Retaining Registers are configured and before any transaction to these devices is initiated.

The clearing bits need to be set by firmware every time RSTDAA CCC is send to corresponding devices (firmware should clear all of them for broadcast RSTDAA) or controller constantly receives NACK response and application recognizes device as no longer connected to the bus.

#### 12.1.4.4.7 I3C Dynamic Address Assignment Procedure

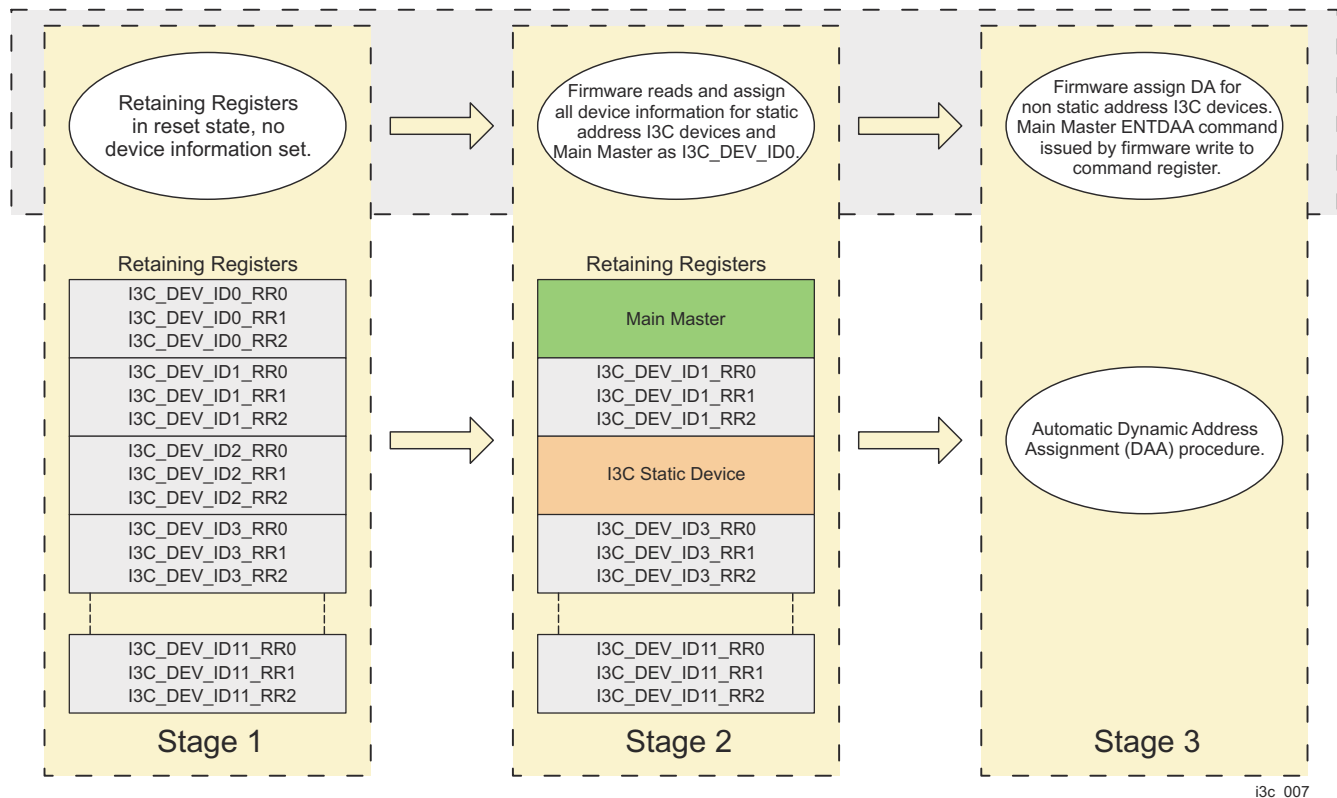
The I3C bus initialization procedure can be decomposed into three main stages, described below:

- **Stage 1:** On startup, the I3C master does not have any information about I<sup>3</sup>C slave devices and all Retaining Registers are set to the default reset vector table;

- **Stage 2:** Is fully maintained by application host. The first register section (Device ID0) is reserved for I3C master self-description. The following registers starting from Device ID1 will be filled by DAA procedure in Stage 3. This implies rules firmware should follow:
  - I<sup>3</sup>C devices with Static Address which firmware assume to enumerate with SETDASA CCC command shall be localized in Retaining Registers table (see [Figure 12-51](#)) after address space that reserved for I<sup>3</sup>C devices intended to be enumerated with ENTDAAs;
  - Corresponding I3C\_DEVS\_CTRL bits for active should be set;
- **Stage 3:** Before attempting DAA procedure, firmware should take following actions:
  - For Static I3C Devices intended to be enumerated with SETDASA CCC command:
    - Configuration registers should be programmed with Static Addresses and other settings relevant for I3C device;
    - Corresponding I3C\_DEVS\_CTRL bits for active should be set;
    - Dynamic Addresses should be assigned using SETDASA CCC command;
    - Configuration registers should be programmed with just assigned Dynamic Addresses;
    - Provisional ID / BCR / DCR Registers (I3C\_DEV\_IDn\_RR1 and I3C\_DEV\_IDn\_RR1, n = 0 to 11) should be programmed with data either predefined by application or obtained by reading PID / BCR / DCR with related CCC;
  - For I<sup>3</sup>C Devices intended to be enumerated with ENTDAAs CCC command:
    - Configuration registers should be programmed with Dynamic Addresses and other settings relevant for I3C Device;

Once all above conditions are satisfied the firmware can issue the ENTDAAs CCC command, which is considered as start condition for Stage 3.

[Figure 12-51](#) presents the I3C bus initialization for DAA procedure.



**Figure 12-51. Power-up I3C Bus Initialization**

i3c\_007

Figure 12-52 presents the I<sup>3</sup>C flow diagram for DAA procedure.



**Figure 12-52. Flow Diagram for DAA Procedure**

Static  $^{13}\text{C}$  devices have already assigned Dynamic Address with SETDASA CCC command and stay quiet during DAA procedure.

Although arbitration continues over BCR and DCR values, it is not guaranteed to be resolved in this phase, if not resolved during Provisional ID period.

- I3C\_CMD0\_FIFO[30] IS\_CCC – should be set to 1;
- I3C\_CMD0\_FIFO[7-0] CCC – specifies the Common Command Code to be sent;
- I3C\_CMD0\_FIFO[7-1] DEV\_ADDR – should address slave device in case of Direct CCC command;
- I3C\_CMD0\_FIFO[23-12] PL\_LEN – specifies the length of the Data field for commands (the number of bytes to be sent for particular CCC);
- I3C\_CMD0\_FIFO[0] RNW – defines the direction of the Data field: 0 for writing data to slave(s) and 1 for reading data from slaves;
- I3C\_CMD0\_FIFO[29] BCH – when two sequential directed CCC messages of the same type are sent; this bit determines whether the broadcast address and message code should be issued before the second address and command code;
- I3C\_CMD0\_FIFO[25] RSBC – use in conjunction with BCH when issuing short CCC messages;



The payload for CCC commands writing data to the slaves needs to be written to the TX FIFO prior to sending the two Command Words. The payload received by reading CCC commands can be read after receiving the COMP interrupt corresponding to the command. Note that ENTDAAC command uses payload of 0, since the data it receives is not stored in RX FIFO; the amount of data depends on the number of slave responding to the command.

When sending a sequence of two identical directed CCC messages to different slaves, the I3C\_CMD0\_FIFO[29] BCH and I3C\_CMD0\_FIFO[25] RSBC Command Word fields can be used to send the second message as short CCC. I3C\_CMD0\_FIFO[29] BCH should be set to 0 for the second CCC message. Instead of sending the broadcast address, waiting for it to be acknowledged and then sending the message code, this will issue restart pattern and immediately transfer the address and payload afterwards. If I3C\_CMD0\_FIFO[29] BCH is set to 1 the second CCC message is sent with broadcast address and command code in the beginning. In all other CCC message cases the I3C\_CMD0\_FIFO[29] BCH field is disregarded. Because the short CCC is preceded by restart pattern in order for this to work the I3C\_CMD0\_FIFO[25] RSBC field for the first command should also be set to 1.

Table 12-54 below summarizes available CCC set along with their payload layout in data FIFOs and Command Word settings.

**Table 12-54. CCC Commands**

CCC <sup>(1)</sup>	Description	Payload Layout <sup>(2)</sup>	PL	RNW
ENEC_BC (0x00) ENEC_DC (0x80)	Enables slave event driven interrupts	TX FIFO: {28'b0, 1'bHJ, 1'b0, 1'bMR, 1'bINT} (HJ – Hot-Join, MR – Mastership Request IBI, INT – regular IBI)	1	0
DISEC_BC (0x01) DISEC_DC (0x81)	Disables slave event driven interrupts	TX FIFO: {28'b0, 1'bHJ, 1'b0, 1'bMR, 1'bINT} (HJ – Hot-Join, MR – Mastership Request IBI, INT – regular IBI)	1	0
ENTAS0_BC (0x02) ENTAS0_DC (0x82)	Set activity to State 0 (normal operation)	No payload	0	0
RSTDAA_BC (0x06) RSTDAA_DC (0x86)	Forget current Dynamic Address and wait for new one	No payload	0	0
ENTDAA_BC (0x07)	Enter DAA procedure to assign Dynamic Addresses to unassigned devices	See Section 12.1.4.5.3, <i>Initiate DAA Procedure</i>	0	0
SETMWL_BC (0x09) SETMWL_DC (0x89)	Set maximum write length of a single private transfer	TX FIFO: {16'b0, LSB, MSB} (LSB and MSB of 16-bit Max Write Length value)	2	0
SETMRL_BC (0x0A) SETMRL_DC (0x8A)	Set maximum read length of a single private transfer	TX FIFO: {16'b0, LSB, MSB} (LSB and MSB of 16-bit Max Write Length value)	2	0
DEFSLVS_BC (0x08)	Broadcast triples of DA, DCR and SA (or 0) for each slave	No payload	0	0
ENTHDR_BC (0x20)	Switch to HD-DDR mode	No payload	0	0
SETDASA_DC (0x87)	Set Dynamic Address to slave with Static Address	TX FIFO: {24'b0, 7'bDA, 1'bP} (DA – new Dynamic Address, P – XNOR of DA bits)	1	0
SETNEWDA_DC (0x88)	Assign new Dynamic Address	TX FIFO: {24'b0, 7'bDA, 1'bP} (DA – new Dynamic Address, P – XNOR of DA bits)	1	0
GETMWL_DC (0x8B)	Get maximum write length of a single private transfer	RX FIFO: {16'b0, LSB, MSB} (LSB and MSB of 16-bit Max Write Length value)	2	1
GETMRL_DC (0x8C)	Get maximum read length of a single private transfer	RX FIFO: {16'b0, LSB, MSB} (LSB and MSB of 16-bit Max Read Length value)	2	1

**Table 12-54. CCC Commands (continued)**

CCC <sup>(1)</sup>	Description	Payload Layout <sup>(2)</sup>	PL	RNW
GETPID_DC (0x8D)	Get Slave's Provisional ID value	RX FIFO[n+1]: {16'b0, <b>PID0</b> , <b>PID1</b> } RX FIFO[n]: { <b>PID2</b> , <b>PID3</b> , <b>PID4</b> , <b>PID5</b> }	6	1
GETBCR_DC (0x8E)	Get Device's Bus Characteristic Register value	RX FIFO: {24'b0, <b>BCR</b> }	1	1
GETDCR_DC (0x8F)	Get Device's Characteristic Register value	RX FIFO: {24'b0, <b>DCR</b> }	1	1
GETSTATUS_DC (0x90)	Read Device's operating status	RX FIFO: {16'b0, 2'b <b>AM</b> , 1'b <b>PE</b> , 1'b0, 4'b <b>INT</b> , 8'b0} ( <b>AM</b> – Activity Mode, <b>PE</b> – Protocol Error, <b>INT</b> – Pending Interrupt number)		
GETACCMST_DC (0x91)	Get accept mastership	RX FIFO: {24'b0, <b>SlaveAddr</b> }	1	1
GETMXDS_DC (0x94)	Read Maximum SCL Frequency of the slave	RX FIFO: {16'b0, <b>MaxRd</b> , <b>MaxWr</b> } ( <b>MaxRd</b> – Clock to Data Turnaround Time and Maximum Sustained Read Data Rate, <b>MaxWr</b> – Maximum Sustained Write Data Rate)	1	1
GETHRCAP (0x96)	Ask slave for HDR modes it supports (can be sent only if BCR identifies HDR support)	RX FIFO: {24'b0, <b>HDRCAP</b> } ( <b>HDRCAP</b> – HDR capability modes)	1	1

(1) CCC column displays:

- The name of the Common Command Code, suffixed with **\_BC** for Broadcast CCC and **\_DC** for Direct CCC;
- In parentheses is value of CCC field to be used in Command Word 1;

(2) Payload Layout column displays:

- {<...>, <field1>, <field0>} - concatenation of 8-bit CCC data fields within 32-bit FIFO cell;
- <width>'b<field\_name> - field <field\_name> takes <width> bits (8 bits if not specified);
- The payload fields are denoted in bold;

#### 12.1.4.4.9 I3C In-Band Interrupt

In order to handle an incoming In-Band Interrupt from particular slave device, the address and configuration of IBI handling for this device needs to be stored in the I3C\_SIR\_MAP0 through I3C\_SIR\_MAP5 registers. This register map holds information on device address, length of payload, device role and response. The map organization is presented in the [Table 12-55](#) below.

**Table 12-55. SIR Map Organization**

Register	[31-16] bit fields	[15-0] bit fields
I3C_SIR_MAP0	Device 1	Device 0
I3C_SIR_MAP1	Device 3	Device 2
I3C_SIR_MAP2	Device 5	Device 4
I3C_SIR_MAP3	Device 7	Device 6
I3C_SIR_MAP4	Device 9	Device 8
I3C_SIR_MAP5	Reserved	Device 10

All devices requesting IBI, but not added to the SIR Map, will receive the NACK response from I3C master device.

Example for IBI Map configuration looks as follows:

- Write 0x80470445 which is translated to the following configurations for first two devices:
  - Regular slave with DA = 0x22 to be ACKed and receive 4 bytes of payload (Device 0)
  - Secondary master with DA = 0x23 to be ACKed and receive no payload (Device 1)
- Write 0x004A0749 which is translated to the following configurations for the third and fourth device:
  - Regular slave with DA = 0x24 to be ACKed and receive 7 bytes of payload (Device 2)



- b. Regular slave with DA = 0x25 to be NACKed (Device 3)
3. Write 0x0000034D which configures the last fifth device as follows:
  - a. Regular slave with DA = 0x26 to be ACKed and receive 3 bytes of payload (Device 4)

After receiving regular or mastership request IBI interrupt, firmware needs to read the I3C\_IBIR register to determine which device requested IBI. The ordering of bits in the I3C\_IBIR register corresponds to ordering of devices stored in I3C\_SIR\_MAP0 through I3C\_SIR\_MAP5 registers. Note that the IBI capable devices are enumerated and ordered independently from device ordering in the retaining registers used during DAA procedure.

#### **12.1.4.4.9.1 Regular I3C Slave In-Band Interrupt**

When regular IBI request appears on the I<sup>3</sup>C bus, I<sup>3</sup>C master hardware automatically handles whole IBI procedure based on settings in I3C\_SIR\_MAP0 through I3C\_SIR\_MAP5 registers. As the IBI procedure acknowledged by I3C master finishes (including payload reception), host is informed by the IBI interrupt that such event occurred and was successfully processed. At this point, firmware should:

1. Read I3C\_IBIR register to determine requesting device
2. Read related payload from IBI RX FIFO
3. Execute related action (application/system dependent), if any designed for particular IBI/device.

#### **12.1.4.4.9.2 Current Master Takeover In-Band Interrupt**

When mastership takeover IBI request appears on the I<sup>3</sup>C bus, I<sup>3</sup>C master hardware automatically sends the response based on settings in I3C\_SIR\_MAP0 through I3C\_SIR\_MAP5 registers. After this host is informed by the I3C\_MST\_ISR interrupt that handoff procedure should be executed. At this point, firmware should:

1. Read I3C\_IBIR register to determine requesting device (optionally, if such information is useful for application host)
2. Send the GETACCMST CCC to confirm mastership handover
3. Wait for the completion of pending I3C master command signaled by I3C\_MST\_ISR interrupt – at this moment I3C master switches to the slave mode
4. Write 0x001FXXXX to FLUSH\_CTRL register to flush the FIFOs as the slave mode reuses TX and RX FIFOs (bits marked X hold RX\_FIFO threshold so should be set/maintained as needed)
5. Depending on application/system, prepare for slave operation

Note that it is firmware and/or system responsibility to provide bus information to Secondary Masters before mastership handover is granted. It can be achieved in several ways:

- Secondary Masters listen to the bus during DAA and store necessary data
- Main Master sends DEFSLVS CCC after each ENTDA / RSTDA / SETNEWD / SETDASA sequence that modifies bus

#### **12.1.4.4.10 I3C Hot-Join Request**

In order to accept any Hot-Join (HJ) request from slave device, application needs to set the I3C\_CTRL[6] HJ\_ACK and I3C\_CTRL[8] HJ\_DISEC control bits that define its response for Hot-Join request. Based on I3C\_CTRL[6] HJ\_ACK value the I3C master accepts or refuses the Hot-Join request by providing respectively ACK or NACK bit right after I3C\_CMD0\_FIFO[0] RNW bit. I3C\_CTRL[8] HJ\_DISEC bit controls whether master starts DAA procedure or sends broadcast DISEC command to disable HJ requests.

If Hot-Join request occurs, the application processor is notified by the I3C\_MST\_ISR interrupt. The application processor is aware how this particular Hot-Join request should be handled by checking current setting (can read the control bits or the stored previously used setting). In case of acknowledged request with subsequent DAA, the application processor is notified of automatically completed enumeration of hot-joined device by subsequent I3C\_MST\_ISR interrupt and simultaneously set I3C\_MST\_STATUS0 flag. In case of DISEC command, only I3C\_MST\_ISR interrupt will be set.

It is recommended to keep the I3C\_CTRL[6] HJ\_ACK bit disabled until the bus initialization procedure is completed in order to prevent slaves from indirect DAA enforcement.

#### 12.1.4.4.11 I3C Immediate Commands

Immediate Command Register is implemented in order to allow the application processor to issue immediate command which has a higher priority over the ordinary command. The immediate commands are supposed to be used in exceptional cases and only CCC commands are applicable. The I3C supports only one immediate command at a time and it is accessed by the host through two APB registers: I3C\_IMD\_CMD0 and I3C\_IMD\_CMD1. If a normal command is currently being executed and new immediate command is loaded, then controller waits to finish the current command and after that starts executing the higher priority command regardless of the content of the ordinary command FIFO. Immediate command can be used only for CCC commands and the corresponding payload must not exceed 4 bytes. As a result the GETPID and DEVSLVS CCC commands cannot be executed using immediate commands. The format for the immediate command words is the same as for the ordinary commands, although some fields are reserved since not used or fixed for CCC commands (writing to these fields has no effect).

#### 12.1.4.4.12 I3C Host Commands

The normal operation of the controller allows the firmware to build a message and then issue the transaction to the I3C controller. The command queue is provided in order to allow the host to load a request for multiple messages. Single message consists of two 32-bit words (Command Word0 and Command Word1). In order to issue the transaction the following procedure should be followed:

- Write the payload to the TX FIFO (applicable for write transfers only);
- Write Command Word1;
- Write Command Word0;

If controller is enabled (I3C\_CTRL[31] DEV\_EN bit is set to 0x01), writing Command Word0 triggers the execution of the command, resulting in start condition on the I3C bus.

#### 12.1.4.4.13 I3C Sending Private Data in SDR Messages

The I3C controller supports private data transfers in Single Data Rate with SCL frequencies up to 10.375 MHz. It supports all types of SDR transfers and provides hardware built-in features for transfers to I3C slave devices with limited maximum SCL frequency. Before any private transfer to the slave devices occurs, bus mode and SCL prescalers shall be programmed properly. Controller also provides vendor extension to the basic transfers which simplifies addressing internal registers of slave devices. To take advantage of this extension, the I3C controller must communicate with the slave devices supporting such transmission scheme.

The extension, called XMIT modes, combines standard private write and read frames to pass the slave register sub-address in addition to the actual payload. Four transmit modes are designed for specific usage:

- XMIT00 – burst (byte-by-byte) transfer with static register sub-address. Sends static sub-address followed by data bytes only. Expecting slave will make self-increment of the address for each data byte or will have payload buffer implemented.
- XMIT01 – single transfer with incremented register sub-address. Each data byte followed by repeated start condition and the slave's internal register address are incremented explicitly.
- XMIT10 – single transfer with static register sub-address. Each data byte followed by repeated start condition, register sub-address remains intact.
- XMIT11 – burst (byte-by-byte) transfer without register sub-address (first data byte after the slave address is a payload byte).

In addition to the multiple transmit modes, I3C controller has an option to provide 16-bit address of the internal register in slave device to support slaves with more than 256 internal addresses.

For arbitration priorities management, private SDR commands may use Broadcast Command Header or not. This is controlled by BCH field of command descriptor.

##### 12.1.4.4.13.1 SDR Private Write Message

The Private Write message transmitted over I3C bus in all supported transmit modes is presented in [Figure 12-53, Master Write Transaction for Single and Multi Address Sequences](#). The provided transmit modes use additional I3C write frame that utilizes the data field to pass slave register address ahead of payload byte(s).

As writing command word triggers start condition immediately, it shall be guaranteed by firmware that the data payload is written to the Tx FIFO in advance before it is going to be transmitted.

The completion of the message frame on the I3C bus is signaled by the I3C\_MST\_ISR[16] IMM\_COMP interrupt.

In case of transfers with data payload greater than the TX FIFO size, firmware can use I3C\_MST\_ISR[15] TX\_THR interrupt to fill the FIFO on time. The system design should prevent the TX FIFO from underflow. During the time elapsed between reading the last four bytes from almost empty TX FIFO and reading the subsequent one, firmware is required to refill the TX FIFO.

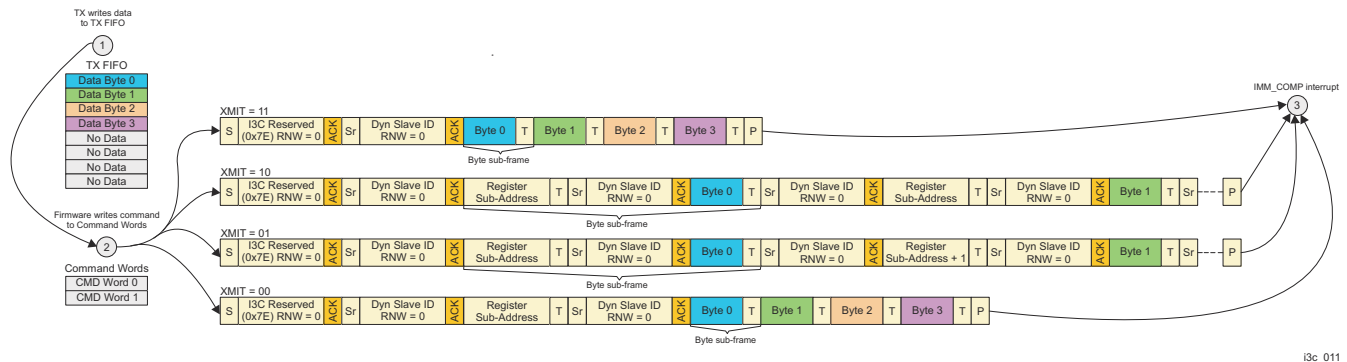


Figure 12-53. Master Write Transaction for Single and Multi Address Sequences

#### 12.1.4.4.13.2 SDR Private Read Message

The Private Read message transmitted over I3C bus in all supported transmit modes is presented in [Figure 12-54](#), *Master Read Transaction for Single and Multi Address Sequences*

Similarly to Private Write message, each I3C read frame is preceded by I3C write frame which sends slave register address to be read. After sending the command, firmware should wait for I3C\_MST\_ISR[16] IMM\_COMP interrupt which signals the end of message and RX FIFO content ready to read.

In case of payload lengths exceeding RX FIFO size, firmware should utilize I3C\_MST\_ISR[7] TX\_THR interrupt to read RX FIFO before it gets filled and prevent from data corruption. For easier management of long FIFO and/or long host read intervals, the amount of data available in FIFO that triggers the interrupt can be programmed according to actual conditions in given system.

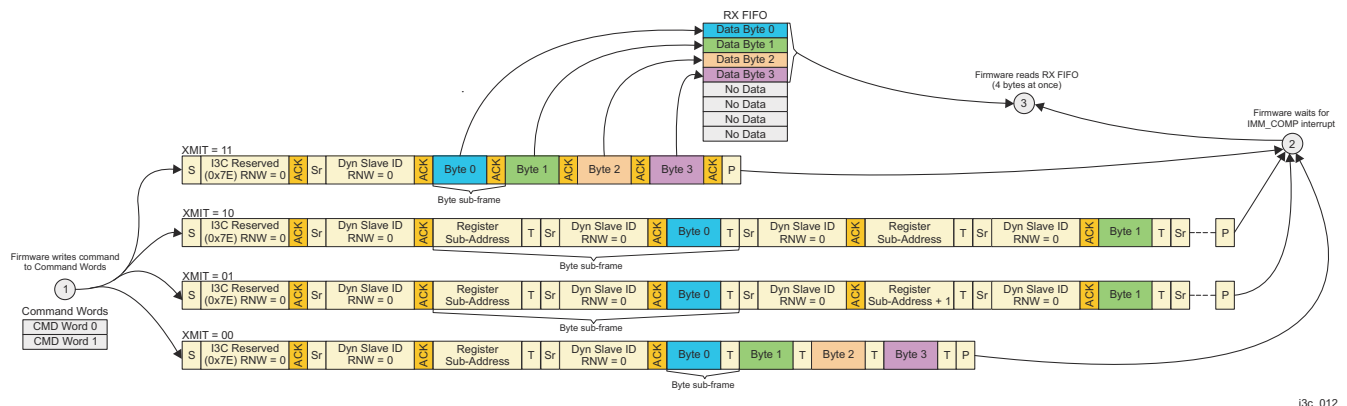


Figure 12-54. Master Read Transaction for Single and Multi Address Sequences

#### 12.1.4.4.13.3 SDR Payload Length Adjustment

In case of Private Write SDR commands, the payload length that needs to be specified is always equal to the number of data bytes sent to slave in Private Write transmission. If the payload specified is greater than slave's capability, the data are simply ignored by slave.

In case of Private Read SDR commands slave is the device which normally decides how many bytes it is able to send and when the read transfer is terminated. This can also result from hardware architecture or from SETMRL CCC command sent earlier to the slave.

For simplified handling I3C design triggers the read abort procedure after receiving specified number of payload bytes. This matters in these transfer modes which send data bytes after one another. If one of these modes is used, the I3C\_CMD0\_FIFO[23-12] PL\_LEN bit-field is interpreted as payload size that should not be reached during transfer rather than the actual number of bytes to be received. This applies only to read transfer from I3C devices.

**Table 12-56. Payload Setting in SDR Mode**

Transmission	I3C_CMD0_FIFO[28-27] XMIT_MODE	I3C_DEV_IDn_RR0[9] IS_I3C (n = 0 to 11)	I3C_CMD0_FIFO[23-12] PL_LEN
I3C, NCA	11	1	Abort limit <sup>(1)</sup>
I3C, Single	00	1	Abort limit <sup>(1)</sup>
I3C, Multi-Static	10	1	Number of bytes to receive
I3C, Multi-Incremental	01	1	Number of bytes to receive

- (1) If payload size is known, it can be set to one more than the known size; otherwise it should be set to MRL+1 where MRL is value programmed with SETMRL CCC in particular slave or its hardware payload size limit, if known.

### 12.1.4.5 I3C Programming Guide

#### 12.1.4.5.1 I3C Power-On Programming Model

Typical power-on programming sequence is as follows:

1. Program bus description into Retaining Registers according [Section 12.1.4.4.6](#) and I3C\_DEVS\_CTRL.
2. If default values of SCL prescalers do not fit applied clock frequencies, program PRESCL\_CTRL0 and PRESCL\_CTRL1 according [Section 12.1.4.4.2](#).
3. If any of following applies:
  - a. Address Header Optimization is to be enabled.
  - b. Halt mechanism should be enabled.

then alter values of I3C\_CTRL[1-0] BUS\_MODE, I3C\_CTRL[3] AHDR\_OPT and I3C\_CTRL[30] HALT\_EN fields, respectively:

- a. Program I3C\_CTRL register with value 0x80000000 to enable the controller.

#### 12.1.4.5.2 I3C Static Devices Programming

I3C devices shipped with Static Address can be enumerated prior to DAA procedure using SETDASA CCC command. In order to enumerate I3C device firmware needs to follow procedure:

1. Prepare device's retaining register:
  - a. Write corresponding I3C\_DEV\_IDn\_RR0 (offset 0x080 + (n × 0x010)) for n = 0 to 11:
    - i. I3C\_DEV\_IDn\_RR0[9] IS\_I3C (n = 0 to 11): 0x01
    - ii. I3C\_DEV\_IDn\_RR0[7-0] DEV\_ADDR bit-field (n = 0 to 11): set to device's Static Address for [7-1] bits and set to Static Address parity for bit 0
2. Issue SETDASA CCC:
  - a. Make sure that controller is enabled.
  - b. Write new Dynamic Address to TX FIFO.
  - c. Write Command Word1 with SETDASA CCC value:
    - i. I3C\_CMD1[7-0] CCC bit-field: 0x87
  - d. Write Command Word0 with following field values:
    - i. I3C\_CMD0\_FIFO[30] IS\_CCC bit: 0x01
    - ii. I3C\_CMD0\_FIFO[23-12] PL\_LEN bit-field: 0x01
    - iii. I3C\_CMD0\_FIFO[7-1] DEV\_ADDR bit-field: Static Address
    - iv. I3C\_CMD0\_FIFO[0] RNW bit: 0x00
3. Enable at least COMP, NACK and INVALID\_DA interrupts by writing 0h to IER register.
4. Wait for COMP interrupt (read I3C\_MST\_ISR).
5. If COMP and no other interrupt occurs, continue. Otherwise, handle the error situation.
6. Fill Dynamic Address into device configuration register:
  - a. Modify I3C\_DEV\_IDn\_RR0 (offset 0x080 + (n × 0x010)) for n = 0 to 11 register by writing Dynamic Address to DEV\_ADDR field and its parity to ADDR\_PAR field (keep values of other fields)
7. If not already known to application host, retrieve BCR with GETBCR CCC:
  - a. Write Command Word1 with GETBCR CCC value:
    - i. I3C\_CMD1[7-0] CCC bit-field: 0x8E
  - b. Write Command Word0 with following field values:
    - i. I3C\_CMD0\_FIFO[30] IS\_CCC bit: 0x01
    - ii. I3C\_CMD0\_FIFO[23-12] PL\_LEN bit-field: 0x01
    - iii. I3C\_CMD0\_FIFO[7-1] DEV\_ADDR bit-field: Dynamic Address
    - iv. I3C\_CMD0\_FIFO[0] RNW bit: 0x01
  - c. Wait for COMP interrupt
  - d. Read RX FIFO register - BCR is stored in bits [7:0]
8. Optionally retrieve Provisional ID and DCR with corresponding CCCs:
  - a. Write Command Word1 with GETDCR CCC value:

- i. I3C\_IMD\_CMD1[7-0] CCC bit-field: 0x8F
- b. Write Command Word0 with following field values:
  - i. I3C\_CMD0\_FIFO[30] IS\_CCC bit: 0x01
  - ii. I3C\_CMD0\_FIFO[23-12] PL\_LEN bit-field: 0x01
  - iii. I3C\_CMD0\_FIFO[7-1] DEV\_ADDR bit-field: Dynamic Address
  - iv. I3C\_CMD0\_FIFO[0] RNW bit: 0x01
- c. Wait for COMP interrupt
- d. Read RX FIFO register - DCR is stored in bits [7:0]
- e. Write Command Word1 with GETPID CCC value:
  - i. I3C\_IMD\_CMD1[7-0] CCC bit-field: 0x8D
- f. Write Command Word0 with following field values:
  - i. I3C\_CMD0\_FIFO[30] IS\_CCC bit: 0x01
  - ii. I3C\_CMD0\_FIFO[23-12] PL\_LEN bit-field: 0x06
  - iii. I3C\_CMD0\_FIFO[7-1] DEV\_ADDR bit-field: Dynamic Address
  - iv. I3C\_CMD0\_FIFO[0] RNW bit: 0x01
- g. Wait for COMP interrupt
- h. Read RX FIFO register twice – PID bytes are stored in two FIFO locations. See [Table 12-54](#)
9. Write retrieved device configuration data to retaining registers:
  - a. Write four MSB of obtained PID value into I3C\_DEV\_IDn\_RR1 (offset 0x084 + (n × 0x010)) for n = 0 to 11:
  - b. Write two LSB of obtained PID value as well as BCR and DCR values into I3C\_DEV\_IDn\_RR2 (offset 0x088 + (n × 0x010)) for n = 0 to 11:

#### Note

SETDASA CCC command has advantage over DAA in shorter time spent on device configuration as retrieving PID and DCR (6 + 1 bytes transmitted over I3C bus) is optional and can be skipped if not used.

#### 12.1.4.5.3 I3C DAA Procedure Initiation

To initiate the DAA procedure I3C needs to issue ENTDAACCC command following the below procedure:

1. Enable at least COMP and NACK interrupts.
2. Issue ENTDAACCC command:
  - a. Make sure controller is enabled.
  - b. Write Command Word1 with ENTDAACCC command value:
    - i. I3C\_IMD\_CMD1[7-0] CCC bit-field: 0x07
  - c. Write Command Word0 with following field values:
    - i. I3C\_CMD0\_FIFO[30] IS\_CCC bit: 0x01
    - ii. I3C\_CMD0\_FIFO[23-12] PL\_LEN bit-field: 0x00
    - iii. I3C\_CMD0\_FIFO[7-1] DEV\_ADDR bit-field: 0x00
    - iv. I3C\_CMD0\_FIFO[0] RNW bit: 0x00
3. Wait for COMP interrupt (read I3C\_MST\_ISR)
4. If COMP and no other interrupt occurs, continue. Otherwise, handle the error situation. NACK interrupt indicates no slave devices available on bus.
5. Read I3C\_MST\_STATUS0[25] DAA\_COMP bit. It is asserted simultaneously with I3C\_MST\_ISR interrupt flag.

#### Note

DAA procedure takes significant amount of time since there are always 9 bytes of data transmitted for each device (PID, BCR, DCR from device and DA to device), so for maximum bus size it will take 99 bytes.



#### 12.1.4.5.4 I3C SDR Write Message Programming Model

To accomplish the SDR write transfer, the host must perform the following steps:

1. Write all payload data bytes into TX FIFO buffer:
  - a. Write first four or less bytes to be sent, with first byte stored at LSB position.
  - b. Repeat writes until whole payload is written.
2. Write Private Write command to command queue:
  - a. Write the Command Word 1 (I3C\_CMD1\_FIFO) with slave CSR address:
    - i. CSR (bits [15:0] or [7:0]): Slave register address
  - b. Write the Command Word0 (I3C\_CMD0\_FIFO) with following data fields:
    - i. I3C\_CMD0\_FIFO[31] IS\_DDR: 0x0
    - ii. I3C\_CMD0\_FIFO[30] IS\_CCC: 0x0
    - iii. I3C\_CMD0\_FIFO[28-27] XMIT\_MODE: Transmit Mode
    - iv. I3C\_CMD0\_FIFO[26] SB\_CA: 0x1 for 16-bit CSR address, 0x0 otherwise
    - v. I3C\_CMD0\_FIFO[23-12] PL\_LEN: Number of bytes to send
    - vi. I3C\_CMD0\_FIFO[7-1] DEV\_ADDR: Slave Address
    - vii. I3C\_CMD0\_FIFO[0] RNW: 0x0
3. Wait for transfer completion notified by I3C\_MST\_ISR[16] IMM\_COMP bit, which indicates that all data of particular command are successfully transmitted to the slave.

#### 12.1.4.5.5 I3C SDR Read Message Programming Model

To accomplish the SDR read transfer, the host must perform the following steps:

1. Write Private Write command to command queue:
  - a. Write the Command Word1 (I3C\_CMD1\_FIFO) with slave CSR address:
    - i. CSR (bits [15:0] or [7:0]): Slave register address
  - b. Write the Command Word0 (I3C\_CMD0\_FIFO) with following data fields:
    - i. I3C\_CMD0\_FIFO[31] IS\_DDR: 0x0
    - ii. I3C\_CMD0\_FIFO[30] IS\_CCC: 0x0
    - iii. I3C\_CMD0\_FIFO[28-27] XMIT\_MODE: Transmit Mode
    - iv. I3C\_CMD0\_FIFO[23-12] PL\_LEN: See [Section 12.1.4.4.13.3](#)
    - v. I3C\_CMD0\_FIFO[7-1] DEV\_ADDR: Slave Address
    - vi. I3C\_CMD0\_FIFO[0] RNW: 0x1
2. Wait for transfer completion notified by I3C\_MST\_ISR[16] IMM\_COMP bit, which indicates that all data of particular command are successfully sent by the slave device and received data are ready to be read by the host.
3. Read all payload data bytes from RX FIFO buffer:
  - a. Read first four or less received bytes, with first received byte stored at LSB position.
  - b. Repeat reads until whole payload is read.

#### 12.1.4.5.6 I3C DDR Write Message Programming Model

To accomplish the DDR write transfer, the host must perform the following steps:

1. Prepare HDR-DDR message words (each 20-bit word is LSB aligned in TX FIFO cell; set bits unspecified below to 0):
  - a. Write command word for write HDR-DDR command to TX FIFO using following field values:
    - i. Preamble (bits [19:18]): 0x01
    - ii. CMD (bits[17:10]): Write Command Code (from 0x00 to 0x7F)
    - iii. DA (bits[9:3]): Slave Dynamic Address
    - iv. Parity (bits[1:0]): Parity (XOR of odd and even bits of 16-bit payload)
  - b. Write first data word to TX FIFO using following field values:
    - i. Preamble (bits [19:18]): 0x10
    - ii. Data (bits[17:2]): 16-bit data

- iii. Parity (bits[1:0]): Parity
  - c. Write zero or more subsequent data words to TX FIFO using following field values:
    - i. Preamble (bits [19:18]): 0x11
    - ii. Data (bits[17:2]): 16-bit data
    - iii. Parity (bits[1:0]): Parity
  - d. Write CRC word to TX FIFO:
    - i. Preamble (bits [19:18]): 0x01
    - ii. Token (bits[17:14]): Ch
    - iii. CRC (bits[13:9]): CRC5
  - e. Repeat steps 1(a) to 1(d) as many write commands are to be sent.
2. Write ENTHDR CCC command to command queue:
  - a. Write the Command Word1 (I3C\_CMD1\_FIFO) of ENTHDR CCC.
  - b. Write the Command Word0 (I3C\_CMD0\_FIFO) of ENTHDR CCC with following data fields:
    - i. I3C\_CMD0\_FIFO[31] IS\_DDR: 0x0
    - ii. I3C\_CMD0\_FIFO[30] IS\_CCC: 0x1
    - iii. I3C\_CMD0\_FIFO[0] RNW: 0x0
3. Write HDR-DDR command(s) to command queue:
  - a. Write the Command Word1 (I3C\_CMD1\_FIFO).
  - b. Write the Command Word0 (I3C\_CMD0\_FIFO) with following data fields:
    - i. I3C\_CMD0\_FIFO[31] IS\_DDR: 0x1
    - ii. I3C\_CMD0\_FIFO[23-12] PL\_LEN: Number of TX FIFO words
  - c. Repeat steps 2(a) and 2(b) as many write commands are prepared in TX FIFO in step 1.
4. Wait for transfer completion notified by I3C\_MST\_ISR[16] IMM\_COMP bit, which indicates that all data of particular command are successfully transmitted to the slave.

If I3C\_MST\_ISR[16] IMM\_COMP interrupt is utilized, the controller allows the application processor to perform other tasks or sleep while message is being sent.

In case of multiple command issued to command queue, I3C\_MST\_ISR[5] CMDD\_EMP interrupt may be utilized instead. It indicates completion of all commands in command queue and allows yet more efficient application processor resources management. It is issued simultaneously with the I3C\_MST\_ISR[5] CMDD\_EMP interrupt corresponding to the last command in the queue.

#### 12.1.4.5.7 I3C DDR Read Message Programming Model

To accomplish the DDR read transfer, the host must perform the following steps:

1. Prepare HDR-DDR message words (each 20-bit word is LSB-aligned in TX FIFO cell, set bits unspecified below to 0):
  - a. Write Command Word for Read HDR-DDR command to TX FIFO using following field values:
    - i. Preamble (bits [19:18]): 2'b01
    - ii. CMD (bits[17:10]): Read Command Code (0x80..0xBF)
    - iii. DA (bits[9:3]): Slave Dynamic Address
    - iv. Parity (bits[1:0]): Parity (XOR of odd and even bits of 16-bit payload)
  - b. Repeat step 1(a) as many read commands are to be sent.
2. Write ENTHDR CCC command to command queue:
  - a. Write the Command Word1 (I3C\_CMD1\_FIFO) of ENTHDR CCC.
  - b. Write the Command Word0 (I3C\_CMD0\_FIFO) of ENTHDR CCC with following data fields:
    - i. I3C\_CMD0\_FIFO[31] IS\_DDR: 0x0
    - ii. I3C\_CMD0\_FIFO[30] IS\_CCC: 0x1
    - iii. I3C\_CMD0\_FIFO[0] RNW: 0x0
3. Write HDR-DDR command(s) to command queue:
  - a. Write the Command Word1 (I3C\_CMD1\_FIFO)
  - b. Write the Command Word0 (I3C\_CMD0\_FIFO) with following data fields:



- i. I3C\_CMD0\_FIFO[31] IS\_DDR: 0x1
  - ii. I3C\_CMD0\_FIFO[23-12] PL\_LEN: 0x1
- c. Repeat steps 2(a) and 2(b) as many read commands are prepared in TX FIFO in step 1.
4. Wait for transfer completion notified by I3C\_MST\_ISR[16] IMM\_COMP bit in interrupt status register, which indicates that all data of particular command are successfully sent by the slave device and received data are ready to be read by the host.
5. Read received payload:
  - a. Read first data word from RX FIFO extracting field values:
    - i. Preamble (bits [19:18]): 0x10
    - ii. Data (bits[17:2]): 16-bit data
    - iii. Parity (bits[1:0]): Parity
  - b. Read zero or more subsequent data words from RX FIFO extracting field values as long as the read preamble matches following value:
    - i. Preamble (bits [19:18]): 0x11
    - ii. Data (bits[17:2]): 16-bit data
    - iii. Parity (bits[1:0]): Parity
  - c. As read preamble changes, read CRC Word:
    - i. Preamble (bits [19:18]): 0x01
    - ii. Token (bits[17:14]): Ch
    - iii. CRC (bits[13:9]): CRC5
  - d. Confirm (by firmware) that CRC5 value matches data payload received.

If I3C\_MST\_ISR[16] IMM\_COMP interrupt is utilized, the controller allows the application processor to perform other tasks or sleep while message is being sent.

In case of multiple command issued to command queue, I3C\_MST\_ISR[5] CMDD\_EMP interrupt may be utilized instead of I3C\_MST\_ISR[16] IMM\_COMP. It indicates completion of all commands in command queue and allows yet more efficient application processor resources management. It is issued simultaneously with the I3C\_MST\_ISR[16] IMM\_COMP interrupt corresponding to the last command in the queue.

## 12.1.5 Multichannel Serial Peripheral Interface (MCSPi)

This section describes the Multichannel Serial Peripheral Interface (MCSPi) modules for the device.

### 12.1.5.1 MCSPi Overview

The MCSPi module is a multichannel transmit/receive, master/slave synchronous serial bus.

There are eleven MCSPi modules in the device. [Table 12-57](#) shows the MCSPi allocation across device domains.

**Table 12-57. MCSPi Allocation Across Device Domains**

Instance	Domain		
	WKUP	MCU	MAIN
MCU_MCSPi0	-	✓	-
MCU_MCSPi1	-	✓	-
MCU_MCSPi2	-	✓	-
MCSPi0	-	-	✓
MCSPi1	-	-	✓
MCSPi2	-	-	✓
MCSPi3	-	-	✓
MCSPi4	-	-	✓
MCSPi5	-	-	✓
MCSPi6	-	-	✓
MCSPi7	-	-	✓

MCSPi3 and MCSPi4 include internal connectivity to MCSPi modules in the MCU domain, as follows:

- MCSPi3 is connected as a master to MCU\_MCSPi1 by default at power-up. MCU\_MCSPi1 and MCSPi3 may be optionally mapped to external device pads.
- MCSPi4 is directly connected as a slave to MCU\_MCSPi2 by default at power-up. MCSPi4 and MCU\_MCSPi2 are not pinned out externally.

[Figure 12-55](#) shows the MCSPi overview.

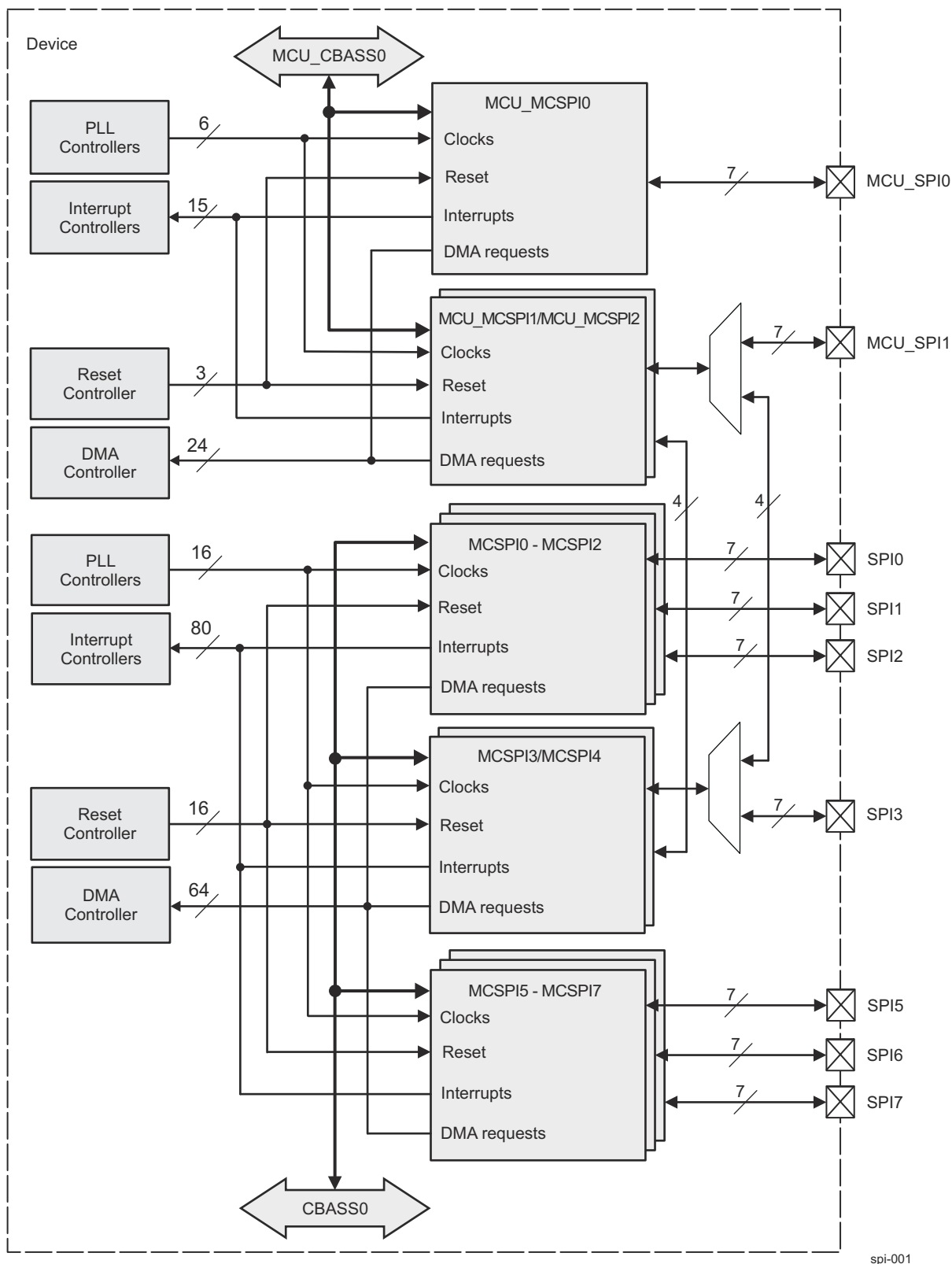


Figure 12-55. MCSPI Overview

#### **12.1.5.1.1 SPI Features**

The SPI module includes the following main features:

- Serial clock with programmable frequency, polarity, and phase for each channel
- Wide selection of SPI word lengths, ranging from 4 to 32 bits
- Up to channels in controller mode, or single channel in receive mode
- Controller multichannel mode:
  - Full duplex/half duplex
  - Transmit-only/receive-only/transmit-and-receive modes
  - Flexible input/output (I/O) port controls per channel
  - Programmable clock granularity
  - Per channel configuration for clock definition, polarity enabling, and word width
- Single interrupt line for multiple interrupt source events
- Enable the addition of a programmable start-bit for MCSPI transfer per channel (start-bit mode)
- Supports start-bit write command
- Programmable timing control between chip select and external clock generation
- Built-in FIFO available for a single channel

#### **12.1.5.1.2 MCSPI Not Supported Features**

The following features are not supported on this family of devices:

- Slave mode wake-up
- Retention during power down
- MCU\_MCSPI2 and MCSPI4 are not pinned out
- MCSPI4 master mode is not supported
- In slave mode only channel 0 is used
- Local power management of clock activity.

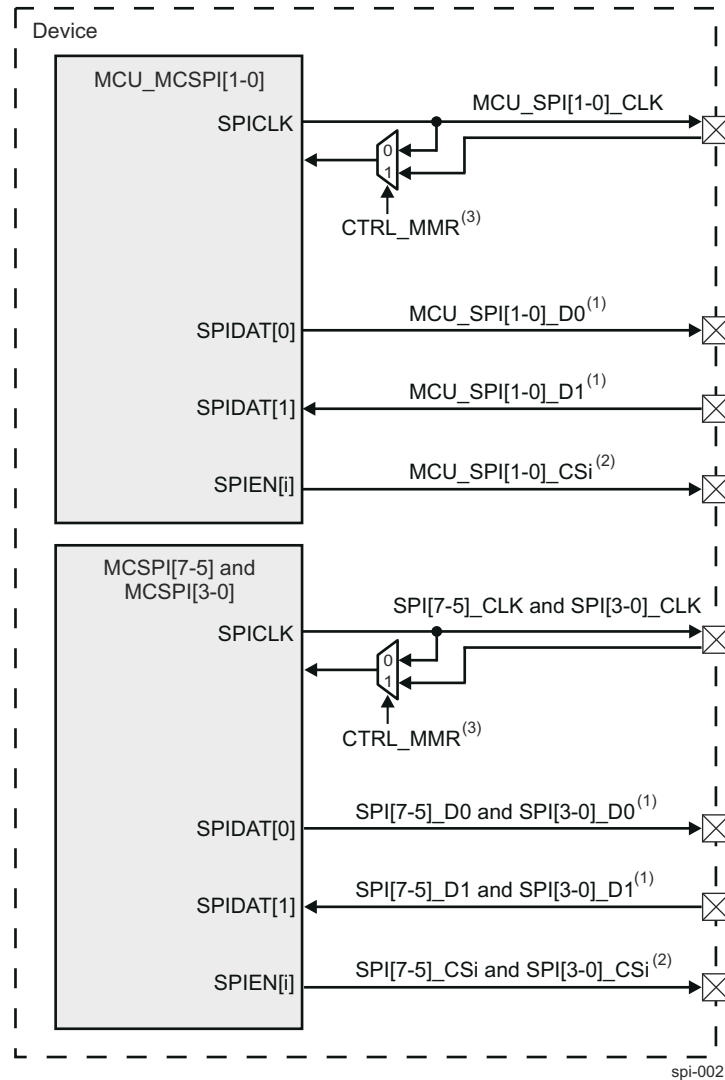
### 12.1.5.2 MCSPI Environment

The MCU\_MCSPI[2-0] and MCSPI[7-0] modules are hereinafter referred to as MCSPI module.

This section describes the MCSPI external connections (environment).

#### 12.1.5.2.1 Basic MCSPI Pins for Master Mode

Figure 12-56 shows all of the MCSPI interface signals in master mode.



- A. Direction depends on MCSPI\_CHCONF\_0/1/2/3[16] DPE0, MCSPI\_CHCONF\_0/1/2/3[17] DPE1 and MCSPI\_CHCONF\_0/1/2/3[18] IS bits
- B.  $i = 0$  to 3
- C. The source of the loopback clock is defined by CTRLMMR\_WKUP\_MCU\_SPI0\_CLKSELCTRLMMR\_WKUP\_MCU\_SPI0\_CLKSEL[16] MSTR\_LB\_CLKSEL bit (and the respective register for MCU\_MCSPI1) and CTRLMMR\_SPI0\_CLKSELCTRLMMR\_SPI0\_CLKSEL[16] MSTR\_LB\_CLKSEL bit (and the respective for MCSPI[3-0] and MCSPI[7-5]) in *Control Module (CTRL\_MMR)*.
- D. MCU\_MCSPI2 and MCSPI4 are not pinned out

**Figure 12-56. MCSPI Interface Signals in Master Mode<sup>(4)</sup>**

Table 12-58 describes the MCSPI I/O signals in master mode.

**Table 12-58. MCSPI I/O Signals (Master Mode)**

Module Pin	Device Level Signal	I/O <sup>(1)</sup>	Description	Module Pin Reset Value <sup>(2)</sup>
<b>MCU_MCSPI[1-0]</b>				
SPICLK	MCU_SPI[1-0]_CLK	O	MCSPI Serial clock output for master mode.	HiZ
SPIDAT[0]	MCU_SPI[1-0]_D0	O <sup>(3)</sup>	MCSPI Data I/O for master mode.	HiZ
SPIDAT[1]	MCU_SPI[1-0]_D1	I <sup>(4)</sup>	MCSPI Data I/O for master mode.	HiZ
SPIEN[i]	MCU_SPI[1-0]_CSi	O	MCSPI Chip-select i output for master mode	HiZ
<b>MCSPi[7-5] and MCSPi[3-0]</b>				
SPICLK	SPI[7-5]_CLK and SPI[3-0]_CLK	O	MCSPI Serial clock output for master mode.	HiZ
SPIDAT[0]	SPI[7-5]_D0 and SPI[3-0]_D0	O <sup>(3)</sup>	MCSPI Data I/O for master mode.	HiZ
SPIDAT[1]	SPI[7-5]_D1 and SPI[3-0]_D1	I <sup>(4)</sup>	MCSPI Data I/O for master mode.	HiZ
SPIEN[i]	SPI[7-5]_CSi and SPI[3-0]_CSi	O	MCSPI Chip-select i output for master mode	HiZ

(1) I = Input; O = Output

(2) HiZ = High Impedance

(3) Example configuration only. Can be configured either as input or as output depending on MCSPI\_CHCONF\_0/1/2/3[18] IS and MCSPI\_CHCONF\_0/1/2/3[16] DPE0.

(4) Example configuration only. Can be configured either as input or as output depending on MCSPI\_CHCONF\_0/1/2/3[18] IS and MCSPI\_CHCONF\_0/1/2/3[17] DPE1.

#### Note

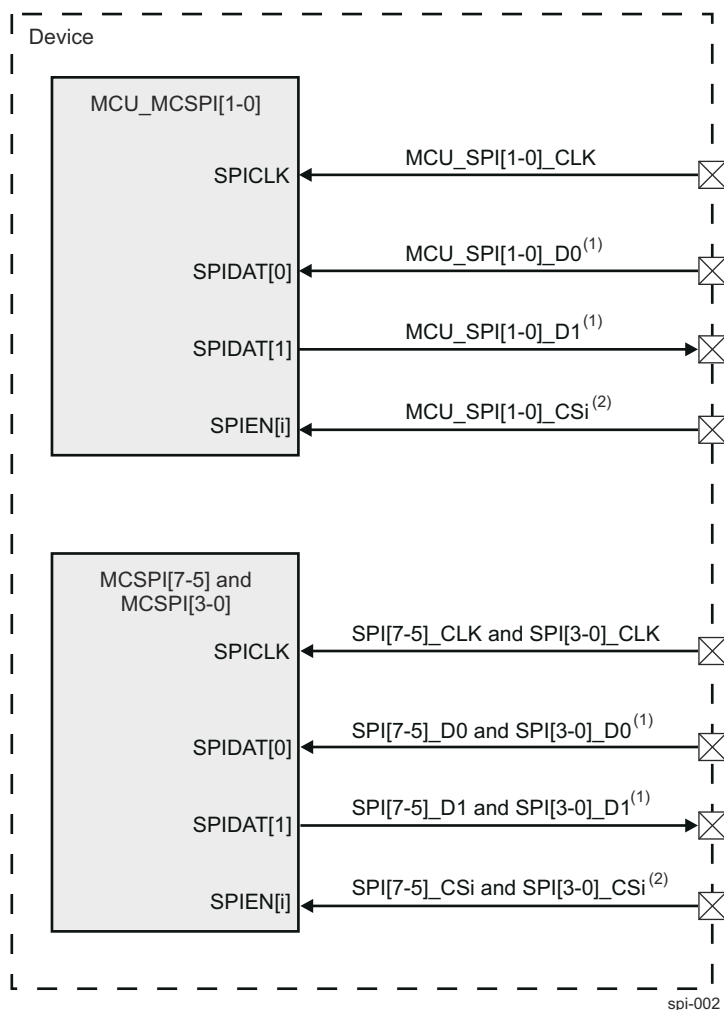
For SPI[7-5]\_CLK, SPI[3-0]\_CLK, and MCU\_SPI[1-0]\_CLK signals to work properly, the RXACTIVE bit of the appropriate CTRLMMR\_WKUP\_PADCONFIGx/ CTRLMMR\_PADCONFIGy registers should be set to 0x1 because of retiming purposes.

#### Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

#### 12.1.5.2.2 Basic MCSPI Pins for Slave Mode

Figure 12-57 shows all of the MCSPI interface signals in slave mode.



- A. Direction depends on MCSPI\_CHCONF\_0/1/2/3[16] DPE0, MCSPI\_CHCONF\_0/1/2/3[17] DPE1 and MCSPI\_CHCONF\_0/1/2/3[18] IS bits
- B.  $i = 0$  to 3
- C. MCU\_MCSPI2 and MCSPI4 are not pinned out

**Figure 12-57. MCSPI Interface Signals in Slave Mode<sup>(3)</sup>**

Table 12-59 describes the MCSPI I/O signals in slave mode.

**Table 12-59. MCSPI I/O Signals (Slave Mode)**

Module Pin	Device Level Signal	I/O <sup>(1)</sup>	Description	Module Pin Reset Value <sup>(1)</sup>
<b>MCU_MCSPI[1-0]</b>				
SPICLK	MCU_SPI[1-0]_CLK	I	MCSPI serial clock input for slave mode.	HiZ
SPIDAT[0]	MCU_SPI[1-0]_D0	I <sup>(2)</sup>	MCSPI Data I/O for slave mode.	HiZ
SPIDAT[1]	MCU_SPI[1-0]_D1	O <sup>(3)</sup>	MCSPI Data I/O for slave mode.	HiZ
SPIEN[i]	MCU_SPI[1-0]_CSi	I <sup>(4)</sup>	MCSPI chip-select i input for slave mode.	HiZ
<b>MCSPI[7-5] and MCSPI[3-0]</b>				
SPICLK	SPI[7-5]_CLK and SPI[3-0]_CLK	I	MCSPI serial clock input for slave mode.	HiZ
SPIDAT[0]	SPI[7-5]_D0 and SPI[3-0]_D0	I <sup>(2)</sup>	MCSPI Data I/O for slave mode.	HiZ

**Table 12-59. MCSPI I/O Signals (Slave Mode) (continued)**

Module Pin	Device Level Signal	I/O <sup>(1)</sup>	Description	Module Pin Reset Value <sup>(1)</sup>
SPIDAT[1]	SPI[7-5]_D1 and SPI[3-0]_D1	O <sup>(3)</sup>	MCSPI Data I/O for slave mode.	HiZ
SPIEN[i]	SPI[7-5]_CSi and SPI[3-0]_CSi	I <sup>(4)</sup>	MCSPI chip-select i input for slave mode.	HiZ

(1) HiZ = High Impedance

(2) Example configuration only. Can be configured either as input or as output depending on MCSPI\_CHCONF\_0/1/2/3[18] IS and MCSPI\_CHCONF\_0/1/2/3[16] DPE0.

(3) Example configuration only. Can be configured either as input or as output depending on MCSPI\_CHCONF\_0/1/2/3[18] IS and MCSPI\_CHCONF\_0/1/2/3[17] DPE1.

(4) The chip-select input in slave mode can be selected through the MCSPI\_CHCONF\_0/1/2/3[22-21] SPIENSLV bit field.

#### Note

For SPI[7-5]\_CLK, SPI[3-0]\_CLK, and MCU\_SPI[1-0]\_CLK signals to work properly, the RXACTIVE bit of the appropriate CTRLMMR\_WKUP\_PADCONFIGx/ CTRLMMR\_PADCONFIGy registers should be set to 0x1 because of retiming purposes.

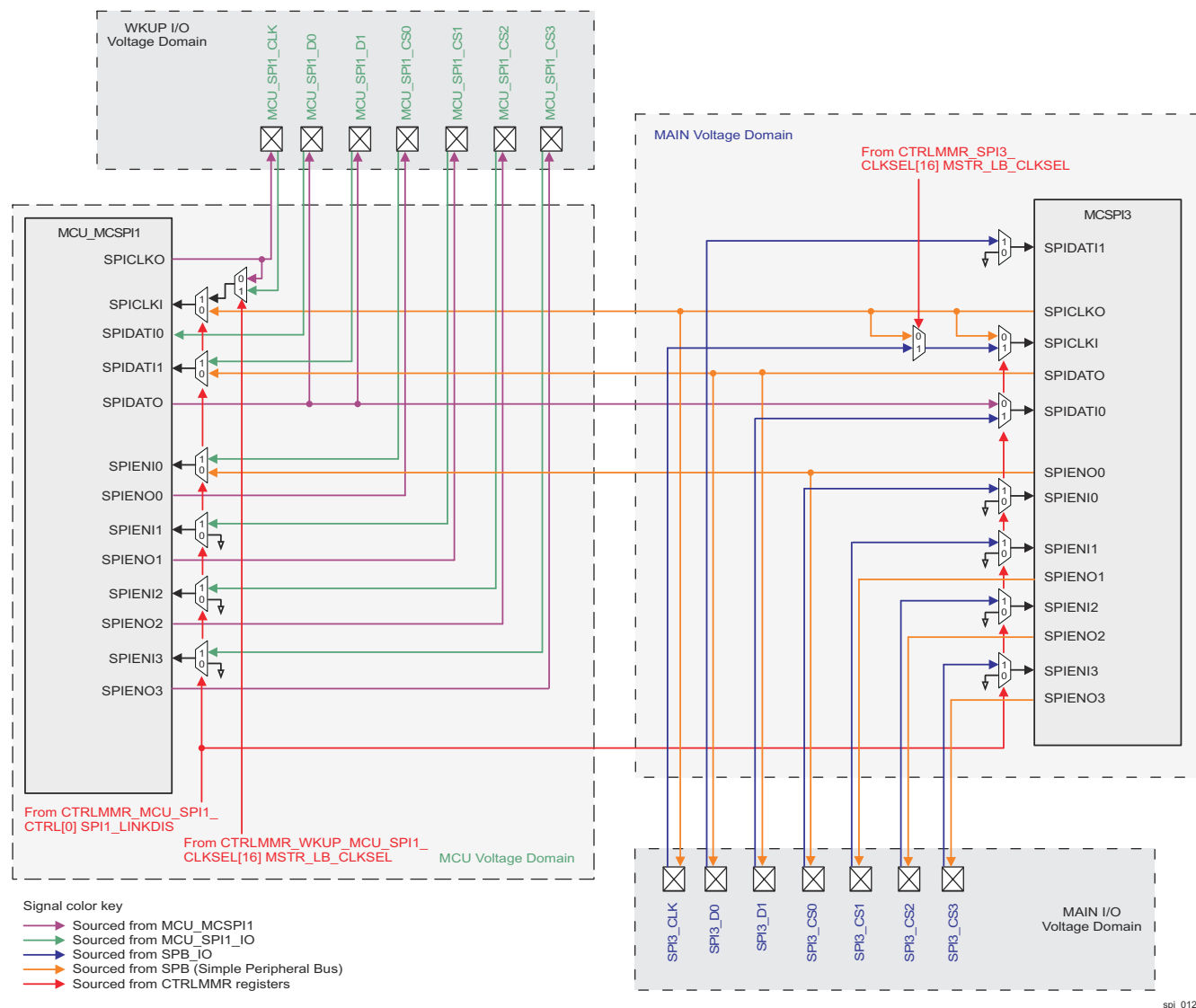
#### Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

#### 12.1.5.2.3 MCSPI Internal Connectivity

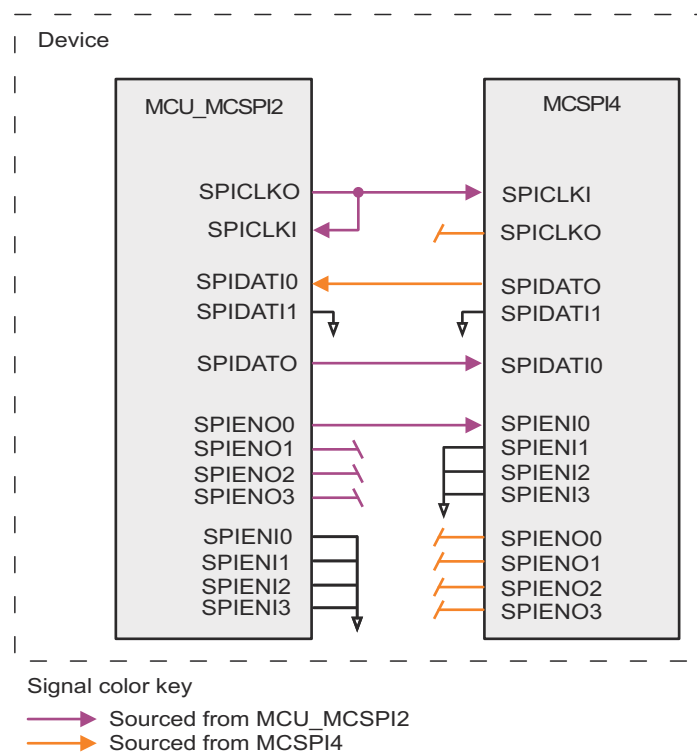
Figure 12-58 shows the MCSPI3 connected as a master to MCU\_MCSPI1 internally in the device. These MCSPIs also support external connectivity when configured in CTRL\_MMR.





**Figure 12-58. MCSPi3 and MCU\_MCSPi1 Connectivity Details**

Figure 12-59 shows the MCSPi4 connected as a slave to MCU\_MCSPi2 internally in the device. MCSPi4 and MCU\_MCSPi2 are not pinned out externally.



spi\_0122

**Figure 12-59. MCU\_MCSPi2 to MCSPI4 Internal Connectivity**

#### 12.1.5.2.4 MCSPI Protocol and Data Format

The synchronous MCSPI protocol allows a controller device to initiate serial data transfers to a peripheral device. A peripheral select line (SPIEN[i]) allows selection of an individual peripheral MCSPI device. Peripheral devices that are not selected do not interfere with MCSPI bus activities.

MCSPI offers the flexibility to modify the following parameters to adapt to the device features:

- Word length

MCSPI supports any MCSPI word ranging from 4 bits to 32 bits long (the MCSPI\_CHCONF\_0/1/2/3[11-7] WL bit field).

MCSPI word length can be changed between transmissions to allow the controller device to communicate with peripheral peripherals that have different requirements.

- MCSPI enable (SPIEN[i], for channel i)

The polarity of the MCSPI enable signals is programmable (the MCSPI\_CHCONF\_0/1/2/3[6] EPOL bit). SPIEN[i] signals can be active high or low.

Assertion of the SPIEN[i] signals is programmable and can be done manually or automatically. The manual assertion mode is available in single controller mode only. SPIEN[i] can be kept active between words with the MCSPI\_CHCONF\_0/1/2/3[20] FORCE bit.

Two consecutive words for two different peripheral devices can go along with active SPIEN[i] signals with different polarity.

- Programmable start-bit

In start-bit mode a start-bit is added before the MCSPI word length to indicate how the next MCSPI word must be handled. The start-bit is enabled by setting the MCSPI\_CHCONF\_0/1/2/3[23] SBE bit to 1. The MCSPI\_CHCONF\_0/1/2/3[24] SBPOL bit defines the polarity of the start-bit.

- Programmable MCSPI clock

- Bit rate

In controller mode, the baud rate of the MCSPI serial clock is programmable using the 50-MHz reference clock (from the device clock management module). [Table 12-60](#) lists the SPICLK bit rates obtained for data transfer when programming the clock divider (the MCSPI\_CHCONF\_0/1/2/3[5-2] CLKD bit field). This is valid when MCSPI\_CHCONF\_0/1/2/3[29] CLKD bit field is 0.

**Table 12-60. MCSPI Controller Clock Rates**

Divider	Clock Rate
1	50 MHz <sup>(1)</sup>
2	25 MHz <sup>(1)</sup>
4	12.5 MHz
8	6.25 MHz
16	3.125 MHz
32	1.5625 MHz
64	781.25 kHz
128	390.625 kHz
256	~195 kHz
512	~97.7 kHz
1024	~48.8 kHz
2048	~24.4 kHz
4096	~12.2 kHz

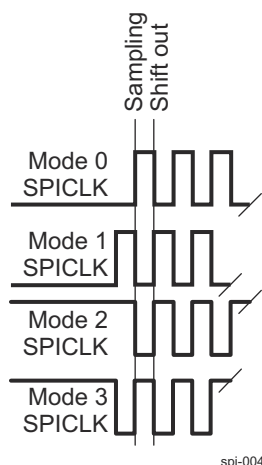
(1) These frequencies are not necessarily supported by all MCSPI modules. For more information, see the *Timing Requirements and Switching Characteristics* chapter in the device-specific Datasheet.

– Polarity and phase

The polarity (the MCSPI\_CHCONF\_0/1/2/3[1] POL bit) and the phase (the MCSPI\_CHCONF\_0/1/2/3[0] PHA bit) of the MCSPI serial clock (SPICLK) are configurable to offer four combinations. Software selects the right combination, depending on the device. See [Table 12-61](#) and [Figure 12-60](#).

**Table 12-61. Phase and Polarity Combinations**

Polarity (POL)	Phase (PHA)	MCSPI Mode	Description
0	0	Mode 0	SPICLK is inactive low and sampling occurs at the rising edge.
0	1	Mode 1	SPICLK is inactive low and sampling occurs at the falling edge.
1	0	Mode 2	SPICLK is inactive high and sampling occurs at the falling edge.
1	1	Mode 3	SPICLK is inactive high and sampling occurs at the rising edge.



**Figure 12-60. Phase and Polarity Combinations**

#### 12.1.5.2.4.1 Transfer Format

In controller and peripheral modes, the MCSPI drives the data lines when SPIEN[i] is asserted.

Each word is transmitted starting with the most-significant bit (MSB).

This section explains the two cases of data transmission determined by the clock phase (PHA) and the type of data transmission using a start-bit (SBE) called the start-bit mode:

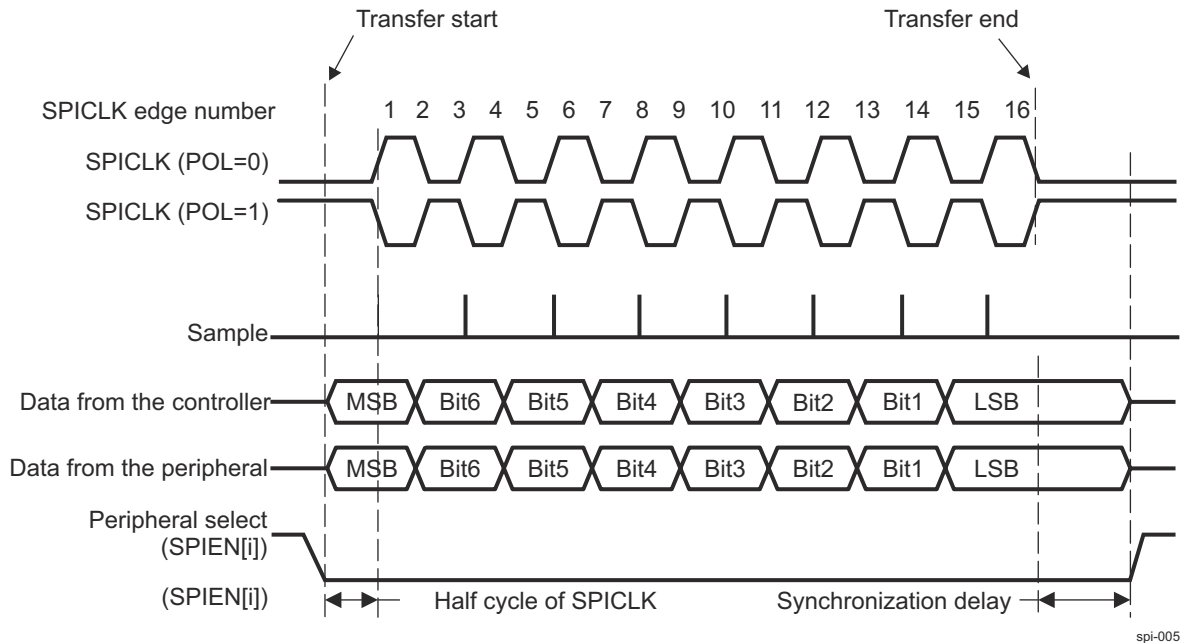
- Transmission in mode 0 and mode 2 (PHA = 0)

When PHA = 0, the first bit of the MCSPI word to transmit (on the controller or the peripheral data output pin) is valid one-half cycle of SPICLK after the assertion of SPIEN[i].

Therefore, the first edge of the SPICLK line is used by the controller to sample the first data bit sent by the peripheral. On the same edge, the first data bit sent by the controller is sampled by the peripheral.

On the next SPICLK edge, the received data bit is shifted into the receive shift register and a new data bit is transmitted on the serial data line.

This process continues for a number of pulses on the SPICLK line defined by the MCSPI word length programmed in the controller device, with data being latched on odd-numbered edges and shifted on even-numbered edges, see [Figure 12-61](#).



**Figure 12-61. Full-Duplex Transfer Format With PHA = 0**

- Transmission in mode 1 and mode 3 (PHA = 1)

When PHA = 1, the first bit of the MCSPI word to transmit (on the controller or the peripheral data output pin) is valid on the following SPICLK edge (one-half cycle later). This is the sampling edge for the controller and peripheral. A synchronization delay is added between the activation of SPIEN[i] and the first SPICLK edge.

The received data bit is shifted into the shift register on the third SPICLK edge.

This process continues for a number of pulses on the SPICLK line defined by the MCSPI word length programmed in the controller device, with data being latched on even-numbered edges and shifted on odd-numbered edges.

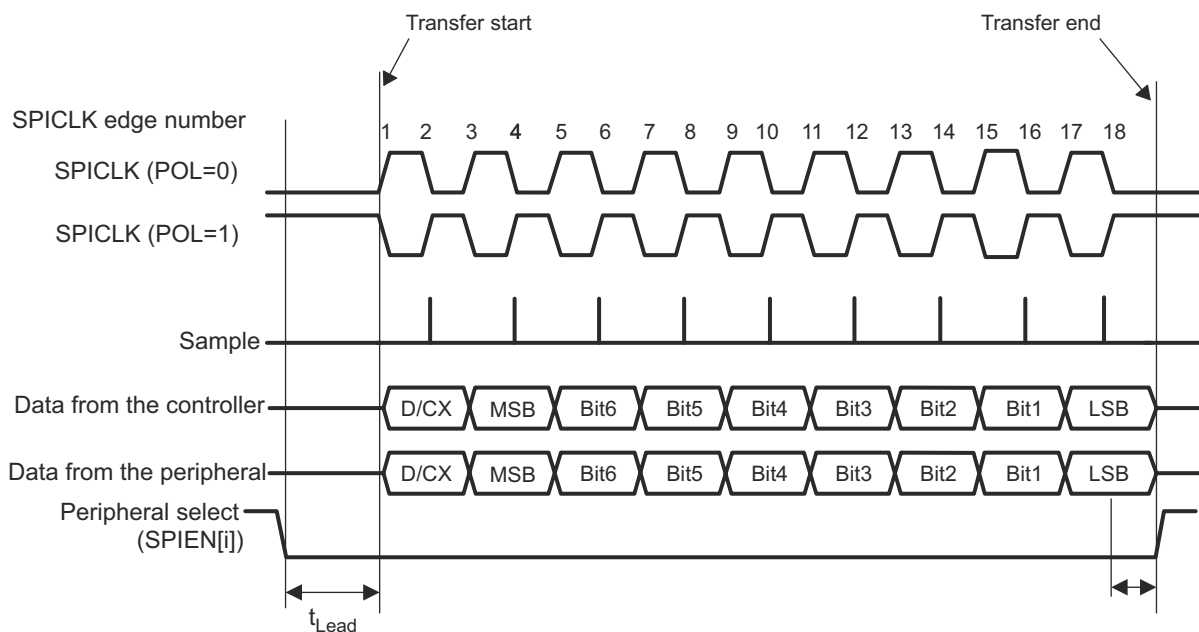
#### Note

The minimum synchronization delay is one cycle of SPICLK, if the frequency of SPICLK equals the frequency of MCSPI\_FCLK (MCSPI functional clock) in controller mode. The minimum synchronization delay is one-half cycle of SPICLK, if the frequency of SPICLK is lower than the frequency of MCSPI\_FCLK in the controller and peripheral modes.

- Transmission with a start-bit (SBE = 1)

When the MCSPI\_CHCONF\_0/1/2/3[23] SBE bit is set to 1, a start-bit is added before the MSB to indicate whether the next MCSPI word must be handled as a command or as data.

Figure 12-62 shows an example of a data transfer with an extra start-bit.

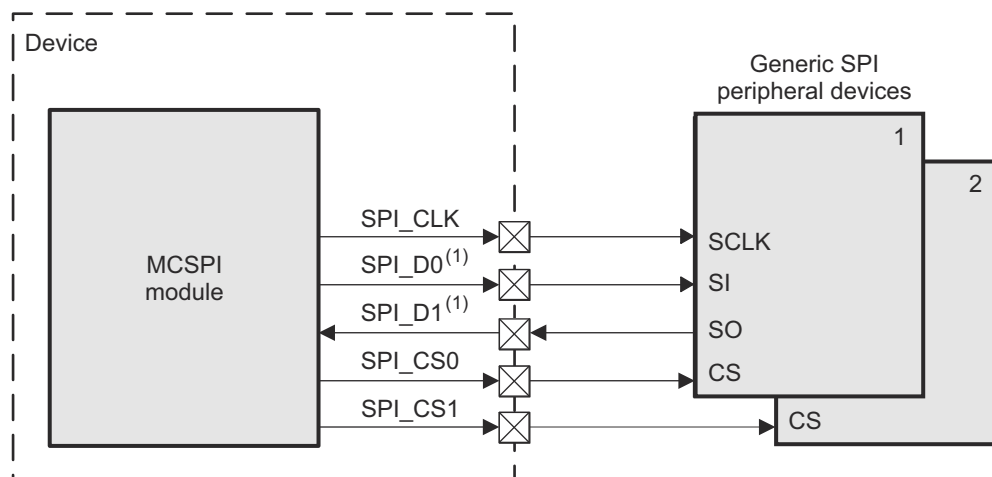


spl-006

**Figure 12-62. Extended MCSPI Transfer With a Start-Bit (SBE = 1)**

#### 12.1.5.2.5 MCSPI in Controller Mode

Figure 12-63 shows a case in controller mode (full-duplex) where the MCSPI module is connected with two peripheral devices.

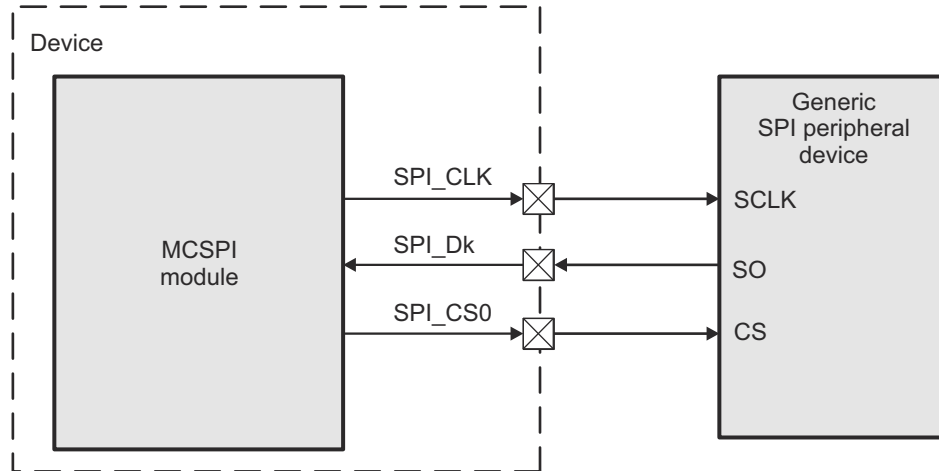


spl-007

- A. Direction depends on MCSPI\_CHCONF\_0/1/2/3[16] DPE0, MCSPI\_CHCONF\_0/1/2/3[17] DPE1 and MCSPI\_CHCONF\_0/1/2/3[18] IS bits

**Figure 12-63. MCSPI Controller Mode (Full Duplex)**

Figure 12-64 shows the controller single mode, which can also be configured in receive-only mode.



spl-008

k = 0 or 1 depending on MCSPI\_CHCONF\_0/1/2/3[16] DPE0, MCSPI\_CHCONF\_0/1/2/3[17] DPE1 and MCSPI\_CHCONF\_0/1/2/3[18] IS bits

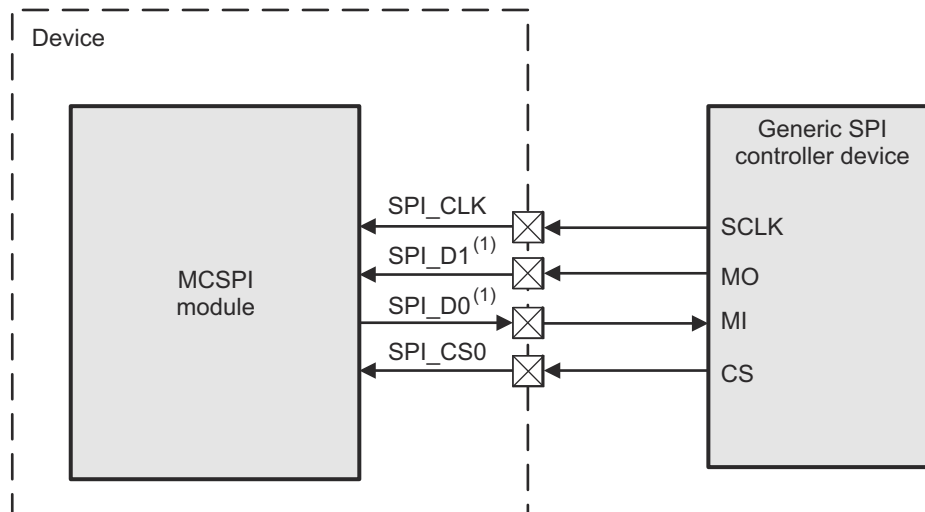
**Figure 12-64. MCSPI Controller Single Mode (Receive Only)**

#### 12.1.5.2.6 MCSPI in Peripheral Mode

Figure 12-65 shows a case in peripheral mode (full-duplex).

#### Note

Only channel 0 can be configured as peripheral, but the chip-enable signal can be connected to any SPIEN[i] pin and then rerouted internally to channel 0 (the MCSPI\_CHCONF\_0[22-21] SPIENSLV bit field). For more information, see [Section 12.1.5.4.4, MCSPI Peripheral Mode](#).

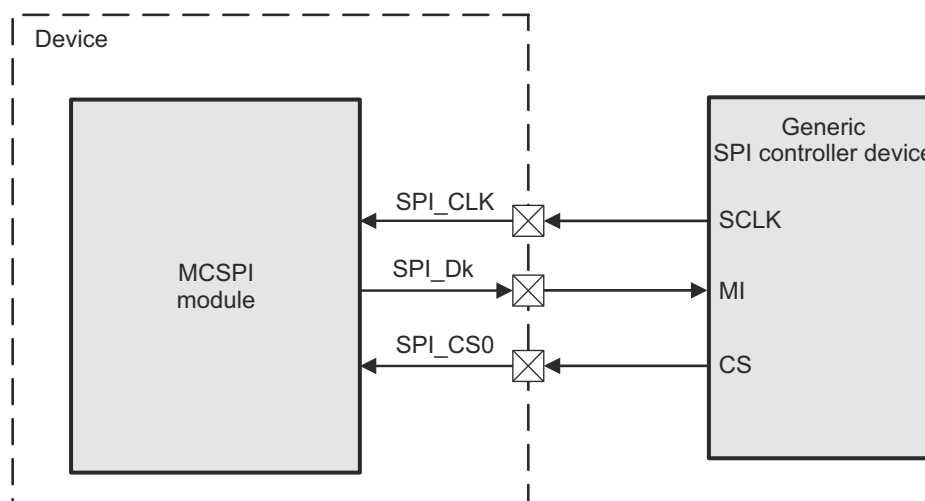


spl-009

A. Direction depends on MCSPI\_CHCONF\_0/1/2/3[16] DPE0, MCSPI\_CHCONF\_0/1/2/3[17] DPE1 and MCSPI\_CHCONF\_0/1/2/3[18] IS bits

**Figure 12-65. MCSPI Peripheral Mode (Full Duplex)**

Figure 12-66 shows the peripheral single mode, which can also be configured in transmit-only mode.



spl-010

k = 0 or 1 depending on MCSPI\_CHCONF\_0/1/2/3[16] DPE0, MCSPI\_CHCONF\_0/1/2/3[17] DPE1 and MCSPI\_CHCONF\_0/1/2/3[18] IS bits

**Figure 12-66. MCSPI Peripheral Single Mode (Transmit Only)**

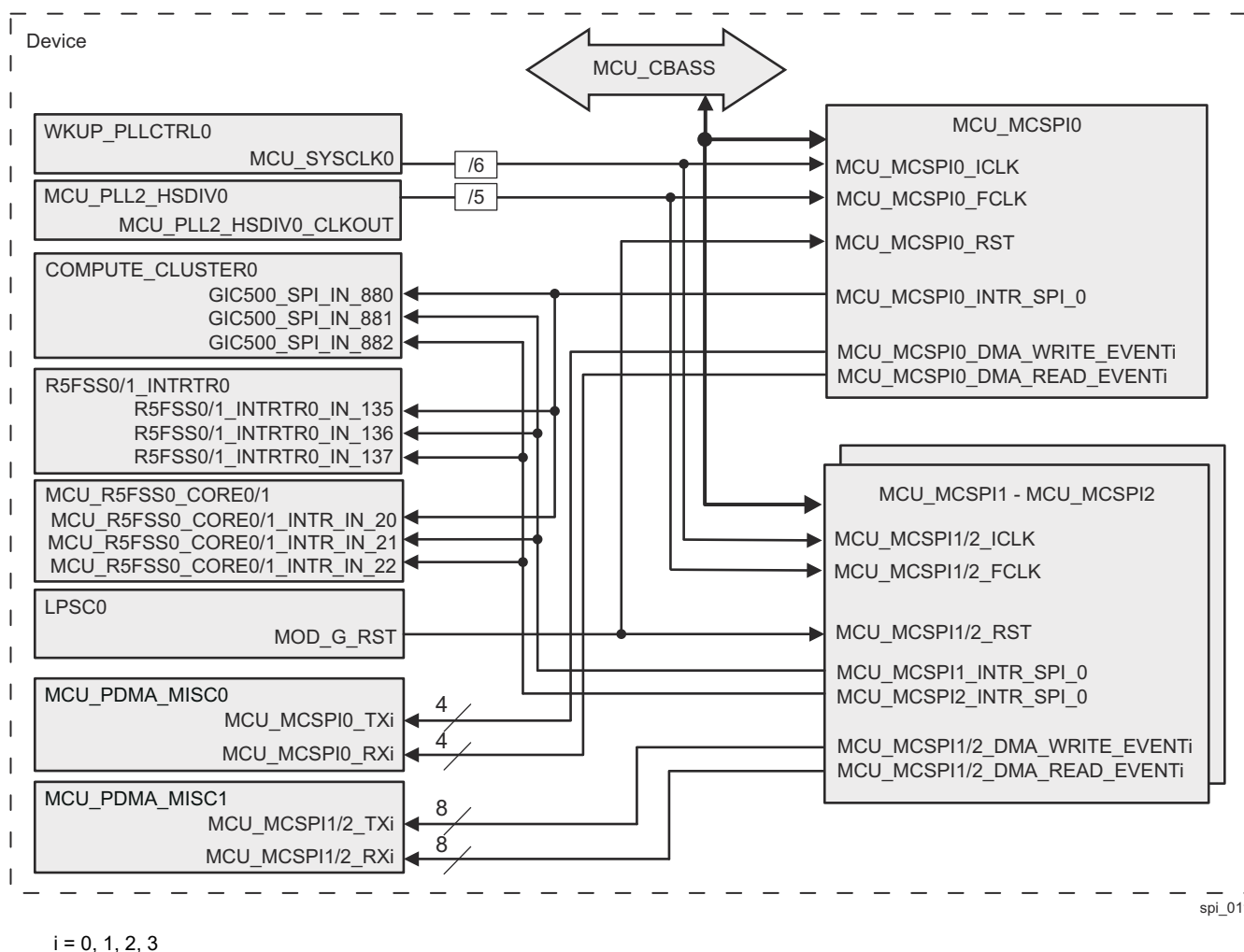


### 12.1.5.3 MCSPI Integration

This section describes module integration in the device, including information about clocks, resets, and hardware requests.

### 12.1.5.3.1 MCSPI Integration in MCU Domain

There are three MCSPI modules integrated in the device MCU domain - MCU\_MCSPi0, MCU\_MCSPi1, and MCU\_MCSPi2. Figure 12-67 shows their integration in the device.



**Figure 12-67. MCU\_MCSPi Integration**

Table 12-62 through Table 12-64 summarize the integration of MCU\_MCSPi0, MCU\_MCSPi1, and MCU\_MCSPi2 in device MCU domain.

**Table 12-62. MCU\_MCSPi Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
MCU_MCSPi0	WKUP_PSC0	PD0	LPSC0	MCU_CBASS0
MCU_MCSPi1	WKUP_PSC0	PD0	LPSC0	MCU_CBASS0
MCU_MCSPi2	WKUP_PSC0	PD0	LPSC0	MCU_CBASS0

**Table 12-63. MCU\_MCSPi Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
MCU_MCSPi0	MCU_MCSPi0_ICLK	MCU_SYSClk0/6	WKUP_PLLCTRL0	MCU_MCSPi0 Interface Clock
	MCU_MCSPi0_FCLK	MCU_PLL2_HSDIV0_CLKOUT T/5	MCU_PLL2	MCU_MCSPi0 Functional Clock

**Table 12-63. MCU\_MCSPI Clocks and Resets (continued)**

MCU_MCSPI1	MCU_MCSPI1_ICLK	MCU_SYCLK0/6	WKUP_PLLCTRL0	MCU_MCSPI1 Interface Clock
	MCU_MCSPI1_FCLK	MCU_PLL2_HSDIV0_CLKOU T/5	MCU_PLL2	MCU_MCSPI1 Functional Clock
MCU_MCSPI2	MCU_MCSPI2_ICLK	MCU_SYCLK0/6	WKUP_PLLCTRL0	MCU_MCSPI2 Interface Clock
	MCU_MCSPI2_FCLK	MCU_PLL2_HSDIV0_CLKOU T/5	MCU_PLL2	MCU_MCSPI2 Functional Clock

Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
MCU_MCSPI0	MCU_MCSPI0_RST	MOD_G_RST	LPSC0	MCU_MCSPI0 Asynchronous Reset
MCU_MCSPI1	MCU_MCSPI1_RST	MOD_G_RST	LPSC0	MCU_MCSPI1 Asynchronous Reset
MCU_MCSPI2	MCU_MCSPI2_RST	MOD_G_RST	LPSC0	MCU_MCSPI2 Asynchronous Reset

**Table 12-64. MCU\_MCSPI Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCU_MCSPI0	MCU_MCSPI0_INTR_SPI_0	GIC500_SPI_IN_880	COMPUTE_CLUSTER0	MCU_MCSPI0 Interrupt Request	Level
		R5FSS0_INTRTR0_IN_135	R5FSS0_INTRTR0	MCU_MCSPI0 Interrupt Request	Level
		R5FSS1_INTRTR0_IN_135	R5FSS1_INTRTR0	MCU_MCSPI0 Interrupt Request	Level
		MCU_R5FSS0_CORE0_INT_R_IN_20	MCU_R5FSS0_CORE0	MCU_MCSPI0 Interrupt Request	Level
		MCU_R5FSS0_CORE1_INT_R_IN_20	MCU_R5FSS0_CORE1	MCU_MCSPI0 Interrupt Request	Level
MCU_MCSPI1	MCU_MCSPI1_INTR_SPI_0	GIC500_SPI_IN_881	COMPUTE_CLUSTER0	MCU_MCSPI1 Interrupt Request	Level
		R5FSS0_INTRTR0_IN_136	R5FSS0_INTRTR0	MCU_MCSPI1 Interrupt Request	Level
		R5FSS1_INTRTR0_IN_136	R5FSS1_INTRTR0	MCU_MCSPI1 Interrupt Request	Level
		MCU_R5FSS0_CORE0_INT_R_IN_21	MCU_R5FSS0_CORE0	MCU_MCSPI1 Interrupt Request	Level
		MCU_R5FSS0_CORE1_INT_R_IN_21	MCU_R5FSS0_CORE1	MCU_MCSPI1 Interrupt Request	Level
MCU_MCSPI2	MCU_MCSPI2_INTR_SPI_0	GIC500_SPI_IN_882	COMPUTE_CLUSTER0	MCU_MCSPI2 Interrupt Request	Level
		R5FSS0_INTRTR0_IN_137	R5FSS0_INTRTR0	MCU_MCSPI2 Interrupt Request	Level
		R5FSS1_INTRTR0_IN_137	R5FSS1_INTRTR0	MCU_MCSPI2 Interrupt Request	Level
		MCU_R5FSS0_CORE0_INT_R_IN_22	MCU_R5FSS0_CORE0	MCU_MCSPI2 Interrupt Request	Level
		MCU_R5FSS0_CORE1_INT_R_IN_22	MCU_R5FSS0_CORE1	MCU_MCSPI2 Interrupt Request	Level

DMA Events					
Module Instance	Module DMA Event	Destination DMA Event Input	Destination	Description	Type

**Table 12-64. MCU\_MCSPI Hardware Requests (continued)**

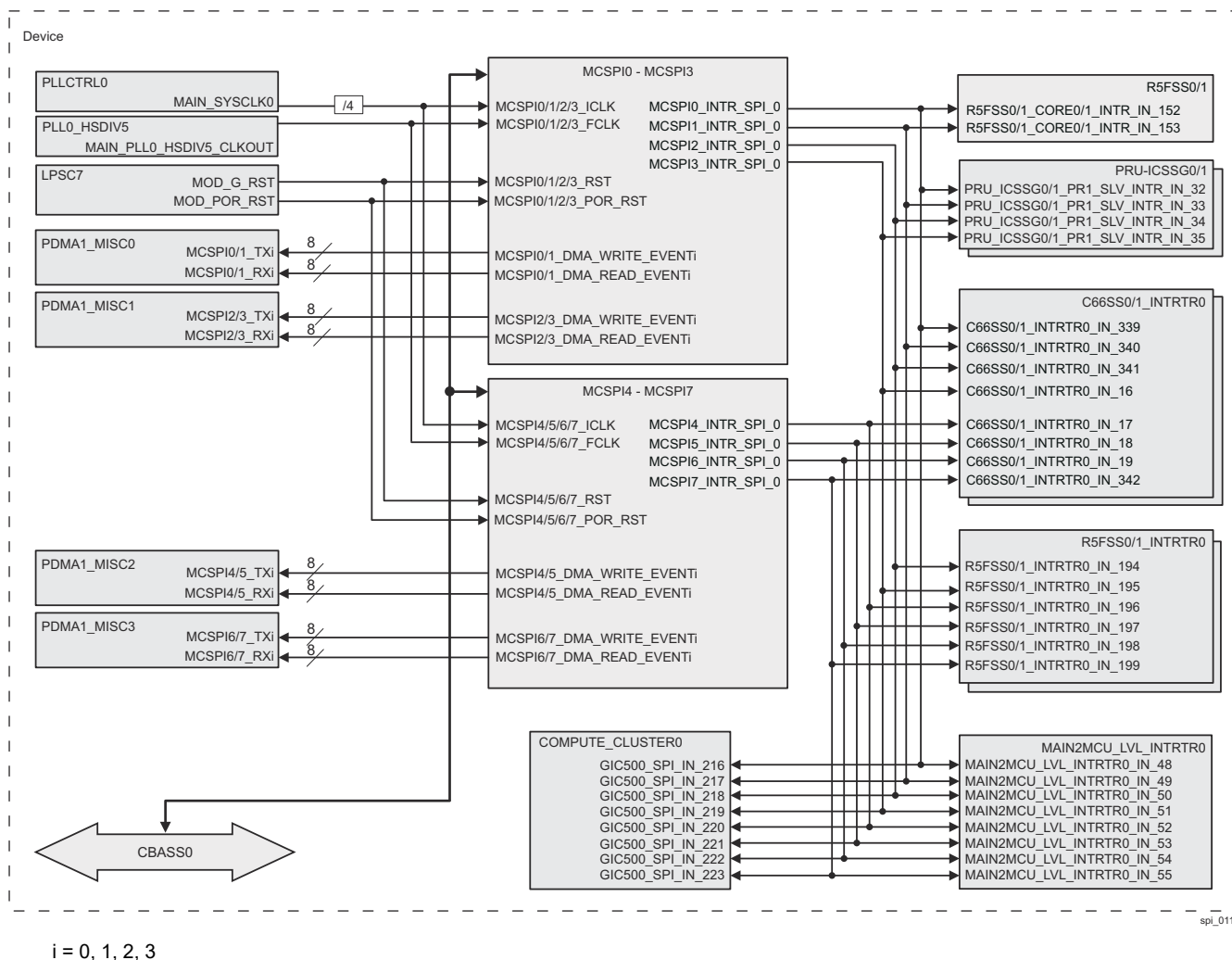
MCU_MCS PI0	MCU_MCSPI0_DMA_WRIT E_EVENT0	MCU_MCSPI0_TX0	MCU_PDMA_MISC0	MCU_MCSPI0 Channel 0 Transmit (Write) Request Line	Pulse
	MCU_MCSPI0_DMA_READ _EVENT0	MCU_MCSPI0_RX0	MCU_PDMA_MISC0	MCU_MCSPI0 Channel 0 Receive (Read) Request Line	Pulse
	MCU_MCSPI0_DMA_WRIT E_EVENT1	MCU_MCSPI0_TX1	MCU_PDMA_MISC0	MCU_MCSPI0 Channel 1 Transmit (Write) Request Line	Pulse
	MCU_MCSPI0_DMA_READ _EVENT1	MCU_MCSPI0_RX1	MCU_PDMA_MISC0	MCU_MCSPI0 Channel 1 Receive (Read) Request Line	Pulse
	MCU_MCSPI0_DMA_WRIT E_EVENT2	MCU_MCSPI0_TX2	MCU_PDMA_MISC0	MCU_MCSPI0 Channel 2 Transmit (Write) Request Line	Pulse
	MCU_MCSPI0_DMA_READ _EVENT2	MCU_MCSPI0_RX2	MCU_PDMA_MISC0	MCU_MCSPI0 Channel 2 Receive (Read) Request Line	Pulse
	MCU_MCSPI0_DMA_WRIT E_EVENT3	MCU_MCSPI0_TX3	MCU_PDMA_MISC0	MCU_MCSPI0 Channel 3 Transmit (Write) Request Line	Pulse
MCU_MCS PI1	MCU_MCSPI1_DMA_WRIT E_EVENT0	MCU_MCSPI1_TX0	MCU_PDMA_MISC1	MCU_MCSPI1 Channel 0 Transmit (Write) Request Line	Pulse
	MCU_MCSPI1_DMA_READ _EVENT0	MCU_MCSPI1_RX0	MCU_PDMA_MISC1	MCU_MCSPI1 Channel 0 Receive (Read) Request Line	Pulse
	MCU_MCSPI1_DMA_WRIT E_EVENT1	MCU_MCSPI1_TX1	MCU_PDMA_MISC1	MCU_MCSPI1 Channel 1 Transmit (Write) Request Line	Pulse
	MCU_MCSPI1_DMA_READ _EVENT1	MCU_MCSPI1_RX1	MCU_PDMA_MISC1	MCU_MCSPI1 Channel 1 Receive (Read) Request Line	Pulse
	MCU_MCSPI1_DMA_WRIT E_EVENT2	MCU_MCSPI1_TX2	MCU_PDMA_MISC1	MCU_MCSPI1 Channel 2 Transmit (Write) Request Line	Pulse
	MCU_MCSPI1_DMA_READ _EVENT2	MCU_MCSPI1_RX2	MCU_PDMA_MISC1	MCU_MCSPI1 Channel 2 Receive (Read) Request Line	Pulse
	MCU_MCSPI1_DMA_WRIT E_EVENT3	MCU_MCSPI1_TX3	MCU_PDMA_MISC1	MCU_MCSPI1 Channel 3 Transmit (Write) Request Line	Pulse
MCU_MCS PI2	MCU_MCSPI2_DMA_WRIT E_EVENT0	MCU_MCSPI2_TX0	MCU_PDMA_MISC1	MCU_MCSPI2 Channel 0 Transmit (Write) Request Line	Pulse
	MCU_MCSPI2_DMA_READ _EVENT0	MCU_MCSPI2_RX0	MCU_PDMA_MISC1	MCU_MCSPI2 Channel 0 Receive (Read) Request Line	Pulse
	MCU_MCSPI2_DMA_WRIT E_EVENT1	MCU_MCSPI2_TX1	MCU_PDMA_MISC1	MCU_MCSPI2 Channel 1 Transmit (Write) Request Line	Pulse

**Table 12-64. MCU\_MCSPI Hardware Requests (continued)**

MCU_MCSPI2_DMA_READ_EVENT1	MCU_MCSPI2_RX1	MCU_PDMA_MISC1	MCU_MCSPI2 Channel 1 Receive (Read) Request Line	Pulse
MCU_MCSPI2_DMA_WRITE_EVENT2	MCU_MCSPI2_TX2	MCU_PDMA_MISC1	MCU_MCSPI2 Channel 2 Transmit (Write) Request Line	Pulse
MCU_MCSPI2_DMA_READ_EVENT2	MCU_MCSPI2_RX2	MCU_PDMA_MISC1	MCU_MCSPI2 Channel 2 Receive (Read) Request Line	Pulse
MCU_MCSPI2_DMA_WRITE_EVENT3	MCU_MCSPI2_TX3	MCU_PDMA_MISC1	MCU_MCSPI2 Channel 3 Transmit (Write) Request Line	Pulse
MCU_MCSPI2_DMA_READ_EVENT3	MCU_MCSPI2_RX3	MCU_PDMA_MISC1	MCU_MCSPI2 Channel 3 Receive (Read) Request Line	Pulse

### 12.1.5.3.2 MCSPI Integration in MAIN Domain

There are eight MCSPI modules integrated in the device MAIN domain - MCSPI0 through MCSPI7. Figure 12-68 shows their integration in the device.



**Figure 12-68. MCSPI Integration**

Table 12-65 through Table 12-67 summarize the integration of MCSPI0 through MCSPI7 in device MAIN domain.

**Table 12-65. MCSPI Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
MCSPi0	PSC0	PD0	LPSC7	CBASS0
MCSPi1	PSC0	PD0	LPSC7	CBASS0
MCSPi2	PSC0	PD0	LPSC7	CBASS0
MCSPi3	PSC0	PD0	LPSC7	CBASS0
MCSPi4	PSC0	PD0	LPSC7	CBASS0
MCSPi5	PSC0	PD0	LPSC7	CBASS0
MCSPi6	PSC0	PD0	LPSC7	CBASS0
MCSPi7	PSC0	PD0	LPSC7	CBASS0

**Table 12-66. MCSPI Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
MCSPI0	MCSPI0_ICLK	MAIN_SYSCLK0/4	PLLCTRL0	MCSPI0 Interface Clock
	MCSPI0_FCLK	MAIN_PLL0_HSDIV5_CLKOUT	PLL0_HSDIV	MCSPI0 Functional Clock
MCSPI1	MCSPI1_ICLK	MAIN_SYSCLK0/4	PLLCTRL0	MCSPI1 Interface Clock
	MCSPI1_FCLK	MAIN_PLL0_HSDIV5_CLKOUT	PLL0_HSDIV	MCSPI1 Functional Clock
MCSPI2	MCSPI2_ICLK	MAIN_SYSCLK0/4	PLLCTRL0	MCSPI2 Interface Clock
	MCSPI2_FCLK	MAIN_PLL0_HSDIV5_CLKOUT	PLL0_HSDIV	MCSPI2 Functional Clock
MCSPI3	MCSPI3_ICLK	MAIN_SYSCLK0/4	PLLCTRL0	MCSPI3 Interface Clock
	MCSPI3_FCLK	MAIN_PLL0_HSDIV5_CLKOUT	PLL0_HSDIV	MCSPI3 Functional Clock
MCSPI4	MCSPI4_ICLK	MAIN_SYSCLK0/4	PLLCTRL0	MCSPI4 Interface Clock
	MCSPI4_FCLK	MAIN_PLL0_HSDIV5_CLKOUT	PLL0_HSDIV	MCSPI4 Functional Clock
MCSPI5	MCSPI5_ICLK	MAIN_SYSCLK0/4	PLLCTRL0	MCSPI5 Interface Clock
	MCSPI5_FCLK	MAIN_PLL0_HSDIV5_CLKOUT	PLL0_HSDIV	MCSPI5 Functional Clock
MCSPI6	MCSPI6_ICLK	MAIN_SYSCLK0/4	PLLCTRL0	MCSPI6 Interface Clock
	MCSPI6_FCLK	MAIN_PLL0_HSDIV5_CLKOUT	PLL0_HSDIV	MCSPI6 Functional Clock
MCSPI7	MCSPI7_ICLK	MAIN_SYSCLK0/4	PLLCTRL0	MCSPI7 Interface Clock
	MCSPI7_FCLK	MAIN_PLL0_HSDIV5_CLKOUT	PLL0_HSDIV	MCSPI7 Functional Clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
MCSPI0	MCSPI0_RST	MOD_G_RST	LPSC7	MCSPI0 Asynchronous Reset
	MCSPI0_POR_RST	MOD_POR_RST	LPSC7	MCSPI0 Power-On Reset
MCSPI1	MCSPI1_RST	MOD_G_RST	LPSC7	MCSPI1 Asynchronous Reset
	MCSPI1_POR_RST	MOD_POR_RST	LPSC7	MCSPI1 Power-On Reset
MCSPI2	MCSPI2_RST	MOD_G_RST	LPSC7	MCSPI2 Asynchronous Reset
	MCSPI3_POR_RST	MOD_POR_RST	LPSC7	MCSPI2 Power-On Reset
MCSPI3	MCSPI3_RST	MOD_G_RST	LPSC7	MCSPI3 Asynchronous Reset
	MCSPI3_POR_RST	MOD_POR_RST	LPSC7	MCSPI3 Power-On Reset
MCSPI4	MCSPI4_RST	MOD_G_RST	LPSC7	MCSPI4 Asynchronous Reset
	MCSPI4_POR_RST	MOD_POR_RST	LPSC7	MCSPI4 Power-On Reset
MCSPI5	MCSPI5_RST	MOD_G_RST	LPSC7	MCSPI5 Asynchronous Reset
	MCSPI5_POR_RST	MOD_POR_RST	LPSC7	MCSPI5 Power-On Reset
MCSPI6	MCSPI6_RST	MOD_G_RST	LPSC7	MCSPI6 Asynchronous Reset
	MCSPI6_POR_RST	MOD_POR_RST	LPSC7	MCSPI6 Power-On Reset
MCSPI7	MCSPI7_RST	MOD_G_RST	LPSC7	MCSPI7 Asynchronous Reset
	MCSPI7_POR_RST	MOD_POR_RST	LPSC7	MCSPI7 Power-On Reset

**Table 12-67. MCSPI Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCSPI0	MCSPI0_INTR_SPI_0	GIC500_SPI_IN_216	COMPUTE_CLUSTER0	MCSPI0 Interrupt Request	Level
		PRU_ICSSG0_PR1_SLV_IN_TR_IN_32	PRU-ICSSG0	MCSPI0 Interrupt Request	Level
		PRU_ICSSG1_PR1_SLV_IN_TR_IN_32	PRU-ICSSG1	MCSPI0 Interrupt Request	Level
		C66SS0_INTRTR0_IN_339	C66SS0_INTRTR0	MCSPI0 Interrupt Request	Level
		C66SS0_INTRTR1_IN_339	C66SS1_INTRTR0	MCSPI0 Interrupt Request	Level
		MAIN2MCU_LVL_INTRTR0_I_N_48	MAIN2MCU_LVL_INTRTR0	MCSPI0 Interrupt Request	Level
		R5FSS0_CORE0_INTR_IN_152	R5FSS0	MCSPI0 Interrupt Request	Level
		R5FSS0_CORE1_INTR_IN_152			
		R5FSS1_CORE0_INTR_IN_152	R5FSS1	MCSPI0 Interrupt Request	Level
		R5FSS1_CORE1_INTR_IN_152			
MCSPI1	MCSPI1_INTR_SPI_0	GIC500_SPI_IN_217	COMPUTE_CLUSTER0	MCSPI1 Interrupt Request	Level
		PRU_ICSSG0_PR1_SLV_IN_TR_IN_33	PRU-ICSSG0	MCSPI1 Interrupt Request	Level
		PRU_ICSSG1_PR1_SLV_IN_TR_IN_33	PRU-ICSSG1	MCSPI1 Interrupt Request	Level
		C66SS0_INTRTR0_IN_340	C66SS0_INTRTR0	MCSPI1 Interrupt Request	Level
		C66SS0_INTRTR1_IN_340	C66SS1_INTRTR0	MCSPI1 Interrupt Request	Level
		MAIN2MCU_LVL_INTRTR0_I_N_49	MAIN2MCU_LVL_INTRTR0	MCSPI1 Interrupt Request	Level
		R5FSS0_CORE0_INTR_IN_153	R5FSS0	MCSPI1 Interrupt Request	Level
		R5FSS0_CORE1_INTR_IN_153			
		R5FSS1_CORE0_INTR_IN_153	R5FSS1	MCSPI1 Interrupt Request	Level
		R5FSS1_CORE1_INTR_IN_153			
MCSPI2	MCSPI2_INTR_SPI_0	GIC500_SPI_IN_218	COMPUTE_CLUSTER0	MCSPI2 Interrupt Request	Level
		PRU_ICSSG0_PR1_SLV_IN_TR_IN_34	PRU-ICSSG0	MCSPI2 Interrupt Request	Level
		PRU_ICSSG1_PR1_SLV_IN_TR_IN_34	PRU-ICSSG1	MCSPI2 Interrupt Request	Level
		C66SS0_INTRTR0_IN_341	C66SS0_INTRTR0	MCSPI2 Interrupt Request	Level
		C66SS0_INTRTR1_IN_341	C66SS1_INTRTR0	MCSPI2 Interrupt Request	Level
		MAIN2MCU_LVL_INTRTR0_I_N_50	MAIN2MCU_LVL_INTRTR0	MCSPI2 Interrupt Request	Level
		R5FSS0_INTRTR0_IN_194	R5FSS0_INTRTR0	MCSPI2 Interrupt Request	Level
		R5FSS1_INTRTR0_IN_194	R5FSS1_INTRTR0	MCSPI2 Interrupt Request	Level
MCSPI3	MCSPI3_INTR_SPI_0	GIC500_SPI_IN_219	COMPUTE_CLUSTER0	MCSPI3 Interrupt Request	Level
		PRU_ICSSG0_PR1_SLV_IN_TR_IN_35	PRU-ICSSG0	MCSPI3 Interrupt Request	Level



**Table 12-67. MCSPI Hardware Requests (continued)**

MCSPI4	MCSPI4_INTR_SPI_0	PRU_ICSSG1_PR1_SLV_IN TR_IN_35	PRU-ICSSG1	MCSPI3 Interrupt Request	Level
		C66SS0_INTRTR0_IN_16	C66SS0_INTRTR0	MCSPI3 Interrupt Request	Level
		C66SS1_INTRTR0_IN_16	C66SS1_INTRTR0	MCSPI3 Interrupt Request	Level
		MAIN2MCU_LVL_INTRTR0_I N_51	MAIN2MCU_LVL_INTRT R0	MCSPI3 Interrupt Request	Level
		R5FSS0_INTRTR0_IN_195	R5FSS0_INTRTR0	MCSPI3 Interrupt Request	Level
		R5FSS1_INTRTR0_IN_195	R5FSS1_INTRTR0	MCSPI3 Interrupt Request	Level
		GIC500_SPI_IN_220	COMPUTE_CLUSTER0	MCSPI4 Interrupt Request	Level
		C66SS0_INTRTR0_IN_17	C66SS0_INTRTR0	MCSPI4 Interrupt Request	Level
		C66SS1_INTRTR0_IN_17	C66SS1_INTRTR0	MCSPI4 Interrupt Request	Level
		MAIN2MCU_LVL_INTRTR0_I N_52	MAIN2MCU_LVL_INTRT R0	MCSPI4 Interrupt Request	Level
		R5FSS0_INTRTR0_IN_196	R5FSS0_INTRTR0	MCSPI4 Interrupt Request	Level
		R5FSS1_INTRTR0_IN_196	R5FSS1_INTRTR0	MCSPI4 Interrupt Request	Level
		GIC500_SPI_IN_221	COMPUTE_CLUSTER0	MCSPI5 Interrupt Request	Level
		C66SS0_INTRTR0_IN_18	C66SS0_INTRTR0	MCSPI5 Interrupt Request	Level
		C66SS1_INTRTR0_IN_18	C66SS1_INTRTR0	MCSPI5 Interrupt Request	Level
		MAIN2MCU_LVL_INTRTR0_I N_53	MAIN2MCU_LVL_INTRT R0	MCSPI5 Interrupt Request	Level
MCSPI5	MCSPI5_INTR_SPI_0	R5FSS0_INTRTR0_IN_197	R5FSS0_INTRTR0	MCSPI5 Interrupt Request	Level
		R5FSS1_INTRTR0_IN_197	R5FSS1_INTRTR0	MCSPI5 Interrupt Request	Level
		GIC500_SPI_IN_222	COMPUTE_CLUSTER0	MCSPI6 Interrupt Request	Level
		C66SS0_INTRTR0_IN_19	C66SS0_INTRTR0	MCSPI6 Interrupt Request	Level
MCSPI6	MCSPI6_INTR_SPI_0	C66SS1_INTRTR0_IN_19	C66SS1_INTRTR0	MCSPI6 Interrupt Request	Level
		MAIN2MCU_LVL_INTRTR0_I N_54	MAIN2MCU_LVL_INTRT R0	MCSPI6 Interrupt Request	Level
		R5FSS0_INTRTR0_IN_198	R5FSS0_INTRTR0	MCSPI6 Interrupt Request	Level
		R5FSS1_INTRTR0_IN_198	R5FSS1_INTRTR0	MCSPI6 Interrupt Request	Level
MCSPI7	MCSPI7_INTR_SPI_0	GIC500_SPI_IN_223	COMPUTE_CLUSTER0	MCSPI7 Interrupt Request	Level
		C66SS0_INTRTR0_IN_342	C66SS0_INTRTR0	MCSPI7 Interrupt Request	Level
		C66SS0_INTRTR1_IN_342	C66SS1_INTRTR0	MCSPI7 Interrupt Request	Level
		MAIN2MCU_LVL_INTRTR0_I N_55	MAIN2MCU_LVL_INTRT R0	MCSPI7 Interrupt Request	Level
		R5FSS0_INTRTR0_IN_199	R5FSS0_INTRTR0	MCSPI7 Interrupt Request	Level
		R5FSS1_INTRTR0_IN_199	R5FSS1_INTRTR0	MCSPI7 Interrupt Request	Level

**DMA Events**

Module Instance	Module DMA Event	Destination DMA Event Input	Destination	Description	Type
MCSPI0	MCSPI0_DMA_WRITE_EV ENT0	MCSPI0_TX0	PDMA1_MISC0	MCSPI0 Channel 0 Transmit (Write) Request Line	Pulse
	MCSPI0_DMA_READ_EV ENT0	MCSPI0_RX0	PDMA1_MISC0	MCSPI0 Channel 0 Receive (Read) Request Line	Pulse
	MCSPI0_DMA_WRITE_EV ENT1	MCSPI0_TX1	PDMA1_MISC0	MCSPI0 Channel 1 Transmit (Write) Request Line	Pulse
	MCSPI0_DMA_READ_EV ENT1	MCSPI0_RX1	PDMA1_MISC0	MCSPI0 Channel 1 Receive (Read) Request Line	Pulse
	MCSPI0_DMA_WRITE_EV ENT2	MCSPI0_TX2	PDMA1_MISC0	MCSPI0 Channel 2 Transmit (Write) Request Line	Pulse

**Table 12-67. MCSPI Hardware Requests (continued)**

MCSPI1	MCSP10_DMA_READ_EV ENT2	MCSP10_RX2	PDMA1_MISC0	MCSP10 Channel 2 Receive (Read) Request Line	Pulse
	MCSP10_DMA_WRITE_EV ENT3	MCSP10_TX3	PDMA1_MISC0	MCSP10 Channel 3 Transmit (Write) Request Line	Pulse
	MCSP10_DMA_READ_EV ENT3	MCSP10_RX3	PDMA1_MISC0	MCSP10 Channel 3 Receive (Read) Request Line	Pulse
	MCSP11_DMA_WRITE_EV ENT0	MCSP11_TX0	PDMA1_MISC0	MCSP11 Channel 0 Transmit (Write) Request Line	Pulse
	MCSP11_DMA_READ_EV ENT0	MCSP11_RX0	PDMA1_MISC0	MCSP11 Channel 0 Receive (Read) Request Line	Pulse
	MCSP11_DMA_WRITE_EV ENT1	MCSP11_TX1	PDMA1_MISC0	MCSP11 Channel 1 Transmit (Write) Request Line	Pulse
	MCSP11_DMA_READ_EV ENT1	MCSP11_RX1	PDMA1_MISC0	MCSP11 Channel 1 Receive (Read) Request Line	Pulse
	MCSP11_DMA_WRITE_EV ENT2	MCSP11_TX2	PDMA1_MISC0	MCSP11 Channel 2 Transmit (Write) Request Line	Pulse
	MCSP11_DMA_READ_EV ENT2	MCSP11_RX2	PDMA1_MISC0	MCSP11 Channel 2 Receive (Read) Request Line	Pulse
	MCSP11_DMA_WRITE_EV ENT3	MCSP11_TX3	PDMA1_MISC0	MCSP11 Channel 3 Transmit (Write) Request Line	Pulse
MCSPI2	MCSP11_DMA_READ_EV ENT3	MCSP11_RX3	PDMA1_MISC0	MCSP11 Channel 3 Receive (Read) Request Line	Pulse
	MCSP12_DMA_WRITE_EV ENT0	MCSP12_TX0	PDMA1_MISC1	MCSP12 Channel 0 Transmit (Write) Request Line	Pulse
	MCSP12_DMA_READ_EV ENT0	MCSP12_RX0	PDMA1_MISC1	MCSP12 Channel 0 Receive (Read) Request Line	Pulse
	MCSP12_DMA_WRITE_EV ENT1	MCSP12_TX1	PDMA1_MISC1	MCSP12 Channel 1 Transmit (Write) Request Line	Pulse
	MCSP12_DMA_READ_EV ENT1	MCSP12_RX1	PDMA1_MISC1	MCSP12 Channel 1 Receive (Read) Request Line	Pulse
	MCSP12_DMA_WRITE_EV ENT2	MCSP12_TX2	PDMA1_MISC1	MCSP12 Channel 2 Transmit (Write) Request Line	Pulse
	MCSP12_DMA_READ_EV ENT2	MCSP12_RX2	PDMA1_MISC1	MCSP12 Channel 2 Receive (Read) Request Line	Pulse
	MCSP12_DMA_WRITE_EV ENT3	MCSP12_TX3	PDMA1_MISC1	MCSP12 Channel 3 Transmit (Write) Request Line	Pulse
MCSPI3	MCSP12_DMA_READ_EV ENT3	MCSP12_RX3	PDMA1_MISC1	MCSP12 Channel 3 Receive (Read) Request Line	Pulse
	MCSP13_DMA_WRITE_EV ENT0	MCSP13_TX0	PDMA1_MISC1	MCSP13 Channel 0 Transmit (Write) Request Line	Pulse
	MCSP13_DMA_READ_EV ENT0	MCSP13_RX0	PDMA1_MISC1	MCSP13 Channel 0 Receive (Read) Request Line	Pulse
	MCSP13_DMA_WRITE_EV ENT1	MCSP13_TX1	PDMA1_MISC1	MCSP13 Channel 1 Transmit (Write) Request Line	Pulse
	MCSP13_DMA_READ_EV ENT1	MCSP13_RX1	PDMA1_MISC1	MCSP13 Channel 1 Receive (Read) Request Line	Pulse
	MCSP13_DMA_WRITE_EV ENT2	MCSP13_TX2	PDMA1_MISC1	MCSP13 Channel 2 Transmit (Write) Request Line	Pulse
	MCSP13_DMA_READ_EV ENT2	MCSP13_RX2	PDMA1_MISC1	MCSP13 Channel 2 Receive (Read) Request Line	Pulse
	MCSP13_DMA_WRITE_EV ENT3	MCSP13_TX3	PDMA1_MISC1	MCSP13 Channel 3 Transmit (Write) Request Line	Pulse
	MCSP13_DMA_READ_EV ENT3	MCSP13_RX3	PDMA1_MISC1	MCSP13 Channel 3 Receive (Read) Request Line	Pulse

**Table 12-67. MCSPI Hardware Requests (continued)**

MCSPI4	MCSPi4_DMA_WRITE_EV	MCSPi4_TX0 ENT0	PDMA1_MISC2	MCSPi4 Channel 0 Transmit (Write) Request Line	Pulse
	MCSPi4_DMA_READ_EV	MCSPi4_RX0 ENT0	PDMA1_MISC2	MCSPi4 Channel 0 Receive (Read) Request Line	Pulse
	MCSPi4_DMA_WRITE_EV	MCSPi4_TX1 ENT1	PDMA1_MISC2	MCSPi4 Channel 1 Transmit (Write) Request Line	Pulse
	MCSPi4_DMA_READ_EV	MCSPi4_RX1 ENT1	PDMA1_MISC2	MCSPi4 Channel 1 Receive (Read) Request Line	Pulse
	MCSPi4_DMA_WRITE_EV	MCSPi4_TX2 ENT2	PDMA1_MISC2	MCSPi4 Channel 2 Transmit (Write) Request Line	Pulse
	MCSPi4_DMA_READ_EV	MCSPi4_RX2 ENT2	PDMA1_MISC2	MCSPi4 Channel 2 Receive (Read) Request Line	Pulse
	MCSPi4_DMA_WRITE_EV	MCSPi4_TX3 ENT3	PDMA1_MISC2	MCSPi4 Channel 3 Transmit (Write) Request Line	Pulse
	MCSPi4_DMA_READ_EV	MCSPi4_RX3 ENT3	PDMA1_MISC2	MCSPi4 Channel 3 Receive (Read) Request Line	Pulse
MCSPI5	MCSPi5_DMA_WRITE_EV	MCSPi5_TX0 ENT0	PDMA1_MISC2	MCSPi5 Channel 0 Transmit (Write) Request Line	Pulse
	MCSPi5_DMA_READ_EV	MCSPi5_RX0 ENT0	PDMA1_MISC2	MCSPi5 Channel 0 Receive (Read) Request Line	Pulse
	MCSPi5_DMA_WRITE_EV	MCSPi5_TX1 ENT1	PDMA1_MISC2	MCSPi5 Channel 1 Transmit (Write) Request Line	Pulse
	MCSPi5_DMA_READ_EV	MCSPi5_RX1 ENT1	PDMA1_MISC2	MCSPi5 Channel 1 Receive (Read) Request Line	Pulse
	MCSPi5_DMA_WRITE_EV	MCSPi5_TX2 ENT2	PDMA1_MISC2	MCSPi5 Channel 2 Transmit (Write) Request Line	Pulse
	MCSPi5_DMA_READ_EV	MCSPi5_RX2 ENT2	PDMA1_MISC2	MCSPi5 Channel 2 Receive (Read) Request Line	Pulse
	MCSPi5_DMA_WRITE_EV	MCSPi5_TX3 ENT3	PDMA1_MISC2	MCSPi5 Channel 3 Transmit (Write) Request Line	Pulse
	MCSPi5_DMA_READ_EV	MCSPi5_RX3 ENT3	PDMA1_MISC2	MCSPi5 Channel 3 Receive (Read) Request Line	Pulse
MCSPI6	MCSPi6_DMA_WRITE_EV	MCSPi6_TX0 ENT0	PDMA1_MISC3	MCSPi6 Channel 0 Transmit (Write) Request Line	Pulse
	MCSPi6_DMA_READ_EV	MCSPi6_RX0 ENT0	PDMA1_MISC3	MCSPi6 Channel 0 Receive (Read) Request Line	Pulse
	MCSPi6_DMA_WRITE_EV	MCSPi6_TX1 ENT1	PDMA1_MISC3	MCSPi6 Channel 1 Transmit (Write) Request Line	Pulse
	MCSPi6_DMA_READ_EV	MCSPi6_RX1 ENT1	PDMA1_MISC3	MCSPi6 Channel 1 Receive (Read) Request Line	Pulse
	MCSPi6_DMA_WRITE_EV	MCSPi6_TX2 ENT2	PDMA1_MISC3	MCSPi6 Channel 2 Transmit (Write) Request Line	Pulse
	MCSPi6_DMA_READ_EV	MCSPi6_RX2 ENT2	PDMA1_MISC3	MCSPi6 Channel 2 Receive (Read) Request Line	Pulse
	MCSPi6_DMA_WRITE_EV	MCSPi6_TX3 ENT3	PDMA1_MISC3	MCSPi6 Channel 3 Transmit (Write) Request Line	Pulse
	MCSPi6_DMA_READ_EV	MCSPi6_RX3 ENT3	PDMA1_MISC3	MCSPi6 Channel 3 Receive (Read) Request Line	Pulse
MCSPI7	MCSPi7_DMA_WRITE_EV	MCSPi7_TX0 ENT0	PDMA1_MISC3	MCSPi7 Channel 0 Transmit (Write) Request Line	Pulse
	MCSPi7_DMA_READ_EV	MCSPi7_RX0 ENT0	PDMA1_MISC3	MCSPi7 Channel 0 Receive (Read) Request Line	Pulse
	MCSPi7_DMA_WRITE_EV	MCSPi7_TX1 ENT1	PDMA1_MISC3	MCSPi7 Channel 1 Transmit (Write) Request Line	Pulse

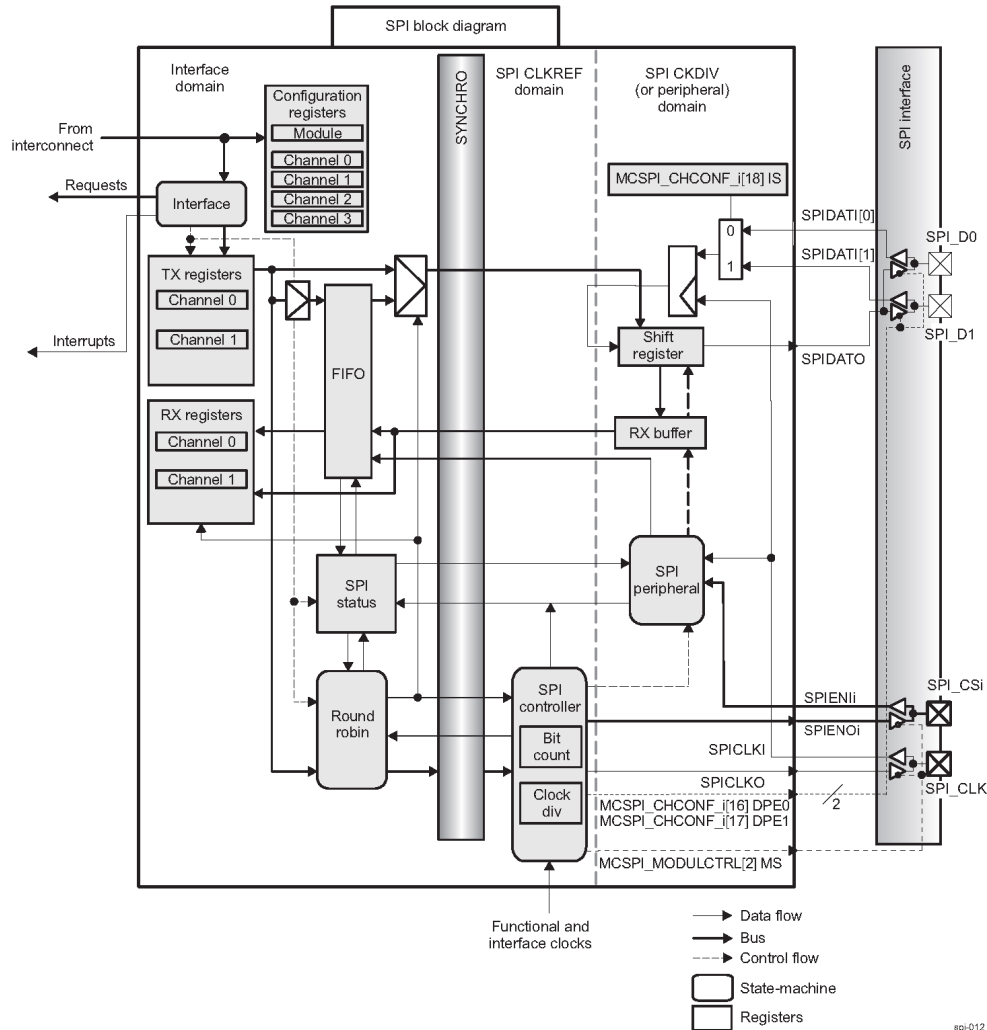
**Table 12-67. MCSPI Hardware Requests (continued)**

MCSPi7_DMA_READ_EV ENT1	MCSPi7_RX1	PDMA1_MISC3	MCSPi7 Channel 1 Receive (Read) Request Line	Pulse
MCSPi7_DMA_WRITE_EV ENT2	MCSPi7_TX2	PDMA1_MISC3	MCSPi7 Channel 2 Transmit (Write) Request Line	Pulse
MCSPi7_DMA_READ_EV ENT2	MCSPi7_RX2	PDMA1_MISC3	MCSPi7 Channel 2 Receive (Read) Request Line	Pulse
MCSPi7_DMA_WRITE_EV ENT3	MCSPi7_TX3	PDMA1_MISC3	MCSPi7 Channel 3 Transmit (Write) Request Line	Pulse
MCSPi7_DMA_READ_EV ENT3	MCSPi7_RX3	PDMA1_MISC3	MCSPi7 Channel 3 Receive (Read) Request Line	Pulse

### 12.1.5.4 MCSPI Functional Description

#### 12.1.5.4.1 SPI Block Diagram

Figure 12-69 shows the SPI module.



#### Note

For single channel operation ( $i=0$  or  $i=1$ ), MCSPI\_CHiCON[20]FORCE is asserted over the SPIENi and SPIENOi lines.

**Figure 12-69. SPI Block Diagram**

#### 12.1.5.4.2 MCSPI Reset

The MCSPI module can be reset either by hardware or by software reset. All configuration registers and all state machines are reset by the hardware reset signal (MCSPI\_RST). MCSPI can be reset by software through the MCSPI\_SYSCONFIG[1] SOFTRESET bit. This bit has the same impact on the module as the hardware reset signal. The only exception is that the MCSPI\_SYSCONFIG register is not affected by that software reset.

### 12.1.5.4.3 MCSPI Controller Mode

#### 12.1.5.4.3.1 Controller Mode Features

The MCSPI controller mode supports multichannel communication with up to four independent MCSPI communication channel contexts. The MCSPI initiates a data transfer on the data lines (SPIDAT[0] and SPIDAT[1]) and generates clock (SPICLK) and control (SPIEN[i]) signals.

Connected to multiple external devices, the MCSPI exchanges data with one MCSPI device at a time through two main modes (available in peripheral mode):

- Two-data-pins interface mode (transmit-and-receive mode for full-duplex transmission)
- Single-data-pin interface mode (recommended for half-duplex transmission)

---

#### Note

There is a fixed chip select line allocation in multichannel controller mode. Channel *i* is mapped to SPIEN[i].

---

Two DMA request events (read and write) allow synchronized accesses of the DMA controller with the activity of MCSPI.

Three interrupt events can be used for data transmission and reception in controller mode (for more information about interrupts, see [Section 13.1.3.4.7.1, Interrupt Events in Controller Mode](#)).

#### 12.1.5.4.3.2 Controller Transmit-and-Receive Mode (Full Duplex)

In full-duplex transmission, data is transmitted (shifted out serially on SPIDAT[0]) and received (shifted in serially on SPIDAT[1]) simultaneously on separate data lines.

The controller transmit-and-receive mode is programmable per channel (the MCSPI\_CHCONF\_0/1/2/3[13-12] TRM bit field).

Channel access to the shift registers for transmission/reception is based on the MCSPI\_TX\_0/1/2/3 transmitter register state, the MCSPI\_RX\_0/1/2/3 receiver register state, and round-robin arbitration.

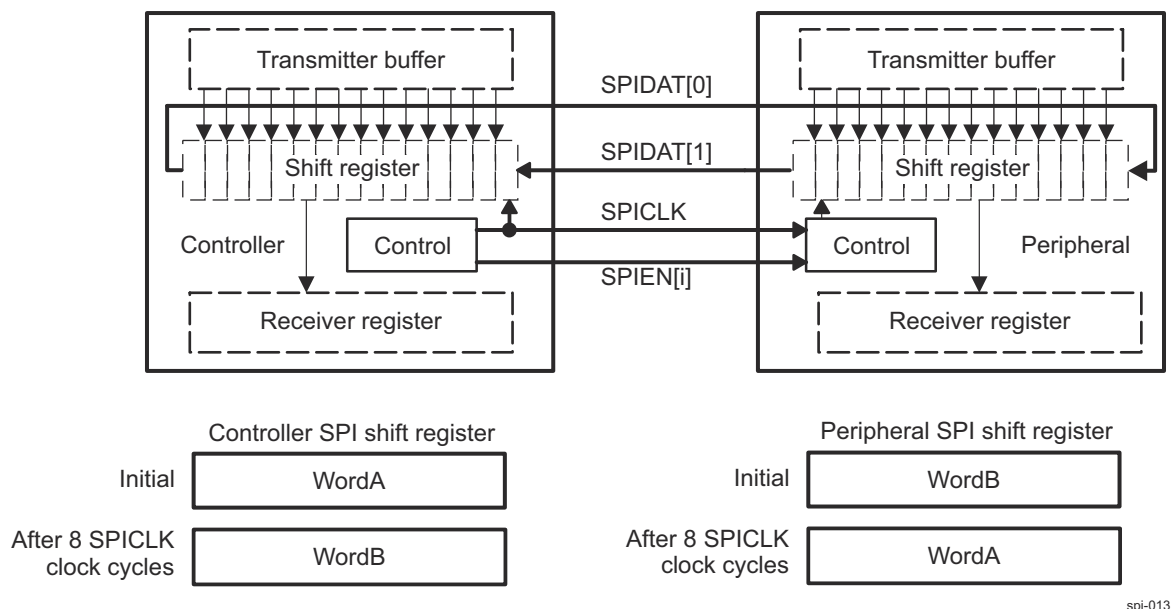
Channels that meet the following rules are included in the round-robin list of active channels scheduled for transmission and/or reception. The arbiter skips channels that do not meet the rules and searches in the rotation for the next enabled channel.

- Rule 1: Only enabled channels (the MCSPI\_CHCTRL\_0/1/2/3[0] EN bit) can be scheduled for transmission and/or reception.
- Rule 2: If its MCSPI\_TX\_0/1/2/3 transmitter register is not empty (the MCSPI\_CHSTAT\_0/1/2/3[1] TXS bit), an enabled channel can be scheduled when the shift register is assigned. If the MCSPI\_TX\_0/1/2/3 register is empty when the shift register is assigned, the TXx\_UNDERFLOW event is activated, and the next enabled channel with new data to transmit is scheduled (see also transmit-only mode).
- Rule 3: An enabled channel can be scheduled if its receive register is not full (the MCSPI\_CHSTAT\_0/1/2/3[0] RXS bit) when the shift register is assigned (see also receive-only mode). Therefore, the MCSPI\_RX\_0/1/2/3 register cannot be overwritten. The MCSPI\_IRQSTATUS[3] RX0\_OVERFLOW bit is never set to this mode.

When MCSPI word transfer completes (the MCSPI\_CHSTAT\_0/1/2/3[2] EOT bit is set), the updated MCSPI\_TX\_0/1/2/3 register of the next scheduled channel is loaded into the shift register. The serialization (transmit-and-receive) starts depending on the channel communication configuration. When serialization completes, the received data transfers to the channel receive register.

The serial clock (SPICLK) synchronizes shifting and sampling of the information on the two serial data lines (SPIDAT[0] and SPIDAT[1]). Each time a bit transfers out from the controller, 1 bit transfers in from the peripheral.

[Figure 12-70](#) shows an example of a full-duplex system with a controller device on the left and a peripheral device on the right. After eight cycles of the serial clock SPICLK, WordA transfers from the controller to the peripheral. At the same time, WordB transfers from the peripheral to the controller.



**Figure 12-70. MCSPI Full-Duplex Transmission (Example)**

#### 12.1.5.4.3.3 Controller Transmit-Only Mode (Half Duplex)

The controller transmit-only mode prevents the processor from reading the MCSPI\_RX\_0/1/2/3 register (minimizing data movement) when only transmission is meaningful.

The controller transmit-only mode is programmable per channel (the MCSPI\_CHCONF\_0/1/2/3[13-12] TRM bit field). Transmission starts only after data is loaded into the MCSPI\_TX\_0/1/2/3 register.

Rule 1 and Rule 2, defined in [Section 12.1.5.4.3.2](#), apply in this mode.

Rule 3, defined in [Section 12.1.5.4.3.2](#), does not apply.

In controller transmit-only mode, the MCSPI\_RX\_0/1/2/3 register state FULL does not prevent transmission and the MCSPI\_RX\_0/1/2/3 register is always overwritten with the new MCSPI word. This event is not significant when only transmission is meaningful. Thus, the RX0\_OVERFLOW bit in the MCSPI\_IRQSTATUS register is never set in this mode.

The hardware automatically disables the RX\_FULL interrupt and the DMA read requests.

The transfer status is given by the MCSPI\_CHSTAT\_0/1/2/3[2] EOT bit.

#### 12.1.5.4.3.4 Controller Receive-Only Mode (Half Duplex)

The controller receive mode prevents the processor from refilling the MCSPI\_TX\_0/1/2/3 register (minimizing data movement) when only reception is meaningful.

The controller receive mode is programmable per channel (the MCSPI\_CHCONF\_0/1/2/3[13-12] TRM bit field).

The controller receive-only mode enables channel scheduling only on the empty state of the MCSPI\_RX\_0/1/2/3 register.

Rule 1 and Rule 3, defined in [Section 12.1.5.4.3.2](#), apply in this mode.

Rule 2, defined in [Section 12.1.5.4.3.2](#), does not apply.

In the controller receive-only mode, software must write dummy data to the MCSPI\_TX\_0/1/2/3 register. Only one dummy write is enough to receive any number of words from the peripheral. Software must ensure that the MCSPI\_TX\_0/1/2/3 register is always full (the TXx\_EMPTY bits of MCSPI\_IRQSTATUS) when receiving. The content of the MCSPI\_TX\_0/1/2/3 register is always loaded into the shift register when the shift



register is assigned. After writing the dummy data to the MCSPI\_TX\_0/1/2/3 register, the TXx\_EMPTY and TXx\_UNDERFLOW bits in the MCSPI\_IRQSTATUS register are never set in receive-only mode.

The MCSPI\_CHSTAT\_0/1/2/3[2] EOT bit gives the status of serialization. The RXx\_FULL bits of the MCSPI\_IRQSTATUS register are set when received data is loaded from the shift register to the corresponding MCSPI\_RX\_0/1/2/3 register. The MCSPI\_IRQSTATUS[3] RX0\_OVERFLOW bit is never set in this mode.

#### 12.1.5.4.3.5 Single-Channel Controller Mode

When the MCSPI is configured as a controller device with a single enabled channel (MCSPI\_MODULCTRL[2] MS = 0 and MCSPI\_MODULCTRL[0] SINGLE = 1), the assertion of the SPIEN[i] signal is optional depending on device connected to the controller. In 3-pin mode (MCSPI\_MODULCTRL[1] PIN34 = 1) the controller starts transmitting data when a write to the MCSPI\_TX\_0/1/2/3 register or the FIFO is performed. In 4-pin mode (MCSPI\_MODULCTRL[1] PIN34 = 0) the assertion and de-assertion of SPIEN[i] is controlled by software using the MCSPI\_CHCONF\_0/1/2/3[20] FORCE bit.

##### 12.1.5.4.3.5.1 Programming Tips When Switching to Another Channel

When a single channel is enabled and data transfer is ongoing:

- Wait for the MCSPI word transfer to complete (wait until the MCSPI\_CHSTAT\_0/1/2/3[2] EOT bit is set to 1) before disabling the current channel and enabling a different channel.
- Disable the current channel, and then enable the other channel.

##### 12.1.5.4.3.5.2 Force SPIEN[i] Mode

Continuous transfers are allowed manually by keeping the SPIEN[i] signal active for successive MCSPI words transfer. Several sequences (configuration/enable/disable of the channel) can be run without deactivating the SPIEN[i] line. This mode is supported by all channels and any controller sequence can be used (transmit-receive, transmit-only, receive-only).

Keeping the SPIEN[i] active mode is supported when:

- A single channel is used (with the MCSPI\_MODULCTRL[0] SINGLE bit set to 1).
- Transfer parameters are loaded in the configuration register of the appropriate channel (MCSPI\_CHCONF\_0/1/2/3).

The state of the SPIEN[i] signal is programmable:

- Writing 1 to the MCSPI\_CHCONF\_0/1/2/3[20] FORCE bit drives the SPIEN[i] line high when the MCSPI\_CHCONF\_0/1/2/3[6] EPOL bit is set to 0. SPIEN[i] is driven low when the MCSPI\_CHCONF\_0/1/2/3[6] EPOL bit is set to 1.
- Writing 0 to the MCSPI\_CHCONF\_0/1/2/3[20] FORCE bit drives the SPIEN[i] line low when the MCSPI\_CHCONF\_0/1/2/3[6] EPOL bit is set to 0. SPIEN[i] is driven high when the MCSPI\_CHCONF\_0/1/2/3[6] EPOL bit is set to 1.
- A single channel is enabled (the MCSPI\_CHCTRL\_0/1/2/3[0] EN bit is set to 1). The first enabled channel activates the SPIEN[i] line.

When the channel is enabled, the SPIEN[i] signal activates with the programmed polarity. As in the multichannel controller mode, the transfer start depends on the status of the MCSPI\_TX\_0/1/2/3 register (the MCSPI\_CHSTAT\_0/1/2/3[1] TXS bit), the status of the MCSPI\_RX\_0/1/2/3 register (the MCSPI\_CHSTAT\_0/1/2/3[1] RXS bit), and the defined mode (the MCSPI\_CHCONF\_0/1/2/3[13-12] TRM bit field) of the channel enabled.

The MCSPI\_CHSTAT\_0/1/2/3[2] EOT bit gives the transfer status of each MCSPI word. The RXx\_FULL bit in the MCSPI\_IRQSTATUS register is set when received data is loaded from the shift register to the MCSPI\_RX\_0/1/2/3 register.

A change in the configuration parameters is propagated directly on the MCSPI interface. If the SPIEN[i] signal is activated, ensure that the configuration is changed only between MCSPI words to avoid corrupting the current transfer.



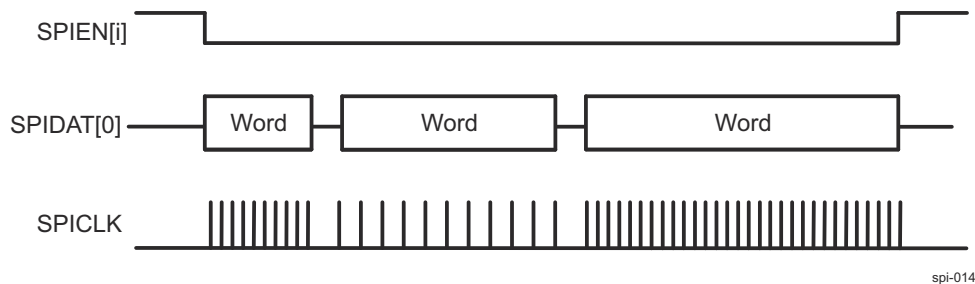
### Note

To avoid data corruption, SPIEN[i] polarity and SPICLK phase and SPICLK polarity must not be modified when the SPIEN[i] signal is activated.

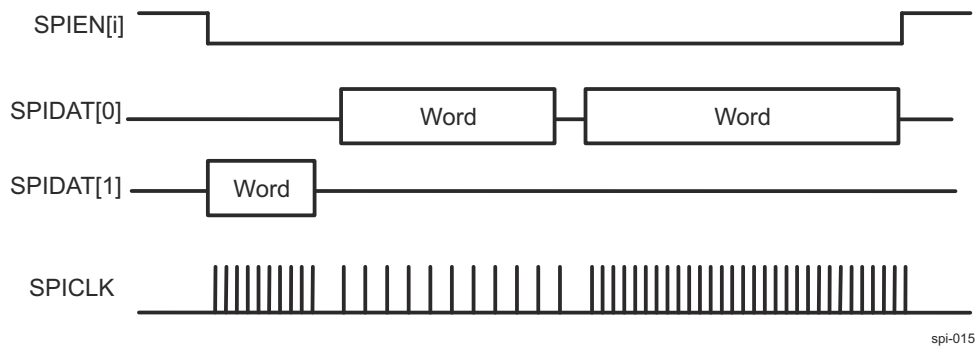
A delay between MCSPI words that requires the connected MCSPI peripheral device to switch from one configuration to another (for instance, from transmit-only to receive-only) must be handled by software.

At the end of the last MCSPI word, the channel must be deactivated (the MCSPI\_CHCTRL\_0/1/2/3[0] EN bit set to 0) and SPIEN[i] can be forced to its INACTIVE state using the MCSPI\_CHCONF\_0/1/2/3[20] FORCE bit.

Figure 12-71 and Figure 12-72 show successive transfers with SPIEN[i] maintained active low with a different configuration for each MCSPI word in single-data-pin and dual-data-pin interface modes, respectively.



**Figure 12-71. Continuous Transfers With SPIEN[i] Maintained Active (Single-Data-Pin Interface Mode)**



**Figure 12-72. Continuous Transfers With SPIEN[i] Maintained Active (Dual-Data-Pin Interface Mode)**

### Note

The SPIEN[i] signal can be maintained active via software using the MCSPI\_CHCONF\_0/1/2/3[20] FORCE bit only when the MCSPI\_MODULCTRL[0] SINGLE bit is set to 0x1.

#### 12.1.5.4.3.5.3 Turbo Mode

Turbo mode improves the throughput of the MCSPI interface when a single channel is enabled by allowing transfers until the shift register and the MCSPI\_RX\_0/1/2/3 register are full. Turbo mode is time saving when a transfer exceeds two words. This mode is programmable per channel (through the MCSPI\_CHCONF\_0/1/2/3[9] TURBO bit).

When several channels are enabled, the TURBO bit has no effect and the channel access to the shift registers remains as previously described.

In turbo mode, Rule 1 and Rule 2 apply, but Rule 3 does not (see [Section 12.1.5.4.3.2, Controller Transmit-and-Receive Mode \(Full Duplex\)](#)). An enabled channel can be scheduled if its receive register is full (the MCSPI\_CHSTAT\_0/1/2/3[0] RXS bit) when the shift-register is assigned until the shift register is full.

The MCSPI\_RX\_0/1/2/3 register cannot be overwritten in turbo mode. Consequently, the MCSPI\_IRQSTATUS[3] RX0\_OVERFLOW bit is never set in this mode.

#### 12.1.5.4.3.6 Start-Bit Mode

In start-bit mode, an extended bit is added before the MCSPI word to indicate whether the next MCSPI word must be handled as a command or as data. This feature is available only in controller mode. Start-bit mode cannot be used at the same time as turbo mode and/or force SPIEN[i] mode. In this case, only one channel can be used; round-robin arbitration is not possible.

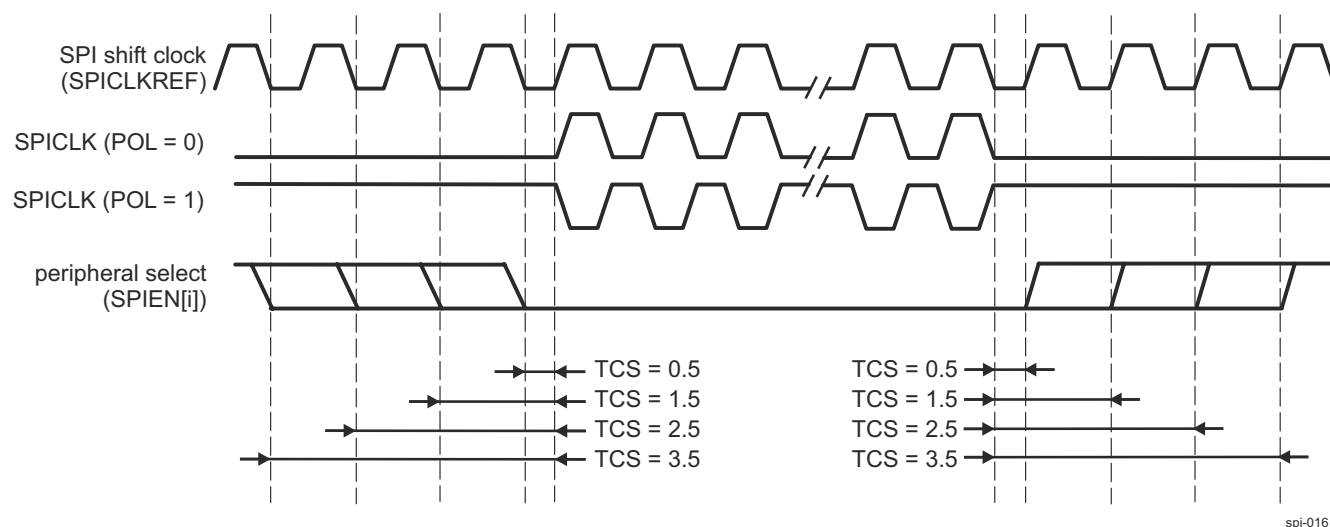
This mode is programmable per channel by setting the MCSPI\_CHCONF\_0/1/2/3[23] SBE bit to 1. The polarity of the extended bit is programmable per channel. When the MCSPI\_CHCONF\_0/1/2/3[24] SBPOL bit is set to 0, the MCSPI word must be handled as a command. When the MCSPI\_CHCONF\_0/1/2/3[24] SBPOL bit is set to 1, the MCSPI word must be handled as data. Moreover, start-bit polarity can be changed dynamically during start-bit transfer without disabling the channel for reconfiguration; in this case, users must configure the MCSPI\_CHCONF\_0/1/2/3[24] SBPOL bit before writing the MCSPI word to be transmitted to the TX register.

#### 12.1.5.4.3.7 Chip-Select Timing Control

The chip-select (CS) timing control is available only in controller mode with automatic CS generation (the MCSPI\_MODULCTRL[0] SINGLE bit set to 0) to add a programmable delay between CS assertion and first clock edge, or CS removal and last clock edge. This option is available only in 4-pin mode when MCSPI\_MODULCTRL[1] PIN34 set to 0.

This mode is programmable per channel through the MCSPI\_CHCONF\_0/1/2/3[26-25] TCS0 bit field.

Figure 12-73 shows the CS SPIEN timing controls.



**Figure 12-73. CS SPIEN Timing Controls**

#### Note

Because of the design implementation for transfers using a clock divider ratio set to 1 (clock bypassed), a half cycle must be added to the value between CS assertion and the first clock edge with PHA = 1 or between CS removal and the last clock edge with PHA = 0.

#### 12.1.5.4.3.8 Programmable MCSPI Clock (SPICLK)

In controller mode, the baud rate of the MCSPI serial clock is programmable.

An internal reference clock, SPICLKREF, is used as input of a programmable divider (the MCSPI\_CHCONF\_0/1/2/3[5-2] CLKD bit field) to generate the bit rate of the serial output clock SPICLK. [Table 12-68](#) summarizes the supported divisor values.

**Table 12-68. MCSPI Controller Clock Rates**

Divider	Clock Rate
1	50 MHz <sup>(1)</sup>
2	25 MHz <sup>(1)</sup>
4	12.5 MHz
8	6.25 MHz
16	3.125 MHz
32	1.5625 MHz
64	781.25 kHz
128	390 kHz <sup>(2)</sup>
256	195 kHz <sup>(2)</sup>
512	97.7 kHz <sup>(2)</sup>
1024	48.8 kHz <sup>(2)</sup>
2048	24.4 kHz <sup>(2)</sup>
4096	12.2 kHz <sup>(2)</sup>
8192 and higher: Division not supported	—

- (1) These frequencies are not necessarily supported by all MCSPI modules. For more information, see the *Timing Requirements and Switching Characteristics* chapter in the device-specific Data sheet.
- (2) Approximate Frequency

#### 12.1.5.4.3.8.1 Clock Ratio Granularity

By default, the clock division ratio is defined by the MCSPI\_CHCONF\_0/1/2/3[5-2] CLKD bit field with power-of-2 granularity leading to a clock division in the range 1 to 4096; in this case, the duty cycle is always 50 percent. With the MCSPI\_CHCONF\_0/1/2/3[29] CLKG bit, clock division granularity can be changed to one clock cycle; in that case the MCSPI\_CHCTRL\_0/1/2/3[15-8] EXTCLK bit field is concatenated with the MCSPI\_CHCONF\_0/1/2/3[5-2] CLKD bit field to give a 12-bit-wide division ratio in the range 1 to 4096.

When granularity is one clock cycle (the CLKG bit set to 1), for the odd value of the clock ratio, the clock high level lasts one clock cycle more than the low level, depending on the MCSPI\_CHCONF\_0/1/2/3[1] POL and MCSPI\_CHCONF\_0/1/2/3[0] PHA bits (see [Table 12-69](#)).

**Table 12-69. CLKSPPIO High/Low Time Computation**

Clock Ratio $F_{RATIO}$	CLKSPPIO High Time	CLKSPPIO Low Time
1	$T_{HIGH\_REF}$	$T_{LOW\_REF}$
Even $\geq 2$	$T_{ref} * (F_{RATIO}/2)$	$T_{ref} * (F_{RATIO}/2)$
Odd $\geq (POL = PHA)$	$T_{ref} * (F_{RATIO} - 1)/2$	$T_{ref} * (F_{RATIO} + 1)/2$
Odd $\geq (POL \neq PHA)$	$T_{ref} * (F_{RATIO} + 1)/2$	$T_{ref} * (F_{RATIO} - 1)/2$

#### Note

$F_{RATIO}$  = SPICLK frequency ( $F_{OUT}$ ) division ratio  
 $T_{HIGH}$  = SPICLK high time period  
 $T_{LOW}$  = SPICLK low time period  
 $T_{ref}$  = MCSPI\_FCLK period  
 $T_{HIGH\_REF}$  = MCSPI\_FCLK high time period  
 $T_{LOW\_REF}$  = MCSPI\_FCLK low time period

If the CLKG bit is set to 1;  $F_{RATIO}$  = EXTCLK concatenated with CLKD + 1.

For odd ratio values, the duty cycle is calculated as follows:

$$\text{Duty\_cycle} = (1 - 1/F_{\text{RATIO}})/2$$

Table 12-70 shows examples of clock granularity with a clock source frequency of 50 MHz.

**Table 12-70. Clock Granularity Examples**

EXTCLK	CLKD	CLKG	F <sub>RATIO</sub>	PHA	POL	T <sub>HIGH</sub> (ns)	T <sub>LOW</sub> (ns)	T <sub>PERIOD</sub> (ns)	Duty Cycle	F <sub>OUT</sub> (MHz)
X	0	0	1	X	X	10.0	10.0	20.0	50–50	50
X	1	0	2	X	X	20.0	20.0	40.0	50–50	25
X	2	0	4	X	X	40.0	40.0	80.0	50–50	12.5
X	3	0	8	X	X	80.0	80.0	160.0	50–50	6.2
0	0	1	1	X	X	10.0	10.0	20.0	50–50	50
0	1	1	2	X	X	20.0	20.0	40.0	50–50	25
0	2	1	3	1	0	40.0	20.0	60.0	66–33	16.6
0	2	1	3	1	1	20.0	40.0	60.0	33–66	16.6
0	3	1	4	X	X	40.0	40.0	80.0	50–50	12.5
5	0	1	81	1	0	820.0	800.0	1620.0	50.6–49.4	0.617
5	7	1	88	X	X	880.0	880.0	1760.0	50–50	0.568

#### 12.1.5.4.4 MCSPI Peripheral Mode

To select the MCSPI peripheral mode, set the MCSPI\_MODULCTRL[2] MS bit.

A MCSPI peripheral device can be connected to up to four external MCSPI controller devices but handles transactions with one MCSPI controller device at a time.

In peripheral mode, the MCSPI initiates data transfer on the data lines (SPIDAT[0] and SPIDAT[1]) when it is selected by an active control signal (SPIEN[i]) and receives an MCSPI clock (SPICLK) from the external MCSPI controller device. Only channel 0 can be configured as a peripheral but through the MCSPI\_CHCONF\_0[22-21] SPIENSLV bit field any of the SPIEN[i] signals can be used to select the MCSPI module. In peripheral mode and when the MCSPI\_MODULCTRL[1] PIN34 is set to 0x0 (default behaviour), the MCSPI uses the edge of SPIEN[i] to detect word length. For this reason, SPIEN[i] must become inactive between each word.

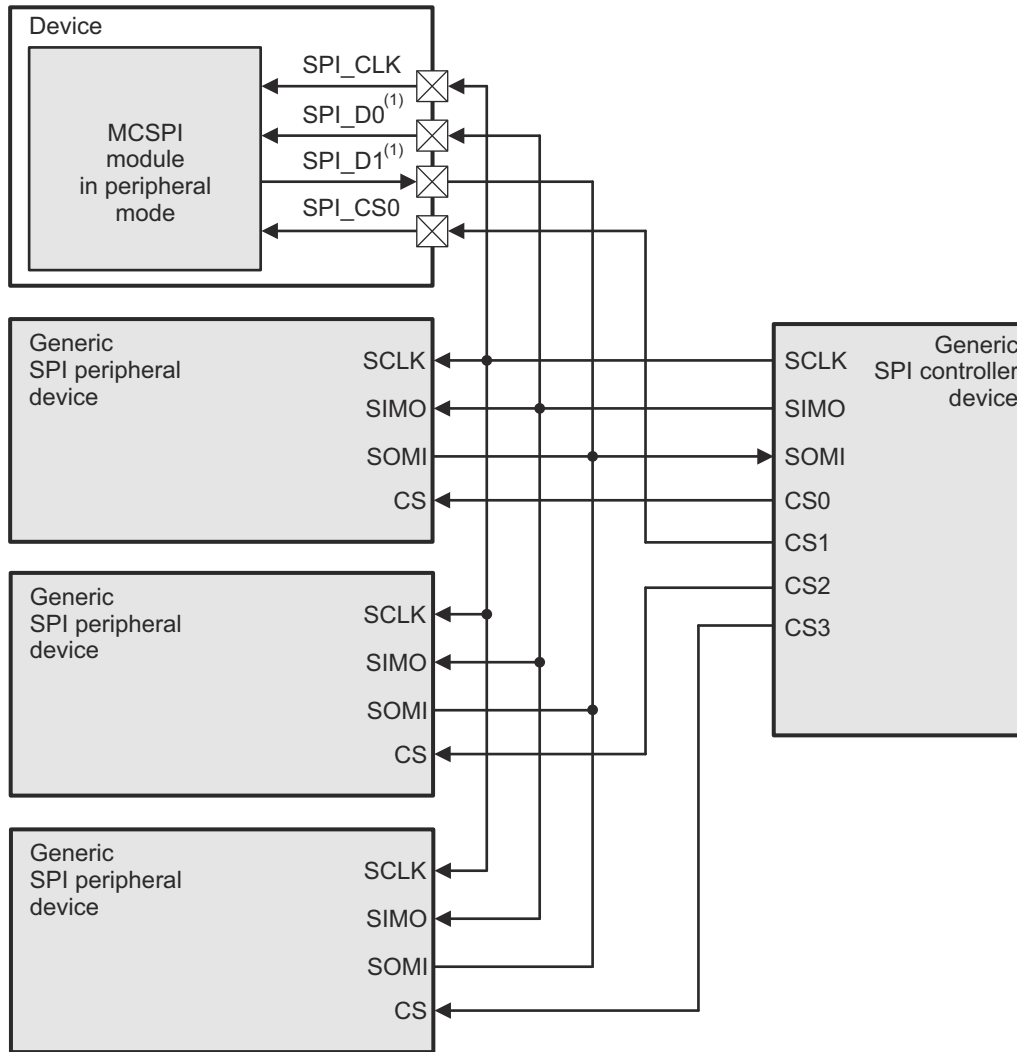
When the MCSPI\_MODULCTRL[1] PIN34 is set to 0x0, the MCSPI does not support SPIEN[i] active between MCSPI words. In this case, the MCSPI uses the edge to detect word length.

When the MCSPI\_MODULCTRL[1] PIN34 is set to 0x1, a multiword transfer can be performed without needing the external MCSPI controller to deactivate SPIEN[i] between each word as in this case the MCSPI module works in 3-pin peripheral mode and SPIEN[i] is not needed.

##### 12.1.5.4.4.1 Dedicated Resources

Only channel 0 can be enabled in peripheral mode.

Figure 12-74 shows an example of four peripherals wired on a single controller device.



spi-017

- A. Direction depends on MCSPI\_CHCONF\_0/1/2/3[16] DPE0, MCSPI\_CHCONF\_0/1/2/3[17] DPE1 and MCSPI\_CHCONF\_0/1/2/3[18] IS bits

**Figure 12-74. Example of MCSPI Peripheral With One Controller and Multiple Peripheral Devices on Channel 0**

Channel 0 in peripheral mode has the following resources:

- Its own channel enable, programmable with the MCSPI\_CHCTRL\_0[0] EN bit. This channel must be enabled before transmission and reception.
- For this mode, the peripheral-select signal can be detected on any of the SPIEN[i] ports. This is programmable with the MCSPI\_CHCONF\_0[22-21] SPIENSLV bit field.
- Its own transmitter register, MCSPI\_TX\_0, on top of the common transmit shift register. If the MCSPI\_TX\_0 register is empty, the MCSPI\_CHSTAT\_0[1] TXS bit is set. If MCSPI is selected by an external controller (the active signal on the SPIEN[i] port assigned to channel 0), the MCSPI\_TX\_0 register content of channel 0 is always loaded into the shift register, whether its content is updated or not. The MCSPI\_TX\_0 register must be loaded before MCSPI is selected by a controller.
- Its own receiver register, MCSPI\_RX\_0, on top of the common receive shift register. If the MCSPI\_RX\_0 register is full, the MCSPI\_CHSTAT\_0[0] RXS bit is set.

### Note

The MCSPI\_TX\_1/2/3 and MCSPI\_RX\_1/2/3 registers are not used. Reading from or writing to a channel register other than channel 0 has no effect.

- Its own communication configuration with the following parameters through the MCSPI\_CHCONF\_0:
  - Transmit and receive modes, programmable with the TRM field
  - Interface mode (two data pins or single data pin) and data pins assignment, both programmable with the IS and DPE bits. (The MCSPI modules are in peripheral mode after reset and must be properly configured for the modules to act in controller mode.)
  - MCSPI word length, programmable with the WL bit
  - SPIEN[i] polarity, programmable with the EPOL bit
  - SPICLK polarity, programmable with the POL bit
  - SPICLK phase, programmable with the PHA bit

The SPICLK frequency of a transfer is controlled by the external MCSPI controller connected to the MCSPI peripheral device. The MCSPI\_CHCONF\_0[5-2] CLKD bit field is not used in peripheral mode.

### Note

The configuration of the channel can be loaded in the MCSPI\_CHCONF\_0 only when the channel is disabled.

- Two DMA request events, read and write, synchronize read/write accesses of the DMA controller with the activity of MCSPI. DMA requests are asserted using the MCSPI\_CHCONF\_0[15] DMAR bit for reading and the MCSPI\_CHCONF\_0[14] DMAW bit for writing.
- Four interrupt events (see [Section 12.1.5.4.7.2, Interrupt Events in Peripheral Mode](#)).

#### 12.1.5.4.4.2 Peripheral Transmit-and-Receive Mode

The peripheral receive mode is programmable (set the MCSPI\_CHCONF\_0[13-12] TRM bit field to 0x0).

In peripheral transmit-and-receive mode, the MCSPI\_TX\_0 register must be loaded before MCSPI is selected by an external MCSPI controller device.

After a channel is enabled, transmission and reception proceed with interrupt and DMA request events.

The MCSPI\_TX\_0 register content is always loaded in the shift register whether it is updated or not. The event TX0\_UNDERFLOW is activated accordingly and does not prevent transmission.

When the MCSPI word transfer completes (the MCSPI\_CHSTAT\_0[2] EOT bit is set to 1), the received data is transferred to the channel receive register.

To use MCSPI as a peripheral transmit-only device, the RX0\_FULL and RX0\_OVERFLOW interrupts and DMA read requests must be disabled due to the state of the MCSPI\_RX\_0 register (see [Section 12.1.5.4.7.2, Interrupt Events in Peripheral Mode](#)).

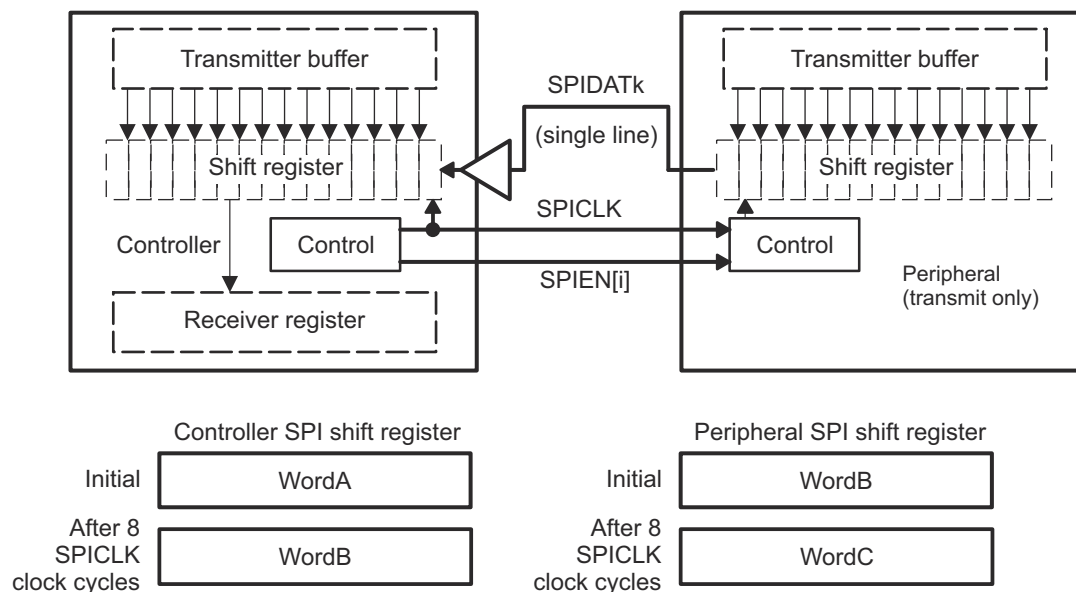
#### 12.1.5.4.4.3 Peripheral Transmit-Only Mode

The peripheral transmit-only mode is programmable (set the MCSPI\_CHCONF\_0[13-12] TRM bit field to 0x2) and avoids the requirement for the processor to read the MCSPI\_RX\_0 register (minimizing data movement) only when transmission is meaningful.

To use the MCSPI as a peripheral transmit-only device, the RX0\_FULL and RX0\_OVERFLOW interrupts and DMA read requests must be disabled due to the state of the MCSPI\_RX\_0 register.

When the MCSPI word transfer completes, the MCSPI\_CHSTAT\_0[2] EOT bit is set.

[Figure 12-75](#) shows a half-duplex system with a controller device on the left and a transmit-only peripheral device on the right. Each time a bit transfers out from the peripheral, 1 bit transfers in the controller. After eight cycles of the serial clock SPICLK, WordB transfers from the peripheral to the controller.



spi-018

k = 0 or 1 depending on MCSPI\_CHCONF\_0/1/2/3[16] DPE0, MCSPI\_CHCONF\_0/1/2/3[17] DPE1 and MCSPI\_CHCONF\_0/1/2/3[18] IS bits

**Figure 12-75. MCSPI Half-Duplex Transmission (Transmit-Only peripheral)**

#### 12.1.5.4.4.4 Peripheral Receive-Only Mode

The peripheral receive mode is programmable (set the MCSPI\_CHCONF\_0[13-12] TRM bit field to 0x1).

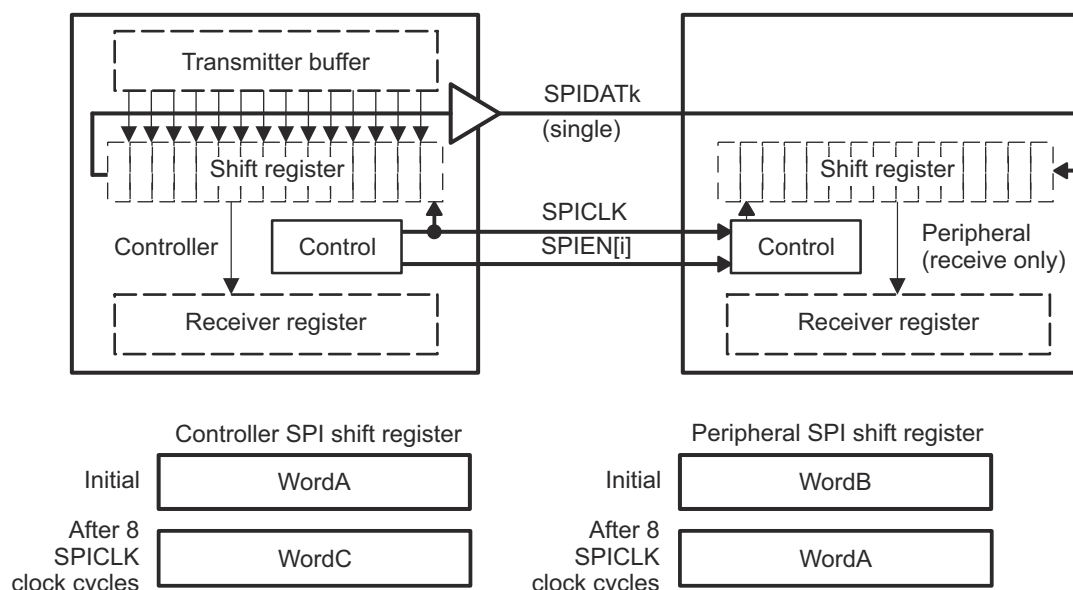
In receive-only mode, the MCSPI\_TX\_0 register must be loaded before the MCSPI is selected by an external MCSPI controller device. The MCSPI\_TX\_0 register content is always loaded into the shift register whether it is updated or not. The TX0\_UNDERFLOW event is activated accordingly and does not prevent transmission.

When the MCSPI word transfer completes (the MCSPI\_CHSTAT\_0[2] EOT bit is set to 1), the received data is transferred to the channel receive register.

To use the MCSPI as a peripheral receive-only device, the TX0\_EMPTY and TX0\_UNDERFLOW interrupts and the DMA write requests must be disabled due to the state of the MCSPI\_TX\_0 register.

For a full-duplex transmission, the serial clock (SPICLK) synchronizes shifting and sampling of the information on the two serial data lines. If SPICLK synchronizes on a single serial data line, the data line should be half-duplex.

Figure 12-76 shows a half-duplex system with a controller device on the left and a receive-only peripheral device on the right. Each time a bit transfers out from the controller, 1 bit transfers in from the peripheral. After eight cycles of the serial clock SPICLK, WordA transfers from the controller to the peripheral.



spi-019

k = 0 or 1 depending on MCSPI\_CHCONF\_0/1/2/3[16] DPE0, MCSPI\_CHCONF\_0/1/2/3[17] DPE1 and MCSPI\_CHCONF\_0/1/2/3[18] IS bits

**Figure 12-76. MCSPI Half-Duplex Transmission (Receive-Only Peripheral)**

#### 12.1.5.4.5 MCSPI 3-Pin or 4-Pin Mode

Depending on targeted application the MCSPI interface can be configured to use 3 or 4 pins through the MCSPI\_MODULCTRL[1] PIN34 bit. If this bit is set to 0, MCSPI is in 4-pin mode using the SPICLK, SPIDAT[0], SPIDAT[1] and SPIEN[i] signals. If PIN34 is set to 1 the controller is in 3-pin mode and SPIEN[i] is not used. In this mode all options related to chip select management are useless (EPOL, FORCE and TCS0 bits of MCSPI\_CHCONF\_0/1/2/3). 3-pin and 4-pin operation applies to both controller and peripheral modes.

#### 12.1.5.4.6 MCSPI FIFO Buffer Management

The MCSPI controller has a built-in 64-byte buffer to unload the DMA or interrupt handler and improve data throughput.

This buffer can be used by only one channel at a time and is selected by setting the MCSPI\_CHCONF\_0/1/2/3[28] FFER or MCSPI\_CHCONF\_0/1/2/3[27] FFEW bit to 1. If several channels are selected and several FIFO enable bit fields are set to 1, the controller forces the buffer not to be used; the driver must set only one FIFO enable bit field.

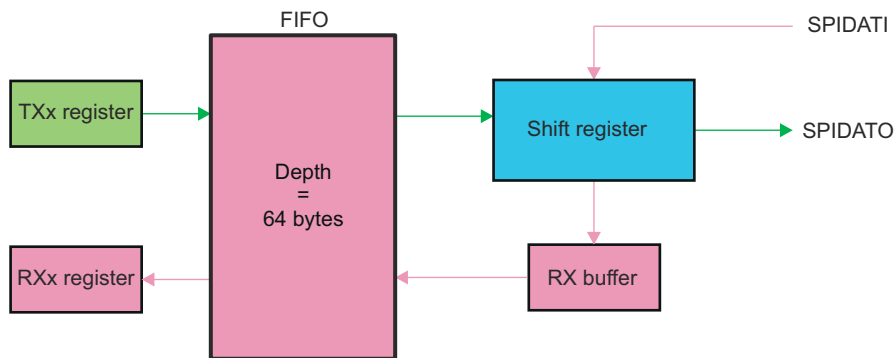
The buffer can be used in the following modes:

- Controller or peripheral mode
- Transmit-only, receive-only, or transmit-and-receive mode
- Single channel or turbo mode, or normal round-robin mode. In round-robin mode the buffer is used by only one channel.

Every word length (MCSPI\_CHCONF\_0/1/2/3[11-7] WL) is supported.

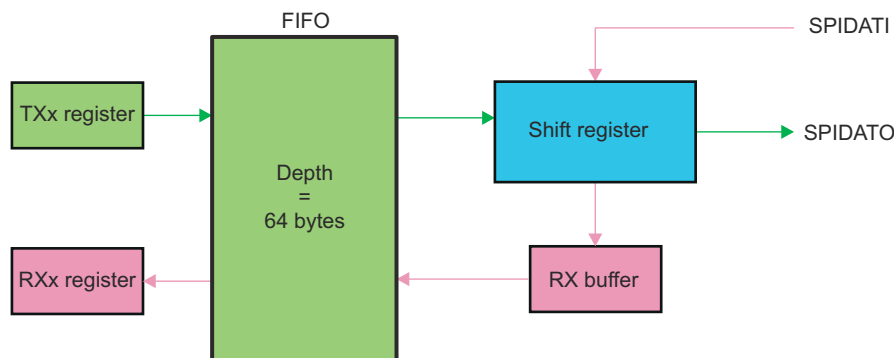
In transmit-and-receive mode, the buffer can be used in transmit (see [Figure 12-77](#)) or receive (see [Figure 12-78](#)) directions, or in both directions. If only one direction is chosen in transmit-and-receive mode, the full buffer is used for this direction. In both directions, the buffer is split into two halves, one for each direction (see [Figure 12-79](#)).





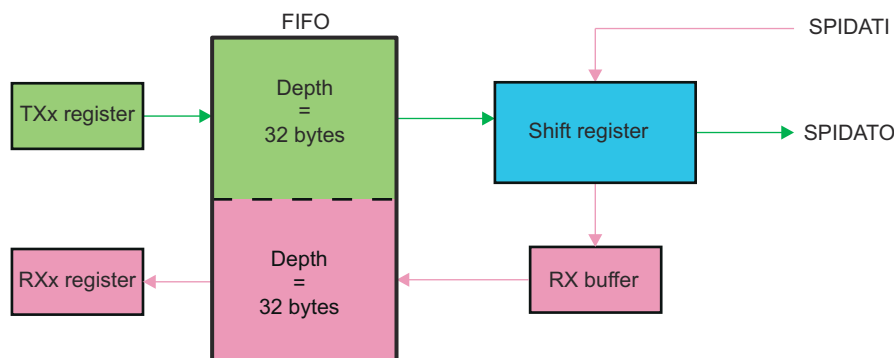
spi-020

**Figure 12-77. Buffer Used in Transmit Direction Only**



spi-021

**Figure 12-78. Buffer Used in Receive Direction Only**



spi-022

**Figure 12-79. Buffer Used for Transmit and Receive Directions**

Two levels (MCSPi\_XFERLEVEL[5-0] AEL and MCSPi\_XFERLEVEL[13-8] AFL) rule the buffer management. The granularity of these levels is 1 byte; it is not aligned with the MCSPi word length. The driver must set these values as a multiple of the MCSPi word length defined in WL. [Table 12-71](#) lists the number of bytes written in the FIFO, depending on the word length.

**Table 12-71. FIFO Writes, Word Length Relationship**

	MCSPi Word Length (WL)		
	$3 \leq WL \leq 7$	$8 \leq WL \leq 15$	$16 \leq WL \leq 31$
Number of bytes written in the FIFO	1 byte	2 bytes	4 bytes

The FIFO buffer pointers are reset when the corresponding channel is enabled or the FIFO configuration changes.

#### 12.1.5.4.6.1 Buffer Almost Full

The MCSPI\_XFERLEVEL[15-8] AFL bit field is needed when the buffer is used to receive an MCSPI word from a peripheral (the MCSPI\_CHCONF\_0/1/2/3[28] FFER bit must be set to 1). It defines the almost-full buffer status. See [Figure 12-80](#).

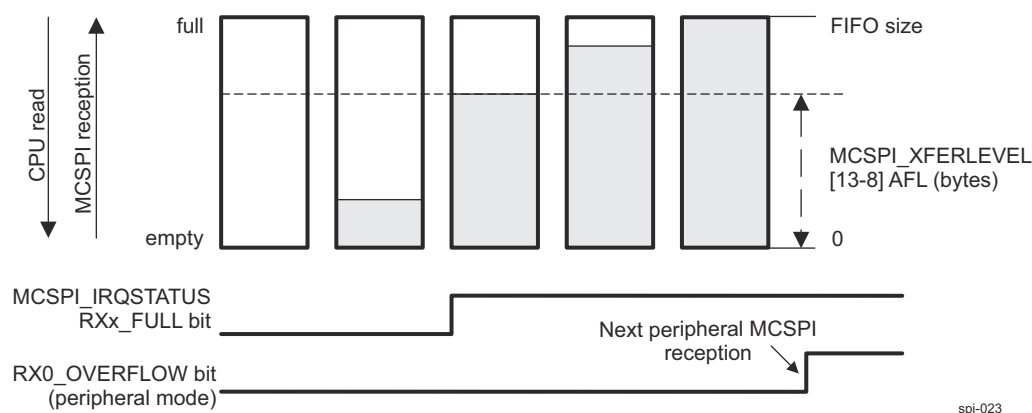
When the FIFO pointer reaches this level, an interrupt or a DMA request is sent to the processor to enable the system to read AFL + 1 bytes from the receive register.

#### Note

AFL + 1 must correspond to a multiple value of the MCSPI\_CHCONF\_0/1/2/3[11-7] WL bit field.

When DMA is used, the request is de-asserted after the first receive register read.

No new request is asserted again as long as the system has not performed the correct number of read accesses.



**Figure 12-80. Buffer Almost Full Level (AFL)**

#### Note

The MCSPI\_IRQSTATUS register bits are not available in DMA mode. In DMA mode, the MCSPI\_DMA\_READ\_EVENTi request is asserted on the same conditions as the MCSPI\_IRQSTATUS RXx\_FULL flag.

#### 12.1.5.4.6.2 Buffer Almost Empty

The MCSPI\_XFERLEVEL[7-0] AEL bit field is needed when the buffer is used to transmit an MCSPI word to a peripheral (the MCSPI\_CHCONF\_0/1/2/3[27] FFEW bit must be set to 1). It defines the almost-empty buffer status. See [Figure 12-81](#).

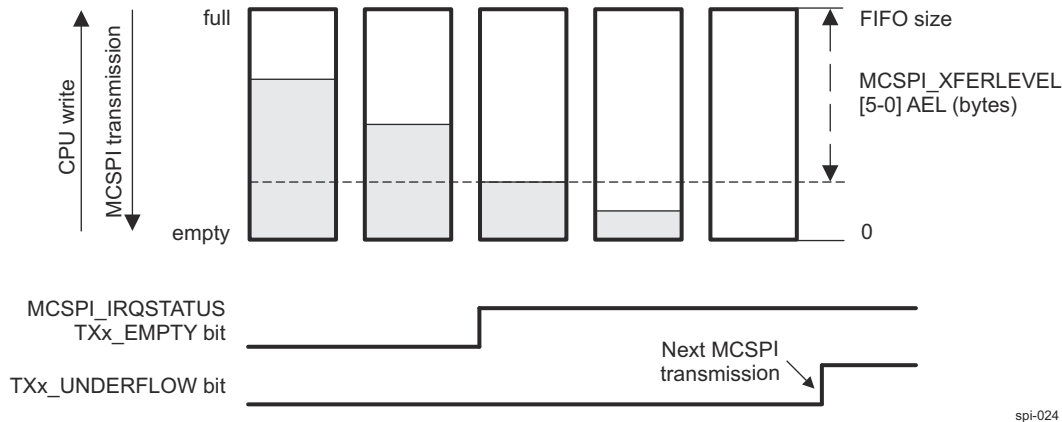
When the FIFO pointer does not reach this level, an interrupt or a DMA request is sent to the processor to enable the system to write AEL + 1 bytes to the transmit register.

#### Note

AEL + 1 must correspond to a multiple value of the MCSPI\_CHCONF\_0/1/2/3[11-7] WL bit field.

When DMA is used, the request is de-asserted after the first transmit register write.

No new request is asserted again as long as the system has not performed the correct number of write accesses.



**Figure 12-81. Buffer Almost Empty Level (AEL)**

#### Note

The MCSPI\_IRQSTATUS register bits are not available in DMA mode. In DMA mode, the MCSPI\_DMA\_WRITE\_EVENTi request is asserted on the same conditions as the MCSPI\_IRQSTATUS TXx\_EMPTY flag.

#### 12.1.5.4.6.3 End of Transfer Management

When the FIFO buffer is enabled for a channel, the user must previously configure in the MCSPI\_XFERLEVEL register the AEL and AFL levels and especially the MCSPI\_XFERLEVEL[31-16] WCNT bit field to define the number of MCSPI words to be transferred using the FIFO before enabling the channel.

This counter lets the controller stop the transfer correctly after a defined number of MCSPI word transfers. If WNCT is set to 0x0000, the counter is not used and the user must stop the transfer manually by disabling the channel; in this case, the user does not know how many MCSPI transfers have been done. For received words, software must poll the MCSPI\_CHSTAT\_i[5] RXFFE bit and read the MCSPI\_RX\_0/1/2/3 receive register to empty the FIFO buffer.

When the end-of-word count interrupt is generated (the MCSPI\_IRQSTATUS[17] EOW bit is set), the user can disable the channel and poll the MCSPI\_CHSTAT\_0/1/2/3[5] RXFFE bit to know the last MCSPI words in the FIFO buffer and read them.

#### 12.1.5.4.6.4 Multiple MCSPI Word Access

The processor has the ability to perform multiple MCSPI word access to the receive or transmit registers within a single 32-bit interface access by setting the MCSPI\_MODULCTRL[7] MOA to 1 under specific conditions:

- The channel selected has the FIFO enable.
- Only FIFO sense enabled support the kind of access.
- MCSPI\_MODULCTRL[7] MOA is set to 1.
- Only 32-bit interface access and data width can be performed to receive or transmit registers, for other kind of access the processor must de-assert MCSPI\_MODULCTRL[7] MOA bit.
- The level MCSPI\_XFERLEVEL[7-0] AEL and MCSPI\_XFERLEVEL[15-8] AFL must be 32-bit aligned, it means that AEL[0] = AEL[1] = 1 or AFL[0] = AFL[1] = 1.
- If MCSPI\_XFERLEVEL[31-16] WCNT is used it must be configured according to MCSPI word length.
- The word length of MCSPI words allows to perform multiple MCSPI access, that means that MCSPI\_CHCONF\_0/1/2/3[11-7] WL is <16.

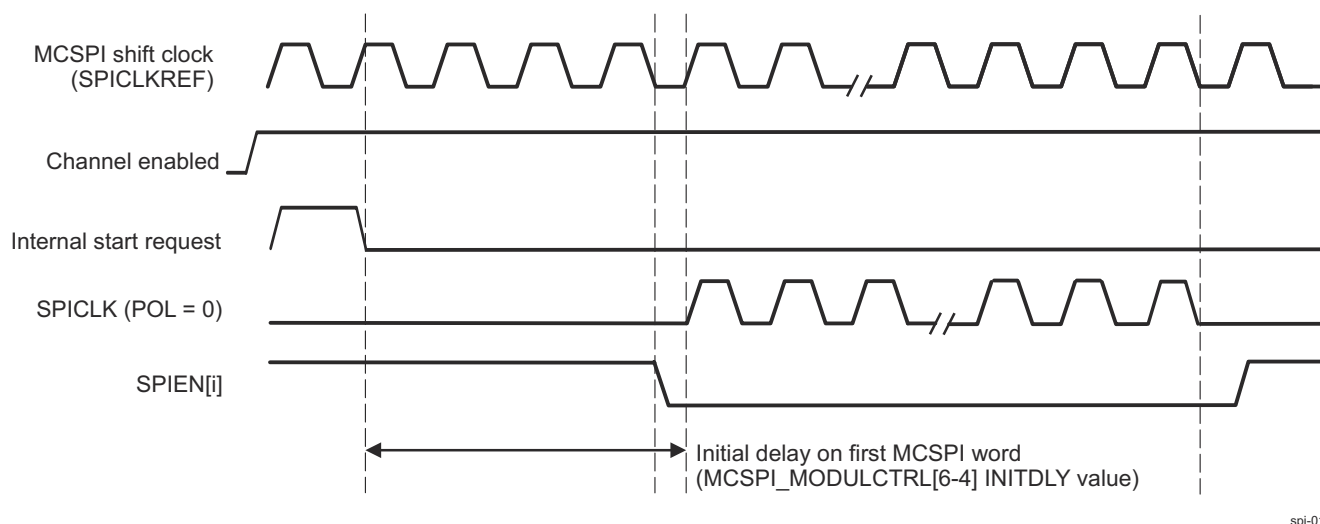
The number of MCSPI word access depends on MCSPI word length:

- $3 \leq WL \leq 7$ , MCSPI word length smaller or equal to byte length, 4 MCSPI words accessed per 32-bit interface read/write. If word count is used (MCSPI\_XFERLEVEL[31-16] WCNT), set the bit field to WCNT[0] = WCNT[1] = 0.
- $8 \leq WL \leq 15$ , MCSPI word length greater than byte or equal to 16-bit length, 2 MCSPI words accessed per 32-bit interface read/write. If word count is used (MCSPI\_XFERLEVEL[31-16] WCNT), set the bit field to WCNT[0] = 0.
- $16 \leq WL$  Multiple MCSPI word access is not applicable.

#### 12.1.5.4.6.5 First MCSPI Word Delay

Figure 12-82 shows the MCSPI controller ability to delay the first MCSPI word transfer to give time for system to complete some parallel processes or fill the FIFO in order to improve transfer bandwidth. This delay is applied only on first MCSPI word after MCSPI channel enabled and first write in transmit register. It is based on output clock frequency.

This option is meaningful in controller mode and single channel mode asserted through MCSPI\_MODULCTRL[0] SINGLE.



**Figure 12-82. Controller Single Channel Initial Delay**

Few delay values are available: No delay, 4/8/16/32 MCSPI cycles.

Its accuracy is half cycle in clock bypass mode and depends on clock polarity and phase.

#### 12.1.5.4.7 MCSPI Interrupts

Each channel can issue interrupt events.

Each interrupt event has status bits in the MCSPI\_IRQSTATUS register (RXx\_FULL, TXx\_UNDERFLOW, TXx\_EMPTY, etc.) (where x = 0, 3) that indicate whether service is required. Each status bit has an interrupt enable bit (a mask) in the MCSPI\_IRQENABLE register (RXx\_FULL\_ENABLE, TXx\_UNDERFLOW\_ENABLE, TXx\_EMPTY\_ENABLE, etc.).

When an interrupt occurs and a mask is later applied on it, the interrupt line is not asserted again, even if the interrupt source is not serviced.

The MCSPI supports interrupt-driven and polling operations.

##### 12.1.5.4.7.1 Interrupt Events in Controller Mode

In controller mode, the interrupt events related to the state of the MCSPI\_TX\_0/1/2/3 register are TXx\_EMPTY and TXx\_UNDERFLOW. The interrupt event related to the state of the MCSPI\_RX\_0/1/2/3 register is RXx\_FULL.

#### 12.1.5.4.7.1.1 TXx\_EMPTY

The TXx\_EMPTY event is activated when a channel is enabled and its MCSPI\_TX\_0/1/2/3 register is empty (transient event). Enabling a channel automatically triggers this event, except in controller receive-only mode (see [Section 12.1.5.4.3.4, Controller Receive-Only Mode](#)). When the FIFO buffer is enabled (the MCSPI\_CHCONF\_0/1/2/3[27] FFEW bit is set to 1), the MCSPI\_IRQSTATUS TXx\_EMPTY bit is set as soon as there is enough space in the buffer to write a number of bytes defined by the MCSPI\_XFERLEVEL[5-0] AEL bit field.

The MCSPI\_TX\_0/1/2/3 register must be loaded with data to remove the source of the interrupt; the MCSPI\_IRQSTATUS TXx\_EMPTY interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as the interrupt source).

When FIFO is enabled, no new TXx\_EMPTY event is asserted as long as the processor has not performed the number of writes into the MCSPI\_TX\_0/1/2/3 register defined by the MCSPI\_XFERLEVEL[5-0] AEL bit field. The processor must perform the correct number of writes.

#### 12.1.5.4.7.1.2 TXx\_UNDERFLOW

The event TXx\_UNDERFLOW is activated when the channel is enabled and if the MCSPI\_TX\_0/1/2/3 register or the FIFO is empty (not updated with new data) when an external controller device starts a data transfer with the MCSPI (transmit and receive).

The TXx\_UNDERFLOW is a harmless warning in controller mode.

To avoid having a TXx\_UNDERFLOW event at the beginning of a transmission, the TXx\_UNDERFLOW event is not activated when no data has been loaded into the MCSPI\_TX\_0/1/2/3 register, because the channel is enabled. To avoid having a TXx\_UNDERFLOW event, the MCSPI\_TX\_0/1/2/3 register must seldom be loaded.

The MCSPI\_IRQSTATUS TXx\_UNDERFLOW interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as the interrupt source).

#### 12.1.5.4.7.1.3 RXx\_FULL

The RXx\_FULL event is activated when a channel is enabled and the MCSPI\_RX\_0/1/2/3 register becomes filled (transient event). When the FIFO buffer is enabled (the MCSPI\_CHCONF\_0/1/2/3[28] FFER bit is set to 1), RXx\_FULL is asserted as soon as the number of bytes held in the FIFO to be read reaches the MCSPI\_XFERLEVEL[13-8] AFL threshold.

The MCSPI\_RX\_0/1/2/3 register must be read to remove the source of the interrupt; the MCSPI\_IRQSTATUS RXx\_FULL interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as the interrupt source).

When FIFO is enabled, no new RXx\_FULL event is asserted as long as the processor has not performed AFL + 1 reads into MCSPI\_RX\_0/1/2/3. The processor must perform the correct number of reads.

#### 12.1.5.4.7.1.4 End Of Word Count

The MCSPI\_IRQSTATUS[17] EOW event (end of word count) is activated when the channel is enabled and configured to use the built-in FIFO. This interrupt is raised when the controller performs the number of transfers defined in the MCSPI\_XFERLEVEL[31-16] WCNT bit field. If WCNT is set to 0x0000, the counter is not enabled and this interrupt is not generated.

The end of word count interrupt also indicates that the MCSPI transfer is halted on the channel using the FIFO buffer as soon as MCSPI\_XFERLEVEL[31-16] WCNT is not reloaded and the channel is not re-enabled.

The MCSPI\_IRQSTATUS[17] EOW interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as the interrupt source).

#### 12.1.5.4.7.2 Interrupt Events in Peripheral Mode

In peripheral mode, the interrupt events related to the state of the MCSPI\_TX\_0/1/2/3 register are TX0\_EMPTY and TX0\_UNDERFLOW. The interrupt events related to the state of the MCSPI\_RX\_0/1/2/3 are RX0\_FULL

and RX0\_OVERFLOW (channels 1, 2, and 3 do not have a receiver overflow status bit). See the MCSPI\_IRQSTATUS register.

#### 12.1.5.4.7.2.1 TXx\_EMPTY

The TXx\_EMPTY event is activated when a channel is enabled and its MCSPI\_TX\_0/1/2/3 register is empty. Enabling the channel automatically raises this event. If the FIFO buffer is enabled (the MCSPI\_CHCONF\_0/1/2/3[27] FFEW bit is set to 1), the TXx\_EMPTY event is asserted as soon as there is enough space in buffer to write a number of bytes defined by the MCSPI\_XFERLEVEL[5-0] AEL bit field.

The MCSPI\_TX\_0/1/2/3 register must be loaded with data to remove the source of the interrupt; the MCSPI\_IRQSTATUS TXx\_EMPTY interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as the interrupt source).

When FIFO is enabled, no new TXx\_EMPTY event is asserted as long as the processor has not performed the number of writes into the MCSPI\_TX\_0/1/2/3 register defined by MCSPI\_XFERLEVEL[5-0] AEL bit field. The processor must perform the correct number of writes.

#### 12.1.5.4.7.2.2 TXx\_UNDERFLOW

The TXx\_UNDERFLOW event is activated when a channel is enabled and if the MCSPI\_TX\_0/1/2/3 register is empty (not updated with new data) when an external controller device starts a data transfer with the MCSPI (transmit and receive).

When FIFO is enabled, the data emitted while the underflow event is raised is not the last data written in the FIFO but the old data in the FIFO (an old transmitted value or a dummy data in the FIFO has been reset).

TXx\_UNDERFLOW indicates an error (data loss) in peripheral mode.

To avoid having a TXx\_UNDERFLOW event at the beginning of a transmission, the TXx\_UNDERFLOW event is not activated when no data has been loaded into the MCSPI\_TX\_0/1/2/3 register because the channel is enabled.

The MCSPI\_IRQSTATUS TXx\_UNDERFLOW interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as the interrupt source).

#### 12.1.5.4.7.2.3 RXx\_FULL

The RXx\_FULL event is activated when a channel is enabled and the MCSPI\_RX\_0/1/2/3 register is being filled (transient event). When the FIFO buffer is enabled (the MCSPI\_CHCONF\_0/1/2/3[28] FFER bit is set to 1), RXx\_FULL is asserted as soon as the number of bytes held in the buffer to read defined by the MCSPI\_XFERLEVEL[13-8] AFL bit field.

The MCSPI\_RX\_0/1/2/3 register must be read to remove the source of the interrupt; the MCSPI\_IRQSTATUS RXx\_FULL interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as the interrupt source).

When FIFO is enabled, no new RXx\_FULL event is asserted as long as the processor has not performed AFL + 1 reads into MCSPI\_RX\_0/1/2/3. The processor must perform the correct number of reads.

#### 12.1.5.4.7.2.4 RX0\_OVERFLOW

The RX0\_OVERFLOW event is activated in peripheral mode in transmit-and-receive mode or receive-only mode when a channel is enabled and the MCSPI\_RX\_0/1/2/3 register or FIFO is full when a new MCSPI word is received. The MCSPI\_RX\_0/1/2/3 register is always overwritten with the new MCSPI word. If the FIFO is enabled, data within the FIFO are overwritten; it must be considered as corrupted. The RX0\_OVERFLOW event should not appear in peripheral mode using the FIFO.

The RX0\_OVERFLOW event indicates an error (data loss) in peripheral mode.

The MCSPI\_IRQSTATUS[3] RX0\_OVERFLOW interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as the interrupt source).

#### 12.1.5.4.7.2.5 End Of Word Count

The MCSPI\_IRQSTATUS[17] EOW event (end of word count) is activated when the channel is enabled and configured to use the built-in FIFO. This interrupt is raised when the controller performs the number of transfers defined in the MCSPI\_XFERLEVEL[31-16] WCNT bit field. If WCNT is set to 0x0000, the counter is not enabled and this interrupt is not generated.

The end of word count interrupt also indicates that the MCSPI transfer is halted on the channel using the FIFO buffer as soon as WCNT is not reloaded and the channel is not re-enabled.

The MCSPI\_IRQSTATUS[17] EOW interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as the interrupt source).

#### 12.1.5.4.7.3 Interrupt-Driven Operation

An interrupt enable bit in the MCSPI\_IRQENABLE register can be set to enable each event to generate interrupt requests when the corresponding event occurs. Status bits are automatically set by hardware logic conditions.

When an event occurs (the single interrupt line is asserted), the processor must:

1. Read the MCSPI\_IRQSTATUS register to identify which event occurred.
2. Read the MCSPI\_RX\_0/1/2/3 register that corresponds to the event to remove the source of an RXx\_FULL event or write into the MCSPI\_TX\_0/1/2/3 register that corresponds to the event to remove the source of a TXx\_EMPTY event. No action is required to remove the source of the TXx\_UNDERFLOW and RX0\_OVERFLOW events.
3. Set the corresponding bit of the MCSPI\_IRQSTATUS register to 1 to clear an interrupt status and then release the interrupt line.

The interrupt status bit must always be reset after channel enabling and before events are enabled as interrupt sources.

#### 12.1.5.4.7.4 Polling

When the interrupt capability of an event is disabled in the MCSPI\_IRQENABLE register, the interrupt line is not asserted, but the status bits in the MCSPI\_IRQSTATUS register can be polled by software to detect when the corresponding event occurs.

Once the expected event occurs:

- RXx\_FULL: To remove the source of the event, the processor must read the corresponding MCSPI\_RX\_0/1/2/3 register.
- TXx\_EMPTY: To remove the source of the event, the processor must write into the corresponding MCSPI\_TX\_0/1/2/3 register.
- TXx\_UNDERFLOW and RX0\_OVERFLOW: No action is required to remove the source of the event.

To clear an interrupt, set the corresponding status bit of the MCSPI\_IRQSTATUS register to 1. This does not affect the interrupt line state.

#### 12.1.5.4.8 MCSPI DMA Requests

Each MCSPI channel, if enabled, can issue DMA requests. There are two DMA request lines per MCSPI channel (one for read and one for write).

The DMA read request line is asserted when the MCSPI channel is enabled and new data is available in the receive register of the MCSPI channel. A DMA read request can be individually masked with the MCSPI\_CHCONF\_0/1/2/3[15] DMAR bit. The DMA read request line is de-asserted when reading of the MCSPI\_RX\_0/1/2/3 register of the MCSPI channel completes.

The DMA write request line is asserted when the MCSPI channel is enabled and the MCSPI\_TX\_0/1/2/3 register of the MCSPI channel is empty. A DMA write request can be individually masked with the MCSPI\_CHCONF\_0/1/2/3[14] DMAW bit. The DMA write request line is de-asserted when loading of the MCSPI\_TX\_0/1/2/3 register of the channel completes.



#### 12.1.5.4.9 MCSPI Power Saving Management

Power consumption can be optimized by switching off internal clocks (interface and functional clock) when there is no activity.

##### 12.1.5.4.9.1 Normal Mode

In normal mode, internal MCSPI module clocks are automatically switched off (autogated) when there is no activity in peripheral or controller mode.

Autogating of the module interface clock and functional clock occurs when the following conditions are met:

- The MCSPI\_SYSCONFIG[0] AUTOIDLE bit is set.
- In controller mode, there is no data to transmit or receive in all channels.
- In peripheral mode, the MCSPI is not selected by the external controller and there are no register accesses.

Autogating of the module interface clock and functional clock stops when the following conditions are met:

- In controller mode, an internal access occurs.
- In peripheral mode, an internal access occurs or the MCSPI is selected by the external controller.

##### 12.1.5.4.9.2 Idle Mode

#### Note

Some of the MCSPI features described in this section may not be supported on this family of devices. For more information, see *MCSPI Not Supported Features*.

At the power management level, when all conditions to shut off the MCSPI\_FCLK or MCSPI\_ICLK clocks are met, the corresponding LPSC asserts a clock stop request to the MCSPI. Although this procedure is completely hardware-oriented and out of software control, the method in which the MCSPI module acknowledges the clock stop request can be configured through the MCSPI\_SYSCONFIG[4-3] SIDLEMODE bit field.

The settings of the SIDLEMODE bit field and the related acknowledgement modes are:

- Force-idle mode (the MCSPI\_SYSCONFIG[4-3] SIDLEMODE bit field is set to 0x0): The MCSPI module acknowledges unconditionally the clock stop request, regardless of its internal operations. This mode must be used carefully in this case because it does not prevent the loss of data when the clock is switched off.
- No-idle mode (the SIDLEMODE bit field is set to 0x1): The MCSPI never acknowledges the clock stop request and is safe from a module point of view because it ensures that the clocks remain active. However, it is not efficient to save power because it does not allow shut off of MCSPI\_FCLK and MCSPI\_ICLK.
- Smart-idle mode (the SIDLEMODE bit field is set to 0x2): The MCSPI acknowledges the clock stop request, depending on its internal activity. MCSPI acknowledges the shut off of MCSPI\_FCLK and MCSPI\_ICLK only when all pending transactions, IRQs, or DMA requests are treated. This is the best approach for efficient system power management.

When configured in smart-idle mode, the MCSPI also offers an additional feature to control gating of MCSPI\_FCLK or MCSPI\_ICLK. The MCSPI\_SYSCONFIG[9-8] CLOCKACTIVITY bit field determines which clock shuts down (MCSPI\_FCLK, MCSPI\_ICLK, neither clock, or both clocks).

The setting of the CLOCKACTIVITY bit field is used internally to the MCSPI to determine on which part of the module the conditions to acknowledge the clock stop request are tested. For example, if MCSPI\_FCLK is not shut off on clock stop request, the MCSPI considers only MCSPI\_ICLK and the associated pending activities before acknowledging the request.

Some MCSPI features are associated with MCSPI\_ICLK and others with MCSPI\_FCLK. Using the CLOCKACTIVITY bit field with the smart-idle mode ensures that the features associated with the clock that remains active are always enabled, even if MCSPI acknowledges the clock stop request.

##### 12.1.5.4.9.2.1 Force-Idle Mode

Force-idle mode is enabled and exited as follows:



- Force-idle mode is enabled when the MCSPI\_SYSCONFIG[4-3] SIDLEMODE bit field is set to 0x0.
  - In force-idle mode, the MCSPI responds unconditionally to the clock stop request by de-asserting unconditionally the interrupt and DMA request lines, if asserted.
  - The transition from normal mode to idle mode does not affect the interrupt event bits of the MCSPI\_IRQSTATUS register.
  - In force-idle mode, because the module must be disabled, the interrupt and DMA request lines are likely de-asserted. The interface clock and MCSPI clock provided to the MCSPI can be switched off.
  - A clock stop request during an MCSPI data transfer can lead to an unexpected and unpredictable result. Software must avoid such a request.

### 12.1.5.5 MCSPI Programming Guide

This section describes the low-level hardware programming sequences for the configuration and use of the MCSPI module.

#### 12.1.5.5.1 MCSPI Global Initialization

##### 12.1.5.5.1.1 Surrounding Modules Global Initialization

This section identifies the requirements for initializing the surrounding modules when the MCSPI module is to be used for the first time after a device reset. This initialization of surrounding modules is based on the integration and environment of the MCSPI. For further information, see *MCSPI Integration* and *MCSPI Environment*.

[Table 12-72](#) lists the information on the global initialization of the surrounding modules.

**Table 12-72. Global Initialization of Surrounding Modules**

Surrounding Modules	Comments
COMPUTE_CLUSTER0	Device INTCs must be configured to enable the interrupt request generation. For information about enabling COMPUTE_CLUSTER0 interrupts, see <i>Interrupts</i> .
R5FSS0/1	Device INTCs must be configured to enable the interrupt request generation. For information about enabling R5FSS0 and R5FSS1 interrupts, see <i>Interrupts</i> .
R5FSS0/1_INTRTR0	Device INTCs must be configured to enable the interrupt request generation. For information about enabling R5FSS0_INTRTR0 and R5FSS1_INTRTR0 interrupts, see <i>Interrupts</i> .
MCU_R5FSS0_CORE0/1	Device INTCs must be configured to enable the interrupt request generation. For information about enabling MCU_R5FSS0_CORE0 and MCU_R5FSS0_CORE1 interrupts, see <i>Interrupts</i> .
C66SS0/1_INTRTR0	Device INTCs must be configured to enable the interrupt request generation. For information about enabling C66SS0_INTRTR0 and C66SS1_INTRTR0 interrupts, see <i>Interrupts</i> .
PRU-ICSSG0/1	Device INTCs must be configured to enable the interrupt request generation. For information about enabling PRU-ICSSG0 and PRU-ICSSG1 interrupts, see <i>Interrupts</i> .
MAIN2MCU_LVL_INTRTR0	Device INTCs must be configured to enable the interrupt request generation. For information about enabling MAIN2MCU_LVL_INTRTR0 interrupts, see <i>Interrupts</i> .
MCU_PDMA_MISC0 and MCU_PDMA_MISC1; PDMA1_MISC0, PDMA1_MISC1, PDMA1_MISC2, and PDMA1_MISC3	Device INTCs must be configured to enable the interrupt request generation.
WKUP_PLLCTRL0 and MCU_PLL2; PLLCTRL0 and MAIN_PLL0_HSDIV5	PLL controller's configuration must be done to enable the module clocks. For more information, see <i>Clocking</i> .

#### 12.1.5.5.1.2 MCSPI Global Initialization

##### 12.1.5.5.1.2.1 Main Sequence – MCSPI Global Initialization

The procedure in [Table 12-73](#) can be used to initialize MCSPI when performing software reset.

**Table 12-73. MCSPI Global Initialization**

Step	Register/Bit Field/Programming Model	Value
Perform a software reset.	MCSPI_SYSCONFIG[1] SOFTRESET	1
Wait until reset is finished?	MCSPI_SYSSTATUS[0] RESETDONE	=1
Configure static settings (such as SPI controller or peripheral) as required.	MCSPI_MODULCTRL[8-0]	0x-
Write MCSPI_SYSCONFIG	MCSPI_SYSCONFIG	0x-

#### 12.1.5.5.2 MCSPI Operational Mode Configuration

##### 12.1.5.5.2.1 MCSPI Operational Modes

The selection of the working mode is done with the MCSPI\_CHCONF\_0/1/2/3 register.

**Table 12-74. MCSPI Receive Mode Initialization**

Step	Register/Bit Field/Programming Model	Value
Set receive mode for the channel.	MCSPI_CHCONF_0/1/2/3[13-12] TRM	0x1

**Table 12-74. MCSPI Receive Mode Initialization (continued)**

Step	Register/Bit Field/Programming Model	Value
Configure SPI clock polarity/phase, clock divider, word length, and others for the channel.	MCSPi_CHCONF_0/1/2/3	0x-
Reset the status bits.	MCSPi_IRQSTATUS	0x0

**Table 12-75. MCSPI Transmit Mode Initialization**

Step	Register/Bit Field/Programming Model	Value
Set transmit mode for the channel.	MCSPi_CHCONF_0/1/2/3[13-12] TRM	0x2
Configure SPI clock polarity/phase, clock divider, word length, and others for the channel.	MCSPi_CHCONF_0/1/2/3	0x-
Reset the status bits.	MCSPi_IRQSTATUS	0x0

**Table 12-76. MCSPI Transmit-and-Receive Mode Initialization**

Step	Register/Bit Field/Programming Model	Value
Set transmit and receive mode for the channel.	MCSPi_CHCONF_0/1/2/3[13-12] TRM	0x0
Configure SPI clock polarity/phase, clock divider, word length, and others for the channel.	MCSPi_CHCONF_0/1/2/3	0x-
Reset the status bits.	MCSPi_IRQSTATUS	0x0

#### 12.1.5.5.2.1.1 Common Transfer Sequence

MCSPi module allows the transfer of one or several words, according to different modes:

- CONTROLLER Normal, CONTROLLER Turbo, PERIPHERAL
- TRANSMIT-RECEIVE, TRANSMIT-ONLY, RECEIVE-ONLY
- Write and Read requests: Interrupts, DMA
- SPIEN[i] lines assertion/deassertion: automatic, manual

For all these sequences, the host process contains the main process and the interrupt routines.

The interrupt routines are called on the interrupt signals or by an internal call if the module is used in polling mode.

[Table 12-77](#) represents the main sequence which is common to all transfers.

In multi-channel controller mode, the sequences of different channels can be run simultaneously.

**Table 12-77. Common Transfer Sequence (Main Process)**

Step	Register/Bit Field/Programming Model	Value
Write MCSPi_IRQSTATUS to reset channel status bits	MCSPi_IRQSTATUS[channel i bits]	0b1111
Write MCSPi_IRQENABLE to enable interrupts	MCSPi_IRQENABLE	0x-
Write MCSPi_CHCONF_0/1/2/3 to configure the channel	MCSPi_CHCONF_0/1/2/3	0x-
Start the channel	MCSPi_CHCTRL_0/1/2/3[0] EN	1
Wait for the first write request (TX empty or DMA write)		
Write the transmitter register with data	MCSPi_TX_0/1/2/3	0x-
Wait for the host event for end of transfer		
Stop the channel	MCSPi_CHCTRL_0/1/2/3[0] EN	0

#### 12.1.5.5.2.1.2 End of Transfer Sequences

The end of transfer depends on the transfer mode. [Table 12-78](#) summarizes the type of end of transfer per transfer mode and gives a reference to the appropriate section for details.

**Table 12-78. End of Transfer Sequences**

		TRANSMIT-AND-RECEIVE		TRANSMIT-ONLY		RECEIVE-ONLY	
		INTERRUPT	DMA	INTERRUPT	DMA	INTERRUPT	DMA
<b>CONTROL LER Normal</b>	End of transfer sequence	See <a href="#">Section 12.1.5.5.2.1.3</a>		See <a href="#">Section 12.1.5.5.2.1.4.1</a>	See <a href="#">Section 12.1.5.5.2.1.4.2</a>	See <a href="#">Section 12.1.5.5.2.1.5.1</a>	See <a href="#">Section 12.1.5.5.2.1.5.2</a>
	Minimum number of word	1	1	1	1	1	2
	DMA transfer size		N		N		N-1
<b>CONTROL LER Turbo</b>	End of transfer sequence	See <a href="#">Section 12.1.5.5.2.1.3</a>		See <a href="#">Section 12.1.5.5.2.1.4.1</a>	See <a href="#">Section 12.1.5.5.2.1.4.2</a>	See <a href="#">Section 12.1.5.5.2.1.6.1</a>	See <a href="#">Section 12.1.5.5.2.1.6.2</a>
	Minimum number of word	1	1	1	1	2	3
	DMA transfer size		N		N		N-2
<b>PERIPHER AL</b>	End of transfer sequence	See <a href="#">Section 12.1.5.5.2.1.3</a>		See <a href="#">Section 12.1.5.5.2.1.4.1</a>	See <a href="#">Section 12.1.5.5.2.1.4.2</a>	See <a href="#">Section 12.1.5.5.2.1.7</a>	
	Minimum number of word	1	1	1	1	1	1
	DMA transfer size		N		N	N	N

The transfer to execute has a size of N words.

The different sequences can be merged in one process to manage transfers of several types. The end of transfer sequences are described from the start of the channel.

In these sequences, some soft variables are used:

- write\_count = 0
- read\_count = 0
- channel\_enable = FALSE
- last\_transfer = FALSE
- last\_request = FALSE

They are initialized before starting the channel.

#### 12.1.5.5.2.1.3 Transmit-and-Receive (Controller and Peripheral)

If the requests are configured in DMA, write\_count and read\_count are assigned with 'N' when the DMA handlers have completed their 'N' CBASS0 accesses.

**Table 12-79. Transmit-and-Receive (Controller and Peripheral) (Main Process)**

Step	Register/Bit Field/Programming Model	Value
Start the channel	MCSPi_CHCTRL_0/1/2/3[0] EN	1
Wait for write_count = N <b>AND</b> read_count = N		
Stop the channel	MCSPi_CHCTRL_0/1/2/3[0] EN	0

**Table 12-80. Transmit-and-Receive (Controller and Peripheral) (Interrupt Routine)**

Step	Register/Bit Field/Programming Model	Value
Read MCSPi_IRQSTATUS	MCSPi_IRQSTATUS	0x-
Write MCSPi_IRQSTATUS to reset channel status bits	MCSPi_IRQSTATUS[channel i bits]	0b1111
<b>IF: TXx_EMPTY</b>		
Write the transmitter register with data	MCSPi_TX_0/1/2/3	0x-
Increment write_count +1		
<b>IF: RXx_FULL</b>		
Read the receiver register	MCSPi_RX_0/1/2/3	

**Table 12-80. Transmit-and-Receive (Controller and Peripheral) (Interrupt Routine) (continued)**

Step	Register/Bit Field/Programming Model	Value
Increment read_count +1		
<b>ENDIF</b>		

#### 12.1.5.5.2.1.4 Transmit-Only (Controller and Peripheral)

##### 12.1.5.5.2.1.4.1 Based on Interrupt Requests

**Table 12-81. Transmit-Only With Interrupts (Controller and Peripheral) (Main Process)**

Step	Register/Bit Field/Programming Model	Value
Start the channel	MCSPi_CHCTRL_0/1/2/3[0] EN	1
Wait until last_transfer = TRUE		
Wait for end of transfer	MCSPi_CHSTAT_0/1/2/3[2] EOT	=1
Stop the channel	MCSPi_CHCTRL_0/1/2/3[0] EN	0

**Table 12-82. Transmit-Only With Interrupts (Controller and Peripheral) (Interrupt Routine)**

Step	Register/Bit Field/Programming Model	Value
Read MCSPi_IRQSTATUS	MCSPi_IRQSTATUS	0x-
Write MCSPi_IRQSTATUS to reset channel status bits	MCSPi_IRQSTATUS[channel i bits]	0b1111
<b>IF: TXx_EMPTY AND write_count &lt; N</b>		
Write the transmitter register with data	MCSPi_TX_0/1/2/3	0x-
Increment write_count +1		
<b>ELSEIF: write_count ≥ N</b>		
last_transfer = TRUE		
<b>ENDIF</b>		

##### 12.1.5.5.2.1.4.2 Based on DMA Write Requests

When the DMA handler has completed its 'N' CBASS0 accesses, write\_count is assigned with 'N'.

**Table 12-83. Transmit-Only With DMA (Controller and Peripheral) (Main Process)**

Step	Register/Bit Field/Programming Model	Value
Start the channel	MCSPi_CHCTRL_0/1/2/3[0] EN	1
Wait until write_count = N		
Disable DMA write request	MCSPi_CHCONF_0/1/2/3[14] DMAW	0
Wait until last_transfer = TRUE		
Wait for end of transfer	MCSPi_CHSTAT_0/1/2/3[2] EOT	=1
Stop the channel	MCSPi_CHCTRL_0/1/2/3[0] EN	0

**Table 12-84. Transmit-Only With DMA (Controller and Peripheral) (Interrupt Routine)**

Step	Register/Bit Field/Programming Model	Value
Read MCSPi_IRQSTATUS	MCSPi_IRQSTATUS	0x-
Write MCSPi_IRQSTATUS to reset channel status bits	MCSPi_IRQSTATUS[channel i bits]	0b1111
<b>IF: TXx_EMPTY AND write_count = N</b>		
last_transfer = TRUE		
<b>ENDIF</b>		

### 12.1.5.5.2.1.5 Controller Normal Receive-Only

#### 12.1.5.5.2.1.5.1 Based on Interrupt Requests

**Table 12-85. Receive-Only With Interrupt (Controller Normal) (Main Process)**

Step	Register/Bit Field/Programming Model	Value
Start the channel	MCSPi_CHCTRL_0/1/2/3[0] EN	1
Wait until last_request = TRUE		
Stop the channel	MCSPi_CHCTRL_0/1/2/3[0] EN	0
Read the receiver register	MCSPi_RX_0/1/2/3	0x-
Increment read_count +1		

**Table 12-86. Receive-Only With Interrupt (Controller Normal) (Interrupt Routine)**

Step	Register/Bit Field/Programming Model	Value
Read MCSPi_IRQSTATUS	MCSPi_IRQSTATUS	0x-
Write MCSPi_IRQSTATUS to reset channel status bits	MCSPi_IRQSTATUS[channel i bits]	0b1111
<b>IF: RXx_FULL AND read_count = N - 1</b>		
last_request = TRUE		
<b>ELSEIF: read_count ≠ N - 1</b>		
Read the receiver register	MCSPi_RX_0/1/2/3	0x-
Increment read_count +1		
<b>ENDIF</b>		

#### 12.1.5.5.2.1.5.2 Based on DMA Read Requests

When the DMA handler has completed its 'N-1' CBASS0 accesses, read\_count is assigned with 'N-1'.

**Table 12-87. Receive-Only With DMA (Controller Normal) (Main Process)**

Step	Register/Bit Field/Programming Model	Value
Start the channel	MCSPi_CHCTRL_0/1/2/3[0] EN	1
Wait until read_count = N - 1		
Disable DMA read request	MCSPi_CHCONF_0/1/2/3[15] DMAR	0
Wait until last_transfer = TRUE		
Stop the channel	MCSPi_CHCTRL_0/1/2/3[0] EN	0
Read the receiver register	MCSPi_RX_0/1/2/3	0x-
Increment read_count +1		

**Table 12-88. Receive-Only With DMA (Controller Normal) (Interrupt Routine)**

Step	Register/Bit Field/Programming Model	Value
Read MCSPi_IRQSTATUS	MCSPi_IRQSTATUS	0x-
Write MCSPi_IRQSTATUS to reset channel status bits	MCSPi_IRQSTATUS[channel i bits]	0b1111
<b>IF: RXx_FULL AND read_count = N-1</b>		
last_transfer = TRUE		
<b>ENDIF</b>		

### 12.1.5.5.2.1.6 Controller Turbo Receive-Only

#### 12.1.5.5.2.1.6.1 Based on Interrupt Requests

**Table 12-89. Receive-Only With Interrupt (Controller Turbo) (Main Process)**

Step	Register/Bit Field/Programming Model	Value
Start the channel	MCSPi_CHCTRL_0/1/2/3[0] EN	1
Wait until channel_enable = TRUE		

**Table 12-89. Receive-Only With Interrupt (Controller Turbo) (Main Process) (continued)**

Step	Register/Bit Field/Programming Model	Value
Wait until last_transfer = TRUE		
Wait for end of transfer	MCSPi_CHSTAT_0/1/2/3[2] EOT	=1
Stop the channel	MCSPi_CHCTRL_0/1/2/3[0] EN	0
Wait until channel_enable = FALSE		

**Table 12-90. Receive-Only With Interrupt (Controller Turbo) (Interrupt Routine)**

Step	Register/Bit Field/Programming Model	Value
Read MCSPi_IRQSTATUS	MCSPi_IRQSTATUS	0x-
Write MCSPi_IRQSTATUS to reset channel status bits	MCSPi_IRQSTATUS[channel i bits]	0b1111
<b>IF: RXx_FULL</b>		
<b>IF: read_count = N - 2</b>		
last_transfer = TRUE		
channel_enable = FALSE		
<b>ENDIF</b>		
<b>IF: read_count &lt; N</b>		
Read the receiver register	MCSPi_RX_0/1/2/3	0x-
Increment read_count +1		
<b>ENDIF</b>		
<b>ENDIF</b>		

#### 12.1.5.5.2.1.6.2 Based on DMA Read Requests

When the DMA handler has completed its 'N-2' CBASS0 accesses read\_count is assigned with 'N-2'.

**Table 12-91. Receive-Only With DMA (Controller Turbo) (Main Process)**

Step	Register/Bit Field/Programming Model	Value
Start the channel	MCSPi_CHCTRL_0/1/2/3[0] EN	1
Wait until channel_enable = TRUE		
Wait until read_count = N-2		
Disable DMA read request	MCSPi_CHCONF_0/1/2/3[15] DMAR	0
Wait until last_transfer = TRUE		
Wait for end of transfer	MCSPi_CHSTAT_0/1/2/3[2] EOT	=1
Stop the channel	MCSPi_CHCTRL_0/1/2/3[0] EN	0
Wait until channel_enable = FALSE		

**Table 12-92. Receive-Only With DMA (Controller Turbo) (Interrupt Routine)**

Step	Register/Bit Field/Programming Model	Value
Read MCSPi_IRQSTATUS	MCSPi_IRQSTATUS	0x-
Write MCSPi_IRQSTATUS to reset channel status bits	MCSPi_IRQSTATUS[channel i bits]	0b1111
<b>IF: RXx_FULL</b>		
<b>IF: read_count = N - 2</b>		
last_transfer = TRUE		
channel_enable = FALSE		
<b>ENDIF</b>		
<b>IF: read_count &lt; N</b>		
Read the receiver register	MCSPi_RX_0/1/2/3	0x-
Increment read_count +1		
<b>ENDIF</b>		

**Table 12-92. Receive-Only With DMA (Controller Turbo) (Interrupt Routine) (continued)**

Step	Register/Bit Field/Programming Model	Value
<b>ENDIF</b>		

**12.1.5.5.2.1.7 Peripheral Receive-Only**

If the requests are configured in DMA, read\_count is assigned with 'N' when the DMA handler has completed its 'N' CBASS0 accesses.

**Table 12-93. Receive-Only (Peripheral) (Main Process)**

Step	Register/Bit Field/Programming Model	Value
Start the channel	MCSPi_CHCTRL_0/1/2/3[0] EN	1
Wait until read_count = N		
Stop the channel	MCSPi_CHCTRL_0/1/2/3[0] EN	0

**Table 12-94. Receive-Only (Peripheral) (Interrupt Routine)**

Step	Register/Bit Field/Programming Model	Value
Read MCSPi_IRQSTATUS	MCSPi_IRQSTATUS	0x-
Write MCSPi_IRQSTATUS to reset channel status bits	MCSPi_IRQSTATUS[channel i bits]	0b1111
<b>IF: RXx_FULL</b>		
Read the receiver register	MCSPi_RX_0/1/2/3	0x-
Increment read_count + 1		
<b>ENDIF</b>		

**12.1.5.5.2.1.8 Transfer Procedures With FIFO**

These flows describe the transfer with FIFO.

The MCSPi module allows the transfer of one or several words, according to different modes:

- CONTROLLER Normal, CONTROLLER Turbo, PERIPHERAL
- TRANSMIT–RECEIVE, TRANSMIT-ONLY, RECEIVE-ONLY
- Write and Read requests: IRQ, DMA

For all these flows, the host process contains the main process and the interrupt routine. This routine is called on the IRQ signals or by an internal call if the module is used in polling mode.

For more information, see [Section 12.1.5.4.6, MCSPi FIFO Buffer Management](#).

**Table 12-95. FIFO Mode Common Sequence (Controller) (Main Process)**

Step	Register/Bit Field/Programming Model	Value
Write MCSPi_IRQSTATUS to reset channel status bits	MCSPi_IRQSTATUS	1
Write MCSPi_IRQENABLE to enable interrupts	MCSPi_IRQENABLE	1
Write MCSPi_CHCONF_0/1/2/3 to configure the channel	MCSPi_CHCONF_0/1/2/3	0x-
Write MCSPi_XFERLEVEL	MCSPi_XFERLEVEL	0x-
Start the channel	MCSPi_CHCTRL_0/1/2/3[0] EN	1
<b>IF: Receive only</b>		
Wait for the write request (TX empty or DMA write)		
Write for the transmitter register with data	MCSPi_TX_0/1/2/3	0x-
<b>ENDIF</b>		
Wait for the host event for end of transfer		
Stop the channel	MCSPi_CHCTRL_0/1/2/3[0] EN	0



#### 12.1.5.5.2.1.8.1 Common Transfer Sequence in FIFO Mode

This flow describes the host sequence for a transfer of any type defined in [Section 12.1.5.5.2.1.8, Transfer Procedures With FIFO](#).

In multi-channel, only one channel can use the FIFO.

Before enabling the FIFO for a channel (MCSPI\_CHCONF\_0/1/2/3[28] FFER and MCSPI\_CHCONF\_0/1/2/3[27] FFEW bits), the host must check that the FIFO is not enabled for another channel, even if these channels are not used.

In transmit-and-receive mode, the FIFO can be enabled for write or read request only, without FIFO for the other request.

In Peripheral mode, the channel 0 only can be activated. The correct SPIEN line is chosen in MCSPI\_CHCONF\_0[22-21] SPIENSLV bits.

The MCSPI module can start the transfer only when the first write request has been released by writing the MCSPI\_TX\_0/1/2/3 register, even in receive-only mode (only one write request occurs in this case).

#### 12.1.5.5.2.1.8.2 End of Transfer Sequences in FIFO Mode

[Table 12-96](#) summarizes the type of end of transfer per transfer mode and gives a reference to the appropriate section for details.

**Table 12-96. End of Transfer Sequences in FIFO Mode**

Word count	TRANSMIT AND RECEIVE	TRANSMIT-ONLY	RECEIVE-ONLY
Yes	See <a href="#">Figure 12-83</a>	See <a href="#">Figure 12-85</a>	See <a href="#">Figure 12-86</a>
No	See <a href="#">Figure 12-84</a>	See <a href="#">Figure 12-85</a>	See <a href="#">Figure 12-87</a>

The end of transfer sequences are described from the start of the channel.

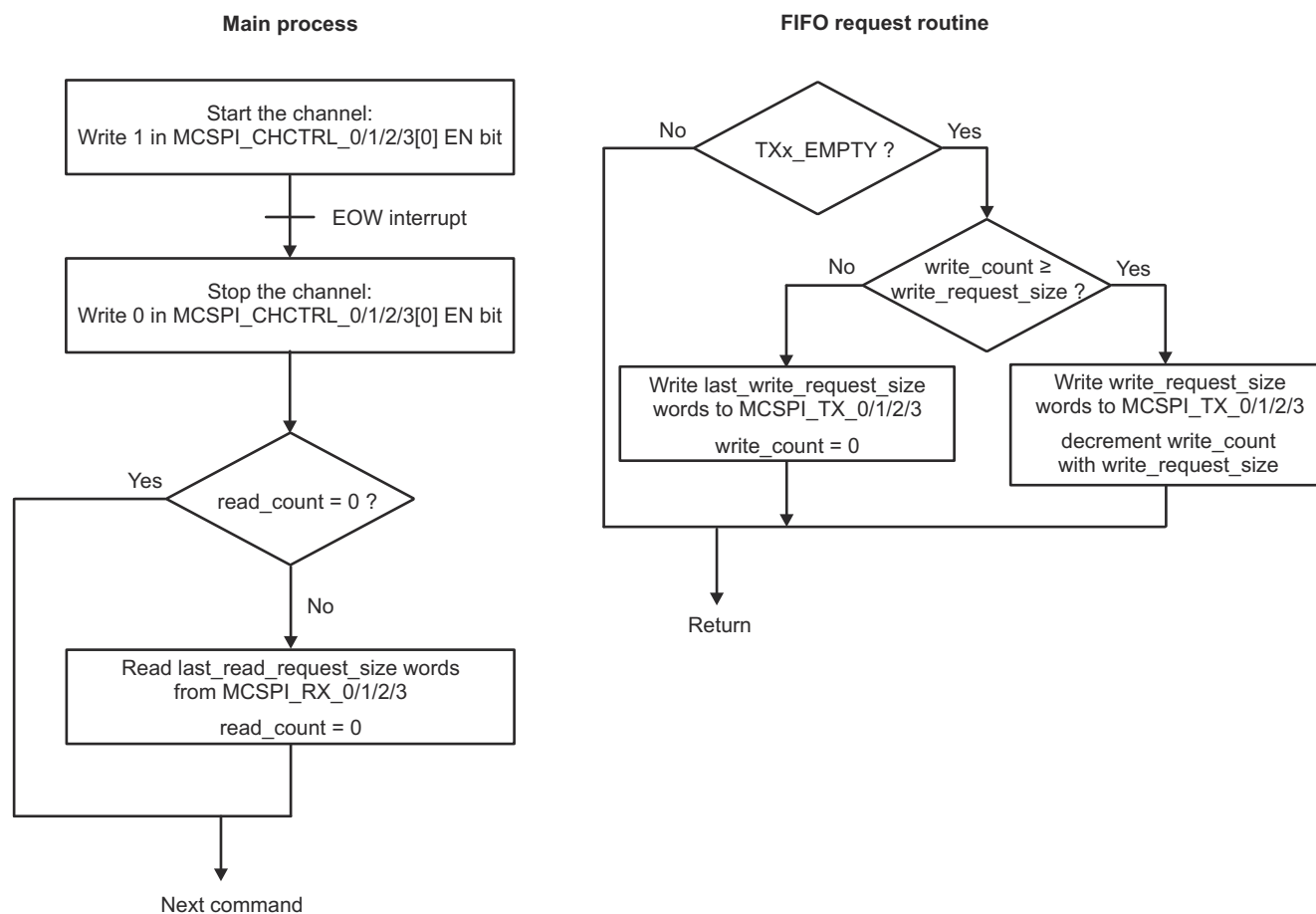
In these sequences, some soft variables are used:

- write\_count = N
- read\_count = N
- last\_request = FALSE

They are initialized before starting the channel.

#### 12.1.5.5.2.1.8.3 Transmit-and-Receive With Word Count

[Figure 12-83](#) shows the flow of a transfer in transmit-and-receive mode, with word count.

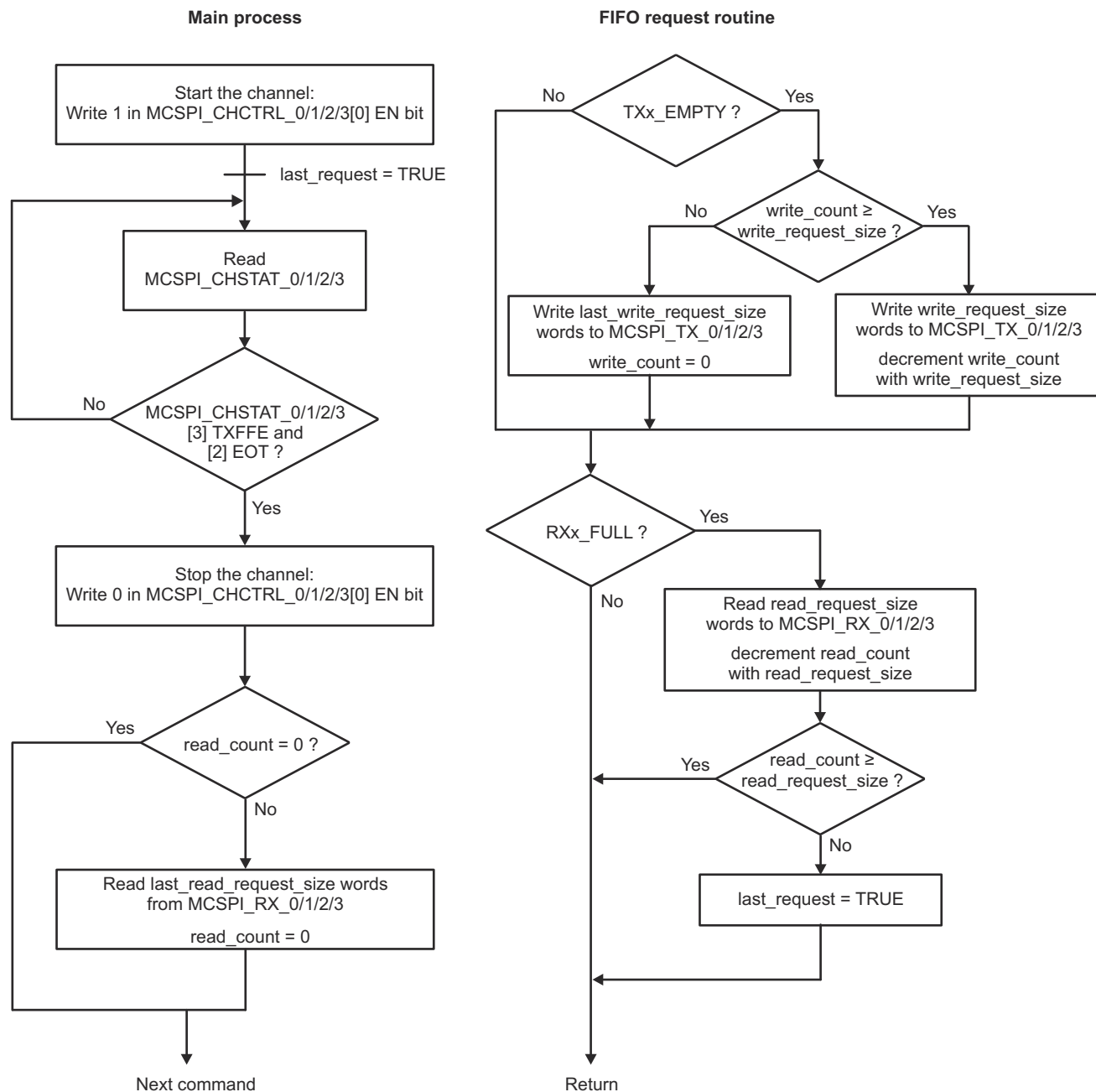


mcspi\_025

**Figure 12-83. FIFO Mode Transmit-and-Receive With Word Count (Controller)**

#### 12.1.5.5.2.1.8.4 Transmit-and-Receive Without Word Count

Figure 12-84 shows the flow of a transfer in transmit-and-receive mode, without word count.



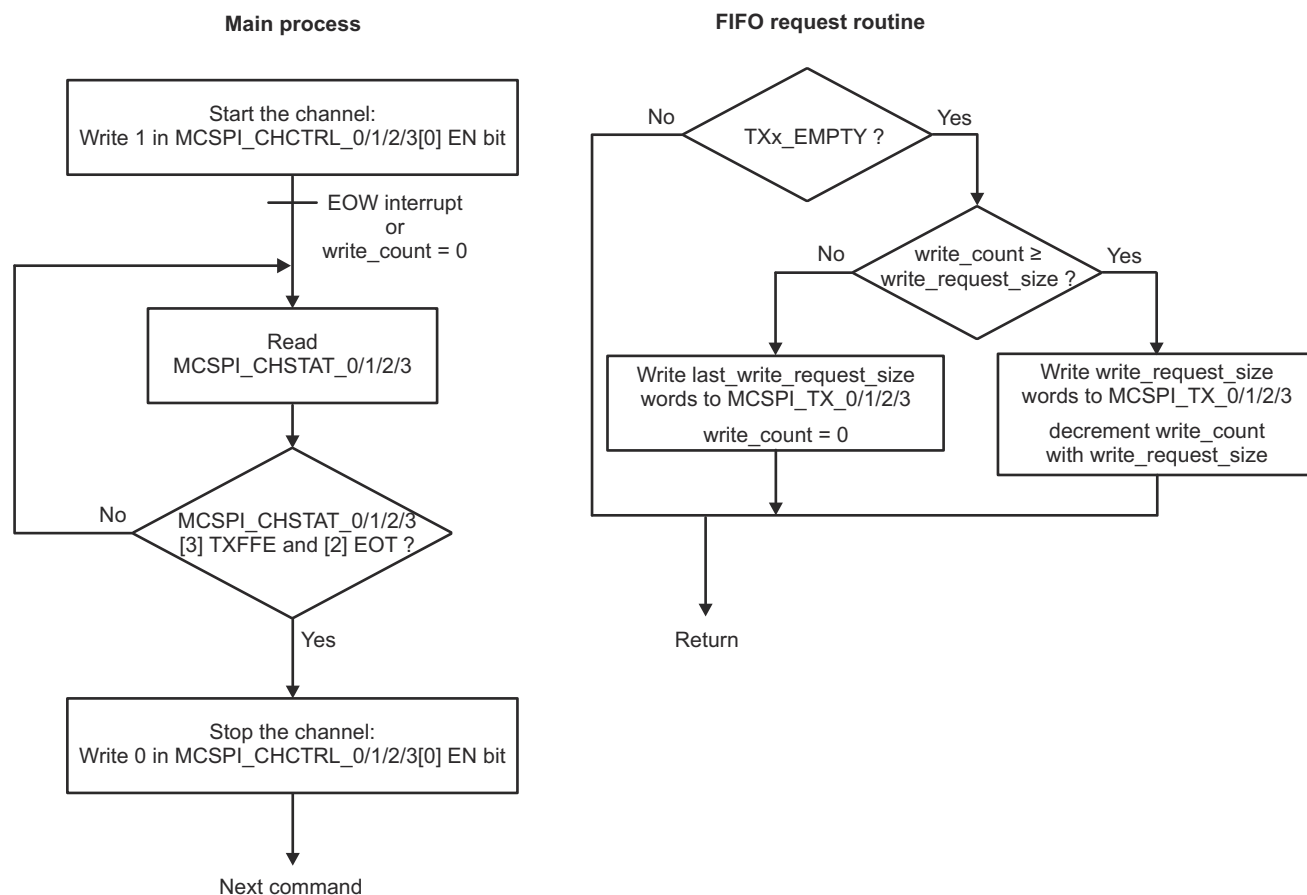
mcspi\_026

**Figure 12-84. FIFO Mode Transmit-and-Receive Without Word Count (Controller)**

#### 12.1.5.5.2.1.8.5 Transmit-Only

Figure 12-85 shows the flow of a transfer in transmit-only mode, with or without word count. The difference between word count enabled or not is just on the condition after starting the channel:

- word count enable: wait for EOW interrupt
- word count disable: wait for write\_count = 0

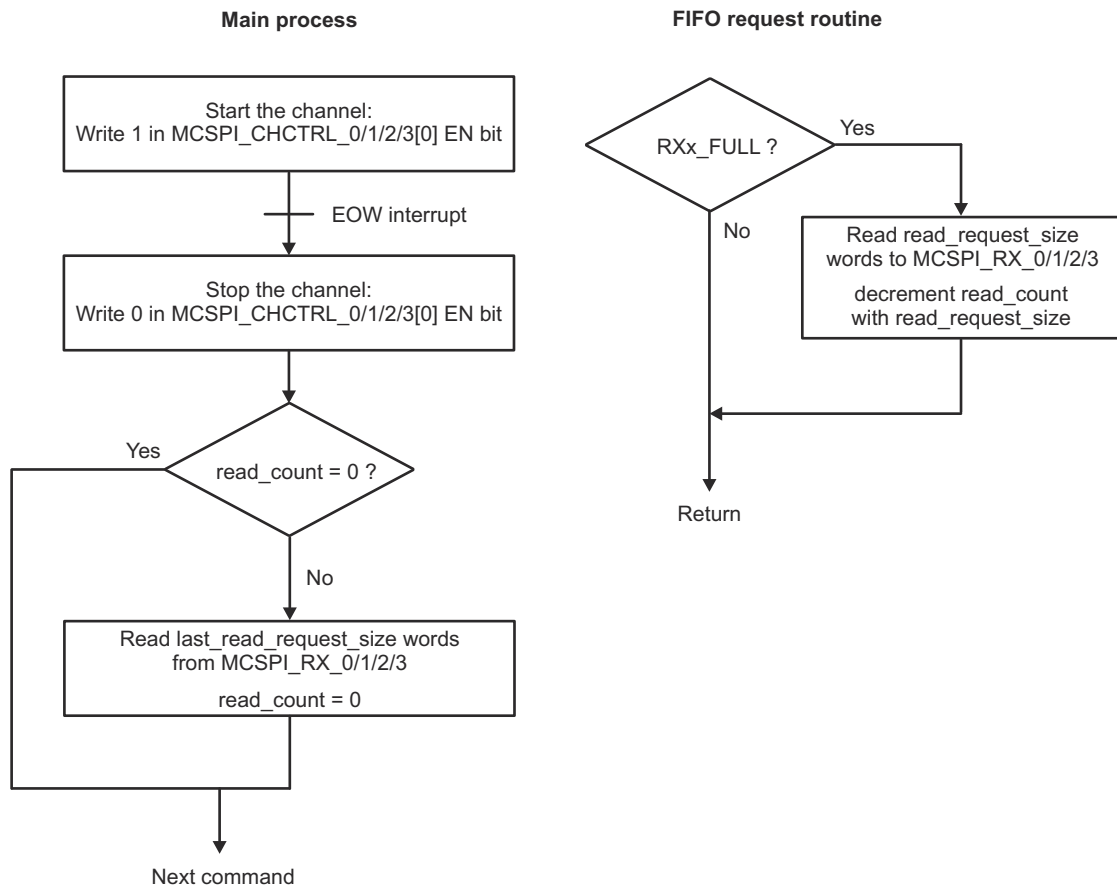


mcspi\_027

**Figure 12-85. FIFO Mode Transmit-Only (Controller)**

#### 12.1.5.5.2.1.8.6 Receive-Only With Word Count

Figure 12-86 shows the flow of a transfer in receive-only mode, with word count.

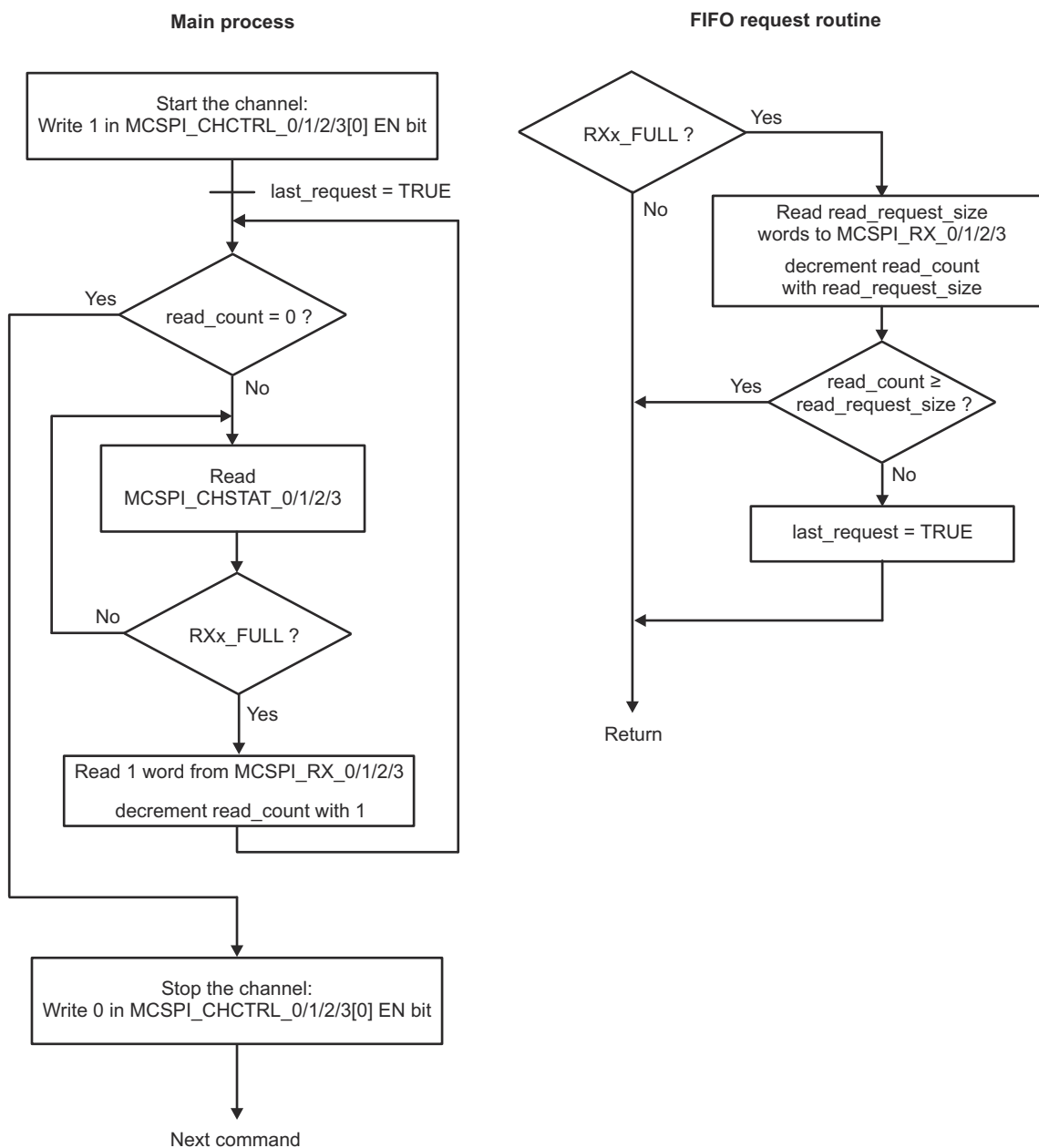


mcspi\_028

**Figure 12-86. FIFO Mode Receive-Only With Word Count (Controller)**

#### 12.1.5.5.2.1.8.7 Receive-Only Without Word Count

Figure 12-87 shows the flow of a transfer in receive-only mode, with word count.



mcspi\_029

**Figure 12-87. FIFO Mode Receive-Only Without Word Count (Controller)**

#### 12.1.5.5.2.1.9 Common Transfer Procedures Without FIFO – Polling Method

##### 12.1.5.5.2.1.9.1 Receive-Only Procedure – Polling Method

Table 12-97 lists the receive-only procedure using the polling method.

**Table 12-97. Receive-Only Procedure – Polling Method**

Step	Register/Bit Field/Programming Model	Value
Configure the channel according to the mode.	See Table 12-74.	
Start the channel.	MCSPI_CHCTRL_0/1/2/3[0] EN	1
Wait for end-of-transfer.	MCSPI_CHSTAT_0/1/2/3[2] EOT	=1
Read the receiver register.	MCSPI_RX_0/1/2/3	0x-

**Table 12-97. Receive-Only Procedure – Polling Method (continued)**

Step	Register/Bit Field/Programming Model	Value
Stop the channel if no more data is expected.	MCSPi_CHCTRL_0/1/2/3[0] EN	0

#### 12.1.5.5.2.1.9.2 Receive-Only Procedure – Interrupt Method

Table 12-98 lists the receive-only procedure using the interrupt method.

**Table 12-98. Receive-Only Procedure – Interrupt Method**

Step	Register/Bit Field/Programming Model	Value
Configure the channel according to the mode.	See Table 12-74.	
Start the channel.	MCSPi_CHCTRL_0/1/2/3[0] EN	1
Enable the interrupt for the receiver register.	MCSPi_IRQENABLE[2] RX_FULL_ENABLE	1
Wait for interrupt.		
Read the status register.	MCSPi_IRQSTATUS[2] RX_FULL	1
Disable the interrupt if no more data is expected.	MCSPi_IRQENABLE[2] RX_FULL_ENABLE	0
Stop the channel if no more data is expected.	MCSPi_CHCTRL_0/1/2/3[0] EN	0
Read the receiver register.	MCSPi_RX_0/1/2/3	0x-

#### 12.1.5.5.2.1.9.3 Transmit-Only Procedure – Polling Method

Table 12-99 lists the transmit-only procedure using the polling method.

**Table 12-99. Transmit-Only Procedure – Polling Method**

Step	Register/Bit Field/Programming Model	Value
Configure the channel according to the mode.	See Table 12-75.	
Start the channel.	MCSPi_CHCTRL_0/1/2/3[0] EN	1
Write the transmitter register with data.	MCSPi_TX_0/1/2/3	0x-
Wait until end of transfer?	MCSPi_CHSTAT_0/1/2/3[2] EOT	=1
Stop the channel.	MCSPi_CHCTRL_0/1/2/3[0] EN	0

#### 12.1.5.5.2.1.9.4 Transmit-and-Receive Procedure – Polling Method

Table 12-100 lists the transmit-and-receive procedure using the polling method.

**Table 12-100. Transmit-and-Receive Procedure – Polling Method**

Step	Register/Bit Field/Programming Model	Value
Configure the channel according to the mode.	See Table 12-76.	
Start the channel.	MCSPi_CHCTRL_0/1/2/3[0] EN	1
Write the transmitter register with data.	MCSPi_TX_0/1/2/3	0x-
Wait until transmit/receive word?	MCSPi_CHSTAT_0/1/2/3[2] EOT	=1
Stop the channel.	MCSPi_CHCTRL_0/1/2/3[0] EN	0
Read the receiver register.	MCSPi_RX_0/1/2/3	0x-

## 12.1.6 Universal Asynchronous Receiver/Transmitter (UART)

This chapter describes the function, operation, and configuration of the Universal Asynchronous Receiver/Transmitter (UART)/RS-485/Infrared Data Association (IrDA)/Consumer Infrared (CIR)/ISO 7816 module in the device.

### Note

UART and USART acronyms are used interchangeably in this section.

### 12.1.6.1 UART Overview

The UART is a peripheral that utilizes the DMA for data transfer or interrupt polling via host CPU. There are twelve UART modules in the device. All UART modules support IrDA and CIR modes when 48 MHz function clock is used. Each UART can be used for configuration and data exchange with a number of external peripheral devices or interprocessor communication between devices.

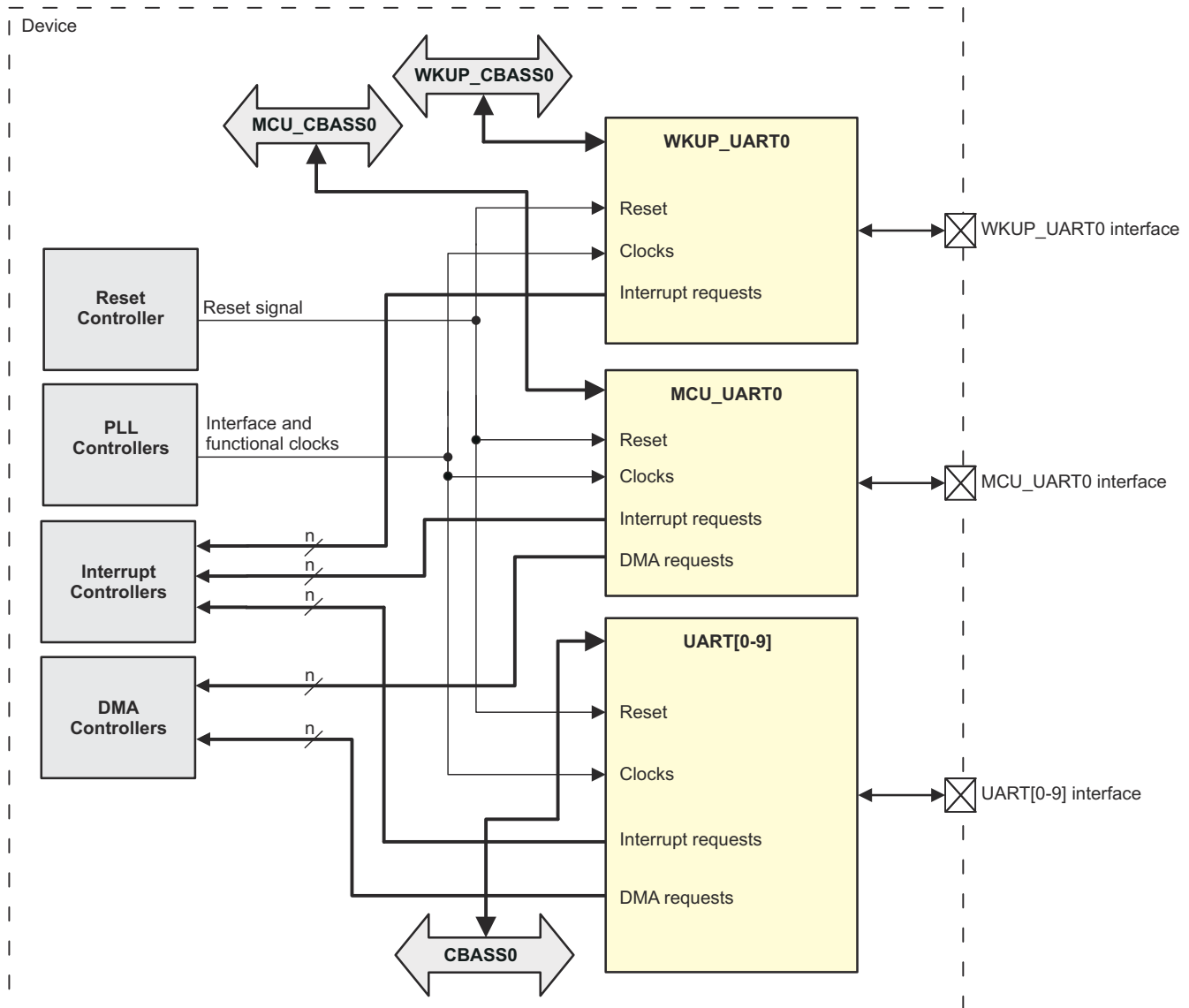
[Table 12-101](#) shows UART modules allocation across device domains.

**Table 12-101. UART Allocation Across Device Domains**

Instance	Domain		
	WKUP	MCU	MAIN
WKUP_UART0	✓	-	-
MCU_UART0	-	✓	-
UART0	-	-	✓
UART1	-	-	✓
UART2	-	-	✓
UART3	-	-	✓
UART4	-	-	✓
UART5	-	-	✓
UART6	-	-	✓
UART7	-	-	✓
UART8	-	-	✓
UART9	-	-	✓

[Figure 12-88](#) shows the UART modules overview.





uart\_001

**Figure 12-88. UART Modules Overview**

#### 12.1.6.1.1 UART Features

The UART includes the following features:

- 16C750-compatible
- RS-485 external transceiver auto flow control support
- 64-byte FIFO buffer for receiver and 64-byte FIFO buffer for transmitter
- Programmable interrupt trigger levels for FIFOs
- Programmable sleep mode
- The 48 MHz functional clock is default option and allows baud rates up to 3.6 Mbps
- Auto-baud between 1200 bits/s and 115.2 Kbits/s (only when 48 MHz function clock is used)
- Optional multi-drop transmission
- Configurable time-guard feature
- Configurable data format:
  - Data bit: 5, 6, 7, 8, or 9 bit
  - Parity bit: Even, Odd, None

- Stop-bit: 1, 1.5, 2 bit
- Flow control: Hardware (RTS/CTS) or software (XON/XOFF)
- False start bit detection
- Line break generation and detection
- Fully prioritized interrupt system controls
- Internal test and loopback capabilities
- Modem control functions (CTS, RTS)
- Only module instance in MAIN domain has extended modem control signals (DCD, RI, DTR, DSR)

#### **12.1.6.1.2 IrDA Features**

The IrDA includes the following features:

- Support of IrDA 1.4 slow infrared (SIR), medium infrared (MIR), and fast infrared (FIR) communications:
  - Slow infrared (SIR 115.2 KBAUD), medium infrared (MIR 0.576 MBAUD) and fast infrared (FIR 4.0 MBAUD) operations (very fast infrared (VFIR) is not supported)
  - Frame formatting: addition of variable beginning-of-frame (xBOF) characters and end-of-frame (EOF) characters
  - 
  - Asynchronous transparency (automatic insertion of break character)
  - Eight-entry status FIFO (with selectable trigger levels) to monitor frame length and frame errors
  - Framing error, CRC error, illegal symbol (FIR), and abort pattern (SIR, MIR) detection
- IrDA mode when 48 MHz function clock is used

#### **12.1.6.1.3 CIR Features**

The CIR mode uses a variable pulse-width modulation (PWM) technique (based on multiples of a programmable  $t$  period) to encompass the various formats of infrared encoding for remote-control applications. The CIR logic transmits data packets based on a user-definable frame structure and packet content.

The CIR includes the following features to provide CIR support for remote-control applications:

- Transmit and receive mode
- Free data format (supports any remote-control private standards)
- Selectable bit rate
- Configurable carrier frequency
- 1/2, 5/12, 1/3, or 1/4 carrier duty cycle
- CIR mode when 48 MHz function clock is used

#### **12.1.6.1.4 UART Not Supported Features**

- Synchronous mode
- ISO7816 mode
- DMA mode 2
- Full modem handshaking is not available on all instances - see device datasheet for details
- Multi-drop transmission
- 9-bit mode
- 12 Mbps operation

### 12.1.6.2 UART Environment

The WKUP\_UART0, MCU\_UART0, UART[0-9] modules are hereinafter referred to as UART module.

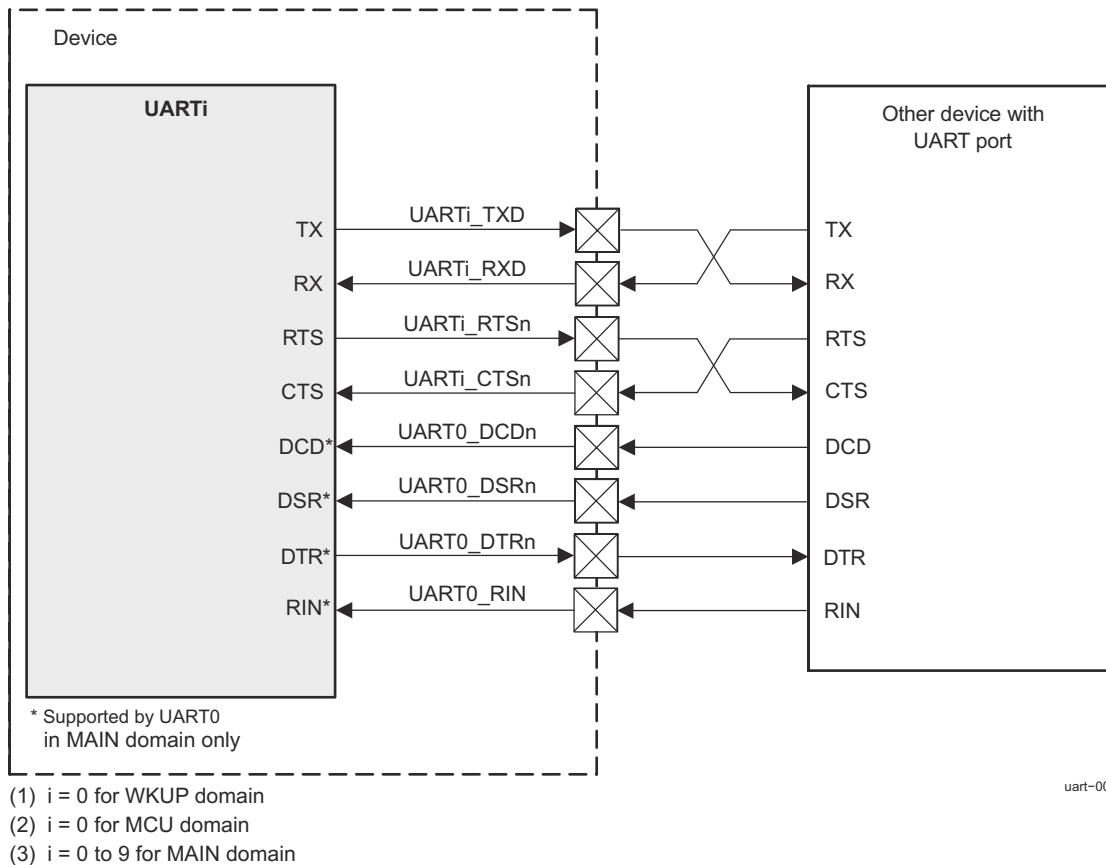
This section describes the UART/RS-485/IrDA/CIR external connections (environment).

- The UART interface is described in [Section 12.1.6.2.1, UART Functional Interfaces](#).
- The RS-485 interface is described in [Section 12.1.6.2.2, RS-485 Functional Interfaces](#).
- The IrDA interface is described in [Section 12.1.6.2.3, IrDA Functional Interfaces](#).
- The CIR interface is described in [Section 12.1.6.2.4, CIR Functional Interfaces](#).

#### 12.1.6.2.1 UART Functional Interfaces

##### 12.1.6.2.1.1 System Using UART Communication With Hardware Handshake

Each UART instance can be easily connected to the UART port of an external IC (see [Figure 12-89](#) ).



**Figure 12-89. UART Mode Interface Signals**

### 12.1.6.2.1.2 UART Interface Description

Table 12-102 lists the UART interface input/output (I/O) signals.

**Table 12-102. UART I/O Signals**

Module Pin Name	Device Level Signal Name	I/O <sup>(1)</sup>	Description	Module Pin Reset Value <sup>(2)</sup>
<b>WKUP_UART0</b>				
RX	WKUP_UART0_RXD	I	Serial data input	HiZ
TX	WKUP_UART0_TXD	O	Serial data output <sup>(3)</sup>	1
CTS	WKUP_UART0_CTS	I	Clear to send <sup>(4)</sup>	HiZ
RTS	WKUP_UART0_RTS	O	Request to send <sup>(5)</sup>	1
<b>UART[0-9]</b>				
RX	UART[0-9]_RXD	I	Serial data input	HiZ
TX	UART[0-9]_TXD	O	Serial data output <sup>(3)</sup>	1
CTS	UART[0-9]_CTS <sub>n</sub>	I	Clear to send <sup>(4)</sup>	HiZ
RTS	UART[0-9]_RTS <sub>n</sub>	O	Request to send <sup>(5)</sup>	1
DCD		I	Data Carrier Detect <sup>(6)</sup>	HiZ
DSR		I	Data Set Ready <sup>(7)</sup>	HiZ
DTR		O	Data Terminal Ready <sup>(8)</sup>	1
RIN		I	Ring Indicator <sup>(9)</sup>	HiZ

(1) I = Input; O = Output

(2) HiZ = High Impedance

(3) Because this pin is active high in IrDA mode and the output is muxed, this pin is set to low on reset (when the UART\_MDR1[2-0] bit field is set to 0x7) and takes the defined inactive level of that signal corresponding to when and how the UART\_MDR1 register is programmed; that is, the output is 1 (inactive for UART modem modes) and 0 (inactive for IrDA modes).

(4) Active-low modem status signal. Reading the UART\_MSR[4] NCTS\_STS bit checks the condition of CTS. Reading the UART\_MSR[0] CTS\_STS bit checks a change of state of CTS since the last read of the modem status register. The auto-CTS mode uses CTS to control the transmitter.

(5) When active (low), the module is ready to receive data. Setting the UART\_MCR[1] RTS bit activates RTS signal, which becomes inactive as the result of a module reset, loopback mode, or clearing the UART\_MCR[1] RTS bit. In auto-RTS mode, RTS signal becomes inactive as a result of the receiver threshold logic.

(6) Active-low modem status signal. The condition of DCD can be checked by reading the UART\_MSR[7] NCD\_STS bit. Any change in its state can be detected by reading the UART\_MSR[3] DCD\_STS bit.

(7) Active-low modem status signal. Reading the UART\_MSR[5] NDSR\_STS bit checks the condition of DSR. Reading the UART\_MSR[1] DSR\_STS bit checks a change of state of DSR since the last read of the UART\_MSR register.

(8) When active (low), this signal informs the modem that the module is ready to communicate. It is activated by setting the UART\_MCR[0] DTR bit.

(9) Active-low modem status signal. The condition of RIN can be checked by reading the UART\_MSR[6] NRI\_STS bit. Any change in its state can be detected by reading the UART\_MSR[2] RI\_STS bit.

#### Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

### 12.1.6.2.1.3 UART Protocol and Data Format

The UART device operates in three modes:

- UART 16× (<= 230.4 kbps)
- UART 16× with autobauding (>= 1200 bps and <= 115.2 kbps)
- UART 13× (>= 460.8 kbps)

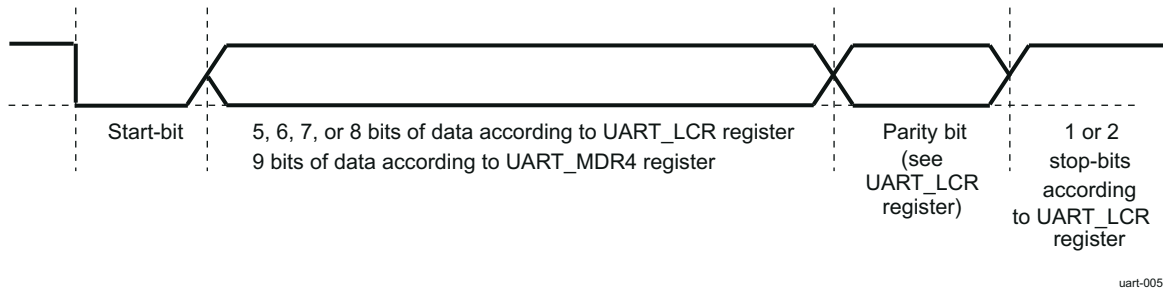
### CAUTION

To be used as a UART, the operating mode must be programmed appropriately in the UART\_MDR1[2-0] MODE\_SELECT bit field to select UART, IrDA, or CIR mode, and the UART\_MDR3[4] DIR\_EN bit field to select RS-485 mode.

The UART uses a wired interface for serial communication with a remote device.

The UART is functionally compatible with the TL16C750 UART and earlier designs such as the TL16C550.

Figure 12-90 shows the UART frame data format.



**Figure 12-90. UART Frame Data Format**

### Note

Not all operating modes are described in the diagram. The UART module is capable of supporting a 9-bit mode that is not shown in the diagram.

#### 12.1.6.2.1.4 UART 9-bit Mode Data Format

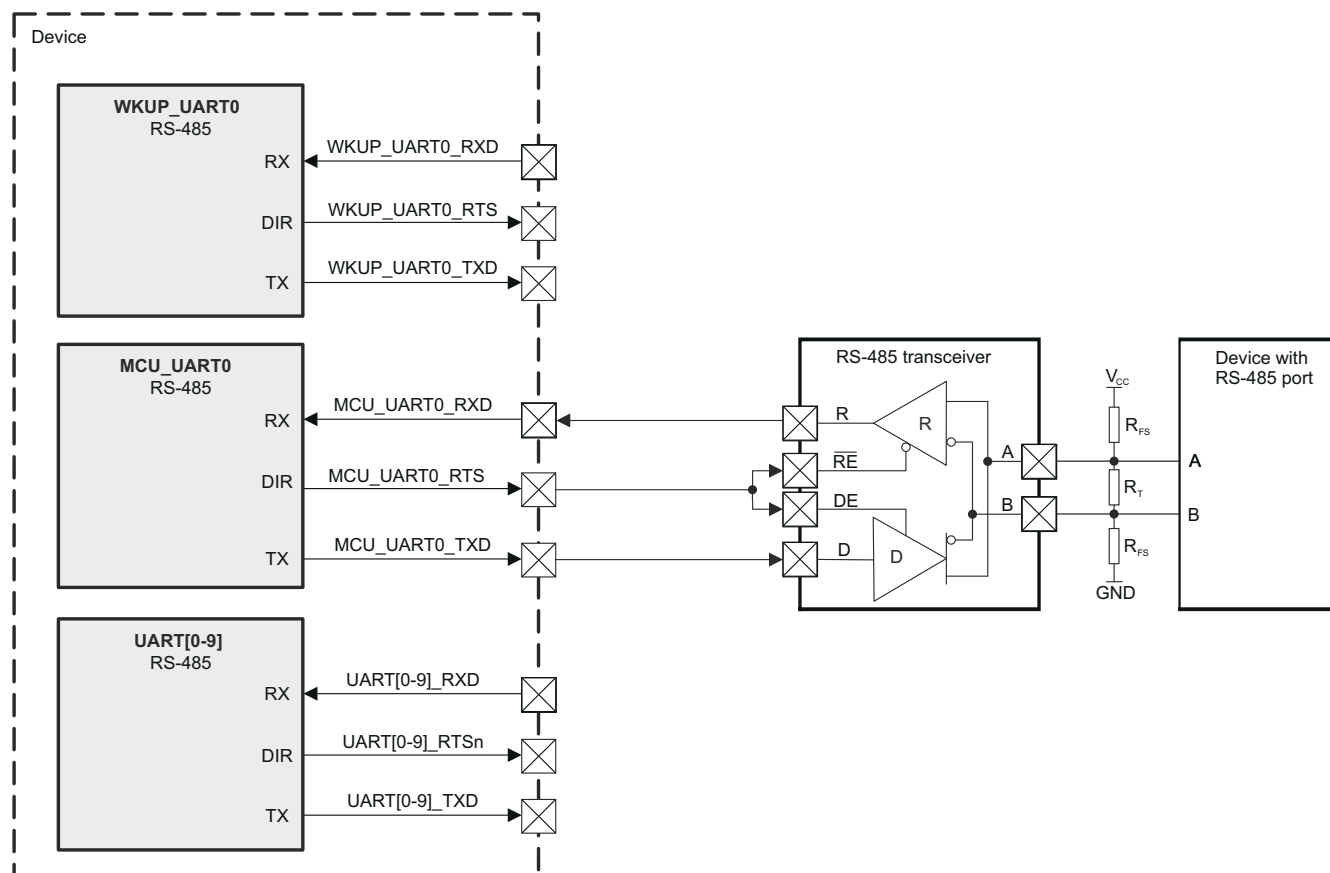
The UART is capable of full 9-bit operation when configured so in UART\_MDR4[6] MODE9 bit. The full 9-bit data can only be written and read through the UART\_ETHR / UART\_ERHR registers. The data can be accessed either with a single or two accesses using appropriate OCP byte enables. In case two accesses are used, make sure to read/write the higher byte first, as reading/writing the lower one will advance the FIFO.

The UART\_RHR / UART\_ERHR and UART\_THR / UART\_ETHR registers point to the same location and either can be used as preferred, in regards to the lower 8 bits.

#### 12.1.6.2.2 RS-485 Functional Interfaces

##### 12.1.6.2.2.1 System Using RS-485 Communication

The RS-485 network physical layer consists of two-wire differential bus, usually twisted pair. External RS-485 transceiver IC is needed to access a RS-485 bus by the RS-485 mode. Figure 12-91 shows an example connection of UART in RS-485 mode.



uart-037

**Figure 12-91. RS-485 Mode Interface Signals**

#### 12.1.6.2.2.2 RS-485 Interface Description

Table 12-103 lists the RS-485 interface input/output (I/O) signals.

**Table 12-103. UART I/O Signals (RS-485 Mode)**

Module Pin	Device Level Signal	I/O <sup>(1)</sup>	Description	Module Pin Reset Value <sup>(2)</sup>
<b>WKUP_UART0</b>				
RX	WKUP_UART0_RXD	I	Serial data input	HiZ
TX	WKUP_UART0_TXD	O	Serial data output	1
DIR	WKUP_UART0_RTS	O	RS-485 Direction	1
<b>MCU_UART0</b>				
RX	MCU_UART0_RXD	I	Serial data input	HiZ
TX	MCU_UART0_TXD	O	Serial data output	1
DIR	MCU_UART0_RTS	O	RS-485 Direction	1
<b>UART[0-9]</b>				
RX	UART[0-9]_RXD	I	Serial data input	HiZ
TX	UART[0-9]_TXD	O	Serial data output	1
DIR	UART[0-9]_TXD	O	RS-485 Direction	1

(1) I = Input; O = Output

(2) HiZ = High Impedance

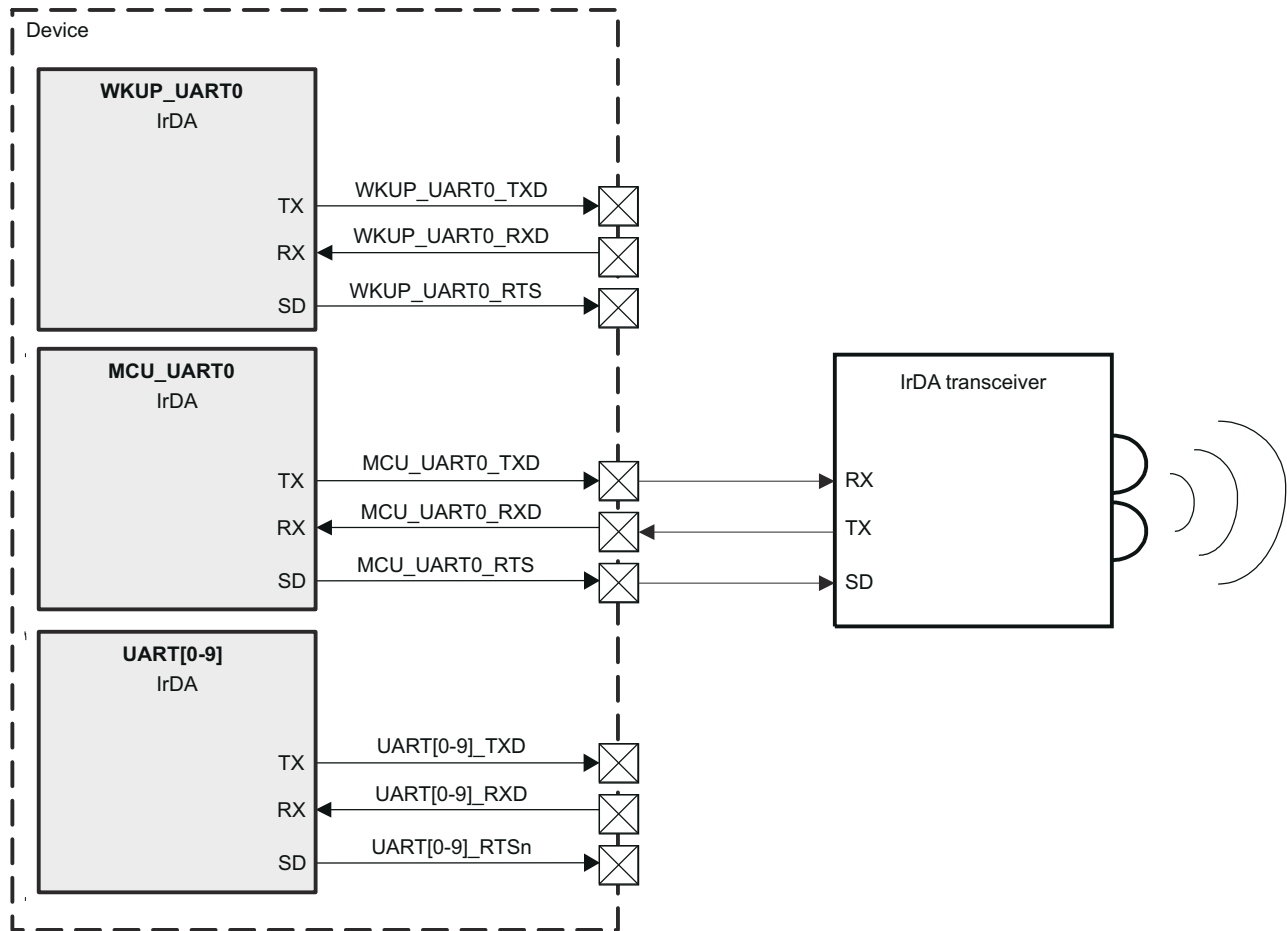
### Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

#### 12.1.6.2.3 IrDA Functional Interfaces

##### 12.1.6.2.3.1 System Using IrDA Communication Protocol

Figure 12-92 shows an example connection of MCU\_UART0 to an external infrared transceiver in the IrDA modes (FIR, SIR, and MIR).



uart-003

Figure 12-92. IrDA Mode Interface Signals

### 12.1.6.2.3.2 IrDA Interface Description

Table 12-104 lists the IrDA interface I/O signals.

**Table 12-104. UART I/O Signals (IrDA Mode)**

Module Pin	Device Level Signal	I/O <sup>(1)</sup>	Description	Module Pin Reset Value <sup>(2)</sup>
<b>WKUP_UART0</b>				
RX	WKUP_UART0_RXD	I	Serial data input	HiZ
TX	WKUP_UART0_TXD	O	Serial data output in IrDA modes (SIR, MIR, and FIR). <sup>(3)</sup>	0
SD	WKUP_UART0_RTS	O	SD mode is used to configure the transceivers. <sup>(4)</sup>	1
<b>MCU_UART0</b>				
RX	MCU_UART0_RXD	I	Serial data input	HiZ
TX	MCU_UART0_TXD	O	Serial data output in IrDA modes (SIR, MIR, and FIR). <sup>(3)</sup>	0
SD	MCU_UART0_RTS	O	SD mode is used to configure the transceivers. <sup>(4)</sup>	1
<b>UART[0-9]</b>				
RX	UART[0-9]_RXD	I	Serial data input	HiZ
TX	UART[0-9]_TXD	O	Serial data output in IrDA modes (SIR, MIR, and FIR). <sup>(3)</sup>	0
SD	UART[0-9]_RTSn	O	SD mode is used to configure the transceivers. <sup>(4)</sup>	1

(1) I = Input; O = Output

(2) HiZ = High Impedance

(3) In other modes, this pin is set to the reset value (inactive state).

(4) The SD pinout (see UART\_ACREG[6] SD\_MOD bit).

#### Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

### 12.1.6.2.3.3 IrDA Protocol and Data Format

#### 12.1.6.2.3.3.1 SIR Mode

In SIR mode, data is transferred between the Host CPU and peripheral devices at speeds of up to 115200 baud. A SIR transmit frame begins with start flags (a single 0xC0, a multiple 0xC0, or a single 0xC0 preceded by a number of 0xFF flags), followed by frame data and a CRC-16, and ends with a stop flag (0xC1).

The bit format for a single word uses 1 start-bit, 8 data bits, and 1 stop-bit, and is unaffected by the use and settings of the UART\_LCR register.

The UART\_BLR[6] XBOF\_TYPE bit selects whether the 0xC0 or 0xFF start patterns are used when multiple start flags are required.

The SIR transmit state-machine attaches start flags, CRC-16, and stop flags, and checks the outgoing data to establish whether data transparency is required.

The SIR transparency is carried out if the outgoing data between the start and stop flags contains 0xC0, 0xC1, or 0x7D. If one of these start flags is about to be transmitted, the SIR state-machine sends an escape character (0x7D), inverts the fifth bit of the real data to be sent, and then sends this data immediately after the 0x7D character.

The SIR receive state-machine recovers the receive clock, removes the start flags and any transparency from the incoming data, and determines the frame boundary with reception of the stop flag. The SIR state-machine also checks for errors such as a frame abort (0x7D character followed immediately by a 0xC1 stop flag without transparency), a CRC error, or a frame-length error. At the end of a frame reception, the Host CPU reads the line status register (UART\_LSR\_IRDA) to find possible errors of the received frame.



### Note

The module can transmit and receive data, but when the device is transmitting, the IR RX circuitry is automatically disabled by hardware. See the description of the UART\_ACREG[5] DIS\_IR\_RX bit. This applies to all three modes: SIR, MIR, and FIR.

Infrared output in SIR mode can be 1.6-μs or 3/16 encoding, selected by the UART\_ACREG[7] PULSE\_TYPE bit. In 1.6-μs encoding, the infrared pulse width is 1.6 μs; and in 3/16th encoding, the infrared pulse width is 3/16th of a bit duration (1/ baud rate).

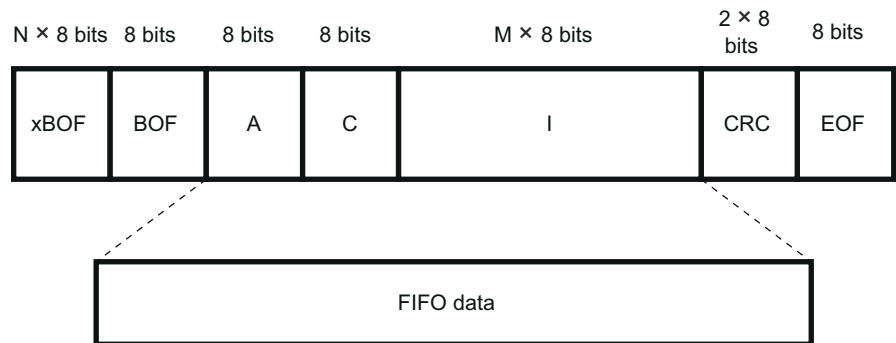
For back-to-back frames, the transmitting device must send at least two start flags at the start of each frame.

### Note

Reception supports variable-length stop-bits.

#### 12.1.6.2.3.3.1.1 Frame Format

Figure 12-93 shows the IrDA SIR frame format.



uart-006

**Figure 12-93. IrDA SIR Frame Format**

The CRC is applied on the address (A), control (C), and information (I) bytes.

### Note

The two words of CRC are written to the FIFO in reception.

#### 12.1.6.2.3.3.1.2 Asynchronous Transparency

Before transmitting a byte, the UART IrDA controller examines each byte of the payload and the CRC field (between BOF and EOF). For each byte equal to 0xC0 (BOF), 0xC1 (EOF), or 0x7D (control escape), the controller performs certain tasks:

- In transmission:
  - Inserts a control escape (CE) byte preceding the byte
  - Complements bit 5 of the byte (that is, exclusive ORs the byte with 0x20)

The byte sent for the CRC computation is the initial byte written in the TX FIFO (before the XOR with 0x20).

- In reception:

For the A, C, I, and CRC fields:

- Compares the byte with the CE byte; if they are not equal, sends the byte to the CRC detector and stores it in the RX FIFO.
- If the byte is equal to the CE byte, discards the CE byte

- Complements bit 5 of the byte following the CE
- Sends the complemented byte to the CRC detector and stores it in the RX FIFO

#### 12.1.6.2.3.3.1.3 Abort Sequence

The transmitter can prematurely close a frame (abort) by sending the sequence 0x7DC1. The abort pattern closes the frame without a CRC field or an ending flag.

When a 0x7D character that is followed immediately by a 0xC1 character is received without transparency, the receiver treats the frame as an aborted frame.

#### 12.1.6.2.3.3.1.4 Pulse Shaping

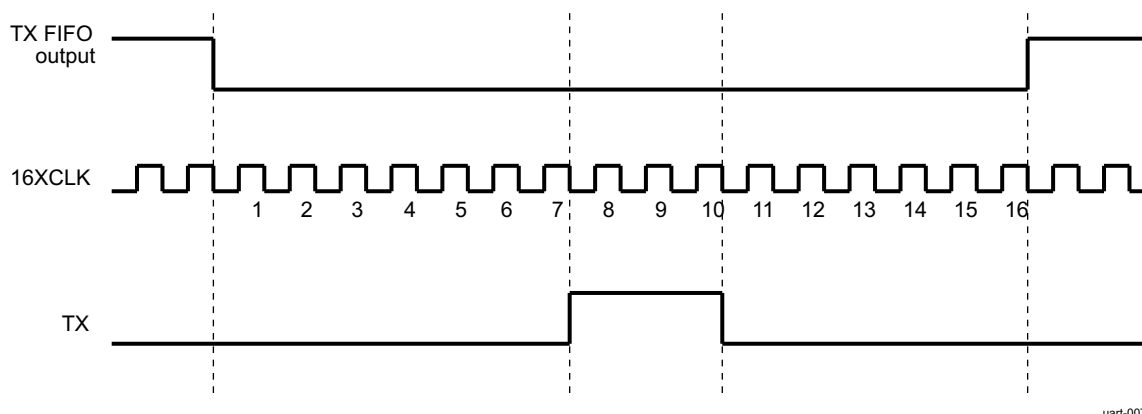
The SIR mode supports the 3/16 and the 1.6-μs pulse duration methods. The UART\_ACREG[7] PULSE\_TYPE bit selects the pulse-width method in transmit mode.

#### 12.1.6.2.3.3.1.5 Encoder

Serial data from the transmit state-machine are encoded to transmit data to the optoelectronics. While the TX FIFO output is high, the TX line is always low, and the counter used to form a pulse on TX is cleared continuously.

After the TX FIFO output resets to 0, TX rises on the falling edge of the seventh 16XCLK. On the falling edge of the tenth 16XCLK pulse, TX falls, creating a 3-clock-wide pulse. While the TX FIFO output stays low, a pulse is transmitted during the seventh clock to the tenth clock of each 16-clock bit cycle.

Figure 12-94 shows the IrDA SIR encoding mechanism.

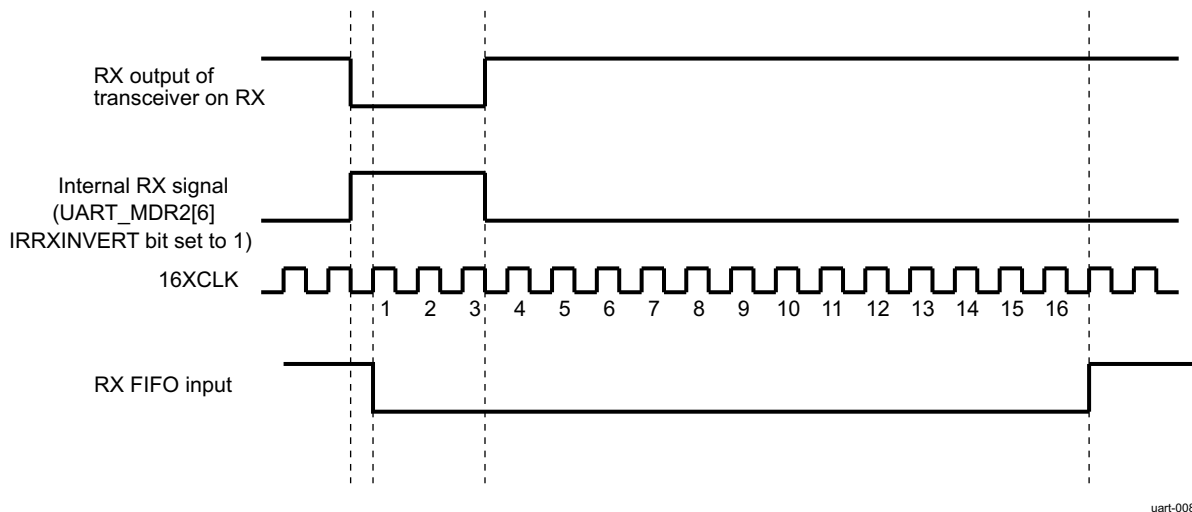


**Figure 12-94. IrDA SIR Encoding Mechanism**

#### 12.1.6.2.3.3.1.6 Decoder

After reset, the RX FIFO input is high and the 4-bit counter is cleared. When a rising edge is detected on RX, the RX FIFO input falls on the next rising edge of 16XCLK with sufficient setup time. The RX FIFO input stays low for 16 cycles (16XCLK) and then returns to high as required by the IrDA specification. As long as no pulses (rising edges) are detected on the RX, the RX FIFO input remains high.

Figure 12-95 shows the IrDA SIR decoding mechanism.



**Figure 12-95. IrDA SIR Decoding Mechanism**

The module can transmit and receive data, but when the device is transmitting, the IR RX circuitry is automatically disabled by hardware. The operation of the RX input can be disabled using the UART\_ACREG[5] DIS\_IR\_RX bit. The UART\_MDR2[6] IRRXINVERT bit can invert the signal from the transceiver (RX) pin to the IR RX logic in the UART. This inversion is performed by default.

#### 12.1.6.2.3.3.1.7 IR Address Checking

In all IR modes, when address checking is enabled by setting the UART\_EFR[1-0] bit field (see [Table 12-105](#)), only frames intended for the device are written to the RX FIFO. This is to avoid receiving frames not meant for this device in a multipoint infrared environment. To program two frame addresses that the UARTi receives in IrDA mode, use the UART\_XON1\_ADDR1[7-0] and UART\_XON2\_ADDR2[7-0] bit fields.

**Table 12-105. UART\_EFR[1-0] IR Address Checking Options**

UART_EFR[1]	UART_EFR[0]	IR Address Checking
0	0	All address-checking operations disabled
0	1	Only address 1 checking enabled
1	0	Only address 2 checking enabled
1	1	All address-checking operations enabled

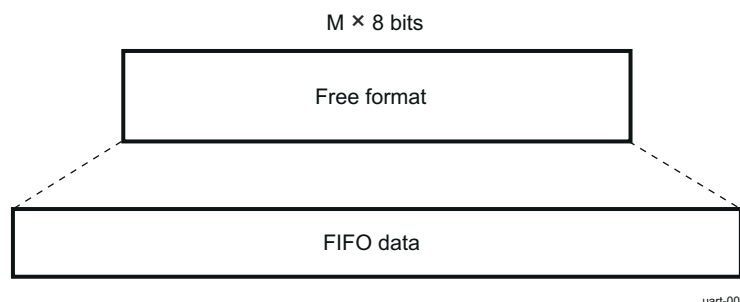
#### 12.1.6.2.3.3.2 SIR Free-Format Mode

To allow complete software flexibility when transmitting and receiving infrared data packets, the SIR free-format (FF) mode is a subfunction of the existing SIR mode. In FF mode, all frames going to and from the FIFO buffers are untouched with respect to appending and removing control characters and CRC values.

The FF mode corresponds to a UART mode with a pulse modulation of 3/16 of baud rate pulse width.

For example, a normal SIR packet has BOF control and CRC error-checking data appended (transmitting) or removed (receiving) from the data going to and from the FIFOs.

[Figure 12-96](#) shows SIR FF mode.



uart-009

**Figure 12-96. SIR FF Mode**

In SIR FF mode, the Host CPU software must construct (that is, encode and decode) the entire FIFO data packet.

The SIR Free Format mode is selected by setting the module in UART mode (MDR1[2:0] = 000) and the MDR2[3] register bit to one to allow the pulse shaping. As the bit format is to remain the same, some UART mode configuration registers need to be set at specific value:

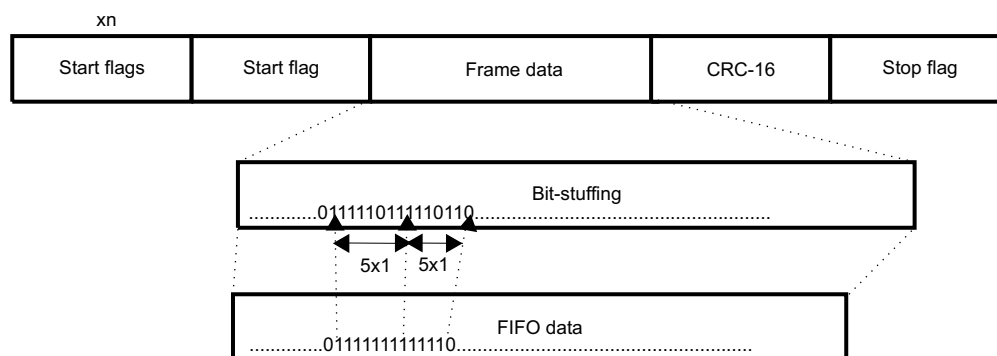
- LCR[1:0] = "11" (8 data bits)
- LCR[2] = 0 (2 stop bit)
- LCR[3] = 0 (no parity)
- ACREG[7] = 0 (3/16 of baud-rate pulse width)

The features defined through MDR2[6] and ACREG[5] are also supported, however:

- All other configuration registers need to be at the reset value
- The same UART mode interrupts used for the SIR FF mode are not necessarily relevant (XOFF, RTS, CETS, Modem Status Register)

#### 12.1.6.2.3.3.3 MIR Mode

In MIR mode, data is transferred between the Host CPU and the peripheral devices at 0.576 Mbps or 1.152 Mbps. A MIR transmit frame starts with at least two start flags, followed by a frame data and a CRC-16, and ends with a stop flag (see [Figure 12-97](#)).



uart-010

**Figure 12-97. MIR Transmit Frame Format**

On transmit, the MIR state-machine attaches start flags, a CRC-16, and stop flags, as in SIR mode. All fields are transmitted least-significant bit (LSB) of each byte first.

In MIR mode:

- The state-machine looks for consecutive 1s in the frame data and automatically inserts 0 after five consecutive 1s (this is called bit-stuffing).
- 0x7E is used for start and stop flags (unambiguously, not data, because of bit-stuffing).

- An abort sequence requires a minimum of seven consecutive 1s (unambiguously, not data, because of bit-stuffing).
- Back-to-back frames are allowed with three or more stop flags between them. If two consecutive frames are not back to back, the gap between the last stop flag of the first frame and the start flag of the second frame must be separated by at least seven bit durations.

On receive, the MIR receive state-machine recovers the receive clock, removes the start flags, destuffs the incoming data, and determines the frame boundary with reception of the stop flag. The state-machine also checks for errors such as frame abort, CRC error, and frame-length error. At the end of a frame reception, the Host CPU reads the line status register (UART\_LSR\_IRDA) to detect errors of the received frame.

The module can transmit and receive data, but when the device is transmitting, the IR RX circuitry is automatically disabled by hardware.

#### 12.1.6.2.3.3.1 MIR Encoder/Decoder

To meet the MIR baud rate tolerance of 0.1 percent with a 48-MHz clock input, a 42-41-42 encoding/decoding adjustment is performed. The reference start point is the first start flag, and the 42-41-42 cyclic pattern is repeated until the stop flag is sent or detected.

The jitter created this way is within MIR tolerances. The pulse width is not exactly 1/4, but it is within the tolerances defined by IrDA specifications.

Figure 12-98 shows the MIR baud rate adjustment mechanism.

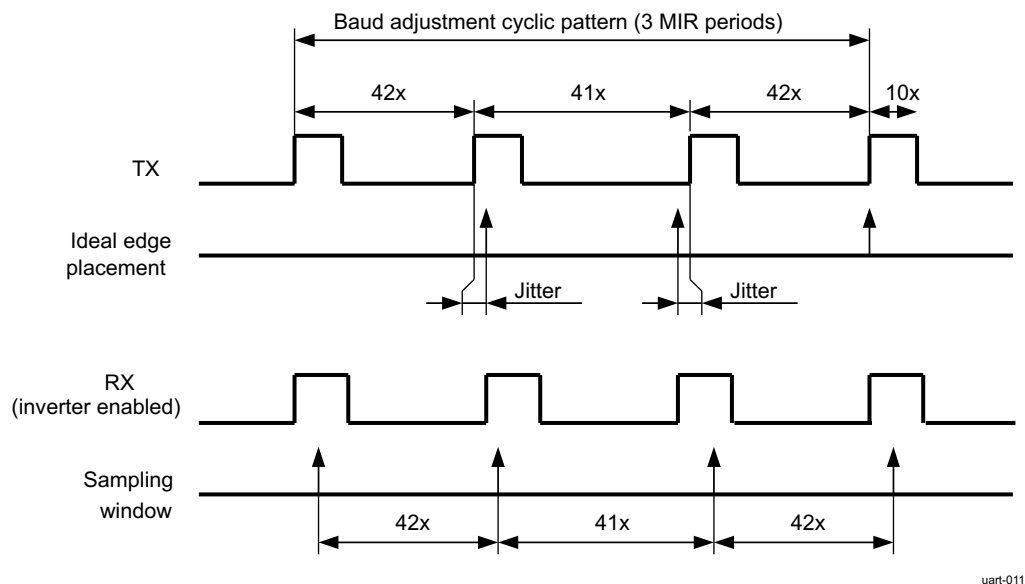
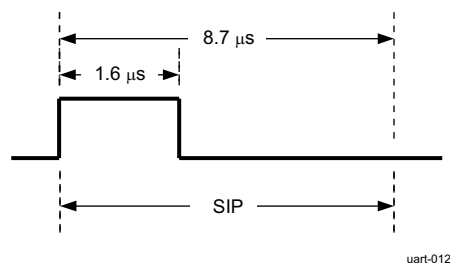


Figure 12-98. MIR Baud Rate Adjustment Mechanism

#### 12.1.6.2.3.3.2 SIP Generation

In the MIR and FIR operation modes, the transmitter must send a serial infrared interaction pulse (SIP) at least once every 500 ms. The SIP informs slow devices (operating in SIR mode) that the medium is occupied.

Figure 12-99 shows the SIP.


**Figure 12-99. SIP**

When the SIP\_MODE bit of the Mode Definition Register 1 is equal to 1 (MDR1[6]), the TX state machine will always send one SIP at the end of the transmission frame. When MDR1[6] is equal to 0, the transmission of the SIP depends on the SEND\_SIP bit of the Auxiliary Control Register (ACREG[3]). The system (Host CPU) can set ACREG[3] at least once every 500ms. The advantage of this approach over the default approach is that the TX state machine does not need to send the SIP at the end of each frame, which may reduce the overhead required.

#### 12.1.6.2.3.3.4 FIR Mode

In FIR mode, data is transferred between the Host CPU and the peripheral devices at 4 Mbps. A FIR transmit frame starts with a preamble that is followed by a start flag, frame data, CRC-32, and ends with a stop flag.

Figure 12-100 shows the FIR transmit frame format.

**Figure 12-100. FIR Transmit Frame Format**

Preamble (16x)	Start flag	Frame data	CRC-32	Stop flag
----------------	------------	------------	--------	-----------

On transmit, the FIR transmit state-machine attaches the preamble, start flag, CRC-32, and stop flag. An abort sequence requires at least two transmissions of 0000. Back-to-back frames are allowed, but each frame must be complete.

The state-machine also encodes the transmit data into 4-PPM format (see Table 12-106) and generates the SIP (see Section 12.1.6.2.3.3.3.2, *SIP Generation*).

**Table 12-106. 4-PPM Format**

Data Bit Pair (Bin)	4-PPM Data Symbol (Bin)
00	1000
01	0100
10	0010
11	0001

The four symbols described in Table 12-106 are the legal, encoded data symbols. All other combinations are illegal for encoding data. Some of these illegal symbols are used in the definition of the preamble, start flag, and stop flag because they are unambiguously not data (see Table 12-107).

**Table 12-107. FIR Preamble, Start Flag, and Stop Flag**

Frame Part	Transmitted Frame (Bin)
Preamble	1000 0000 1010 1000 (16 repeated transmissions)
Start flag	0000 1100 0000 1100 0110 0000 0110 0000
Stop flag	0000 1100 0000 1100 0000 0110 0000 0110

All fields are transmitted LSBs of each byte first (see Table 12-108).

**Table 12-108. FIR Data Byte Transmission Order Example**

Data Byte (Hex)	Data Byte Pair (Bin)	4-PPM Data Symbol (Bin)	Transmission Order
0x0B	00	1000	4
	00	1000	3
	10	0010	2
	11	0001	1

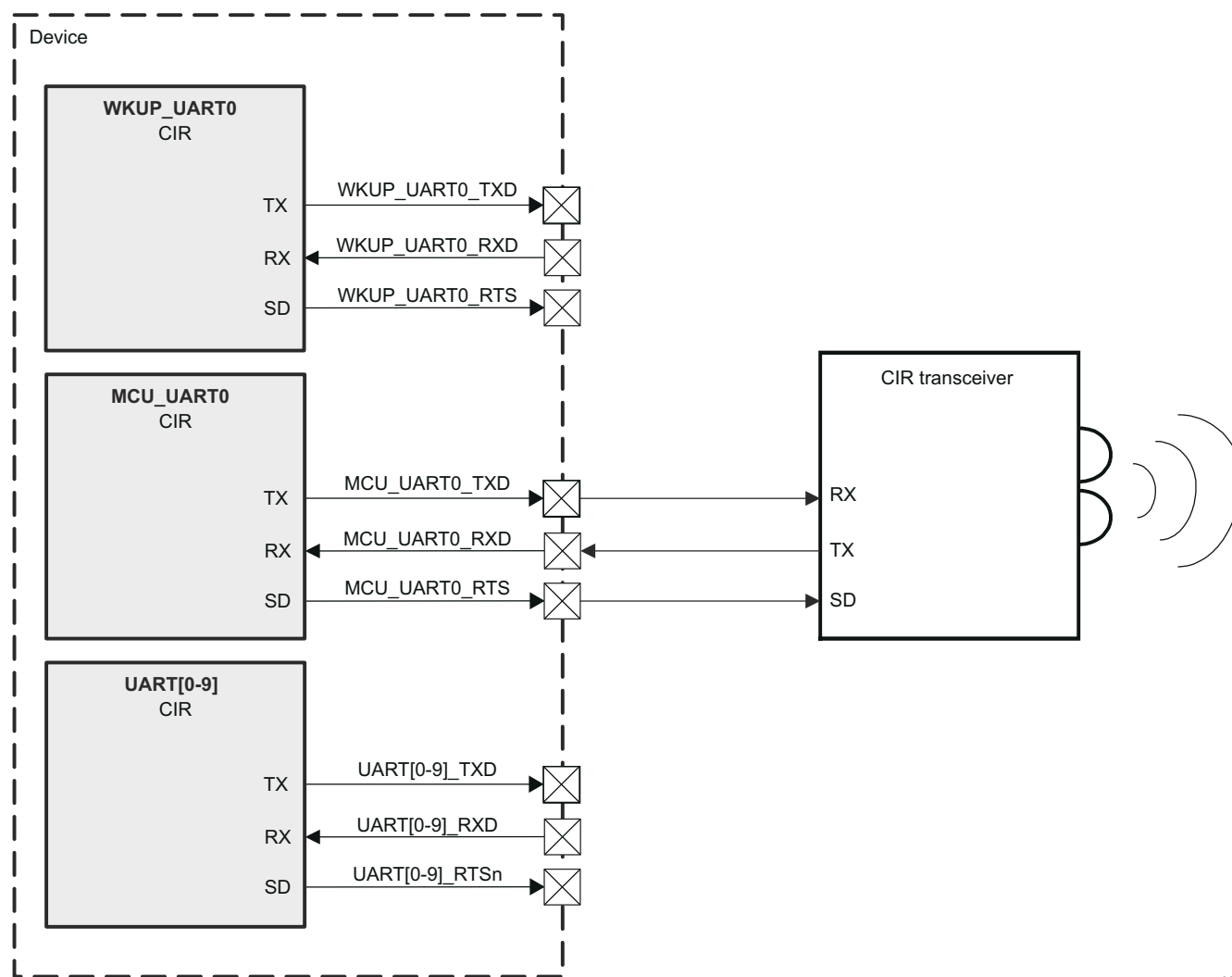
On receive, the FIR receive state-machine recovers the receive clock, removes the preamble and the start flag, decodes the 4-PPM incoming data, and determines the frame boundary with reception of the stop flag. The state-machine also checks for errors such as illegal symbol, CRC error, and frame-length error. At the end of a frame reception, the Host CPU reads the line status register (UART\_LSR\_IRDA) to detect errors of the received frame.

The module can transmit and receive data, but when the device is transmitting, the IR RX circuitry is automatically disabled by hardware.

#### 12.1.6.2.4 CIR Functional Interfaces

##### 12.1.6.2.4.1 System Using CIR Communication Protocol With Remote Control

All UART modules can be connected to an external infrared transceiver in CIR mode. [Figure 12-101](#) shows an example connection of MCU\_UART0 in CIR mode.



uart-004

**Figure 12-101. CIR Mode Interface Signals**

#### 12.1.6.2.4.2 CIR Interface Description

Table 12-109 lists the CIR interface I/O signals.

**Table 12-109. UART I/O Signals (CIR Mode)**

Module Pin	Device Level Signal	I/O <sup>(1)</sup>	Description	Module Pin Reset Value <sup>(2)</sup>
<b>WKUP_UART0</b>				
RX	WKUP_UART0_RXD	I	Serial data input	HiZ
TX	WKUP_UART0_TXD	O	Serial data output in CIR mode. <sup>(3)</sup>	0
SD	WKUP_UART0_RTS	O	SD mode is used to configure the transceivers. <sup>(4)</sup>	1
<b>MCU_UART0</b>				
RX	MCU_UART0_RXD	I	Serial data input	HiZ
TX	MCU_UART0_TXD	O	Serial data output in CIR mode. <sup>(3)</sup>	0
SD	MCU_UART0_RTS	O	SD mode is used to configure the transceivers. <sup>(4)</sup>	1
<b>UART[0-9]</b>				
RX	UART[0-9]_RXD	I	Serial data input	HiZ



**Table 12-109. UART I/O Signals (CIR Mode) (continued)**

Module Pin	Device Level Signal	I/O <sup>(1)</sup>	Description	Module Pin Reset Value <sup>(2)</sup>
TX	UART[0-9]_TXD	O	Serial data output in CIR mode. <sup>(3)</sup>	0
SD	UART[0-9]_RTSn	O	SD mode is used to configure the transceivers. <sup>(4)</sup>	1

(1) I = Input; O = Output

(2) HiZ = High Impedance

(3) In other modes, this pin is set to the reset value (inactive state).

(4) The SD pinout is an inverted value of the UART\_ACREG[6] SD\_MOD bit.

#### Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

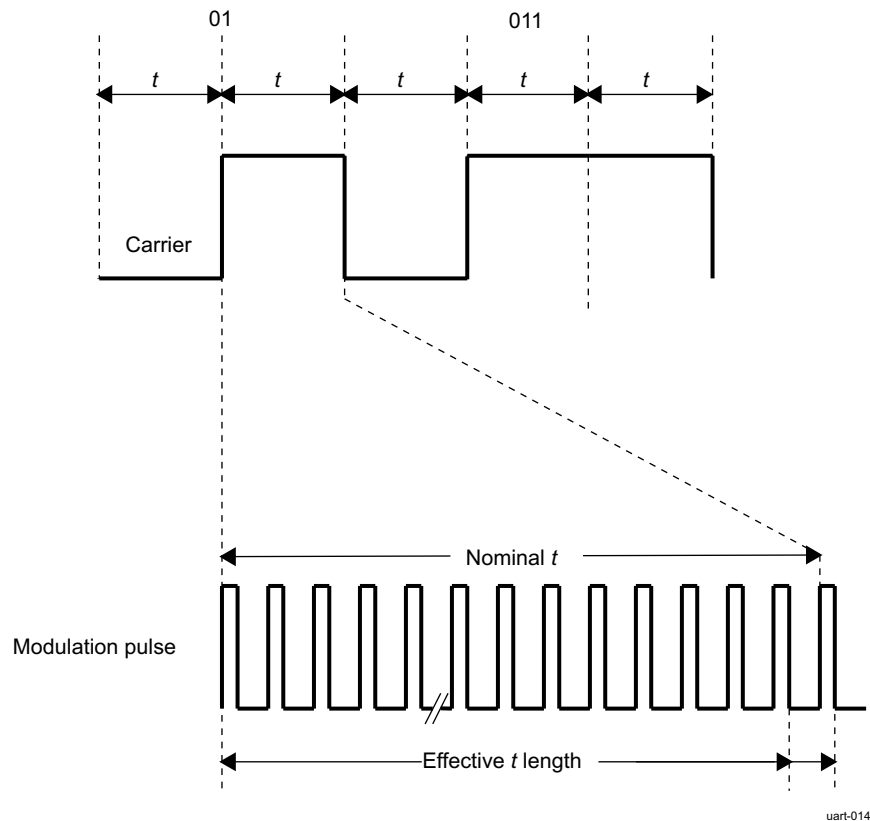
#### 12.1.6.2.4.3 CIR Protocol and Data Format

In CIR mode, the infrared operation functions as a programmable (universal) remote control.

The CIR mode uses a variable PWM technique (based on multiples of a programmable  $t$  period) to encompass the various formats of infrared encoding for remote-control applications. The CIR logic transmits data packets based on user-defined frame structure and packet content.

##### 12.1.6.2.4.3.1 Carrier Modulation

Each modulated pulse that constitutes a digit is a train of on/off pulses (see [Figure 12-102](#)).

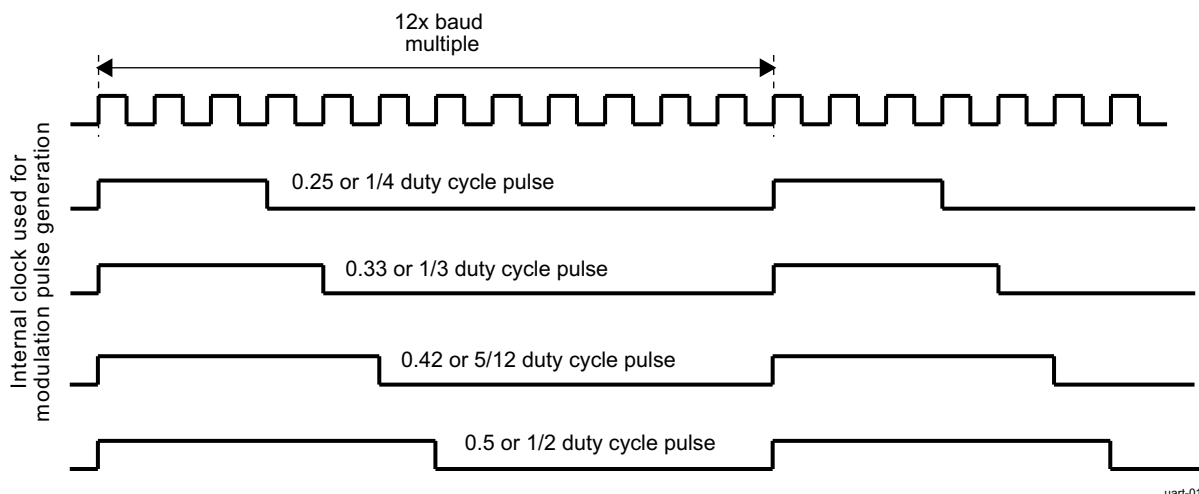


**Figure 12-102. CIR Pulse Modulation**

### 12.1.6.2.4.3.2 Pulse Duty Cycle

The programmer can choose one of four duty cycles for modulation pulses by setting the appropriate value in the UART\_MDR2[5-4] CIR\_PULSE\_MODE bit field (1/4, 1/3, 5/12, or 1/2).

Figure 12-103 shows the CIR modulation duty cycles.



**Figure 12-103. CIR Modulation Duty Cycle**

The transmission logic ensures that all pulses are transmitted completely (no cutoff during transmission). While transmitting continuous bytes back-to-back, no delay is inserted between 2 transmitted bytes. Thus, software must handle the delay between consecutively transmitted bytes if the receiving end requires it.

### 12.1.6.2.4.3.3 Consumer IR Encoding/Decoding

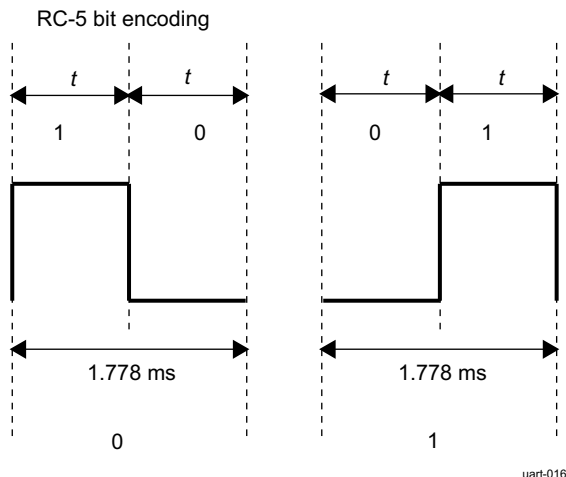
There are two methods of encoding for remote-control applications:

- Pulse duration encoding (time-extended bit forms): A variable pulse distance, or duration, in which the difference between logic 1 and logic 0 is the length of the pulse width
- Biphase encoding: The encoding of logic 0 and logic 1 is in the change of signal level from 1 to 0 or 0 to 1, respectively.

Japanese manufacturers favor pulse duration encoding; European manufacturers favor biphase encoding.

CIR mode uses a completely flexible free-format encoding in which 1 is transmitted from the TX FIFO as a modulated pulse with duration  $t$ .

Similarly, 0 is transmitted as a blank duration  $T$ . The Host CPU constructs and deciphers the protocol of the data. For example, the RC-5 protocol using Manchester encoding can be emulated as using a 01 pair for 1 and a 10 pair for 0 (see Figure 12-104).

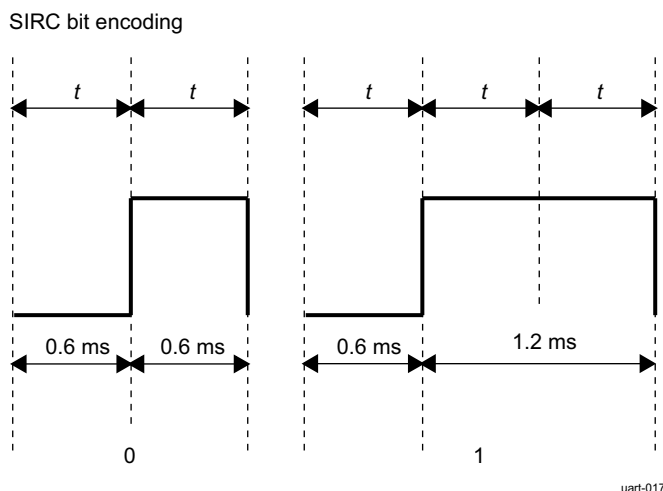


**Figure 12-104. UART RC-5 Bit Encoding**

Because CIR mode logic does not impose a fixed format for infrared packets of data, the Host CPU software can define the format using simple data structures that are then modulated into an industry standard, such as RC-5 or SIRC. To send a sequence of 0101 in RC-5, the Host CPU software must write an 8-bit binary character of 10011001 to the data FIFO of the UART.

For SIRC, the modulation length (multiples of  $t$ ) is used to distinguish between 1 and 0. The subsequent SIRC digits show the difference in encoding between this and, for example, RC-5. The pulse width is extended for one digit.

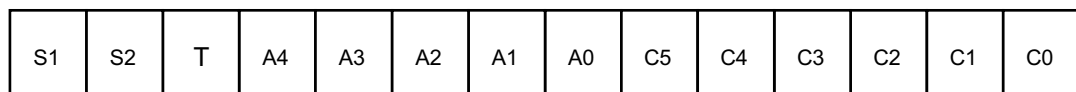
Figure 12-105 shows SIRC bit encoding.



**Figure 12-105. UART SIRC Bit Encoding**

To construct comprehensive packets constituting remote-control commands, the Host CPU software must combine a number of 8-bit data characters in a sequence that follows one of the universally accepted formats.

Figure 12-106 shows a standard RC-5 frame as detected by UART in CIR mode (the SIRC format follows this). Each field in RC-5 can be considered as two  $t$  pulses (digital bits) from the TX FIFO.



uart-018

**Figure 12-106. UART RC-5 Standard Packet Format**

Where:

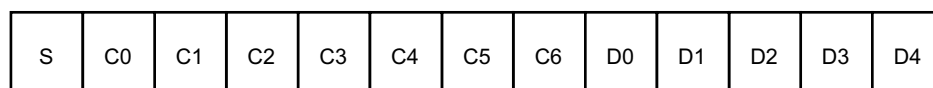
S1, S2: Start-bits (always 1)  
T: Toggle bit  
A4..A0: Address (or system) bits  
C5..C0: Command bits

The toggle bit T changes when a new command is transmitted to detect when the same key is pressed twice (effectively receiving the same data from the host consecutively). A brief delay in the transmission of the same command is detected by the use of the toggle bit because a code is sent while the Host CPU transmits characters to the UART for transmission. The address bits define the machine or device for which the infrared transmission is intended, and the command defines the operation.

To accommodate an extended RC-5 format, the S2 bit is replaced by an additional command bit (C6) that lets the command range increase to 7 bits. This format is known as the extended RC-5 format.

The SIRC encoding uses the duration of modulation for mark and space; therefore, the duration of data bits in the standard frame length varies.

Figure 12-107 shows the packet format and bit encoding. As Figure 12-108 shows, 1 start-bit of 2.4 ms and control codes are followed by data that constitute the entire frame.

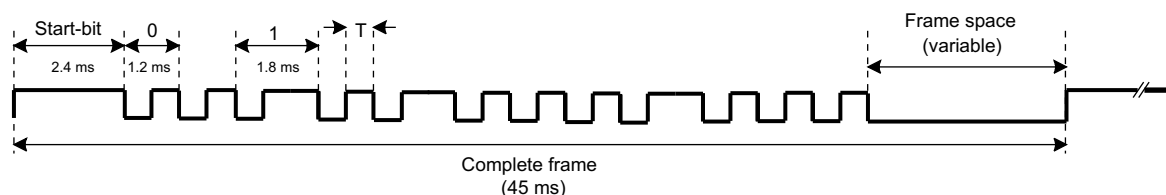


uart-019

**Figure 12-107. UART SIRC Packet Format**

### Note

The encoding must take a standard duration, but the contents of the data can vary. This implies that the control software for sending and receiving data packets must exercise a scheme of interpacket delay, where successive packets can be sent only after a real-time delay expires.



uart-020

**Figure 12-108. UART SIRC Bit Transmission Example**

---

**Note**

This document does not describe all encoding methods and techniques; the previous information discusses the considerations required to employ different encoding methods for different industry-standard protocols. See industry-standard documentation for specific methods of encoding and protocol use.

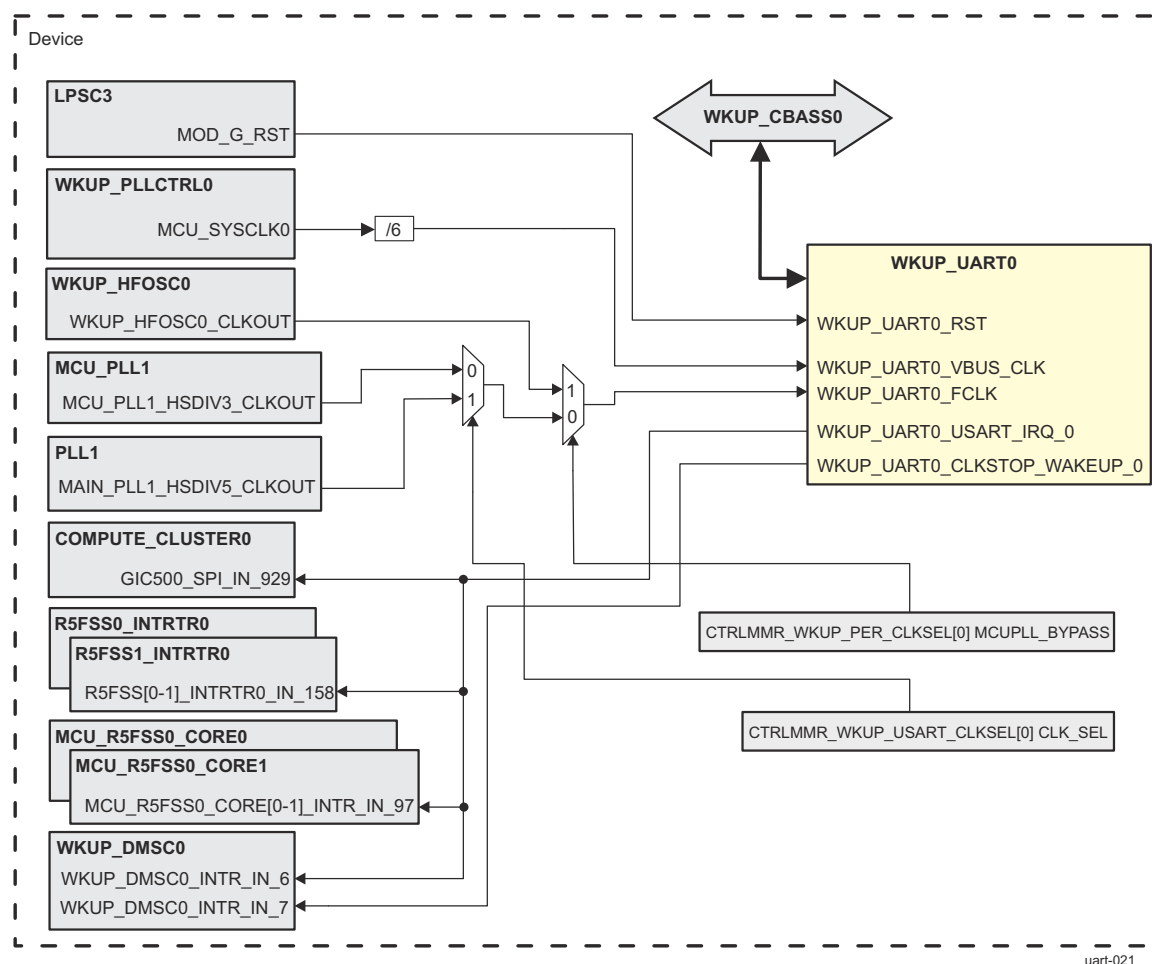
---

### 12.1.6.3 UART Integration

This section describes UART integration in the device, including information about clocks, resets, and hardware requests.

#### 12.1.6.3.1 UART Integration in WKUP Domain

A single WKUP\_UART0 module is integrated in the device WKUP domain. [Figure 12-109](#) shows the integration of WKUP\_UART0.



**Figure 12-109. WKUP\_UART0 Integration**

[Table 12-110](#) through [Table 12-112](#) summarize the integration of WKUP\_UART0 in the device WKUP domain.

**Table 12-110. WKUP\_UART0 Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
WKUP_UART0	WKUP_PSC0	PD0	LPSC3	WKUP_CBASS0

**Table 12-111. WKUP\_UART0 Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
WKUP_UART0	WKUP_UART0_CLK	MCU_SYSCCLK0/6	WKUP_PLLCTRL0	WKUP_UART0 interface clock
	WKUP_UART0_FCLK	MCU_PLL1_HSDIV3_CLKOUT	MCU_PLL1	WKUP_UART0 functional clock. Output of multiplexer, see <a href="#">Figure 12-109</a> , <i>WKUP_UART0 Integration</i> . Multiplexers control is provided via CTRLMMR_WKUP_USART_CLKSEL[0] CLK_SEL and CTRLMMR_WKUP_PER_CLKSEL[0] MCUPLL_BYPASS bit fields.
		MAIN_PLL1_HSDIV5_CLKOUT	PLL1	
		WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
WKUP_UART0	WKUP_UART0_RST	MOD_G_RST	LPSC3	WKUP_UART0 reset

**Table 12-112. WKUP\_UART0 Hardware Requests**

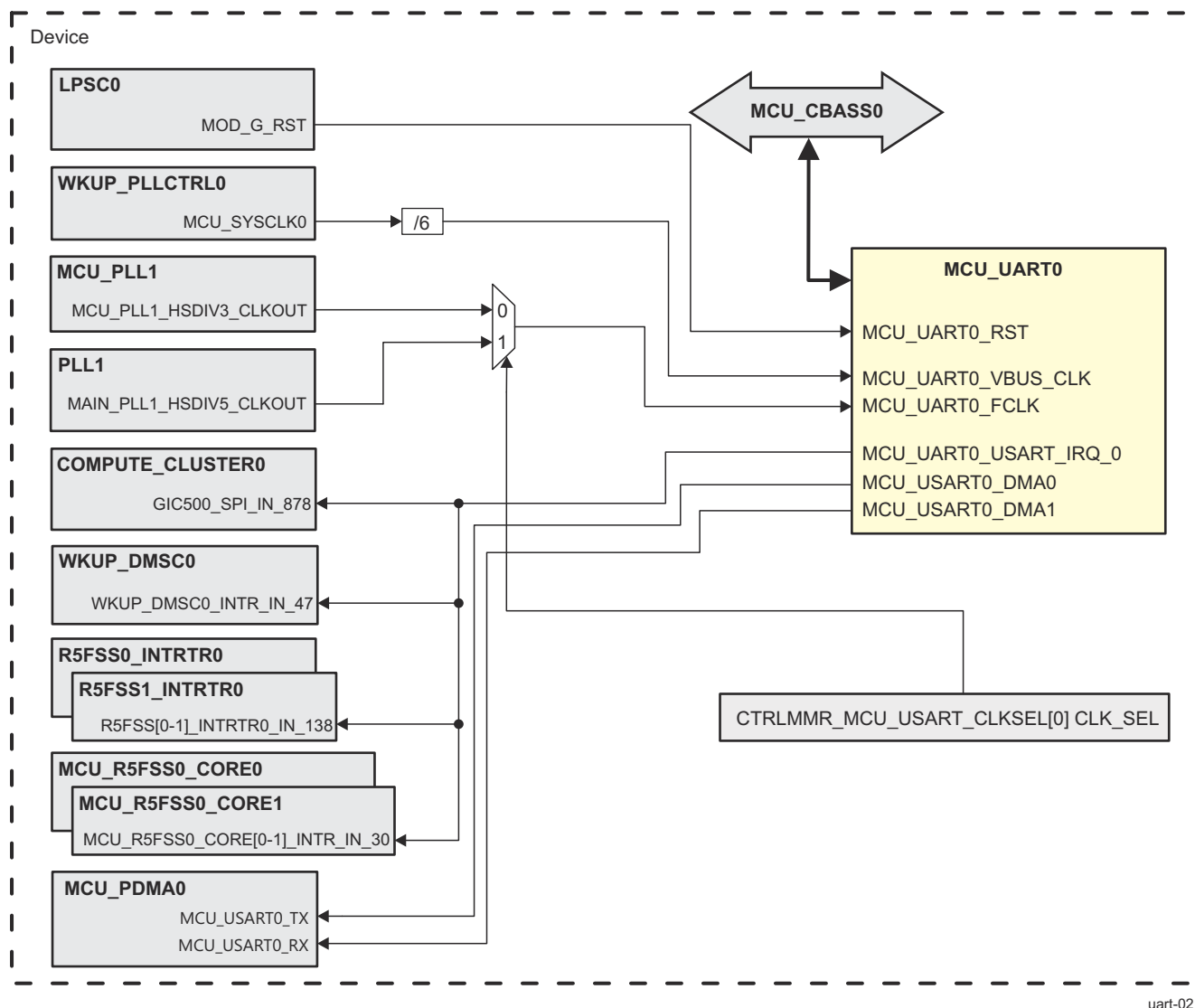
Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
WKUP_UART0	WKUP_UART0_USART_IRQ_0	GIC500_SPI_IN_929	COMPUTE_CLUSTER0	WKUP_UART0 interrupt request.	Level
		WKUP_DMSC0_INTR_IN_6	WKUP_DMSC0		
		R5FSS0_INTRTR0_IN_158	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_158	R5FSS1_INTRTR0		
		MCU_R5FSS0_CORE0_INTR_IN_97	MCU_R5FSS0_CORE0		
		MCU_R5FSS0_CORE1_INTR_IN_97	MCU_R5FSS0_CORE1		
	WKUP_UART0_CLKSTOP_WAKEUP_0	WKUP_DMSC0_INTR_IN_7	WKUP_DMSC0	WKUP_UART0 wakeup interrupt.	Pulse

### Note

UART interrupts are further described in [Section 12.1.6.4.5](#), *UART Interrupt Requests*.

### 12.1.6.3.2 UART Integration in MCU Domain

A single MCU\_UART0 module is integrated in the device MCU domain. Figure 12-110 shows the integration of MCU\_UART0.



uart-021

Figure 12-110. MCU\_UART0 Integration



Table 12-113 through Table 12-115 summarize the integration of MCU\_UART0 in the device MCU domain.

**Table 12-113. MCU\_UART0 Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
MCU_UART0	WKUP_PSC0	PD0	LPSC0	MCU_CBASS0

**Table 12-114. MCU\_UART0 Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
MCU_UART0	MCU_UART0_CLK	MCU_SYSCLK0/6	WKUP_PLLCTRL0	MCU_UART0 interface clock
	MCU_UART0_FCLK	MCU_PLL1_HSDIV3_CLKOUT	MCU_PLL1	MCU_UART0 functional clock. Output of multiplexer, see <a href="#">Figure 12-110</a> , <i>MCU_UART0 Integration</i> . Multiplexer control is provided via CTRLMMR_MCU_USART_CLKSEL[0] CLK_SEL bit field.
		MAIN_PLL1_HSDIV5_CLKOUT	PLL1	
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
MCU_UART0	MCU_UART0_RST	MOD_G_RST	LPSC0	MCU_UART0 reset

**Table 12-115. MCU\_UART0 Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCU_UART0	MCU_UART0_USART_IRQ_0	GIC500_SPI_IN_878	COMPUTE_CLUSTER0	MCU_UART0 interrupt request	Level
		WKUP_DMSC0_INTR_IN_47	WKUP_DMSC0		
		R5FSS0_INTRTR0_IN_138	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_138	R5FSS1_INTRTR0		
		MCU_R5FSS0_CORE0_INTR_IN_30	MCU_R5FSS0_CORE0		
		MCU_R5FSS0_CORE1_INTR_IN_30	MCU_R5FSS0_CORE1		
DMA Events					
Module Instance	Module DMA Event	Destination DMA Event Input	Destination	Description	Type
MCU_UART0	MCU_USART0_DMA0	MCU_USART0_TX	MCU_PDMA0	MCU_UART0 transmit request line	Level
	MCU_USART0_DMA1	MCU_USART0_RX	MCU_PDMA0	MCU_UART0 receive request line	Level

---

**Note**

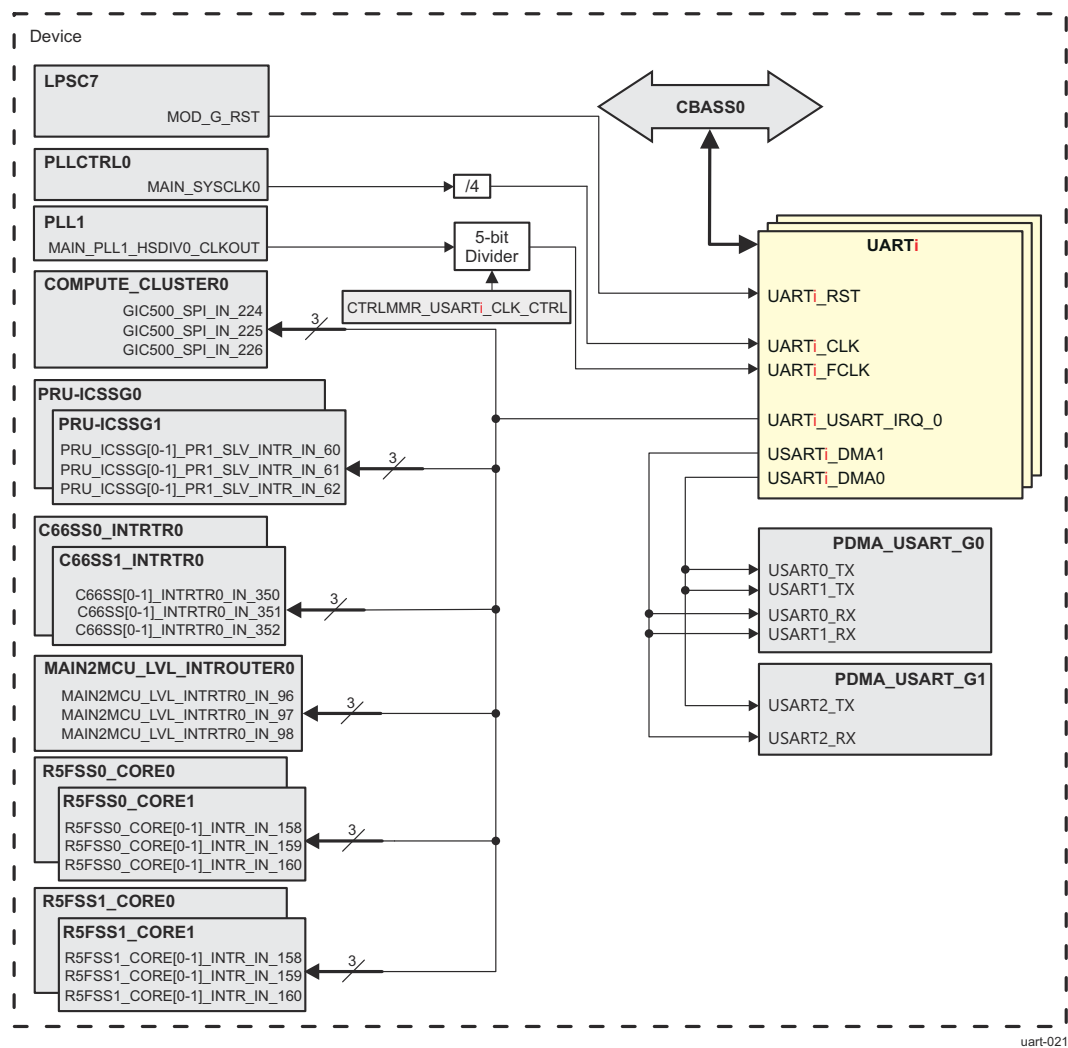
UART interrupts are further described in [Section 12.1.6.4.5](#), *UART Interrupt Requests*.

UART DMA events are further described in [Section 12.1.6.4.6.4](#), *FIFO DMA Mode Operation*.

---

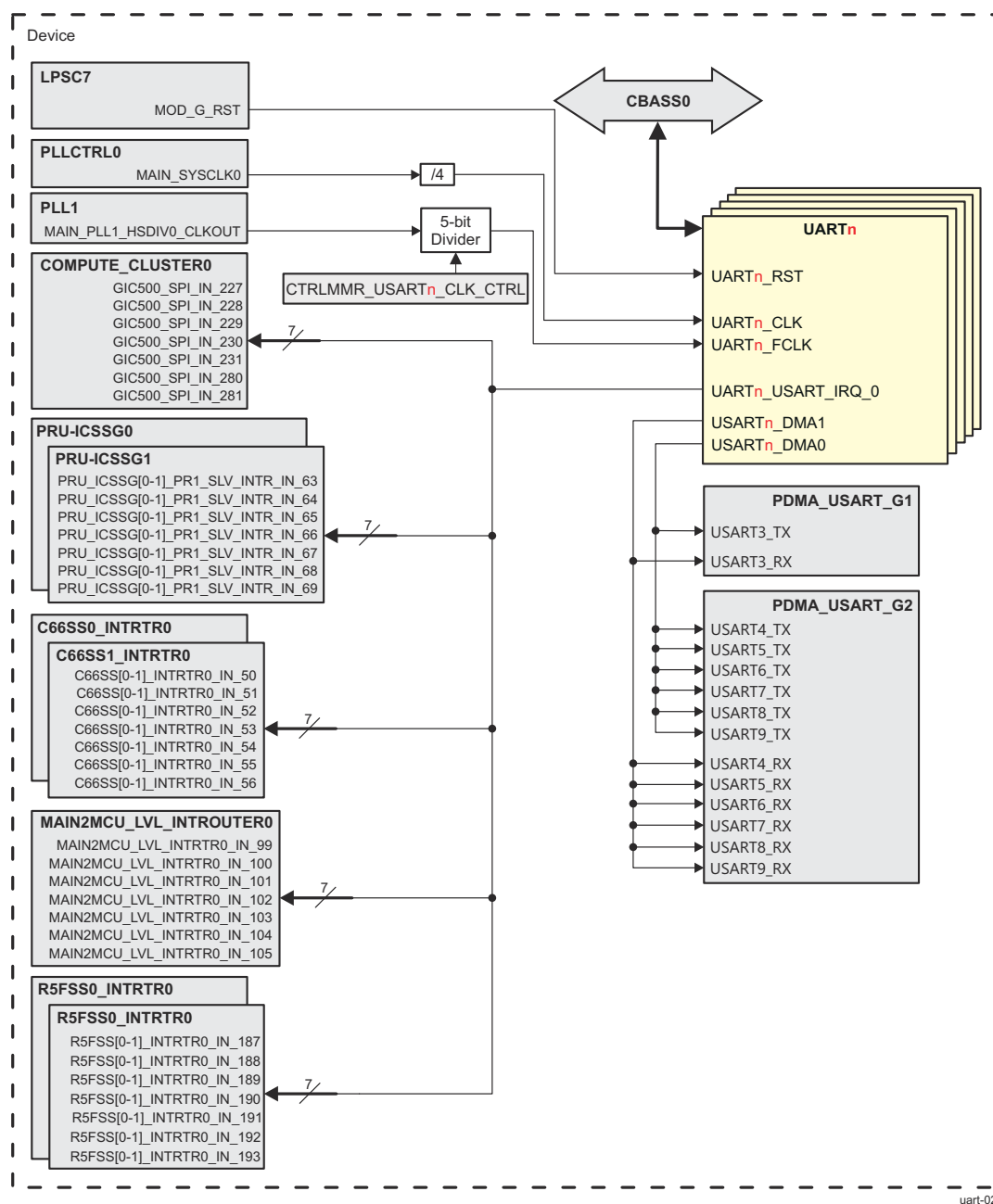
**12.1.6.3.3 UART Integration in MAIN Domain**

There are ten UART modules integrated in the device MAIN domain - UART0, UART1, UART2, UART3, UART4, UART5, UART6, UART7, UART8, and UART9. [Figure 12-111](#) shows the integration of UART[0-2]. [Figure 12-112](#) shows the integration of UART[3-9].



A.  $i = 0$  to 2

Figure 12-111. UART[0-2] Integration



A n = 3 to 9

Table 12-116 through Table 12-118 summarize the integration of UART[0-9] in the device MAIN domain.

**Table 12-116. UART Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
UART0	PSC0	PD0	LPSC7	CBASS0
UART1	PSC0	PD0	LPSC7	CBASS0
UART2	PSC0	PD0	LPSC7	CBASS0
UART3	PSC0	PD0	LPSC7	CBASS0
UART4	PSC0	PD0	LPSC7	CBASS0
UART5	PSC0	PD0	LPSC7	CBASS0
UART6	PSC0	PD0	LPSC7	CBASS0
UART7	PSC0	PD0	LPSC7	CBASS0
UART8	PSC0	PD0	LPSC7	CBASS0
UART9	PSC0	PD0	LPSC7	CBASS0

**Table 12-117. UART Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
UART0	UART0_CLK	MAIN_SYSCLK0/4	PLLCTRL0	UART0 interface clock
	UART0_FCLK	MAIN_PLL1_HSDIV0_CLKOUT/DIV <sup>(1)</sup>	PLL1	UART0 functional clock
UART1	UART1_CLK	MAIN_SYSCLK0/4	PLLCTRL0	UART1 interface clock
	UART1_FCLK	MAIN_PLL1_HSDIV0_CLKOUT/DIV <sup>(1)</sup>	PLL1	UART1 functional clock
UART2	UART2_CLK	MAIN_SYSCLK0/4	PLLCTRL0	UART2 interface clock
	UART2_FCLK	MAIN_PLL1_HSDIV0_CLKOUT/DIV <sup>(1)</sup>	PLL1	UART2 functional clock
UART3	UART3_CLK	MAIN_SYSCLK0/4	PLLCTRL0	UART3 interface clock
	UART3_FCLK	MAIN_PLL1_HSDIV0_CLKOUT/DIV <sup>(1)</sup>	PLL1	UART3 functional clock
UART4	UART4_CLK	MAIN_SYSCLK0/4	PLLCTRL0	UART4 interface clock
	UART4_FCLK	MAIN_PLL1_HSDIV0_CLKOUT/DIV <sup>(1)</sup>	PLL1	UART4 functional clock
UART5	UART5_CLK	MAIN_SYSCLK0/4	PLLCTRL0	UART5 interface clock
	UART5_FCLK	MAIN_PLL1_HSDIV0_CLKOUT/DIV <sup>(1)</sup>	PLL1	UART5 functional clock
UART6	UART6_CLK	MAIN_SYSCLK0/4	PLLCTRL0	UART6 interface clock
	UART6_FCLK	MAIN_PLL1_HSDIV0_CLKOUT/DIV <sup>(1)</sup>	PLL1	UART6 functional clock
UART7	UART7_CLK	MAIN_SYSCLK0/4	PLLCTRL0	UART7 interface clock
	UART7_FCLK	MAIN_PLL1_HSDIV0_CLKOUT/DIV <sup>(1)</sup>	PLL1	UART7 functional clock

**Table 12-117. UART Clocks and Resets (continued)**

UART8	UART8_CLK	MAIN_SYSCLK0/4	PLLCTRL0	UART8 interface clock
	UART8_FCLK	MAIN_PLL1_HSDIV0_CLKOUT/DIV <sup>(1)</sup>	PLL1	UART8 functional clock
UART9	UART9_CLK	MAIN_SYSCLK0/4	PLLCTRL0	UART9 interface clock
	UART9_FCLK	MAIN_PLL1_HSDIV0_CLKOUT/DIV <sup>(1)</sup>	PLL1	UART9 functional clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
UART0	UART0_RST	MOD_G_RST	LPSC7	UART0 reset
UART1	UART1_RST	MOD_G_RST	LPSC7	UART1 reset
UART2	UART2_RST	MOD_G_RST	LPSC7	UART2 reset
UART3	UART3_RST	MOD_G_RST	LPSC7	UART3 reset
UART4	UART4_RST	MOD_G_RST	LPSC7	UART4 reset
UART5	UART5_RST	MOD_G_RST	LPSC7	UART5 reset
UART6	UART6_RST	MOD_G_RST	LPSC7	UART6 reset
UART7	UART7_RST	MOD_G_RST	LPSC7	UART7 reset
UART8	UART8_RST	MOD_G_RST	LPSC7	UART8 reset
UART9	UART9_RST	MOD_G_RST	LPSC7	UART9 reset

(1) Programmable 5-bit divider. Default set to DIV by 4 ("11") to get 48 MHz clock.

**Table 12-118. UART Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
UART0	UART0_USART_IRQ_0	GIC500_SPI_IN_224	COMPUTE_CLUSTER0	UART0 interrupt request	Level
		PRU_ICSSG0_PR1_SLV_INTR_IN_60	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INTR_IN_60	PRU-ICSSG1		
		C66SS0_INTRTR0_IN_350	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_350	C66SS1_INTRTR0		
		MAIN2MCU_LVL_INTRTR0_IN_96	MAIN2MCU_LVL_INTRTR0		
		R5FSS0_CORE0_INTR_IN_158	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_158	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_158	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_158	R5FSS1_CORE1		
UART1	UART1_USART_IRQ_0	GIC500_SPI_IN_225	COMPUTE_CLUSTER0	UART1 interrupt request	Level
		PRU_ICSSG0_PR1_SLV_INTR_IN_61	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INTR_IN_61	PRU-ICSSG1		
		C66SS0_INTRTR0_IN_351	C66SS0_INTRTR0		

**Table 12-118. UART Hardware Requests (continued)**

UART2	UART2_USART_IRQ_0	C66SS1_INTRTR0_IN_351	C66SS1_INTRTR0	UART2 interrupt request	Level
		MAIN2MCU_LVL_INTRTR0_IN_97	MAIN2MCU_LVL_INTRTR0		
		R5FSS0_CORE0_INTR_IN_159	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_159	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_159	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_159	R5FSS1_CORE1		
		GIC500_SPI_IN_226	COMPUTE_CLUSTER0		
		PRU_ICSSG0_PR1_SLV_INTR_IN_62	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INTR_IN_62	PRU-ICSSG1		
		C66SS0_INTRTR0_IN_352	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_352	C66SS1_INTRTR0		
		MAIN2MCU_LVL_INTRTR0_IN_98	MAIN2MCU_LVL_INTRTR0		
		R5FSS0_CORE0_INTR_IN_160	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_160	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_160	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_160	R5FSS1_CORE1		
UART3	UART3_USART_IRQ_0	GIC500_SPI_IN_227	COMPUTE_CLUSTER0	UART3 interrupt request	Level
		PRU_ICSSG0_PR1_SLV_INTR_IN_63	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INTR_IN_63	PRU-ICSSG1		
		C66SS0_INTRTR0_IN_50	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_50	C66SS1_INTRTR0		
		R5FSS0_INTRTR0_IN_187	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_187	R5FSS1_INTRTR0		
		MAIN2MCU_LVL_INTRTR0_IN_99	MAIN2MCU_LVL_INTRTR0		
UART4	UART4_USART_IRQ_0	GIC500_SPI_IN_228	COMPUTE_CLUSTER0	UART4 interrupt request	Level
		PRU_ICSSG0_PR1_SLV_INTR_IN_64	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INTR_IN_64	PRU-ICSSG1		
		C66SS0_INTRTR0_IN_51	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_51	C66SS1_INTRTR0		
		R5FSS0_INTRTR0_IN_188	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_188	R5FSS1_INTRTR0		
		MAIN2MCU_LVL_INTRTR0_IN_100	MAIN2MCU_LVL_INTRTR0		
UART5	UART5_USART_IRQ_0	GIC500_SPI_IN_229	COMPUTE_CLUSTER0	UART5 interrupt request	Level
		PRU_ICSSG0_PR1_SLV_INTR_IN_65	PRU-ICSSG0		

**Table 12-118. UART Hardware Requests (continued)**

		PRU_ICSSG1_PR1_SLV_INTR_IN_65	PRU-ICSSG1		
		C66SS0_INTRTR0_IN_52	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_52	C66SS1_INTRTR0		
		R5FSS0_INTRTR0_IN_189	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_189	R5FSS1_INTRTR0		
		MAIN2MCU_LVL_INTRTR0_IN_101	MAIN2MCU_LVL_INTRTR0		
UART6	UART6_USART_IRQ_0	GIC500_SPI_IN_230	COMPUTE_CLUSTER0	UART6 interrupt request	Level
		PRU_ICSSG0_PR1_SLV_INTR_IN_66	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INTR_IN_66	PRU-ICSSG1		
		C66SS0_INTRTR0_IN_53	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_53	C66SS1_INTRTR0		
		R5FSS0_INTRTR0_IN_190	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_190	R5FSS1_INTRTR0		
		MAIN2MCU_LVL_INTRTR0_IN_102	MAIN2MCU_LVL_INTRTR0		
UART7	UART7_USART_IRQ_0	GIC500_SPI_IN_231	COMPUTE_CLUSTER0	UART7 interrupt request	Level
		PRU_ICSSG0_PR1_SLV_INTR_IN_67	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INTR_IN_67	PRU-ICSSG1		
		C66SS0_INTRTR0_IN_54	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_54	C66SS1_INTRTR0		
		R5FSS0_INTRTR0_IN_191	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_191	R5FSS1_INTRTR0		
		MAIN2MCU_LVL_INTRTR0_IN_103	MAIN2MCU_LVL_INTRTR0		
UART8	UART8_USART_IRQ_0	GIC500_SPI_IN_280	COMPUTE_CLUSTER0	UART8 interrupt request	Level
		PRU_ICSSG0_PR1_SLV_INTR_IN_68	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INTR_IN_68	PRU-ICSSG1		
		C66SS0_INTRTR0_IN_55	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_55	C66SS1_INTRTR0		
		R5FSS0_INTRTR0_IN_192	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_192	R5FSS1_INTRTR0		
		MAIN2MCU_LVL_INTRTR0_IN_104	MAIN2MCU_LVL_INTRTR0		
UART9	UART9_USART_IRQ_0	GIC500_SPI_IN_281	COMPUTE_CLUSTER0	UART9 interrupt request	Level
		PRU_ICSSG0_PR1_SLV_INTR_IN_69	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INTR_IN_69	PRU-ICSSG1		
		C66SS0_INTRTR0_IN_56	C66SS0_INTRTR0		



**Table 12-118. UART Hardware Requests (continued)**

C66SS1_INTRTR0_IN_56	C66SS1_INTRTR0
R5FSS0_INTRTR0_IN_193	R5FSS0_INTRTR0
R5FSS1_INTRTR0_IN_193	R5FSS1_INTRTR0
MAIN2MCU_LVL_INTRTR0_IN_105	MAIN2MCU_LVL_INTRTR0

DMA Events					
Module Instance	Module DMA Event	Destination DMA Event Input	Destination	Description	Type
UART0	USART0_DMA0	USART0_TX	PDMA_USART_G0	UART0 transmit request line	Level
	USART0_DMA1	USART0_RX	PDMA_USART_G0	UART0 receive request line	Level
UART1	USART1_DMA0	USART1_TX	PDMA_USART_G0	UART1 transmit request line	Level
	USART1_DMA1	USART1_RX	PDMA_USART_G0	UART1 receive request line	Level
UART2	USART2_DMA0	USART2_TX	PDMA_USART_G1	UART2 transmit request line	Level
	USART2_DMA1	USART2_RX	PDMA_USART_G1	UART2 receive request line	Level
UART3	USART3_DMA0	USART3_TX	PDMA_USART_G1	UART3 transmit request line	Level
	USART3_DMA1	USART3_RX	PDMA_USART_G1	UART3 receive request line	Level
UART4	USART4_DMA0	USART4_TX	PDMA_USART_G2	UART4 transmit request line	Level
	USART4_DMA1	USART4_RX	PDMA_USART_G2	UART4 receive request line	Level
UART5	USART5_DMA0	USART5_TX	PDMA_USART_G2	UART5 transmit request line	Level
	USART5_DMA1	USART5_RX	PDMA_USART_G2	UART5 receive request line	Level
UART6	USART6_DMA0	USART6_TX	PDMA_USART_G2	UART6 transmit request line	Level
	USART6_DMA1	USART6_RX	PDMA_USART_G2	UART6 receive request line	Level
UART7	USART7_DMA0	USART7_TX	PDMA_USART_G2	UART7 transmit request line	Level
	USART7_DMA1	USART7_RX	PDMA_USART_G2	UART7 receive request line	Level
UART8	USART8_DMA0	USART8_TX	PDMA_USART_G2	UART8 transmit request line	Level
	USART8_DMA1	USART8_RX	PDMA_USART_G2	UART8 receive request line	Level
UART9	USART9_DMA0	USART9_TX	PDMA_USART_G2	UART9 transmit request line	Level
	USART9_DMA1	USART9_RX	PDMA_USART_G2	UART9 receive request line	Level

## 12.1.6.4 UART Functional Description

### 12.1.6.4.1 UART Block Diagram

The UART module can be divided into three main blocks:

- FIFO management
- Mode selection
- Protocol formatting

FIFO management is common to all functions and enables the transmission and reception of data from the host processor point of view.

There are two modes:

- Function mode: Routes the data to the chosen function (UART, RS-485, IrDA, or CIR) and enables the mechanism corresponding to the chosen function.
- Register mode: Enables conditional access to registers.

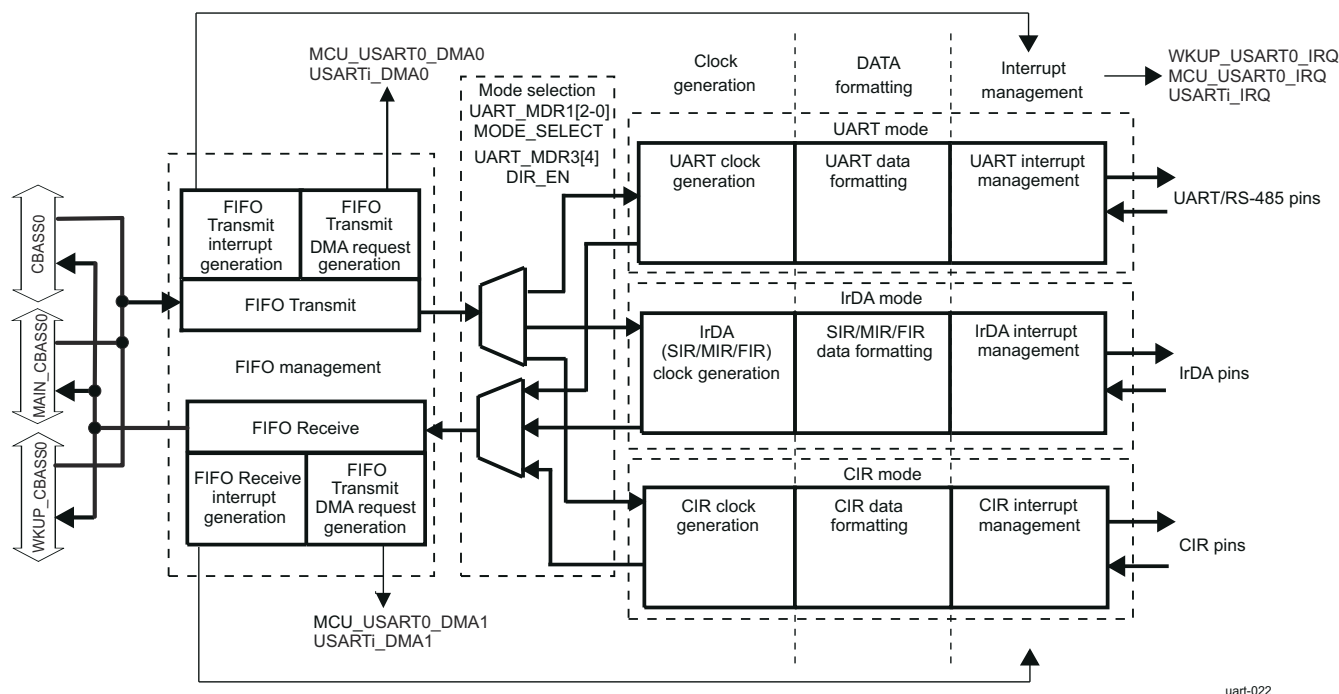
For more information about mode configuration, see *Mode Selection*.

Protocol formatting has three subcategories:

- Clock generation: The 48-MHz input clock generates all necessary clocks.
- Data formatting: Each function uses a dedicated state-machine that is responsible for the transition between FIFO data and the associated frame data.
- Interrupt management: Different interrupt types are generated depending on the chosen function. In each mode, when an interrupt is generated, the UART\_IIR\_UART register indicates the interrupt type.
  - UART mode interrupts: Seven interrupts prioritized in six different levels
  - IrDA mode interrupts: Eight interrupts. The interrupt line is activated when any interrupt is generated (there is no priority).
  - CIR mode interrupts: A subset of existing IrDA mode interrupts is used.

In parallel with these functional blocks, a power-saving strategy exists for each function.

The UART block diagram is shown below.



**Figure 12-113. UART Functional Block Diagram**

#### 12.1.6.4.2 UART Clock Configuration

Each UART uses a 48-MHz functional clock for its logic and to generate external interface signals. Each UART uses an interface clock for register accesses.

#### 12.1.6.4.3 UART Software Reset

The UART\_SYSC[1] SOFTRESET bit controls the software reset; setting this bit to 1 triggers a software reset functionally equivalent to hardware reset.

##### 12.1.6.4.3.1 Independent TX/RX

The receiver and transmitter are enabled by default after reset. Software can choose to disable, re-enable or to reset either the RX or the TX side independently of the other through the UART\_ECR register.

#### 12.1.6.4.4 UART Power Management

##### 12.1.6.4.4.1 UART Mode Power Management

##### 12.1.6.4.4.1.1 Module Power Saving

In UART modes, sleep mode is enabled by setting the UART\_IER\_UART[4] SLEEP\_MODE bit to 1 (when the UART\_EFR[4] ENHANCED\_EN bit is set to 1).

Sleep mode is entered when all of the following conditions exist:

- The serial data input line, RX, is idle.
- The TX FIFO and TX shift register are empty.
- The RX FIFO is empty.
- The only pending interrupts are THR interrupts.

Sleep mode is a good way to lower UART power consumption, but this state can be achieved only when the UART is set to modem mode. Therefore, even if the UART has no key role functionally, it must be initialized in a functional mode to take advantage of sleep mode.

In sleep mode, the module clock and baud rate clock are stopped internally. Because most registers are clocked by these clocks, this greatly reduces power consumption. The module wakes up when a change is detected on the RX line, when data is written to the TX FIFO, and when there is a change in the state of the modem input pins.

An interrupt can be generated on a wake-up event by setting the UART\_SCR[4] RX\_CTS\_WU\_EN bit to 1. To understand how to manage the interrupt, see [Section 12.1.6.4.5.1.2, Wake-Up Interrupt](#).

---

#### Note

There must be no writing to the divisor latches, UART\_DLL and UART\_DLH, to set the baud clock (BCLK) while in sleep mode. It is advisable to disable sleep mode using the UART\_IER\_UART[4] SLEEP\_MODE bit before writing to the UART\_DLL or UART\_DLH register.

---

##### 12.1.6.4.4.1.2 System Power Saving

Sleep and auto-idle modes are embedded power-saving features. Power-reduction techniques can be applied at the system level by shutting down certain internal clock and power domains of the device.

For more information, see *Power*, in the *Device Configuration*.

##### 12.1.6.4.4.2 IrDA Mode Power Management

##### 12.1.6.4.4.2.1 Module Power Saving

In IrDA modes, sleep mode is enabled by setting the UART\_MDR1[3] IR\_SLEEP bit to 1.

Sleep mode is entered when all of the following conditions exist:

- The serial data input line, RXD, is idle.
- The TX FIFO and TX shift register are empty.

- The RX FIFO is empty.
- No interrupts are pending except THR interrupts.

The module wakes up when a change is detected on the RXD line or when data is written to the TX FIFO.

#### 12.1.6.4.4.2.2 System Power Saving

System power saving for the IrDA mode has the same function as for the UART mode (see [Section 12.1.6.4.4.1.2, System Power Saving](#)).

#### 12.1.6.4.4.3 CIR Mode Power Management

##### 12.1.6.4.4.3.1 Module Power Saving

Module power saving for the CIR mode has the same function as for the IrDA mode (see [Section 12.1.6.4.4.2.1, Module Power Saving](#)).

##### 12.1.6.4.4.3.2 System Power Saving

System power saving for the CIR mode has the same function as for the UART mode (see [Section 12.1.6.4.4.1.2, System Power Saving](#)).

#### 12.1.6.4.4.4 Local Power Management

[Table 12-119](#) describes power-management features available for the UART.

#### Note

For information about source clock gating and the sleep/wake-up transitions description, see *Power*, in the *Device Configuration*.

**Table 12-119. UART Local Power-Management Features**

Feature	Registers	Description
Clock autogating		
Peripheral idle modes		
Clock activity	N/A	Feature not available
Controller standby modes	N/A	Feature not available
Global wake-up enable	UART_SYSC[2] ENAWAKEUP	This bit enables the wake-up feature at module level.
Wake-Up sources enable	N/A	Feature not available

#### 12.1.6.4.5 UART Interrupt Requests

##### 12.1.6.4.5.1 UART Mode Interrupt Management

##### 12.1.6.4.5.1.1 UART Interrupts

The UART mode includes seven possible interrupts prioritized to six levels.

When an interrupt is generated, the interrupt identification register (UART\_IIR\_UART) sets the UART\_IIR\_UART[0] IT\_PENDING bit to 0 to indicate that an interrupt is pending, and indicates the type of interrupt through the UART\_IIR\_UART[5-1] bit field. [Table 12-120](#) summarizes the interrupt control functions.

**Table 12-120. UART Mode Interrupts**

IIR[5:0]	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Method
000001	N/A	No Interrupt	N/A	N/A
000110	1	Receiver line status	OE, FE, PE, or BI errors occur in characters in the RX FIFO.	FE, PE, BI: Read the UART_RHR register. OE: Read the UART_LSR_UART register.

**Table 12-120. UART Mode Interrupts (continued)**

IIR[5:0]	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Method
001100	2	RX time-out	Stale data in RX FIFO	Read the UART_RHR register if using the default timeout behavior: EFR2[6]=0 Cleared by reading its value (IIR) if using the periodic timeout behavior: EFR2[6]=1
000100	2	RHR interrupt	DRDY (data ready) (FIFO disabled) RX FIFO above trigger level (FIFO enabled)	Read the UART_RHR register until the interrupt condition disappears
000010	3	THR interrupt	TFE (THR empty) (FIFO disabled) TX FIFO below trigger level (FIFO enabled)	Write to the UART_THR until the interrupt condition disappears
000000	4	Modem status	See the UART_MSR register.	Read the MSR register
010000	5	XOFF interrupt/ special character interrupt	Receive XOFF characters/special character	Receive XON character(s), if XOFF interrupt/read of the UART_IIR_UART register, if special character interrupt
100000	6	CTS, RTS	RTS pin or CTS pin change state from active (low) to inactive (high)	Read the UART_IIR_UART register

For the receiver-line status interrupt, the UART\_LSR\_UART[7] RX\_FIFO\_STS bit generates the interrupt.

For the XOFF interrupt, if an XOFF flow character detection caused the interrupt, the interrupt is cleared by an XON flow character detection. If special character detection caused the interrupt, the interrupt is cleared by a read of the UART\_IIR\_UART register.

#### 12.1.6.4.5.1.2 Wake-Up Interrupt

Wake-up interrupt is a special interrupt that works differently from other interrupts. This interrupt is enabled when the RX\_CTS\_DSR\_WAKE\_UP\_ENABLE bit of the Supplementary Control Register (SCR[4]) is set to 1. The IIR register is not modified when it occurs, SSR[1] must be checked to detect a wake-up event. When a wake-up event occurs, the only way to clear it is to reset SCR[4] to 0. Wake-up can also occur if the WER[7] TX\_WAKEUP\_EN is set to 1 and one of the following events occurs:

1. THR interrupt is enabled and occurs (omitted if TX DMA request is enabled)
2. TX DMA request is enabled and occurs
3. TX\_STATUS\_IT is enabled and occurs (only IrDA and CIR modes). Cannot be used with THR Interrupt.

#### 12.1.6.4.5.2 IrDA Mode Interrupt Management

##### 12.1.6.4.5.2.1 IrDA Interrupts

The IrDA function generates interrupts. All interrupts can be enabled and disabled by writing to the appropriate bit in the interrupt enable register (UART\_IER\_IRDA). The interrupt status of the device can be checked by reading the interrupt identification register (UART\_IIR\_IRDA).

The UART, IrDA, and CIR modes have different interrupts in the UART module and, therefore, different UART\_IER\_IRDA and UART\_IIR\_IRDA mappings, depending on the selected mode.

The IrDA modes have eight possible interrupts (see [Table 12-121](#)). The interrupt line is activated when any interrupt is generated (there is no priority).

**Table 12-121. IrDA Mode Interrupts**

IIR_IRDA Bit	Interrupt Type	Interrupt Source	Interrupt Reset Method
0	RHR interrupt	DRDY (data ready) (FIFO disabled) RX FIFO above trigger level (FIFO enabled)	Read the UART_RHR register until the interrupt condition disappears.
1	THR interrupt	TFE (UART_THR empty) (FIFO disabled) TX FIFO below trigger level (FIFO enabled)	Write to the UART_THR until the interrupt condition disappears.

**Table 12-121. IrDA Mode Interrupts (continued)**

IIR_IRDA Bit	Interrupt Type	Interrupt Source	Interrupt Reset Method
2	Last byte in RX FIFO	Last byte of frame in RX FIFO is available to be read at the UART_RHR port.	Read the UART_RHR register.
3	RX overrun	Write to the UART_RHR register when the RX FIFO is full.	Read UART_RESUME register.
4	Status FIFO interrupt	Status FIFO triggers level reached.	Read STATUS FIFO.
5	TX status	UART_THR empty before EOF sent. Last bit of transmission of the IrDA frame occurred, but with an underrun error OR Transmission of the last bit of the IrDA frame completed successfully.	Read the UART_RESUME register OR Read the UART_IIR_IRDA register.
6	Receiver line status interrupt	CRC, ABORT, or frame-length error is written into the STATUS FIFO.	Read the STATUS FIFO (read until empty - maximum of eight reads required).
7	Received EOF	Received end-of-frame	Read the UART_IIR_IRDA register.

**12.1.6.4.5.2.2 Wake-Up Interrupts**

The wake-up interrupt for IrDA mode has the same function as that for UART mode (see [Section 12.1.6.4.5.1.2, Wake-Up Interrupt](#)).

**12.1.6.4.5.3 CIR Mode Interrupt Management****12.1.6.4.5.3.1 CIR Interrupts**

The CIR function generates interrupts that can be enabled and disabled by writing to the appropriate bit in the interrupt enable register (UART\_IER\_CIR). The interrupt status of the device can be checked by reading the interrupt identification register (UART\_IIR\_CIR).

The UART, IrDA, and CIR modes have different interrupts in the UART module and, therefore, different UART\_IER\_CIR and UART\_IIR\_CIR mappings, depending on the selected mode.

[Table 12-122](#) lists the interrupt modes to be maintained. In CIR mode, the sole purpose of the UART\_IIR\_CIR[5] TX\_STATUS\_IT bit is to indicate that the last bit of infrared data was passed to the TX pin.

**Table 12-122. CIR Mode Interrupts**

IIR_CIR Bit Number	Interrupt Type	Interrupt Source	Interrupt Reset Method
0			
1	THR interrupt	TFE (THR empty) (FIFO disabled) TX FIFO below trigger level (FIFO enabled)	Write to the UART_THR register until the interrupt condition disappears
2			
3			
4	N/A for CIR mode	N/A for CIR mode	N/A for CIR mode
5	TX status	Transmission of the last bit of the frame is complete successfully	Read the UART_IIR_CIR register
6	N/A for CIR mode	N/A for CIR mode	N/A for CIR mode
7	N/A for CIR mode	N/A for CIR mode	N/A for CIR mode

**12.1.6.4.5.3.2 Wake-Up Interrupts**

The wake-up interrupt for CIR mode has the same function as that for UART mode (see [Section 12.1.6.4.5.1.2, Wake-Up Interrupt](#)).

**12.1.6.4.6 UART FIFO Management**

The FIFO is accessed by reading and writing the UART\_RHR and UART\_THR registers. Parameters are controlled using the FIFO control register (UART\_FCR) and supplementary control register (UART\_SCR). Reading the UART\_SSR[0] TX\_FIFO\_FULL bit at 1 means the FIFO is full.

The UART\_TLR register controls the FIFO trigger level, which enables DMA and interrupt generation. After reset, transmit (TX) and receive (RX) FIFOs are disabled; thus, the trigger level is the default value of 1 byte. [Figure 12-114](#) shows the FIFO management registers.

---

**Note**

Data in the UART\_RHR register is not overwritten when an overflow occurs.

---

---

**Note**

The UART\_SFLSR, UART\_SFREGL, and UART\_SFREGH status registers are used in IrDA mode only. For information about their use, see [Section 12.1.6.4.8.3.3, IrDA Data Formatting](#).

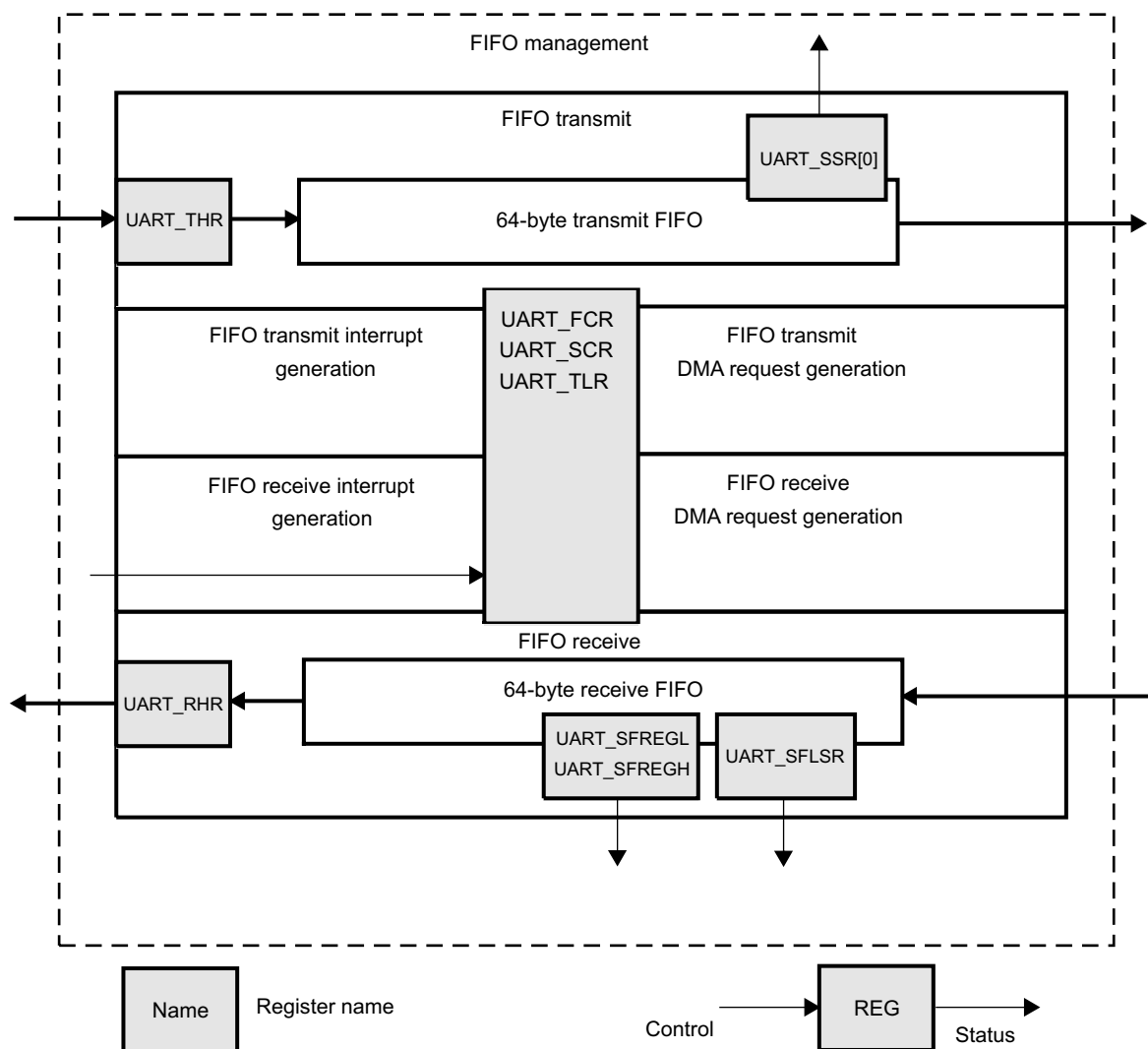
---

---

**Note**

Bits UART\_FCR[2] TX\_FIFO\_CLEAR and UART\_FCR[1] RX\_FIFO\_CLEAR are automatically cleared by hardware after  $4 \times \text{UARTi\_CLK} + 5 \times \text{UARTi\_FCLK}$  clock cycles. This delay is needed to finish the resetting of the corresponding FIFO and DMA control registers.

---



uart-023

**Figure 12-114. UART FIFO Management Registers**

#### 12.1.6.4.6.1 FIFO Trigger

##### 12.1.6.4.6.1.1 Transmit FIFO Trigger

Table 12-123 lists the TX FIFO trigger level settings.

**Table 12-123. UART TX FIFO Trigger Level Setting Summary**

SCR[6]	TLR[3:0]	TX FIFO Trigger Level
0	= 0x0	Defined by the UART_FCR[5-4] TX_FIFO_TRIG bit field (8, 16, 32, or 56 spaces)
0	!= 0x0	Defined by the UART_TLR[3-0] TX_FIFO_TRIG_DMA bit field (from 4 to 60 spaces with a granularity of 4 spaces)
1	Value	Defined by the concatenated value of TLR[3:0] (higher bits) and FCR[5:4] (lower bits) from 1 to 63 spaces with a granularity of 1 space.

**Note:** The combination of TLR[3:0]=0000 and FCR[5:4]=00 (all zeros) is not supported (min 1 space required). All zeros will result in unsupported behavior.

##### 12.1.6.4.6.1.2 Receive FIFO Trigger

Table 12-124 lists the RX FIFO trigger-level settings.



**Table 12-124. UART RX FIFO Trigger-Level Setting Summary**

SCR[7]	TLR[7:4]	RX FIFO Trigger Level
0	= 0x0	Defined by the UART_FCR[7:6] RX_FIFO_TRIG bit field (8, 16, 56, or 60 characters)
0	!= 0x0	Defined by the UART_TLR[7:4] RX_FIFO_TRIG_DMA bit field (from 4 to 60 characters with a granularity of 4 characters)
1	Value	Defined by the concatenated value of TLR[7:4] and FCR[7:6] from 1 to 63 characters with a granularity of one character.  <b>Note:</b> The combination of TLR[7:4]=0000 and FCR[7:6]=00 (all zeros) is not supported (min 1 character required). All zeros will result in unsupported behavior.

The receive threshold is programmed using the UART\_TCR[7:4] RX\_FIFO\_TRIG\_START and UART\_TCR[3:0] RX\_FIFO\_TRIG\_HALT bit fields:

- Trigger levels from 0 to 60 bytes are available with a granularity of 4 (trigger level = 4 × [4-bit register value]).
- To ensure correct device operation, ensure that RX\_FIFO\_TRIG\_HALT > RX\_FIFO\_TRIG when auto-RTS is enabled.

$$\text{Delay} = [4 + 16 \times (1 + \text{CHAR\_LENGTH} + \text{Parity} + \text{Stop} - 0.5)] \times \text{Baud\_rate} + 4 \times \text{FCLK}$$

#### Note

The RTS signal is deasserted after the UART module receives the data over RX\_FIFO\_TRIG\_HALT. Delay means how long the UART module takes to deassert the RTS signal after reaching RX\_FIFO\_TRIG\_HALT.

- In FIFO interrupt mode with flow control, ensure that the trigger level to HALT transmission is greater than or equal to the RX FIFO trigger level (the UART\_TCR[7:4] RX\_FIFO\_TRIG\_START bit field or the UART\_FCR[7:6] RX\_FIFO\_TRIG bit field); otherwise, FIFO operation stalls. In FIFO DMA mode with flow control, this concept does not exist, because a DMA request is sent when a byte is received.

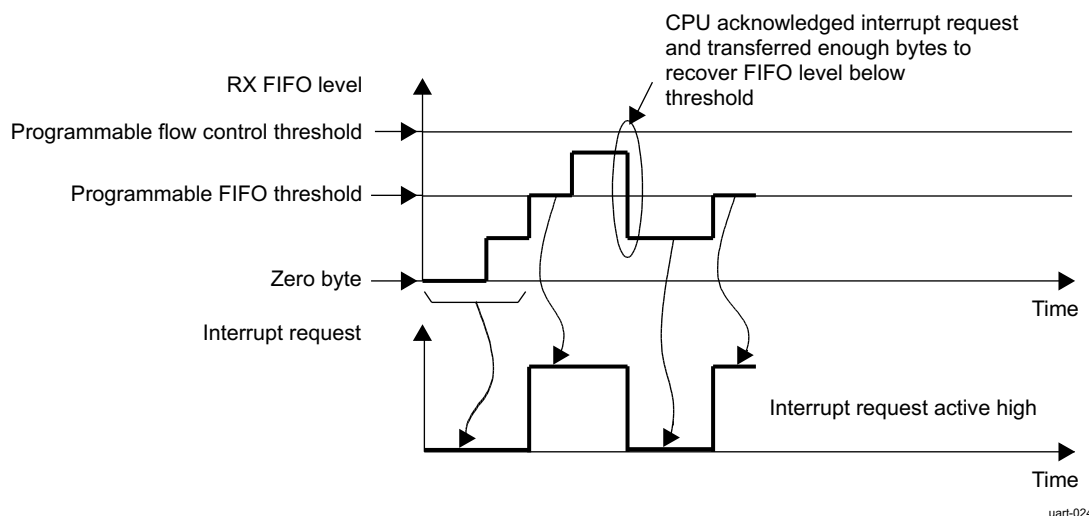
#### 12.1.6.4.6.2 FIFO Interrupt Mode

In FIFO interrupt mode (the FIFO control register UART\_FCR[0] FIFO\_EN bit is set to 1 and relevant interrupts are enabled by the UART\_IER\_UART register), an interrupt signal informs the processor of the status of the receiver and transmitter. These interrupts are raised when the RX/TX FIFO threshold (the UART\_TLR[7:4] RX\_FIFO\_TRIG\_DMA and UART\_TLR[3:0] TX\_FIFO\_TRIG\_DMA bit fields or the UART\_FCR[7:6] RX\_FIFO\_TRIG and UART\_FCR[5:4] TX\_FIFO\_TRIG bit fields, respectively) is reached.

The interrupt signals instruct the Host CPU to transfer data to the destination (from the UART in receive mode and/or from any source to the UART FIFO in transmit mode).

When UART flow control is enabled with interrupt capabilities, the UART flow control FIFO threshold (the UART\_TCR[3:0] RX\_FIFO\_TRIG\_HALT bit field) must be greater than or equal to the RX FIFO threshold.

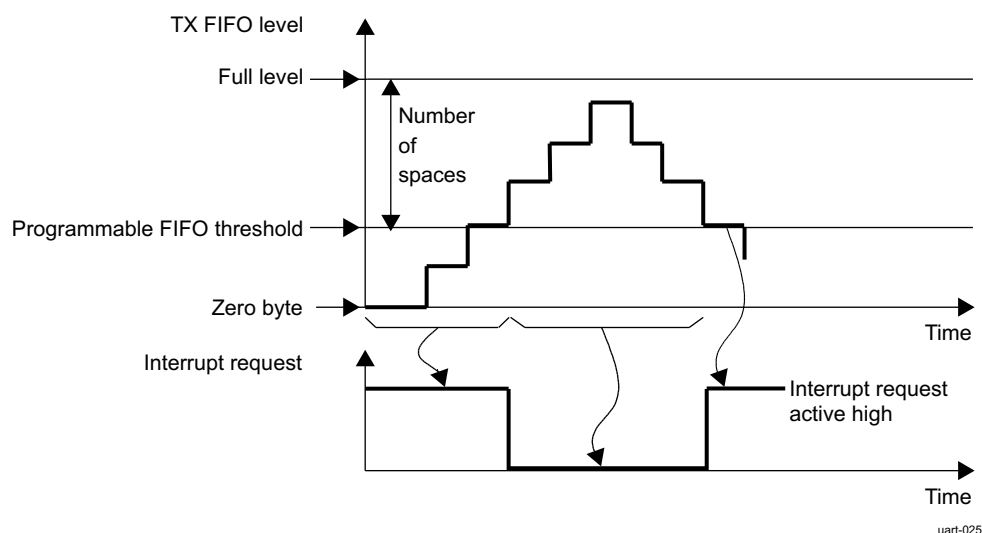
Figure 12-115 shows the generation of the RX FIFO interrupt request.



**Figure 12-115. UART RX FIFO Interrupt Request Generation**

In receive mode, no interrupt is generated until the RX FIFO reaches its threshold. Once low, the interrupt can be deasserted only when the Host CPU has handled enough bytes to put the FIFO level below threshold. The flow control threshold is set at a higher value than the FIFO threshold.

Figure 12-116 shows the generation of the TX FIFO interrupt request.



**Figure 12-116. UART TX FIFO Interrupt Request Generation**

In transmit mode, an interrupt request is automatically asserted when the TX FIFO is empty. This request is deasserted when the TX FIFO crosses the threshold level. The interrupt line is deasserted until a sufficient number of elements is transmitted to go below the TX FIFO threshold.

#### 12.1.6.4.6.3 FIFO Polled Mode Operation

In FIFO polled mode (the UART\_FCR[0] FIFO\_EN bit is set to 0 and the relevant interrupts are disabled by the UART\_IER\_UART register), the status of the receiver and transmitter can be checked by polling the line status register (UART\_LSR\_UART).

This mode is an alternative to the FIFO interrupt mode of operation in which the status of the receiver and transmitter is automatically determined by sending interrupts to the Host CPU.

#### 12.1.6.4.6.4 FIFO DMA Mode Operation

Although the DMA operation includes four modes (DMA modes 0 through 3), the information in *UART Hardware Requests*, assumes that mode 1 is used. (Mode 2 and mode 3 are legacy modes that use only one DMA request for each module.)

In mode 2, the remaining DMA request is used for RX. In mode 3, the remaining DMA request is used for TX.

DMA requests in mode 2 and mode 3 use the USARTi\_DMA0, and MCU\_USART0\_DMA0 signals (where i = 0 to 9).

The USARTi\_DMA1 (where i = 0 to 9), and MCU\_USART0\_DMA1 signals are not used by the module in mode 2 and mode 3:

The DMA mode and signals usage can be selected as follows:

- When SCR[0]=0:
  - Setting FCR[3] to 0 enables DMA mode 0
  - Setting FCR[3] to 1 enables DMA mode 1
- When SCR[0]=1:
  - SCR[2:1] determines DMA mode 0 to 3 according to the Supplementary Control Register (SCR) description.

For example:

- If no DMA operation is desired: set SCR[0] to 1 and SCR[2:1] to 00 (FCR[3] is discarded)
- If DMA mode 1 is desired: either set SCR[0] to 0 and FCR[3] to 1 or set SCR[0] to 1 and SCR[2:1] to 01 (FCR[3] is discarded)

If the FIFOs are disabled (FCR[0]=0), DMA operations occur in single character transfers.

Note that when DMA Mode 0 has been programmed, the signals associated with DMA operation are not active.

Depending on UART\_MDR3[2] SET\_DMA\_TX\_THRESHOLD, the threshold can be programmed different ways:

- SET\_TX\_DMA\_THRESHOLD = 1:
 

The threshold value will be the value of the UART\_TX\_DMA\_THRESHOLD register. If SET\_TX\_DMA\_THRESHOLD + TX trigger spaces 64, then the default method of threshold is used: threshold value = TX FIFO size.
- SET\_TX\_DMA\_THRESHOLD = 0:
 

The threshold value = TX FIFO size TX trigger space. The TX DMA line is asserted if the TX FIFO level is lower then the threshold. It remains asserted until TX trigger spaces number of bytes are written into the FIFO. The DMA line is then deasserted and the FIFO level is compared with the threshold value.

##### 12.1.6.4.6.4.1 DMA sequence to disable TX DMA

In order to disable TX DMA if it is not needed anymore (e.g. all transfers are done and UART idle mode is desired), the following sequence must be use

1. DMA mode 1 is set (both TX/RX DMA) by registers UART\_SCR[0] DMA\_MODE\_CTL = 0 and UART\_FCR[3] DMA\_MODE = 1:
  - a. Set the UART\_SCR[2-1] DMA\_MODE\_2 bit fields to 01 (DMA mode 1)
  - b. Set the UART\_SCR[0] DMA\_MODE\_CTL bit to 1 (this setting of UART\_SCR[0] DMA\_MODE-CTL will ignores UART\_FCR[3] DMA\_MODE\_CTL bit)

#### Note

It is strongly suggested to do steps 'a' and 'b' in two separate write in order to avoid malfunction of the device.

- c. Set the UART\_FCR[3] DMA\_MODE bit to 0. It is not necessary but suggested to avoid restore of DMA mode 1 during accidental reset of UART\_SCR[0] DMA\_MODE\_CTL bit. Be sure that all data was read

out from RX FIFO and if it possible disable the RX side. In UART mode the RTS/CTS or XOFF/XON protocol can be used. In IrDA modes RX can be forcibly disabled by setting UART\_ACREG[5] DIS\_IR\_RX bit

---

#### Note

There can be RX DATA loss during the next steps if all DATA was not read out or there was an ongoing reception!

- d. Set the UART\_FCR[2-1] DMA\_MODE bit field to 11 (clear TX and RX FIFO and resets its counter logic to 0. Returns to 0 after clearing FIFO).
- e. Set the UART\_SCR[2-1] DMA\_MODE\_2 bit field to 10 (DMA mode 2, RX only).
- f. Set the UART\_FCR[2-1] DMA\_MODE bit field to 11 (clear TX and RX FIFO and the DMA request again).
- g. Set the UART\_SCR[2-1] DMA\_MODE\_2 bit field to 00 (no DMA) or keep 10 if RX DMA is needed.
2. DMA mode 1 is set (both TX/RX DMA) by registers UART\_FCR[3] DMA\_MODE = 0 and UART\_SCR[0] DMA\_MODE\_CTL = 1, UART\_SCR[2-1] DMA\_MODE\_2 = 01. It is almost the same as above, but steps 'a', and 'b' can be skipped:
  - a. Set the UART\_FCR[3] DMA\_MODE bit to 0. It is not necessary but suggested to avoid restore of DMA mode 1 during accidental reset of UART\_SCR[0] DMA\_MODE\_CTL bit. Be sure that all data was read out from RX FIFO and if it possible disable the RX side. In UART mode the RTS/CTS or XOFF/XON protocol can be used. In IrDA modes RX can be forcibly disabled by setting UART\_ACREG[5] DIS\_IR\_RX bit

---

#### Note

There can be RX DATA loss during the next steps if all DATA was not read out or there was an ongoing reception!

- b. Set the UART\_FCR[2-1] DMA\_MODE bit field to 11 (clear TX and RX FIFO and resets its counter logic to 0. Returns to 0 after clearing FIFO).
- c. Set the UART\_SCR[2-1] DMA\_MODE\_2 bit field to 10 (DMA mode 2, RX only).
- d. Set the UART\_FCR[2-1] DMA\_MODE bit field to 11 (clear TX and RX FIFO and the DMA request again).
- e. Set the UART\_SCR[2-1] DMA\_MODE\_2 bit field to 00 (no DMA) or keep 10 if RX DMA is needed.
3. DMA mode 3 is set (TX DMA only) by registers UART\_FCR[3] DMA\_MODE = 0 and UART\_SCR[0] DMA\_MODE\_CTL = 1, UART\_SCR[2-1] DMA\_MODE\_2 = 11. It is the same as above:
  - a. Set the UART\_FCR[3] DMA\_MODE bit to 0. It is not necessary but suggested to avoid restore of DMA mode 1 during accidental reset of UART\_SCR[0] DMA\_MODE\_CTL bit. Be sure that all data was read out from RX FIFO and if it possible disable the RX side. In UART mode the RTS/CTS or XOFF/XON protocol can be used. In IrDA modes RX can be forcibly disabled by setting UART\_ACREG[5] DIS\_IR\_RX bit

---

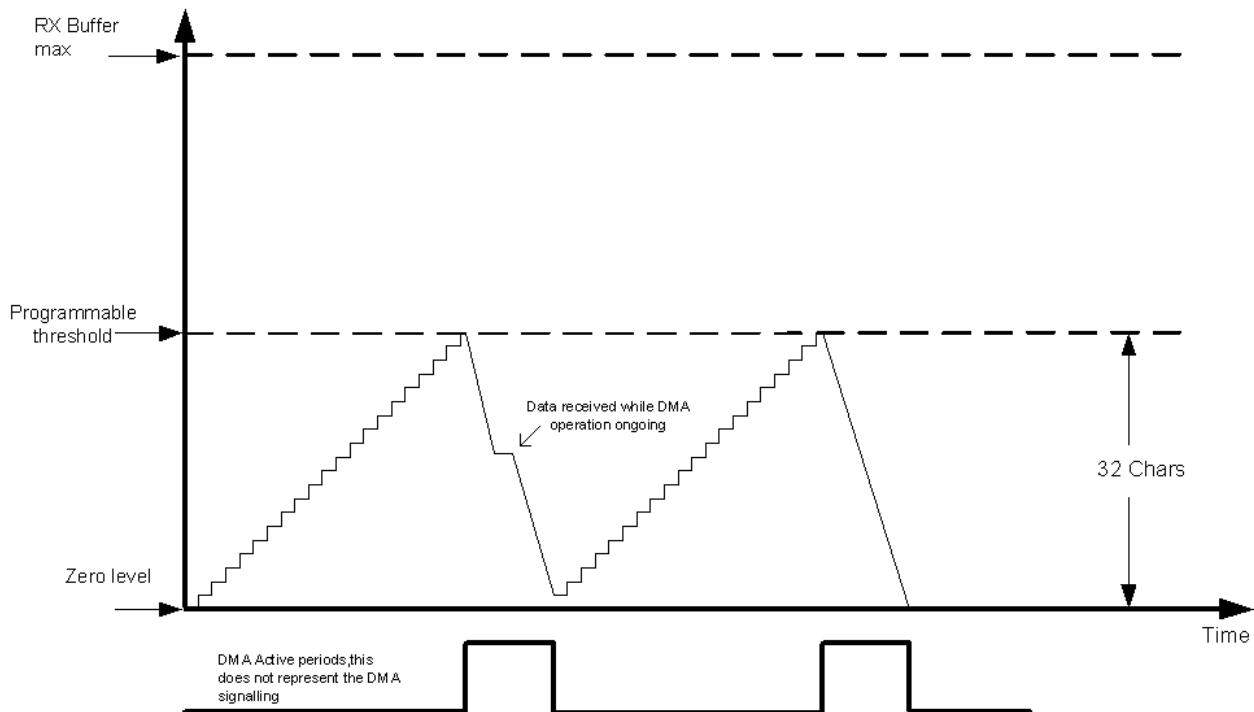
#### Note

There can be RX DATA loss during the next steps if all DATA was not read out or there was an ongoing reception!

- b. Set the UART\_FCR[2-1] DMA\_MODE bit field to 11 (clear TX and RX FIFO and resets its counter logic to 0. Returns to 0 after clearing FIFO).
- c. Set the UART\_SCR[2-1] DMA\_MODE\_2 bit field to 10 (DMA mode 2, RX only).
- d. Set the UART\_FCR[2-1] DMA\_MODE bit field to 11 (clear TX and RX FIFO and the DMA request again).
- e. Set the UART\_SCR[2-1] DMA\_MODE\_2 bit field to 00 (no DMA) or keep 10 if RX DMA is needed.

#### 12.1.6.4.6.4.2 DMA Transfers (DMA Mode 1, 2, or 3)

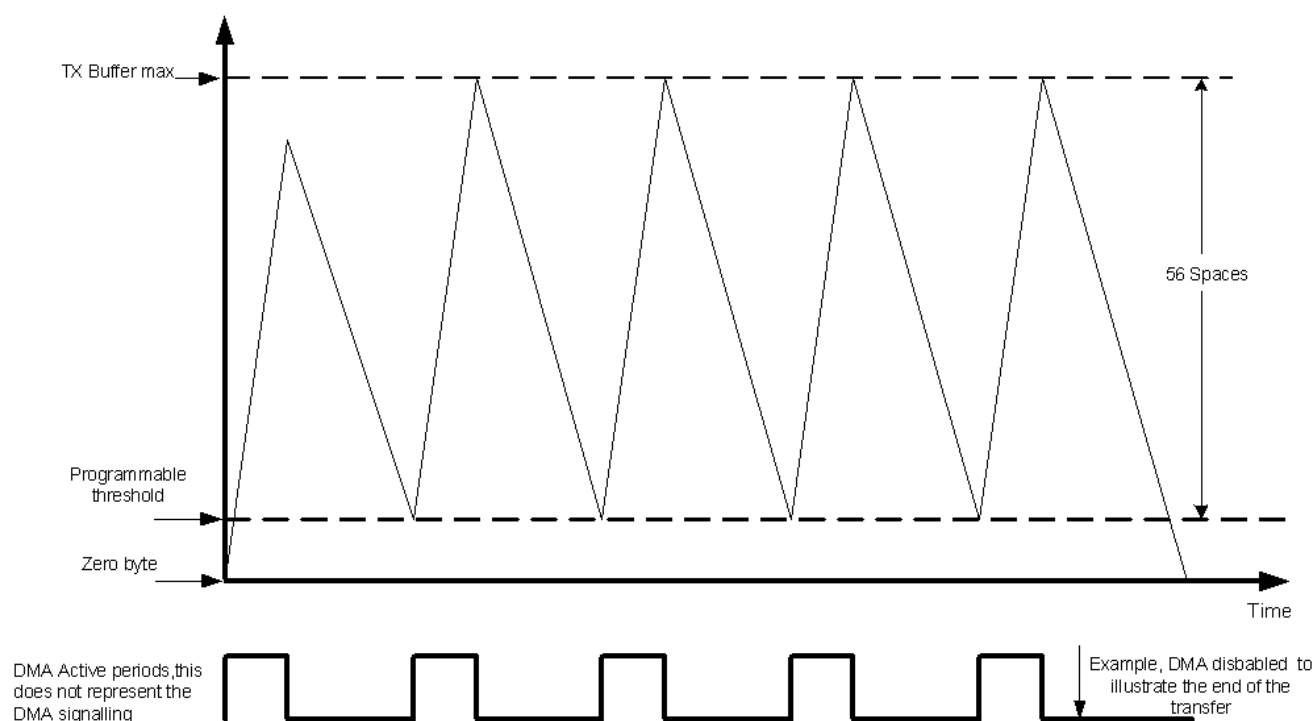
Figure 12-117 through Figure 12-120 show the supported DMA operations.



**Figure 12-117. UART Receive FIFO DMA Request Generation (32 Characters)**

In receive mode, a DMA request is generated when the RX FIFO reaches its threshold level defined in the trigger level register (UART\_TLR). This request is deasserted when the number of bytes defined by the threshold level is read by the device DMA controllers.

In transmit mode, a DMA request is automatically asserted when the TX FIFO is empty. This request is deasserted when the number of bytes defined by the number of spaces in the UART\_TLR register is written by the device DMA controllers. If an insufficient number of characters is written, the DMA request stays active.



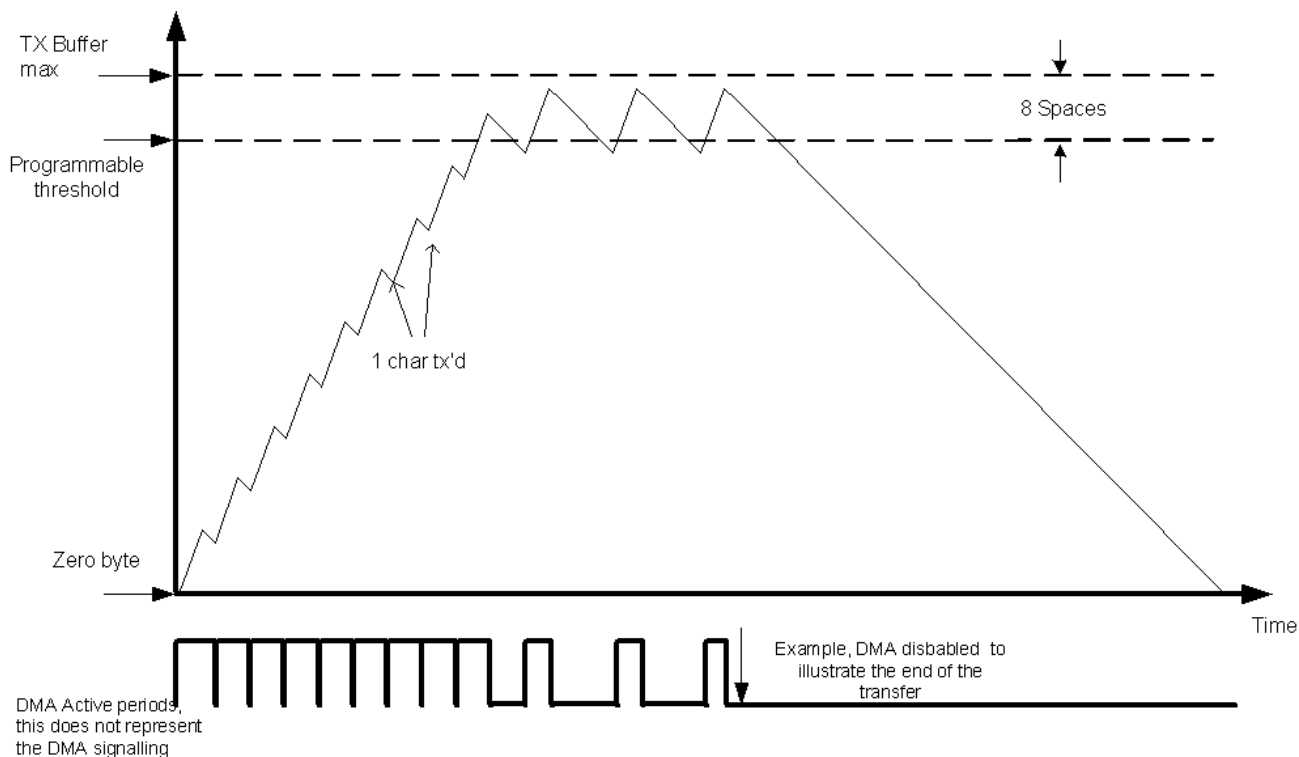
**Figure 12-118. UART Transmit FIFO DMA Request Generation (56 Spaces)**

The DMA request is again asserted if the FIFO can receive the number of bytes defined by the UART\_TLR register.

The threshold can be programmed in a number of ways. [Figure 12-118](#) shows a DMA transfer operating with a space setting of 56 that can arise from using the auto settings in the UART\_FCR[5-4] TX\_FIFO\_TRIG bit field or the UART\_TLR[3-0] TX\_FIFO\_TRIG\_DMA bit field concatenated with the TX\_FIFO\_TRIG bit field.

The setting of 56 spaces in the UART module must correlate with the settings of the device DMA controllers, so that the buffer does not overflow (program the DMA request size of the LH controller to equal the number of spaces in the UART module).

[Figure 12-119](#) shows an example with eight spaces to show the buffer level crossing the space threshold. The LH DMA controller settings must correspond to those of the UART module.

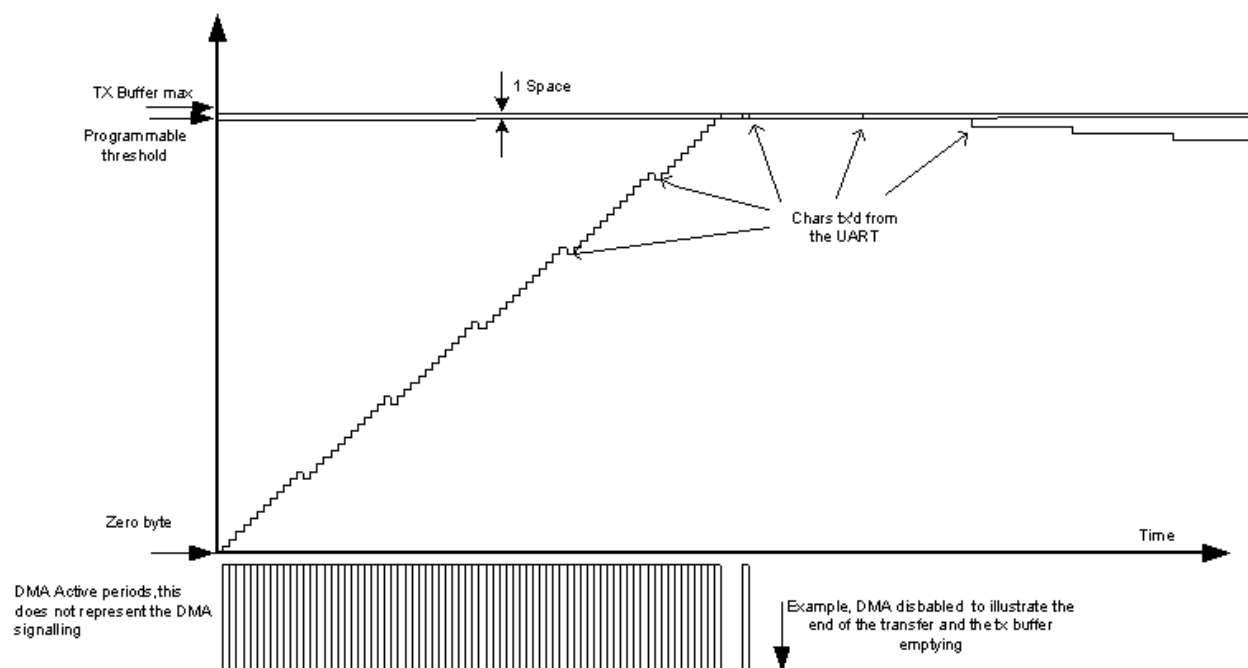


**Figure 12-119. UART Transmit FIFO DMA Request Generation (8 Spaces)**

The next example shows the setting of one space that uses the DMA for each transfer of one character to the transmit buffer (see [Figure 12-120](#)). The buffer is filled faster than the baud rate at which data is transmitted to the TX pin. Eventually, the buffer is completely full and the DMA operations stop transferring data to the transmit buffer.

On two occasions, the buffer holds the maximum amount of data words; shortly after this, the DMA is disabled to show the slower transmission of the data words to the TX pin. Eventually, the buffer is emptied at the rate specified by the baud rate settings of the UART\_DLL and UART\_DLH registers.

The DMA settings must correspond to the system LH DMA controller settings to ensure correct operation of this logic.



**Figure 12-120. UART Transmit FIFO DMA Request Generation (1 Space)**

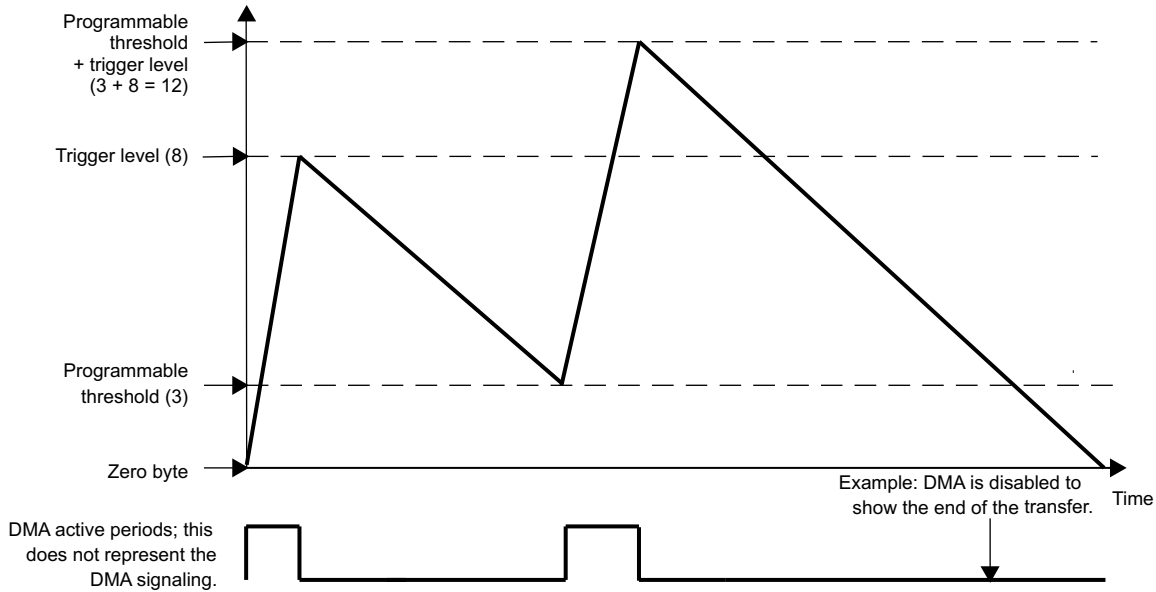
The final example illustrates the setting of eight spaces but setting the TX DMA threshold directly by setting UART\_MDR3[1] NONDEFAULT\_FREQ bit and UART\_TX\_DMA\_THRESHOLD register (see [Figure 12-121](#)). In the example, the UART\_TX\_DMA\_THRESHOLD[5-0] TX\_DMA\_THRESHOLD = 3 and the trigger level is 8. The buffer is filled at a faster rate than the BAUD rate transmits data to the TX pin. The buffer is filled with 8 bytes and the DMA operations stop transferring data to the transmit buffer. When the buffer is emptied to the threshold level by transmission, the DMA operation activates again to fill the buffer with 8 bytes.

Eventually, the buffer will be emptied at the rate specified by the BAUD Rate settings of the UART\_DLL and UART\_DLH registers.

If the selected threshold level + trigger level exceeds max buffer size, then the original TX DMA threshold method is used to prevent TX overrun, regardless of the UART\_MDR3[1] NONDEFAULT\_FREQ value.

The DMA settings should correspond to the system Local Host DMA controller settings in order to ensure the correct operation of this logic.



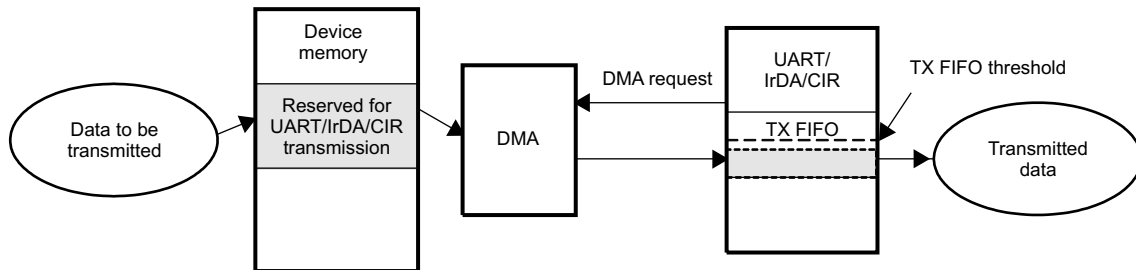


uart-036

**Figure 12-121. UART Transmit FIFO DMA Request Generation Using Direct TX DMA Threshold Programming. (Threshold = 3; Spaces = 8)**

#### 12.1.6.4.6.4.3 DMA Transmission

Figure 12-122 shows DMA transmission.



uart-030

**Figure 12-122. DMA Transmission**

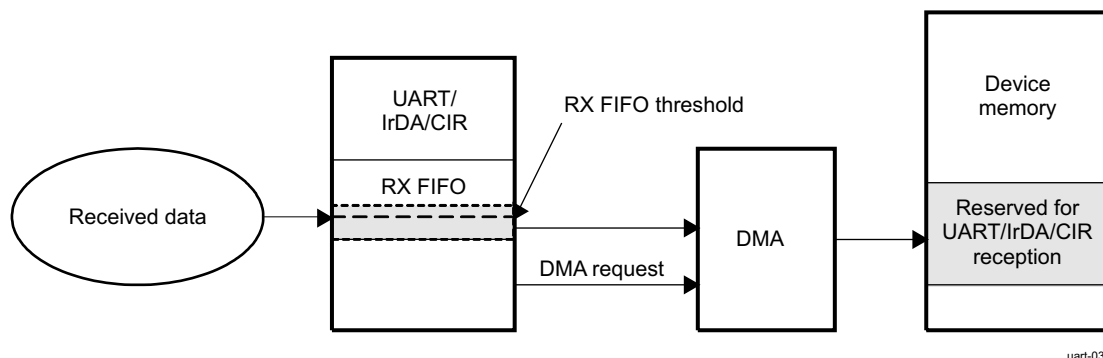
1. Data to be transmitted are put in the device memory reserved for UART transmission by the DMA:
  - a. Until the TX FIFO trigger level is not reached, a DMA request is generated
  - b. An element (1 byte) is transferred from the SDRAM to the TX FIFO at each DMA request (DMA element synchronization).
2. Data in the TX FIFO are automatically transmitted.
3. The end of the transmission is signaled by the UART\_THR empty (TX FIFO empty).

#### Note

In IrDA mode, the transmission does not end immediately after the TX FIFO empties, at which point the last data byte, the CRC field, and the stop flag still must be transmitted; thus, the end of transmission occurs a few milliseconds after the UART\_THR register empties.

#### 12.1.6.4.6.4.4 DMA Reception

Figure 12-123 shows DMA reception.



**Figure 12-123. DMA Reception**

1. Enable the reception.
2. Received data are put in the RX FIFO.
3. Data are transferred from the RX FIFO to the device memory by the DMA:
  - a. At each received byte, the RX FIFO trigger level (one character) is reached and a DMA request is generated.
  - b. An element (1 byte) is transferred from the RX FIFO to the SDRAM at each DMA request (DMA element synchronization).
4. The end of the reception is signaled by the EOF interrupt.

#### 12.1.6.4.7 UART Mode Selection

##### 12.1.6.4.7.1 Register Access Modes

##### 12.1.6.4.7.1.1 Operational Mode and Configuration Modes

Register access depends on the register access mode, although register access modes are not correlated to functional mode selection. Three different modes are available:

- Operational mode
- Configuration mode A
- Configuration mode B

Operational mode is the selected mode when the function is active; serial data transfer can be performed in this mode.

Configuration mode A and configuration mode B are used during module initialization steps. These modes enable access to configuration registers, which are hidden in the operational mode. The modes are used when the module is inactive (no serial data transfer processed) and only for initialization or reconfiguration of the module.

The value of the UART\_LCR register determines the register access mode (see [Table 12-125](#)).

**Table 12-125. UART Register Access Mode Programming (Using UART\_LCR)**

Mode	Condition
Configuration mode A	UART_LCR[7] = 0x1 and UART_LCR[7-0] != 0xBF
Configuration mode B	UART_LCR[7] = 0x1 and UART_LCR[7-0] = 0xBF
Operational mode	UART_LCR[7] = 0x0

#### 12.1.6.4.7.1.2 Register Access Submode

In each access register mode (operational mode or configuration mode A/B), some register accesses are conditional on the programming of a submode (MSR\_SPR, TCR\_TLR, and XOFF). These registers are identified in [Table 12-149](#), *UART Load FIFO Triggers Defined by the Concatenated Value*.

[Table 12-126](#) through [Table 12-128](#) summarize the register access submodes.

**Table 12-126. UART Subconfiguration Mode A Summary**

Mode	Condition
MSR_SPR	(UART_EFR[4] = 0x0 or UART_MCR[6] = 0x0)
TCR_TLR	UART_EFR[4] = 0x1 and UART_MCR[6] = 0x1

**Table 12-127. UART Subconfiguration Mode B Summary**

Mode	Condition
TCR_TLR	UART_EFR[4] = 0x1 and UART_MCR[6] = 0x1
XOFF	(UART_EFR[4] = 0x0 or UART_MCR[6] = 0x0)

**Table 12-128. UART Suboperational Mode Summary**

Mode	Condition
MSR_SPR	UART_EFR[4] = 0x0 or UART_MCR[6] = 0x0
TCR_TLR	UART_EFR[4] = 0x1 and UART_MCR[6] = 0x1

#### 12.1.6.4.7.1.3 Registers Available for the Register Access Modes

[Table 12-129](#) lists the names of the register bits in each access register mode. Gray shading indicates that the register does not depend on the register access mode (available in all modes).

**Table 12-129. UART Register Access Mode Overview**

Address Offset	Registers					
	Configuration Mode A		Configuration Mode B		Operational Mode	
	Read	Write	Read	Write	Read	Write
0x000	UART_DLL	UART_DLL	UART_DLL	UART_DLL	UART_RHR	UART_THR
0x004	UART_DLH	UART_DLH	UART_DLH	UART_DLH	UART_IER_UART	UART_IER_UART
0x008	UART_IIR_UART	UART_FCR	UART_EFR	UART_EFR	UART_IIR_UART	UART_FCR
0x00C	UART_LCR	UART_LCR	UART_LCR	UART_LCR	UART_LCR	UART_LCR
0x010	UART_MCR	UART_MCR	UART_XON1_AD DR1	UART_XON1_AD DR1	UART_MCR	UART_MCR
0x014	UART_LSR_UART	–	UART_XON2_AD DR2	UART_XON2_AD DR2	UART_LSR_UART	–
0x018	UART_MSR / UART_TCR	UART_TCR	UART_TCR / UART_XOFF1	UART_TCR / UART_XOFF1	UART_MSR / UART_TCR	UART_TCR
0x01C	UART_SPR / UART_TLR	UART_SPR / UART_TLR	UART_TLR / UART_XOFF2	UART_TLR / UART_XOFF2	UART_SPR / UART_TLR	UART_SPR / UART_TLR
0x020	UART_MDR1	UART_MDR1	UART_MDR1	UART_MDR1	UART_MDR1	UART_MDR1
0x024	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2
0x028	UART_SFLSR	UART_TXFLL	UART_SFLSR	UART_TXFLL	UART_SFLSR	UART_TXFLL
0x02C	UART_RESUME	UART_TXFLH	UART_RESUME	UART_TXFLH	UART_RESUME	UART_TXFLH
0x030	UART_SFREGL	UART_RXFLL	UART_SFREGL	UART_RXFLL	UART_SFREGL	UART_RXFLL
0x034	UART_SFREGH	UART_RXFLH	UART_SFREGH	UART_RXFLH	UART_SFREGH	UART_RXFLH
0x038	UART_UASR	–	UART_UASR	–	UART_BLR	UART_BLR
0x03C	–	–	–	–	UART_ACREG	UART_ACREG
0x040	UART_SCR	UART_SCR	UART_SCR	UART_SCR	UART_SCR	UART_SCR

**Table 12-129. UART Register Access Mode Overview (continued)**

Address Offset	Registers					
	Configuration Mode A		Configuration Mode B		Operational Mode	
	Read	Write	Read	Write	Read	Write
0x044	UART_SSR	–	UART_SSR	–	UART_SSR	–
0x048	–	–	–	–	UART_EBLR	UART_EBLR
0x050	UART_MVR	–	UART_MVR	–	UART_MVR	–
0x054	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC
0x058	UART_SYSS	–	UART_SYSS	–	UART_SYSS	–
0x05C	UART_WER	UART_WER	UART_WER	UART_WER	UART_WER	UART_WER
0x060	UART_CFPS	UART_CFPS	UART_CFPS	UART_CFPS	UART_CFPS	UART_CFPS
0x064	UART_RXFIFO_L VL	UART_RXFIFO_L VL	UART_RXFIFO_L VL	UART_RXFIFO_L VL	UART_RXFIFO_L VL	UART_RXFIFO_L VL
0x068	UART_TXFIFO_LV L	UART_TXFIFO_LV L	UART_TXFIFO_LV L	UART_TXFIFO_LV L	UART_TXFIFO_LV L	UART_TXFIFO_LV L
0x06C	UART_IER2	UART_IER2	UART_IER2	UART_IER2	UART_IER2	UART_IER2
0x070	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2
0x074	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL
0x080	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3
0x084	UART_TX_DMA_T HRESHOLD	UART_TX_DMA_T HRESHOLD	UART_TX_DMA_T HRESHOLD	UART_TX_DMA_T HRESHOLD	UART_TX_DMA_T HRESHOLD	UART_TX_DMA_T HRESHOLD

#### 12.1.6.4.7.2 UART/RS-485/IrDA (SIR, MIR, FIR)/CIR Mode Selection

To select a mode, set the UART\_MDR1[2:0] MODE\_SELECT bit field (see [Table 12-130](#)).

**Table 12-130. UART Mode Selection**

Value	Mode
0x0:	UART 16× mode
0x1:	SIR mode
0x2:	UART 16× auto-baud
0x3:	UART 13× mode
0x4:	MIR mode
0x5:	FIR mode
0x6:	CIR mode
0x7:	Disable (default state)

MODE\_SELECT is effective when the module is in operational mode (see [Section 12.1.6.4.7.1, Register Access Modes](#)).

To select a RS-485 mode, set the UART\_MDR3[4] DIR\_EN bit field to 0x1.

#### 12.1.6.4.7.2.1 Registers Available for the UART Function

Only the registers listed in [Table 12-131](#) are used for the UART function.

**Table 12-131. UART Mode Register Overview**

Address Offset	Registers <sup>(1) (2)</sup>					
	Configuration Mode A		Configuration Mode B		Operational Mode	
	Read	Write	Read	Write	Read	Write
0x000	UART_DLL	UART_DLL	UART_DLL	UART_DLL	UART_RHR	UART_THR
0x004	UART_DLH	UART_DLH	UART_DLH	UART_DLH	UART_IER_UART (UART)	UART_IER_UART (UART)

**Table 12-131. UART Mode Register Overview (continued)**

Address Offset	Registers <sup>(1) (2)</sup>					
	Configuration Mode A		Configuration Mode B		Operational Mode	
	Read	Write	Read	Write	Read	Write
0x008	UART_IIR_UART	UART_FCR	UART_EFR [4]	UART_EFR [4]	UART_IIR_UART (UART)	UART_FCR (UART)
0x00C	UART_LCR	UART_LCR	UART_LCR	UART_LCR	UART_LCR	UART_LCR
0x010	UART_MCR	UART_MCR	UART_XON1_AD DR1	UART_XON1_AD DR1	UART_MCR	UART_MCR
0x014	UART_LSR_UART (UART)	–	UART_XON2_AD DR2	UART_XON2_AD DR2	UART_LSR_UART (UART)	–
0x018	UART_MSR/ UART_TCR	UART_TCR	UART_XOFF1/ UART_TCR	UART_XOFF1/ UART_TCR	UART_MSR/ UART_TCR	UART_TCR
0x01C	UART_TLR/ UART_SPR	UART_TLR/ UART_SPR	UART_TLR/ UART_XOFF2	UART_TLR/ UART_XOFF2	UART_TLR/ UART_SPR	UART_TLR/ UART_SPR
0x020	UART_MDR1	UART_MDR1 [2-0]	UART_MDR1 [2-0]	UART_MDR1 [2-0]	UART_MDR1 [2-0]	UART_MDR1 [2-0]
0x024	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2
0x028	–	–	–	–	–	–
0x02C	–	–	–	–	–	–
0x030	–	–	–	–	–	–
0x034	–	–	–	–	–	–
0x038	UART_UASR	–	UART_UASR	–	–	–
0x03C	–	–	–	–	–	–
0x040	UART_SCR	UART_SCR	UART_SCR	UART_SCR	UART_SCR	UART_SCR
0x044	UART_SSR	–	UART_SSR	–	UART_SSR	–
0x048	–	–	–	–	–	–
0x050	UART_MVR	–	UART_MVR	–	UART_MVR	–
0x054	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC
0x058	UART_SYSS	–	UART_SYSS	–	UART_SYSS	–
0x05C	UART_WER	UART_WER	UART_WER	UART_WER	UART_WER	UART_WER
0x060	–	–	–	–	–	–
0x064	UART_RXFIFO_L VL	UART_RXFIFO_L VL	UART_RXFIFO_L VL	UART_RXFIFO_L VL	UART_RXFIFO_L VL	UART_RXFIFO_L VL
0x068	UART_TXFIFO_LV L	UART_TXFIFO_LV L	UART_TXFIFO_LV L	UART_TXFIFO_LV L	UART_TXFIFO_LV L	UART_TXFIFO_LV L
0x06C	UART_IER2	UART_IER2	UART_IER2	UART_IER2	UART_IER2	UART_IER2
0x070	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2
0x074	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL
0x080	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3
0x084	UART_TX_DMA_T HRESHOLD	UART_TX_DMA_T HRESHOLD	UART_TX_DMA_T HRESHOLD	UART_TX_DMA_T HRESHOLD	UART_TX_DMA_T HRESHOLD	UART_TX_DMA_T HRESHOLD

(1) REGISTER\_NAME(UART) notation indicates that the register exists for other functions (IrDA or CIR), but fields have different meanings for other functions (described separately in *UART Registers*).

(2) REGISTER\_NAME[m:n] notation indicates that only register bits numbered m to n apply to the UART function.

#### 12.1.6.4.7.2.2 Registers Available for the IrDA Function

Only the registers listed in [Table 12-132](#) are used for the IrDA function.

**Table 12-132. IrDA Mode Register Overview**

Address Offset	Registers <sup>(1) (2)</sup>					
	Configuration Mode A		Configuration Mode B		Operational Mode	
	Read	Write	Read	Write	Read	Write
0x000	UART_DLL	UART_DLL	UART_DLL	UART_DLL	UART_RHR	UART_THR
0x004	UART_DLH	UART_DLH	UART_DLH	UART_DLH	UART_IER_UART (IrDA)	UART_IER_UART (IrDA)
0x008	UART_IIR_UART	UART_FCR	UART_EFR [4]	UART_EFR [4]	UART_IIR_UART (IrDA)	UART_FCR (IrDA)
0x00C	UART_LCR [7]	UART_LCR [7]	UART_LCR [7]	UART_LCR [7]	UART_LCR [7]	UART_LCR [7]
0x010	–	–	UART_XON1_AD DR1	UART_XON1_AD DR1	–	–
0x014	UART_LSR_UART – (IrDA )	–	UART_XON2_AD DR2	UART_XON2_AD DR2	UART_LSR_UART – (IrDA)	–
0x018	UART_MSR/ UART_TCR	UART_TCR	UART_TCR	UART_TCR	UART_MSR/ UART_TCR	UART_TCR
0x01C	UART_TLR/ UART_SPR	UART_TLR/ UART_SPR	UART_TLR	UART_TLR	UART_TLR/ UART_SPR	UART_TLR/ UART_SPR
0x020	UART_MDR1	UART_MDR1	UART_MDR1	UART_MDR1	UART_MDR1	UART_MDR1
0x024	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2
0x028	UART_SFSLR	UART_TXFLL	UART_SFSLR	UART_TXFLL	UART_SFSLR	UART_TXFLL
0x02C	UART_RESUME	UART_TXFLH	UART_RESUME	UART_TXFLH	UART_RESUME	UART_TXFLH
0x030	UART_SFREGL	UART_RXFLL	UART_SFREGL	UART_RXFLL	UART_SFREGL	UART_RXFLL
0x034	UART_SFREGH	UART_RXFLH	UART_SFREGH	UART_RXFLH	UART_SFREGH	UART_RXFLH
0x038	–	–	–	–	UART_BLR	UART_BLR
0x03C	–	–	–	–	UART_ACREG	UART_ACREG
0x040	UART_SCR	UART_SCR	UART_SCR	UART_SCR	UART_SCR	UART_SCR
0x044	UART_SSR	–	UART_SSR	–	UART_SSR	–
0x048	–	–	–	–	UART_EBLR	UART_EBLR
0x050	UART_MVR	–	UART_MVR	–	UART_MVR	–
0x054	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC
0x058	UART_SYSS	–	UART_SYSS	–	UART_SYSS	–
0x05C	UART_WER [6-4]	UART_WER [6-4]	UART_WER [6-4]	UART_WER [6-4]	UART_WER [6-4]	UART_WER [6-4]
0x060	–	–	–	–	–	–
0x064	UART_RXFIFO_L VL	UART_RXFIFO_L VL	UART_RXFIFO_L VL	UART_RXFIFO_L VL	UART_RXFIFO_L VL	UART_RXFIFO_L VL
0x068	UART_TXFIFO_LV L	UART_TXFIFO_LV L	UART_TXFIFO_LV L	UART_TXFIFO_LV L	UART_TXFIFO_LV L	UART_TXFIFO_LV L
0x06C	UART_IER2	UART_IER2	UART_IER2	UART_IER2	UART_IER2	UART_IER2
0x070	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2
0x074	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL
0x080	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3
0x084	UART_TX_DMA_T HRESHOLD	UART_TX_DMA_T HRESHOLD	UART_TX_DMA_T HRESHOLD	UART_TX_DMA_T HRESHOLD	UART_TX_DMA_T HRESHOLD	UART_TX_DMA_T HRESHOLD

(1) REGISTER\_NAME(IrDA) notation indicates that the register exists for other functions (UART or CIR), but fields have different meanings for other functions (described separately in *UART Registers*).

(2) REGISTER\_NAME[m:n] notation indicates that only register bits numbered m to n apply to the IrDA function.

#### 12.1.6.4.7.2.3 Registers Available for the CIR Function

Only the registers listed in [Table 12-133](#) are used for the CIR function.

**Table 12-133. CIR Mode Register Overview**

Address Offset	Registers <sup>(1) (2)</sup>					
	Configuration Mode A		Configuration Mode B		Operational Mode	
	Read	Write	Read	Write	Read	Write
0x000	UART_DLL	UART_DLL	UART_DLL	UART_DLL	–	UART_THR
0x004	UART_DLH	UART_DLH	UART_DLH	UART_DLH	UART_IER_UART (CIR)	UART_IER_UART (CIR)
0x008	UART_IIR_UART	UART_FCR	UART_EFR	UART_EFR	UART_IIR_UART (CIR)	UART_FCR (CIR)
0x00C	UART_LCR	UART_LCR [7]	UART_LCR [7]	UART_LCR [7]	UART_LCR [7]	UART_LCR [7]
0x010	–	–	–	–	–	–
0x014	UART_LSR_UART (CIR)	–	–	–	UART_LSR_UART (CIR)	–
0x018	UART_MSR/ UART_TCR	UART_TCR	UART_TCR	UART_TCR	UART_MSR/ UART_TCR	UART_TCR
0x01C	UART_TLR/ UART_SPR	UART_TLR/ UART_SPR	UART_TLR	UART_TLR	UART_TLR/ UART_SPR	UART_TLR/ UART_SPR
0x020	UART_MDR1 [3-0]	UART_MDR1 [3-0]	UART_MDR1 [3-0]	UART_MDR1 [3-0]	UART_MDR1 [3-0]	UART_MDR1 [3-0]
0x024	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2
0x028	–	–	–	–	–	–
0x02C	UART_RESUME	–	UART_RESUME	–	UART_RESUME	–
0x030	–	–	–	–	–	–
0x034	–	–	–	–	–	–
0x038	–	–	–	–	–	–
0x03C	–	–	–	–	UART_ACREG	UART_ACREG
0x040	UART_SCR	UART_SCR	UART_SCR	UART_SCR	UART_SCR	UART_SCR
0x044	UART_SSR	–	UART_SSR	–	UART_SSR	–
0x048	–	–	–	–	UART_EBLR	UART_EBLR
0x050	UART_MVR	–	UART_MVR	–	UART_MVR	–
0x054	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC
0x058	UART_SYSS	–	UART_SYSS	–	UART_SYSS	–
0x05C	UART_WER [6-4]	UART_WER [6-4]	UART_WER [6-4]	UART_WER [6-4]	UART_WER [6-4]	UART_WER [6-4]
0x060	UART_CFPS	UART_CFPS	UART_CFPS	UART_CFPS	UART_CFPS	UART_CFPS
0x064	UART_RXFIFO_L VL	UART_RXFIFO_L VL	UART_RXFIFO_L VL	UART_RXFIFO_L VL	UART_RXFIFO_L VL	UART_RXFIFO_L VL
0x068	UART_TXFIFO_LV L	UART_TXFIFO_LV L	UART_TXFIFO_LV L	UART_TXFIFO_LV L	UART_TXFIFO_LV L	UART_TXFIFO_LV L
0x06C	UART_IER2	UART_IER2	UART_IER2	UART_IER2	UART_IER2	UART_IER2
0x070	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2
0x074	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL
0x080	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3
0x084	UART_TX_DMA_T HRESHOLD	UART_TX_DMA_T HRESHOLD	UART_TX_DMA_T HRESHOLD	UART_TX_DMA_T HRESHOLD	UART_TX_DMA_T HRESHOLD	UART_TX_DMA_T HRESHOLD

(1) REGISTER\_NAME(CIR) notation indicates that the register exists for other functions (IrDA or UART), but fields have different meanings for other functions (described separately in *UART Registers*).

(2) REGISTER\_NAME[m:n] notation indicates that only register bits numbered m to n apply to the CIR function.

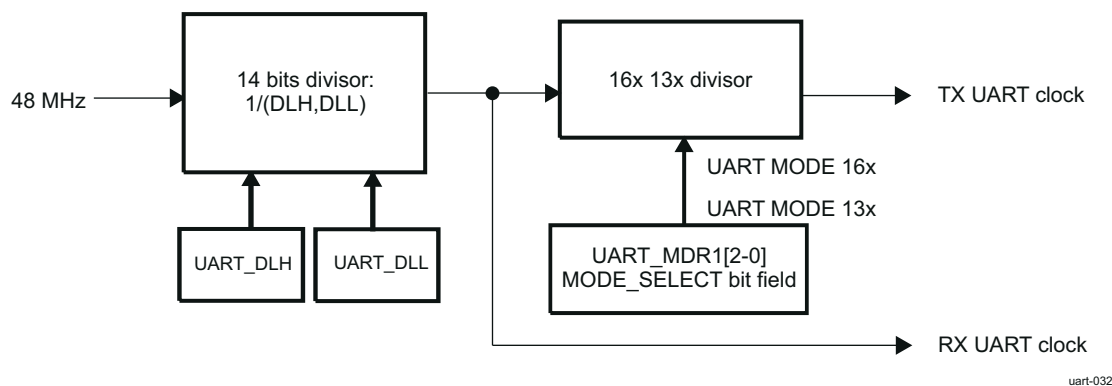
### 12.1.6.4.8 UART Protocol Formatting

#### 12.1.6.4.8.1 UART Mode

##### 12.1.6.4.8.1.1 UART Clock Generation: Baud Rate Generation

The UART function contains a programmable baud generator and a set of fixed dividers that divide the 48-MHz clock input down to the expected baud rate.

Figure 12-124 shows the baud rate generator and associated controls.



**Figure 12-124. UART Baud Rate Generation**

#### CAUTION

Before initializing or modifying clock parameter controls (UART\_DLH, UART\_DLL), UART\_MDR1[2-0] MODE\_SELECT = DISABLE must be set to 0x7. Failure to observe this rule can result in unpredictable module behavior.

##### 12.1.6.4.8.1.2 Choosing the Appropriate Divisor Value

Two divisor values are:

- UART 16× mode: Divisor value = Operating frequency / (16× baud rate)
- UART 13× mode: Divisor value = Operating frequency / (13× baud rate)

Table 12-134 describe the UART baud rate settings.

**Table 12-134. UART Baud Rate Settings (48-MHz Clock)**

Baud Rate	Baud Multiple	DLH, DLL (Decimal)	DLH, DLL (Hex)	Actual Baud Rate	Error (%)
0.3 kbps	16x	10000	0x27, 0x10	0.3 kbps	0
0.6 kbps	16x	5000	0x13, 0x88	0.6 kbps	0
1.2 kbps	16x	2500	0x09, 0xC4	1.2 kbps	0
2.4 kbps	16x	1250	0x04, 0xE2	2.4 kbps	0
4.8 kbps	16x	625	0x02, 0x71	4.8 kbps	0
9.6 kbps	16x	312	0x01, 0x39	9.6153 kbps	+0.16
14.4 kbps	16x	208	0x00, 0xD0	14.423 kbps	+0.16
19.2 kbps	16x	156	0x00, 0x9C	19.231 kbps	+0.16
28.8 kbps	16x	104	0x00, 0x68	28.846 kbps	+0.16
38.4 kbps	16x	78	0x00, 0x4E	38.462 kbps	+0.16
57.6 kbps	16x	52	0x00, 0x34	57.692 kbps	+0.16
115.2 kbps	16x	26	0x00, 0x1A	115.38 kbps	+0.16
230.4 kbps	16x	13	0x00, 0x0D	230.77 kbps	+0.16
460.8 kbps	13x	8	0x00, 0x08	461.54 kbps	+0.16
921.6 kbps	13x	4	0x00, 0x04	923.08 kbps	+0.16



**Table 12-134. UART Baud Rate Settings (48-MHz Clock) (continued)**

Baud Rate	Baud Multiple	DLH, DLL (Decimal)	DLH, DLL (Hex)	Actual Baud Rate	Error (%)
1.843 Mbps	13x	2	0x00, 0x02	1.846 Mbps	+0.16
3.6884 Mbps	13x	1	0x00, 0x01	3.6923 Mbps	+0.16

#### 12.1.6.4.8.1.3 UART Data Formatting

The UART can use hardware flow control to manage transmission and reception. Hardware flow control significantly reduces software overhead and increases system efficiency by automatically controlling serial data flow using the RTS output and CTS input signals.

The UART is enhanced with the autobauding function. In control mode, autobauding lets the speed, the number of bits per character, and the parity selected be set automatically.

##### 12.1.6.4.8.1.3.1 Frame Formatting

When autobauding is not used, frame format attributes must be defined in the UART\_LCR register.

Character length is specified using the UART\_LCR[1-0] CHAR\_LENGTH bit field.

The number of stop-bits is specified using the UART\_LCR[2] NB\_STOP bit.

The parity bit is programmed using the UART\_LCR[5-3] PARITY\_EN, UART\_LCR[5-3] PARITY\_TYPE\_1, and UART\_LCR[5-3] PARITY\_TYPE\_2 bit fields (see [Table 12-135](#)).

**Table 12-135. UART Parity Bit Encoding**

PARITY_EN	PARITY_TYPE_1	PARITY_TYPE_2	Parity
0	N/A	N/A	No parity
1	0	0	Odd parity
1	1	0	Even parity
1	0	1	Forced 1
1	1	1	Forced 0

##### 12.1.6.4.8.1.3.2 Hardware Flow Control

Hardware flow control is composed of auto-CTS and auto-RTS. Auto-CTS and auto-RTS can be enabled and disabled independently by programming the UART\_EFR[7] AUTO\_CTS\_EN and UART\_EFR[6] AUTO\_RTS\_EN bit fields, respectively.

With auto-CTS, CTS signal must be active before the module can transmit data.

Auto-RTS activates the RTS output only when there is enough room in the RX FIFO to receive data. It deactivates the RTS output when the RX FIFO is sufficiently full. The HALT and RESTORE trigger levels in the UART\_TCR register determine the levels at which RTS is activated and deactivated.

If auto-CTS and auto-RTS are enabled, data transmission does not occur unless the RX FIFO has empty space. Thus, overrun errors are eliminated during hardware flow control. If auto-CTS and auto-RTS are not enabled, overrun errors occur if the transmit data rate exceeds the RX FIFO latency.

- Auto-RTS:

Auto-RTS data flow control originates in the receiver block. The RX FIFO trigger levels used in auto-RTS are stored in the UART\_TCR register. RTS is active if the RX FIFO level is below the HALT trigger level in the UART\_TCR[3-0] RX\_FIFO\_TRIG\_HALT bit field. When the RX FIFO HALT trigger level is reached, RTS is deasserted. The sending device (for example, another UART) can send an additional byte after the trigger level is reached because it may not recognize the deassertion of RTS until it begins sending the additional byte.

RTS is automatically reasserted when the RX FIFO reaches the RESUME trigger level programmed by the UART\_TCR[7-4] RX\_FIFO\_TRIG\_START bit field. This reassertion requests the sending device to resume transmission.

In this case, RTS is an active-low signal.

- Auto-CTS:

The transmitter circuitry checks CTS before sending the next data byte. When CTS is active, the transmitter sends the next byte. To stop the transmitter from sending the next byte, CTS must be deasserted before the middle of the last stop-bit currently sent.

The auto-CTS function reduces interrupts to the host system. When auto-CTS flow control is enabled, the CTS state changes do not have to trigger host interrupts because the device automatically controls its own transmitter. Without auto-CTS, the transmitter sends any data present in the transmit FIFO, and a receiver overrun error can result.

In this case, CTS is an active-low signal.

#### 12.1.6.4.8.1.3.3 Software Flow Control

Software flow control is enabled through the enhanced feature register (UART\_EFR) and the modem control register (UART\_MCR). Different combinations of software flow control can be enabled by setting different combinations of the UART\_EFR[3-0] bit field (see [Table 12-136](#)).

Two other enhanced features relate to software flow control:

- XON-any function (UART\_MCR[5] XON\_EN): Operation resumes after receiving any character after the XOFF character is recognized. If special character detect is enabled and special character is received after XOFF1, it does not resume transmission. The special character is stored in the RX FIFO.

#### Note

The XON-any character is written into the RX FIFO even if it is a software flow character.

- Special character (UART\_EFR[5] SPECIAL\_CHAR\_DETECT): Incoming data is compared to XOFF2. When the special character is detected, the XOFF interrupt (UART\_IIR\_UART) is set, but it does not halt transmission. The XOFF interrupt is cleared by a read of UART\_IIR\_UART. The special character is transferred to the RX FIFO. Special character does not work with XON2, XOFF2, or sequential XOFFs.

**Table 12-136. UART\_EFR[3:0] Software Flow Control Options**

Bit 3	Bit 2	Bit 1	Bit 0	TX, RX Software Flow Controls
0	0	X	X	No transmit flow control
1	0	X	X	Transmit XON1, XOFF1
0	1	X	X	Transmit XON2, XOFF2
1	1	X	X	Transmit XON1, XON2: XOFF1, XOFF2 <sup>(1)</sup>
X	X	0	0	No receive flow control
X	X	1	0	Receiver compares XON1, XOFF1
X	X	0	1	Receiver compares XON2, XOFF2
X	X	1	1	Receiver compares XON1, XON2: XOFF1, XOFF2 <sup>(1)</sup>

- (1) In these cases, the XON1 and XON2 characters or the XOFF1 and XOFF2 characters must be transmitted/received sequentially with XON1/XOFF1 followed by XON2/XOFF2.  
XON1 is defined in the UART\_XON1\_ADDR1[7-0] XON\_WORD1 bit field. XON2 is defined in the UART\_XON2\_ADDR2[7-0] XON\_WORD2 bit field.  
XOFF1 is defined in the UART\_XOFF1[7-0] XOFF\_WORD1 bit field. XOFF2 is defined in the UART\_XOFF2[7-0] XOFF\_WORD2 bit field.

#### 12.1.6.4.8.1.3.3.1 Receive (RX)

When software flow control operation is enabled, the UART compares incoming data with XOFF1/2 programmed characters (in certain cases, XOFF1 and XOFF2 must be received sequentially). When the correct XOFF characters are received, transmission stops after transmission of the current character completes. Detection of

XOFF also sets the UART\_IIR\_UART[4] bit (if enabled by UART\_IER\_UART[5]) and causes the interrupt line to go low.

To resume transmission, an XON1/2 character must be received (in certain cases, XON1 and XON2 must be received sequentially). When the correct XON characters are received, the UART\_IIR\_UART[4] bit is cleared and the XOFF interrupt disappears.

---

#### Note

When a parity, framing, or break error occurs while receiving a software flow control character, this character is treated as normal data and is written to the RX FIFO.

---

When XON-any and special character detect are disabled and software flow control is enabled, no valid XON or XOFF characters are written to the RX FIFO. For example, when UART\_EFR[1-0] = 0x2, if XON1 and XOFF1 characters are received, they are not written to the RX FIFO.

When pairs of software flow characters are programmed to be received sequentially (UART\_EFR[1-0] = 0x3), the software flow characters are not written to the RX FIFO if they are received sequentially. However, received XON1/XOFF1 characters must be written to the RX FIFO if the subsequent character is not XON2/XOFF2.

#### 12.1.6.4.8.1.3.3.2 Transmit (TX)

Two XOFF1 characters are transmitted when the RX FIFO passes the trigger level programmed by UART\_TCR[3-0] RX\_FIFO\_TRIG\_HALT. As soon as the RX FIFO reaches the trigger level programmed by UART\_TCR[7-4] RX\_FIFO\_TRIG\_START, two XON1 characters are sent, so the data transfer recovers.

---

#### Note

If software flow control is disabled after an XOFF character is sent, the module transmits XON characters automatically to enable normal transmission.

---

The transmission of XOFF(s)/XON(s) follows the same protocol as transmission of an ordinary byte from the TX FIFO. This means that even if the word length is 5, 6, or 7 characters, the 5, 6, or 7 LSBs of XOFF1/2 and XON1/2 are transmitted. The 5, 6, or 7 bits of a character are seldom transmitted, but this function is included to maintain compatibility with earlier designs.

It is assumed that software flow control and hardware flow control are never enabled simultaneously.

#### 12.1.6.4.8.1.3.4 Autobauding Modes

In autobauding mode, the UART can extract transfer characteristics (speed, length, and parity) from an "at" (AT) command (ASCII code). These characteristics are used to receive data after an AT and to send data.

The following AT commands are valid:

AT	DATA	<CR>
at	DATA	<CR>
A/		
a/		

A line break during the acquisition of the sequence AT is not recognized, and an echo function is not implemented in hardware.

A/ and a/ are not used to extract characteristics, but they must be recognized because of their special meaning. A/ or a/ is used to instruct the software to repeat the last received AT command; therefore, an a/ always follows an AT, and transfer characteristics are not expected to change between an AT and an a/.

When a valid AT is received, AT and all subsequent data, including the final <CR> (0x0D), are saved to the RX FIFO. The autobaud state-machine waits for the next valid AT command. If an a/ (A/) is received, the a/ (A/) is saved in the RX FIFO and the state-machine waits for the next valid AT command.

On the first successful detection of the baud rate, the UART activates an interrupt to signify that the AT (upper or lower case) sequence is detected. The UART\_UASR register reflects the correct settings for the baud rate detected. Interrupt activity can continue in this fashion when a subsequent character is received. Therefore, it is recommended that the software enable the RHR interrupt when using the autobaud mode.

The following settings are detected in autobaud mode with a module clock of 48 MHz:

- Speed:
  - 115.2K baud
  - 57.6K baud
  - 38.4K baud
  - 28.8K baud
  - 19.2K baud
  - 14.4K baud
  - 9.6K baud
  - 4.8K baud
  - 2.4K baud
  - 1.2K baud
- Length: 7 or 8 bits
- Parity: Odd, even, or space

---

#### Note

The combination of 7-bit character plus space parity is not supported.

---

Autobauding mode is selected when the UART\_MDR1[2-0] MODE\_SELECT bit field is set to 0x2. In UART autobauding mode, UART\_DLL, UART\_DLH, and UART\_LCR[5-0] bit field settings are not used; instead, the UART\_UASR register is updated with the configuration detected by the autobauding logic.

#### UASR Autobauding Status Register Use

This register is used to set up transmission according to the characteristics of the previous reception instead of the UART\_LCR, UART\_DLL, and UART\_DLH registers when the UART is in autobauding mode.

To reset the autobauding hardware (to start a new AT detection) or to set the UART in standard mode (no autobaud), the UART\_MDR1[2-0] MODE\_SELECT bit field must be set to reset state (0x7) and then to the UART in autobauding mode (0x2) or to the UART in standard mode (0x0).

Use limitation:

- Only 7- and 8-bit characters (5- and 6-bit not supported)
- 7-bit character with space parity not supported
- Baud rate between 1200 and 115.2 bps (10 possibilities)

#### 12.1.6.4.8.1.3.5 Error Detection

When the UART\_LSR\_UART register is read, the UART\_LSR\_UART[4:2] bit field reflects the error bits (BI: break condition, FE: framing error, PE: parity error) of the character at the top of the RX FIFO (the next character to be read). Therefore, reading the UART\_LSR\_UART register and then reading the UART\_RHR register identifies errors in a character.

Reading the UART\_RHR register updates the BI, FE, and PE bits (see [Table 12-120](#) for the UART mode interrupts).

The UART\_LSR\_UART[7] RX\_FIFO\_STS bit is set when there is an error in the RX FIFO and is cleared only when no errors remain in the RX FIFO.

### Note

Reading the UART\_LSR\_UART register does not cause an increment of the RX FIFO read pointer. The RX FIFO read pointer is incremented by reading the UART\_RHR register.

Reading the UART\_LSR\_UART register clears the OE bit if it is set (see [Table 12-120](#) for the UART mode interrupts).

#### 12.1.6.4.8.1.3.6 Overrun During Receive

Overrun during receive occurs if the RX state-machine tries to write data into the RX FIFO when it is already full. When overrun occurs, the device interrupts the Host CPU with the UART\_IIR\_UART[5-1] IT\_TYPE bit field set to 0x3 (receiver line status error) and discards the remaining portion of the frame.

Overrun also causes an internal flag to be set, which disables further reception. Before the next frame can be received, the Host CPU must:

- Reset the RX FIFO.
- Read the UART\_RESUME register, which clears the internal flag.

#### 12.1.6.4.8.1.3.7 Time-Out and Break Conditions

##### 12.1.6.4.8.1.3.7.1 Time-Out Counter

An RX idle condition is detected when the receiver line (RX) is high for a time that equals 4x the programmed word length + 12 bits or manually configured amount of baud clocks, if a value other zero is set in the timeout register. RX is sampled midway through each bit.

For sleep mode, the counter is reset when there is activity on RX.

There are two modes of operation:

- In default operation on the UART\_EFR2[6] TIMEOUT\_BEHAVE is set to 0. For the time-out interrupt, the counter counts only when there is data in the RX FIFO, and the count is reset when there is activity on RX or when the UART\_RHR register is read.
- Optionally, for choose to enable the timeout counter even if no character has been received by setting UART\_EFR2[6] TIMEOUT\_BEHAVE bit. This will generate periodic interrupts if the RX line remains idle. In this mode the counter will auto-reset when a timeout has been reached. Reading the UART\_IIR\_UART will clear the interrupt, but not the counter.

##### 12.1.6.4.8.1.3.7.2 Break Condition

When a break condition occurs, TX is pulled low. A break condition is activated by setting the UART\_LCR[6] BREAK\_EN bit. The break condition is not aligned on word stream (a break condition can occur in the middle of a character). The only way to send a break condition on a full character is:

1. Reset the TX FIFO (if enabled).
2. Wait for the transmit shift register to empty (the UART\_LSR\_UART[6] TX\_SR\_E bit is set to 1).
3. Take a guard time according to stop-bit definition.
4. Set the BREAK\_EN bit to 1.

The break condition is asserted while the BREAK\_EN bit is set to 1.

The time-out counter and break condition apply only to UART modem operation and not to IrDA/CIR mode operation.

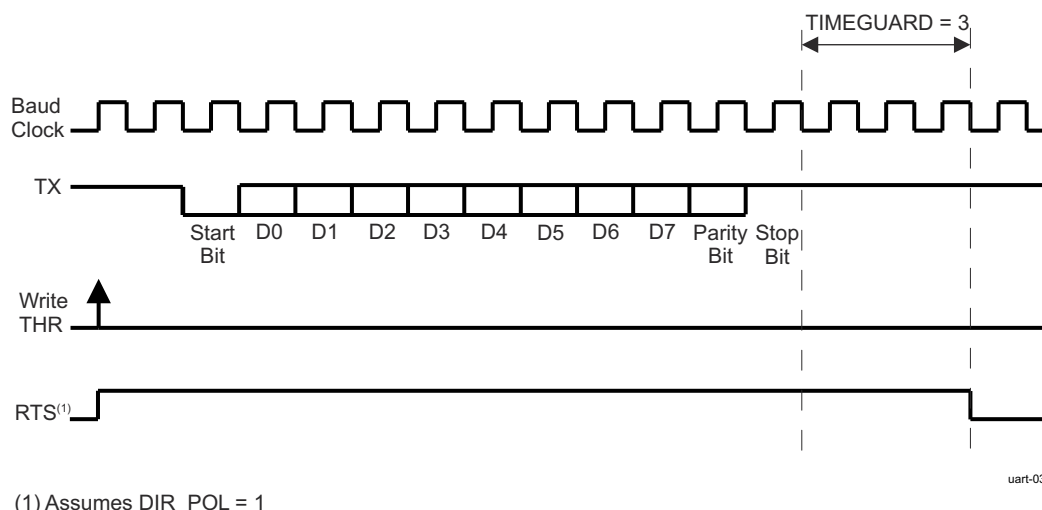
#### 12.1.6.4.8.2 RS-485 Mode

##### 12.1.6.4.8.2.1 RS-485 External Transceiver Direction Control

The UART\_MDR3[4] DIR\_EN bit enables hardware control over an external transceiver to support RS-485. The direction signal comes across the DIR port. The direction polarity is controlled by the UART\_MDR3[3] DIR\_POL bit. The direction is determined by the hardware monitoring the TX FIFO and the TX shift register. When both are empty the transceiver is set to RX. There is a guard band delay counter of 3 bit clock cycles after the TX

shift register is going empty to allow time for the stop bit to transition through the transceiver before a direction change to receive might be applied.

Figure 12-125 shows the direction control.



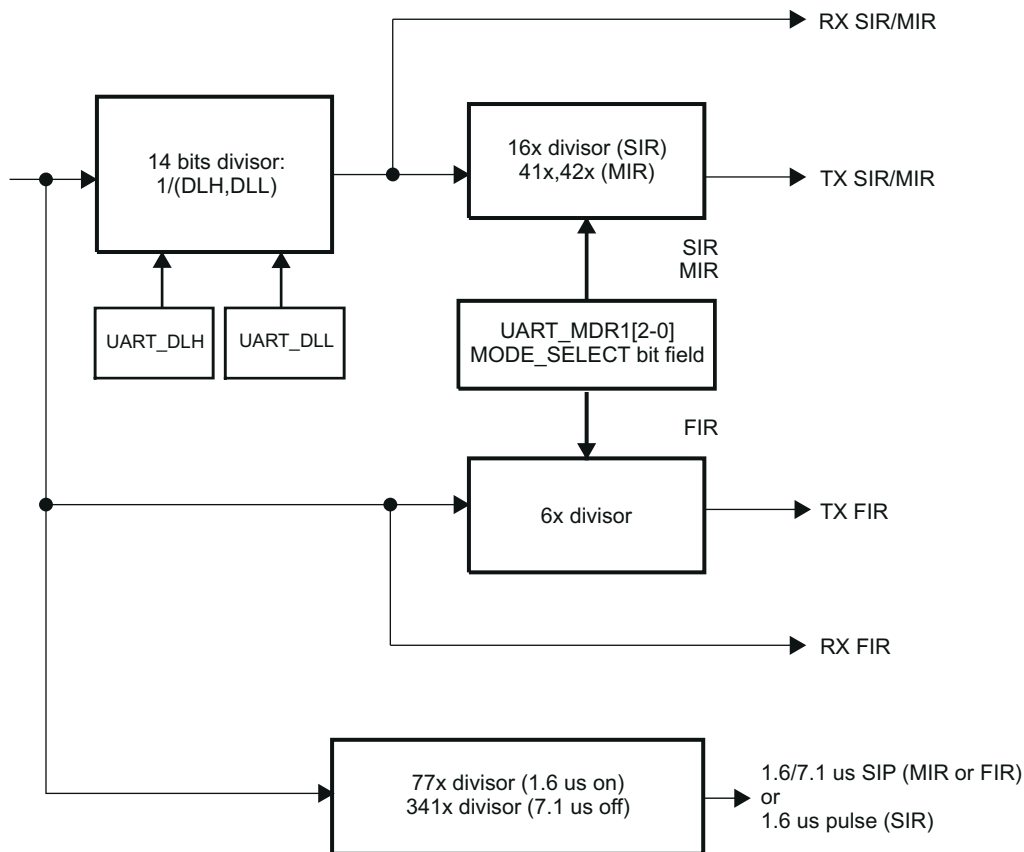
**Figure 12-125. RS-485 External Transceiver Direction Control**

#### 12.1.6.4.8.3 IrDA Mode

##### 12.1.6.4.8.3.1 IrDA Clock Generation: Baud Generator

The IrDA function contains a programmable baud generator and a set of fixed dividers that divide the 48-MHz clock input down to the expected baud rate.

Figure 12-126 shows the baud rate generator and associated controls.



uart-033

**Figure 12-126. IrDA Baud Rate Generator**

**CAUTION**

Before initializing or modifying clock parameter controls (UART\_DLH, UART\_DLL), MODE\_SELECT=DISABLE (UART\_MDR1[2-0] MODE\_SELECT) must be set to 0x7). Failure to observe this rule can result in unpredictable module behavior.

**12.1.6.4.8.3.2 Choosing the Appropriate Divisor Value**

Three divisor values are:

- SIR mode: Divisor value = Operating frequency/(16× baud rate)
- MIR mode: Divisor value = Operating frequency/(41×/42× baud rate)
- FIR mode: Divisor value = None

Table 12-137 lists the IrDA baud rate settings.

**Table 12-137. IrDA Baud Rate Settings**

Baud Rate	IR Mode	Baud Multiple	Encoding	DLH, DLL (Decimal)	Actual Baud Rate	Error (%)	Source Jitter (%)	Pulse Duration
2.4 kbps	SIR	16x	3/16	1250	2.4 kbps	0	0	78.1 μs
9.6 kbps	SIR	16x	3/16	312	9.6153 kbps	+0.16	0	19.5 μs
19.2 kbps	SIR	16x	3/16	156	19.231 kbps	+0.16	0	9.75 μs
38.4 kbps	SIR	16x	3/16	78	38.462 kbps	+0.16	0	4.87 μs
57.6 kbps	SIR	16x	3/16	52	57.692 kbps	+0.16	0	3.25 μs
115.2 kbps	SIR	16x	3/16	26	115.38 kbps	+0.16	0	1.62 μs



**Table 12-137. IrDA Baud Rate Settings (continued)**

Baud Rate	IR Mode	Baud Multiple	Encoding	DLH, DLL (Decimal)	Actual Baud Rate	Error (%)	Source Jitter (%)	Pulse Duration
0.576 Mbps	MIR	41×/42×	1/4	2	0.5756 Mbps <sup>(1)</sup>	0	+1.63/-0.80	416 ns
1.152 Mbps	MIR	41×/42×	1/4	1	1.1511 Mbps <sup>(1)</sup>	0	+1.63/-0.80	208 ns
4 Mbps	FIR	6×	4 PPM	–	4 Mbps	0	0	125 ns

(1) Average value

#### Note

Baud rate error and source jitter table values do not include 48-MHz reference clock error and jitter.

#### 12.1.6.4.8.3.3 IrDA Data Formatting

The methods described in this section apply to all IrDA modes (SIR, MIR, and FIR).

##### 12.1.6.4.8.3.3.1 IR RX Polarity Control

The UART\_MDR2[6] IRRXINVERT bit provides the flexibility to invert the RX pin in the UART to ensure that the protocol at the output of the transceiver has the same polarity at module level. By default, the RX pin is inverted because most transceivers invert the IR receive pin.

##### 12.1.6.4.8.3.3.2 IrDA Reception Control

The module can transmit and receive data, but when the device is transmitting, the IR RX circuitry is automatically disabled by hardware.

Operation of the RX input can be disabled by the UART\_ACREG[5] DIS\_IR\_RX bit.

##### 12.1.6.4.8.3.3.3 IR Address Checking

In all IR modes, when address checking is enabled, only frames intended for the device are written to the RX FIFO. This restriction avoids receiving frames not meant for this device in a multipoint infrared environment. It is possible to program two frame addresses that the UART IrDA receives, with the UART\_XON1\_ADDR1[7-0] XON\_WORD1 and UART\_XON2\_ADDR2[7-0] XON\_WORD2 bit fields.

Setting the UART\_EFR[0] bit to 1 selects address1 checking. Setting the UART\_EFR[1] bit to 1 selects address2 checking. Setting the UART\_EFR[1-0] bit field to 0 disables all address checking operations. If both bits are set, the incoming frame is checked for private and public addresses.

If address checking is disabled, all received frames write to the RX FIFO.

##### 12.1.6.4.8.3.3.4 Frame Closing

A transmission frame can be terminated in two ways:

- Frame-length method: Set the UART\_MDR1[7] FRAME\_END\_MODE bit to 0. The Host CPU writes the value of the frame length to the UART\_TXFLH and UART\_TXFLL registers. The device automatically attaches end flags to the frame when the number of bytes transmitted equals the value of the frame length.
- Set-EOT bit method: Set the UART\_MDR1[7] FRAME\_END\_MODE bit to 1. The Host CPU writes 1 to the UART\_ACREG[0] EOT bit just before it writes the last byte to the TX FIFO. When the Host CPU writes the last byte to the TX FIFO, the device internally sets the tag bit for that character in the TX FIFO. As the TX state-machine reads data from the TX FIFO, it uses this tag-bit information to attach end flags and correctly terminate the frame.

##### 12.1.6.4.8.3.3.5 Store and Controlled Transmission

In store and controlled transmission (SCT) mode, the Host CPU starts writing data to the TX FIFO. Then, after writing a part of a frame (for a bigger frame) or an entire frame (a small frame; that is, a supervisory frame), the Host CPU writes 1 to the UART\_ACREG[2] SCTX\_EN bit (deferred TX start) to start transmission.



SCT mode is enabled by setting the UART\_MDR1[5] SCT bit to 1. This transmission method differs from normal mode, in which data transmission starts immediately after data is written to the TX FIFO. SCT mode is useful for sending short frames without TX underrun.

#### **12.1.6.4.8.3.3.6 Error Detection**

When the UART\_LSR\_UART register is read, the UART\_LSR\_UART[4-2] bit field reflects the error bits [FL, CRC, ABORT] of the frame at the top of the STATUS FIFO (the next frame status to be read).

The error is triggered by an interrupt (for IrDA mode interrupts, see [Table 12-121](#)). The STATUS FIFO must be read until empty (a maximum of eight reads is required).

#### **12.1.6.4.8.3.3.7 Underrun During Transmission**

Underrun during transmission occurs when the TX FIFO is empty before the end of the frame is transmitted. When underrun occurs, the device closes the frame with end flags but attaches an incorrect CRC value. The receiving device detects a CRC error and discards the frame; it can then ask for a retransmission.

Underrun also causes an internal flag to be set, which disables additional transmissions. Before the next frame can be transmitted, the Host CPU must:

- Reset the TX FIFO.
- Read the UART\_RESUME register, which clears the internal flag.

This function can be disabled by the UART\_ACREG[4] DIS\_TX\_UNDERRUN bit, compensated by the extension of the stop-bit in transmission if the TX FIFO is empty.

#### **12.1.6.4.8.3.3.8 Overrun During Receive**

Overrun during receive for the IrDA mode has the same function as that for the UART mode (see [Section 12.1.6.4.8.1.3.6, Overrun During Receive](#)).

#### **12.1.6.4.8.3.3.9 Status FIFO**

In IrDA modes, a status FIFO records the received frame status. When a complete frame is received, the length of the frame and the error bits associated with the frame are written to the status FIFO.

Reading the UART\_SFREGH[3-0] MSB and UART\_SFREGL[3-0] (LSB) bit fields obtains the frame length. The frame error status is read in the UART\_SFLSR register. Reading the UART\_SFLSR register increments the status FIFO read pointer. Because the status FIFO is eight entries deep, it can hold the status of eight frames.

The Host CPU uses the frame-length information to locate the frame boundary in the received frame data. The Host CPU can screen bad frames using the error status information and can later request the sender to resend only the bad frames.

This status FIFO can be used effectively in DMA mode because the Host CPU must be interrupted only when the programmed status FIFO trigger level is reached, not each time a frame is received.

#### **12.1.6.4.8.3.3.10 Multi-drop Parity Mode with Address Match**

Multi-drop mode is enabled in the UART\_EFR2 register.

Address matching mode is only available with 8 bit character length setting. UART\_LCR[1-0] CHAR\_LENGTH bit fields should always be set to 0x11 (8 bits) prior to enabling the feature.

This mode allows the transmitter to send data on a line where multiple receivers are connected, when supported. In this mode, a set parity bit is used to mark an address, and a parity of 0 denotes data.

This setting affects how the parity is generated. Writing a 0x1 into the UART\_ECR[0] A\_MULTIDROP bit will set the parity bit for the next byte to be sent, which will then be considered an address, for sending a data frame, the UART\_ECR[0] A\_MULTIDROP bit has to be cleared.

On reception if the feature is enabled by setting the UART\_EFR2[2] MULTIDROP bit to 0x1 incoming frames with parity set to 0x1 are treated as address frames and with parity set to 0x0 as data frames. The receiver will drop all data frames until a matching address frame was found.

The matching address is determined by the values set in UART\_MAR, UART\_MMR and UART\_MBR registers and the value set in UART\_EFR2[7] BROADCAST bit.

Table 12-138 summarizes the operation of address matching based on the mentioned values.

**Table 12-138. Details of address matching**

Received frame	Received parity	Frame type	UART_MAR	UART_MMR	UART_MBR	UART_EFR2[7] BROADCAST	Operation of receiver	Address matching
0xXX <sup>(2)</sup>	0	DATA	X <sup>(1)</sup>	X <sup>(1)</sup>	0xXX <sup>(2)</sup>	X <sup>(1)</sup>	Drops data until matching address found	N/A
0xXX <sup>(2)</sup>	1	ADDRESS	0xXX <sup>(2)</sup>	0x00	0xXX <sup>(2)</sup>	0	Matches any address	Yes
0xEF	1	ADDRESS	0xXX <sup>(2)</sup>	0xXX <sup>(2)</sup>	0xEF	1	Matches broadcast address	Yes
0x1A	1	ADDRESS	0x1A	0xFF	0xXX <sup>(2)</sup>	0	Single address match	Yes
0xF5	1	ADDRESS	0xF3	0xF9	0xXX <sup>(2)</sup>	0	Group address match	Yes

(1) X indicates a do not care bit value

(2) 0xXX indicates a do not care 8 bit hexadecimal value

The possible values for matching address can be calculated in the following way:

- Single and Group addresses can be formed by masking the UART\_MAR registers value with the value set in the UART\_MMR register, bits set to 0x0 in the UART\_MMR register result in do not care values.
- Broadcast addresses can be set in the UART\_MBR register if broadcast address is enabled in the UART\_EFR2[7] BROADCAST bit, the module will match on received address frames containing the broadcast address.
- For more details, see example below:
  - UART\_MAR: 0xF3, UART\_MMR: 0xF9, UART\_MBR: 0xFF
  - Single and Group addresses: 0xF1, 0xF3, 0xF5, 0xF7
  - Broadcast addresses: 0xFF

If an address match occurred the matching address value can be obtained from the UART\_RHR register in the following way:

- If the FIFO is disabled or the threshold is set to 0x1, the matching address can be directly read from UART\_RHR as the FIFO will not be overwritten.
- If the FIFO is enabled or the threshold is greater than 0x1, the matching address will be the latest frame in the FIFO with a parity error bit set.

For received data, the parity error bit in the UART\_LSR\_UART register is set when a bit with a parity of 0x1 is received indicating an address frame and the received address matches based on the values of UART\_MAR, UART\_MMR, UART\_MBR and UART\_EFR2[2] MULTIDROP bit.

In Multi-drop mode no parity is used, as the parity bit is used to differentiate address and data frames. The parity error bit is used for indicating an address match.

For enabling the interrupt generation for address matching UART\_IER\_UART[2] LINE\_STS\_IT bit has to be set to 0x1.

An interrupt for the matching address can be identified by reading the UART\_IIR\_UART[5-1] IT\_TYPE bit fields, a value of 0x00011 indicates a receiver line status error. After the UART\_LSR\_UART[2] RX\_PE bit has to be read, a value of 0x1 indicates that an address match occurred. The reception of a frame is indicated with a value

of 0x1 in the UART\_LSR\_UART[0] RX\_FIFO\_E bit as the matching value is written into the FIFO regardless of the frame type (data or address). UART\_LSR\_UART[7] RX\_FIFO\_STS bit will also be set to 0x1 as the parity error bit is used to indicate a matching address.

Note that the operation of the UART\_LSR\_UART[2] RX\_PE bit depends on the value set in UART\_EFR2[2] MULTIDROP bit. If UART\_EFR2[2] MULTIDROP bit is set to 0x0, UART\_LSR\_UART[2] RX\_PE bit is used to indicate a received parity error. If UART\_EFR2[2] MULTIDROP bit is set to 0x1, the receiver is in Multi-drop Address Match mode, thus the value in UART\_LSR\_UART[2] RX\_PE bit is used to indicate an address match.

The interrupt is cleared the same way in both operation modes: reading the UART\_LSR\_UART register updates the values.

This feature is available in UART and synchronous modes. The ISO7816 has not defined Multidrop Parity Mode, so the feature should be left off.

#### **12.1.6.4.8.3.3.11 Time-guard**

The time-guard feature enables the UART interface to operate with slow remote devices.

When set, it will insert a number of idle states between transmitting two characters, the length of which can be set in the UART\_TIMEGUARD register. The value in the register defines the number of baud clocks of idle period to insert.

This idle state essentially acts like a long stop bit. In UART and synchronous modes, a Timeguard is added in addition to the stop bit. In ISO7816 there is a waiting period rather than an actual stop bit. Software should set 1-2 or more Timeguard cycles according to the protocol used and the card requirements.

#### **12.1.6.4.8.3.4 SIR Mode Data Formatting**

This section provides specific instructions for SIR mode programming.

##### **12.1.6.4.8.3.4.1 Abort Sequence**

The transmitter can prematurely close a frame (abort) by sending the sequence 0x7DC1. The abort pattern closes the frame without a CRC field or an ending flag.

A transmission frame can be aborted by setting the UART\_ACREG[1] ABORT\_EN bit to 1. When this bit is set to 1, 0x7D and 0xC1 are transmitted and the frame is not terminated with CRC or stop flags.

When a 0x7D character followed immediately by a 0xC1 character is received without transparency, the receiver treats a frame as an aborted frame.

#### **CAUTION**

When the TX FIFO is not empty and the UART\_MDR1[5] SCT bit is set to 1, the UART IrDA starts a new transfer with data of a previous frame when the aborted frame is sent. Therefore, the TX FIFO must be reset before sending an aborted frame.

##### **12.1.6.4.8.3.4.2 Pulse Shaping**

SIR mode supports the 3/16 or the 1.6-μs pulse duration methods. The UART\_ACREG[7] PULSE\_TYPE bit selects the pulse width method in the transmit mode.

##### **12.1.6.4.8.3.4.3 SIR Free Format Programming**

The SIR FF mode is selected by setting the module in the UART mode (UART\_MDR1[2-0] MODE\_SELECT = 0x0) and the UART\_MDR2[3] PULSE bit to 1 to allow pulse shaping.

Because the bit format stays the same, some UART mode configuration registers must be set at specific values:

- UART\_LCR[1-0] CHAR\_LENGTH bit field = 0x3 (8 data bits)
- UART\_LCR[2] NB\_STOP bit = 0x0 (1 stop-bit)
- UART\_LCR[3] PARITY\_EN bit = 0x0 (no parity)

The UART mode interrupts are used for the SIR FF mode, but many are not relevant (XOFF, RTS, CTS, modem status register, etc.).

#### **12.1.6.4.8.3.5 MIR and FIR Mode Data Formatting**

This section describes common instructions for FIR and MIR mode programming.

At the end of a frame reception, the CPU reads the line status register (UART\_LSR\_UART) to detect errors in the received frame.

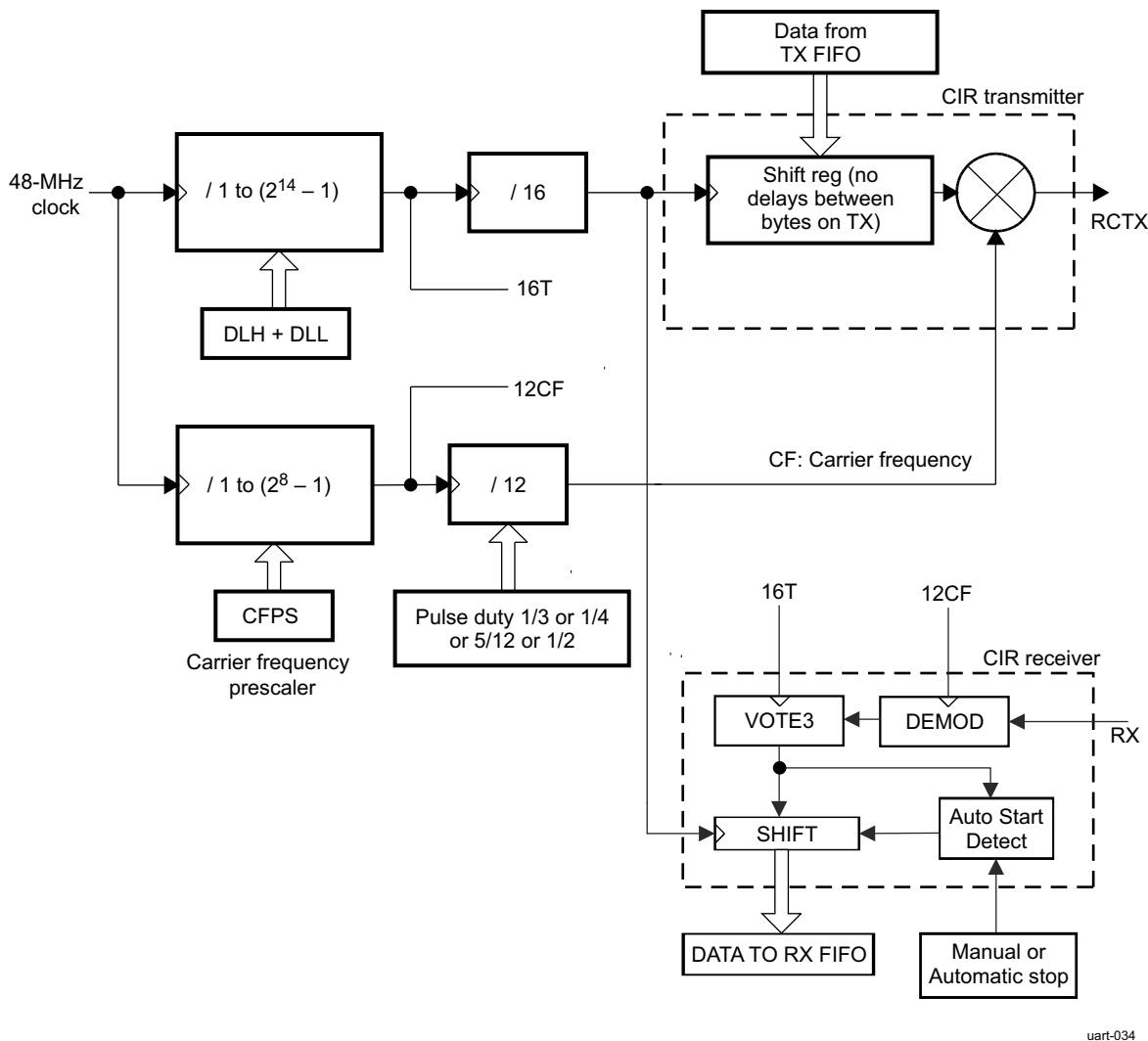
When the UART\_MDR1[6] SIP\_MODE bit is set to 1, the TX state-machine always sends one SIP at the end of a transmission frame. However, when the SIP\_MODE bit is set to 0, SIP transmission depends on the UART\_ACREG[3] SEND\_SIP bit.

The CPU can set the SEND\_SIP bit at least once every 500 ms. The advantage of this approach over the default approach is that the TX state-machine does not have to send the SIP at the end of each frame, thus reducing the overhead required.

#### **12.1.6.4.8.4 CIR Mode**

##### **12.1.6.4.8.4.1 CIR Mode Clock Generation**

Depending on the encoding method (variable pulse distance/biphase), the Host CPU must develop a data structure that combines 1 and 0 with a  $t$  period to encode the complete frame to transmit. This can then be transmitted to the infrared output with a modulation method, as shown in [Figure 12-127](#).



uart-034

**Figure 12-127. CIR Mode Block Components**

Based on the requested modulation frequency, the UART\_CFPS register must be set with the correct dividing value to provide an accurate pulse frequency:

$$\text{Dividing value} = (\text{FCLK} / 12) / \text{MODfreq}$$

Where:

FCLK = System clock frequency (48 MHz)

12 = Real value of baud multiple

MODfreq = Effective frequency of the modulation (MHz)

Example: For a targeted modulation frequency of 36 kHz, the value of CFPS must be set to 0x7 (decimal), which provides a modulation frequency of 36.04 kHz.

#### Note

The UART\_CFPS register starts with a reset value of 105 (decimal), which translates to a frequency of 38.1 kHz.

The duty cycle of these pulses is user-defined by the pulse duty register bits in the UART\_MDR2 register. [Table 12-139](#) shows the duty cycle.

**Table 12-139. CIR Duty Cycle**

UART_MDR2[5-4] CIR_PULSE_MODE	Duty Cycle (High-Level)
00	1/4
01	1/3
10	5/12
11	1/2

#### 12.1.6.4.8.4.2 CIR Data Formatting

The methods described in this section apply to all CIR modes.

##### 12.1.6.4.8.4.2.1 IR RX Polarity Control

The IR RX polarity control for CIR mode has the same function as that for IrDA mode (see [Section 12.1.6.4.8.3.3.1, IR RX Polarity Control](#)).

##### 12.1.6.4.8.4.2.2 CIR Transmission

In transmission, the Host CPU software must exercise an element of real-time control to transmit data packets, each of which must be emitted at a constant delay from the start-bits of each individual packet. Thus, when sending a series of packets, the packet-to-packet delay must respect a specific delay. Two methods can be used to control this delay:

- Filling the TX FIFO with a number of zero bits that are transmitted with a  $t$  period
- Using an external system timer to control the delay between each start-of-frame or between the end of a frame and the start of the next one. This can be performed by:
  - Controlling the start of the frame using the UART\_MDR1[5] SCT bit and the UART\_ACREG[2] SCTX\_EN bit, depending on the timer status
  - Using the UART\_IIR\_UART[5] TX\_STATUS\_IT interrupt bit to preload the next frame in the TX FIFO and to control the start of the timer (in case of control delay between the end of a frame and the start of the next frame)

##### 12.1.6.4.8.4.2.3 CIR Reception

There are 2 ways to stop a CIR reception:

- The Host CPU can disable the reception by setting the UART\_ACREG[5] DIS\_IR\_RX bit to 1. When it considers that the reception is finished because a large number of 0 has been received. To receive a new frame, the UART\_ACREG[5] DIS\_IR\_RX bit must be set to 0.
- An automatic stop mechanism can be configured by setting a value in the BOF length register (UART\_EBLR). If the value set in the UART\_EBLR register is different than 0, this feature is enabled and the number of bits received will begin counting from 0. When the counter reaches the value defined in the UART\_EBLR register, reception is automatically disabled and UART\_IIR\_CIR[2] RX\_STOP\_IT bit is set. When a 1 is detected on the RX pin, reception is automatically re-enabled.

There is a limitation when receiving data in UART CIR mode. Certain IrDA transceivers on the market have a characteristic that causes shrinking of the received modulation pulse hold-time. The UART receive filtering schema is based on the same encoding mechanism used for transmission.

For the following scenario:

- Shift register period: 0.9μs
- Modulation frequency: 36kHz
- Duty cycle: 1/4 of a modulation frequency period

Data sent with these conditions would contain 7µs pulses within a 28µs period. The UART expects to receive similar incoming data on receive, but various transceiver timing characteristics typically only send 2µs modulated pulses. These 2µs pulses will be filtered out and RX FIFO will not receive data. This does not affect UART CIR mode in transmission.

CIR RX demodulation can be bypassed by setting the UART\_MDR3[0] DISABLE\_CIR\_RX\_DEMOD bit.

### 12.1.6.5 UART Programming Guide

This section describes the procedure for operating the UART with FIFO and DMA or interrupts. This three-part procedure ensures the quick start of the UART. It does not cover every UART feature.

The first programming model covers software reset of the UART. The second programming model describes FIFO and DMA configuration. The last programming model describes protocol, baud rate, and interrupt configuration.

#### Note

Each programming model can be used independently of the other two; for instance, reconfiguring the FIFOs and DMA settings only.

Each programming model can be executed starting from any UART register access mode (register modes, submodes, and other register dependencies). However, if the UART register access mode is known before executing the programming model, some steps that enable or restore register access are optional. For more information, see [Section 12.1.6.4.7.1, Register Access Modes](#).

#### 12.1.6.5.1 UART Global Initialization

##### 12.1.6.5.1.1 Surrounding Modules Global Initialization

This section identifies the requirements for initializing the surrounding modules when the UART module is to be used for the first time after a device reset. This initialization of surrounding modules is based on the integration of the UART.

For more information, see [Table 12-140](#).

**Table 12-140. UART Global Initialization of Surrounding Modules**

Surrounding Modules	Comments
LPSC0	Module reset must be enabled. For more information about the module configuration, see <i>Reset</i> .
LPSC3	Module reset must be enabled. For more information about the module configuration, see <i>Reset</i> .
LPSC7	Module reset must be enabled. For more information about the module configuration, see <i>Reset</i> .
PLLCTRL0	PLLCTRL0 configuration must be done to enable the clocks to the UART modules, see <i>Clocking</i> .
WKUP_PLLCTRL0	WKUP_PLLCTRL0 configuration must be done to enable the clocks to the UART modules, see <i>Clocking</i> .
WKUP_HFOSC0	WKUP_HFOSC0 configuration must be done to enable the clocks to the UART modules, see <i>Clocking</i> .
PLL1	PLL1 configuration must be done to enable the clocks to the UART modules, see <i>Clocking</i> .
MCU_PLL1	MCU_PLL1 configuration must be done to enable the clocks to the UART modules, see <i>Clocking</i> .
COMPUTE_CLUSTER0	Device INTCs must be configured to enable the interrupt request generation. For information about enabling COMPUTE_CLUSTER0 interrupts, see <i>Interrupts</i> .
R5FSS0/1_INTRTR0	Device INTCs must be configured to enable the interrupt request generation. For information about enabling R5FSS0/1_INTRTR0 interrupts, see <i>Interrupts</i> .
MCU_R5FSS0_CORE0/1	Device INTCs must be configured to enable the interrupt request generation. For information about enabling MCU_R5FSS0_CORE0/1 interrupts, see <i>Interrupts</i> .
WKUP_DMSC0	Device INTCs must be configured to enable the interrupt request generation. For information about enabling WKUP_DMSC0 interrupts, see <i>Interrupts</i> .
PRU-ICSSG0/1	Device INTCs must be configured to enable the interrupt request generation. For information about enabling PRU_ICSSG0/1 interrupts, see <i>Interrupts</i> .
C66SS0/1_INTRTR0	Device INTCs must be configured to enable the interrupt request generation. For information about enabling C66SS0/1_INTRTR0 interrupts, see <i>Interrupts</i> .
MAIN2MCU_LVL_INTROUTER0	Device INTCs must be configured to enable the interrupt request generation. For information about enabling MAIN2MCU_LVL_INTROUTER0 interrupts, see <i>Interrupts</i> .
R5FSS0_CORE0/1	Device INTCs must be configured to enable the interrupt request generation. For information about enabling R5FSS0_CORE0/1 interrupts, see <i>Interrupts</i> .



**Table 12-140. UART Global Initialization of Surrounding Modules (continued)**

Surrounding Modules	Comments
R5FSS1_CORE0/1	Device INTCs must be configured to enable the interrupt request generation. For information about enabling R5FSS1_CORE0/1 interrupts, see <i>Interrupts</i> .
MCU_PDMA0	MCU_PDMA0 controllers configuration must be done to enable the module MCU_PDMA0 channel request, see <i>Data Movement Architecture (DMA)</i> .
PDMA_UART_G0	PDMA_UART_G0 controllers configuration must be done to enable the module PDMA_UART_G0 channel request, see <i>Data Movement Architecture (DMA)</i> .
PDMA_UART_G1	PDMA_UART_G1 controllers configuration must be done to enable the module PDMA_UART_G1 channel request, see <i>Data Movement Architecture (DMA)</i> .
PDMA_UART_G2	PDMA_UART_G2 controllers configuration must be done to enable the module PDMA_UART_G2 channel request, see <i>Data Movement Architecture (DMA)</i> .
Interconnects	For information about the WKUP_CBASS0, MCU_CBASS0, and CBASS0 interconnects configuration, see <i>System Interconnect</i> .

#### 12.1.6.5.1.2 UART Module Global Initialization

The procedure in [Table 12-141](#) can be used to initialize UART when performing software reset.

**Table 12-141. UART Global Initialization**

Step	Register/Bit Field/Programming Model	Value
Perform a software reset.	UART_SYSC[1] SOFTRESET	1
Wait until reset is finished.	UART_SYSS[0] RESETDONE	=1

#### 12.1.6.5.2 UART Mode selection

[Table 12-142](#) describes how to set different register access mode.

**Table 12-142. UART Configure Register Access Mode**

Step	Register/Bit Field/Programming Model	Value
Set the register access mode A	UART_LCR[7] DIV_EN	1
	UART_LCR[7-0]	≠0xBF
Set the register access mode B	UART_LCR[7-0]	0xBF
Set the operational mode	UART_LCR[7] DIV_EN	0

#### 12.1.6.5.3 UART Submode selection

This section describes how to set different register access submode.

**Table 12-143. UART Configure Register Access Submode TCR\_TLR**

Step	Register/Bit Field/Programming Model	Value
Configure the submode TCR_TLR		
Configure mode B	see <a href="#">Table 12-142</a>	
Enable writing to register bits UART_MCR[7-5]	UART_EFR[4] ENHANCED_EN	1
Configure mode A	see <a href="#">Table 12-142</a>	0x1
Set the submode TCR_TLR	UART_MCR[6] TCR_TLR	1

**Table 12-144. UART Configure Register Access Submode MSR\_SPR**

Step	Register/Bit Field/Programming Model	Value
First option: configure the submode MSR_SPR		
Configure mode B	see <a href="#">Table 12-142</a>	
Set the submode MSR_SPR	UART_EFR[4] ENHANCED_EN	0
Second option: configure the submode MSR_SPR		
Configure mode B	see <a href="#">Table 12-142</a>	
Enable writing to register bits UART_MCR[7-5]	UART_EFR[4] ENHANCED_EN	1

**Table 12-144. UART Configure Register Access Submode MSR\_SPR (continued)**

Step	Register/Bit Field/Programming Model	Value
Set the submode MSR_SPR	UART_MCR[6] TCR_TLR	0

**Table 12-145. UART Configure Register Access Submode XOFF**

Step	Register/Bit Field/Programming Model	Value
Configure of the XOFF		
Configure B	see <a href="#">Table 12-142</a>	
Set the submode XOFF	UART_EFR[4] ENHANCED_EN	0

**12.1.6.5.4 UART Load FIFO trigger and DMA mode settings****12.1.6.5.4.1 DMA mode Settings**

To enable and configure program the DMA mode, perform the following steps:

**Table 12-146. DMA Mode Settings**

Step	Register/Bit Field/Programming Model	Value
Set the option of DMA mode configuration	UART_SCR[0] DMA_MODE_CTL	-
IF Configure DMA mode 0 and 1	UART_SCR[0] DMA_MODE_CTL	=0
Select the DMA mode, for more information see <a href="#">Section 12.1.6.4.6.4</a>	UART_FCR[3] DMA_MODE	-
IF Configure DMA mode from 0 to 3	UART_SCR[0] DMA_MODE_CTL	=1
Select the DMA mode, for more information see <a href="#">Section 12.1.6.4.6.4</a>	UART_SCR[2-1] DMA_MODE_2	-

**12.1.6.5.4.2 FIFO Trigger Settings**

In this section is described configuration and settings of FIFO trigger level, which enable DMA and interrupt generation.

**Table 12-147. Load FIFO Triggers Defined by the FCR**

Step	Register/Bit Field/Programming Model	Value
Configure register submode TCR_TLR	see <a href="#">Table 12-143</a>	0x-
Set the desire RX FIFO trigger level	UART_FCR[5-4] TX_FIFO_TRIG	0x-
Set the desire TX FIFO trigger level	UART_FCR[7-6] RX_FIFO_TRIG	0x-

**Table 12-148. Load FIFO Triggers Defined by the TLR**

Step	Register/Bit Field/Programming Model	Value
Configure register submode TCR_TLR	see <a href="#">Table 12-143</a>	0x-
Set the desire RX FIFO trigger level	UART_TLR[7-4] RX_FIFO_TRIG_DMA	0x-
Set the desire TX FIFO trigger level	UART_TLR[3-0] TX_FIFO_TRIG_DMA	0x-

**Table 12-149. Load FIFO Triggers Defined by the Concatenated Value**

Step	Register/Bit Field/Programming Model	Value
Configure register submode TCR_TLR	see <a href="#">Table 12-143</a>	0x-
Set the register bit	UART_SCR[7] RX_TRIG_GRANU1	1
Set the desire RX FIFO trigger level	UART_TLR[7-4] RX_FIFO_TRIG_DMA UART_FCR[7-6] RX_FIFO_TRIG	0x-
Set the register bit	UART_SCR[6] TX_TRIG_GRANU1	1
Set the desire TX FIFO trigger level	UART_TLR[3-0] TX_FIFO_TRIG_DMA UART_FCR[5-4] TX_FIFO_TRIG	0x-

### 12.1.6.5.5 UART Protocol, Baud rate and interrupt settings

#### 12.1.6.5.5.1 Baud rate settings

**Table 12-150. UART Baud Rate Settings**

Step	Register/Bit Field/Programming Model	Value
Disable UART mode	UART_MDR1[2-0] MODE_SELECT	0x7
Switch to register configuration mode B	see <a href="#">Table 12-142</a>	
Enable access to UART_IER_UART[7-4]	UART_EFR[4] ENHANCED_EN	1
Switch register operational mode	see <a href="#">Table 12-142</a>	
Disable sleep mode	UART_IER_UART[4] SLEEP_MODE	0
Switch to register configuration mode A or B	see <a href="#">Table 12-142</a>	
Set the appropriate divisor value	UART_DLL[7-0] CLOCK_LSB UART_DLH[5-0] CLOCK_MSB	0x-

#### 12.1.6.5.5.2 Interrupt settings

**Table 12-151. UART Interrupt Settings**

Step	Register/Bit Field/Programming Model	Value
Switch to register configuration mode B	see <a href="#">Table 12-142</a>	0x7
Enable access to UART_IER_UART[7-4]	UART_EFR[4] ENHANCED_EN	1
Switch register operational mode	see <a href="#">Table 12-142</a>	
Set the desired interrupt configuration (0: Disable the interrupt; 1: Enable the interrupt)	UART_IER_UART[7] CTS_IT UART_IER_UART[6] RTS_IT UART_IER_UART[5] XOFF_IT UART_IER_UART[4] SLEEP_MODE UART_IER_UART[3] MODEM_STS_IT UART_IER_UART[2] LINE_STS_IT UART_IER_UART[1] THR_IT UART_IER_UART[0] RHR_IT	0x-

#### 12.1.6.5.5.3 Protocol settings

Load the desired protocol formatting (parity, stop-bit, character length) and switch to register operational mode.

**Table 12-152. UART Protocol Settings**

Step	Register/Bit Field/Programming Model	Value
Load desired protocol formatting, see <a href="#">Section 12.1.6.4.8.1.3.1</a> , <i>Frame Formatting</i>	UART_LCR[5] PARITY_TYPE_2 UART_LCR[4] PARITY_TYPE_1 UART_LCR[3] PARITY_EN UART_LCR[2] NB_STOP UART_LCR[1-0] PARITY_LENGTH	0x-
Switch to register operational mode	UART_LCR[7] DIV_EN UART_LCR[6] BREAK_EN	0

#### 12.1.6.5.5.4 UART/RS-485/IrDA(SIR/MIR/FIR)/CIR

**Table 12-153. UART Mode Selection**

Step	Register/Bit Field/Programming Model	Value
Load the desired UART/IrDA (SIR, MIR, FIR)/CIR modes, see <a href="#">Section 12.1.6.4.7.2</a> , <i>UART/RS-485/IrDA (SIR, MIR, FIR)/CIR Mode Selection</i>	UART_MDR1[2-0] MODE_SELECT	0x-

**Table 12-153. UART Mode Selection (continued)**

Step	Register/Bit Field/Programming Model	Value
Load the desired RS-485 mode, see <a href="#">Section 12.1.6.4.7.2, UART/RS-485/IrDA (SIR, MIR, FIR)/CIR Mode Selection</a>	UART_MDR3[4] DIR_EN	0x1

**12.1.6.5.5.5 UART Multi-drop Parity Address Match Mode Configuration****Table 12-154. UART Multi-drop Parity Address Match Mode Configuration**

Step	Register/Bit Field/Programming Model	Value
Disable receive mode	UART_ECR[3] RX_EN	0
Enable Multi-drop parity Address match mode	UART_EFR2[2] MULTIDROP	1
Set the matching device address	UART_MAR[7-0] ADDRESS	0x-
Set the address match masking	UART_MMR[7-0] MASK	0x-
Set the broadcast address match	UART_MBR[7-0] BROADCAST_ADDRESS	0x-
Enable broadcast address matching if needed	UART_EFR2[7] BROADCAST	1
Enable receive mode	UART_ECR[3] RX_EN	1

**12.1.6.5.6 UART Hardware and Software Flow Control Configuration**

This section describes the programming steps to enable and configure hardware and software flow control. Hardware and software flow control cannot be used at the same time.

**12.1.6.5.6.1 Hardware Flow Control Configuration****Table 12-155. UART Hardware Flow Control Configuration**

Step	Register/Bit Field/Programming Model	Value
Configure register submode TCR_TLR	see <a href="#">Table 12-143</a>	0x7
Load the start and halt trigger value.	UART_TCR[7-4] AUTO_RTS_START UART_TCR[3-0] AUTO_RTS_HALT	0x-
Enable or disable receive and transmit hardware flow control mode.	UART_EFR[7] AUTO_CTS_EN UART_EFR[6] AUTO_RTS_EN	0x-

**12.1.6.5.6.2 Software Flow Control Configuration****Table 12-156. UART Software Flow Control Configuration**

Step	Register/Bit Field/Programming Model	Value
Set the register access submode XOFF	see <a href="#">Table 12-145</a>	
Load the software control characters	UART_XON1_ADDR1[7-0] XON_WORD1 UART_XON2_ADDR2[7-0] XON_WORD2 UART_XOFF1[7-0] XOFF_WORD1 UART_XOFF2[7-0] XOFF_WORD2	0x-
Set the register access submode TCR_TLR	see <a href="#">Table 12-143</a>	
Enable or disable XON any function (0: Disable; 1: Enable).	UART_MCR[5] XON_EN	--
Load start and halt trigger value for software flow control	UART_TCR[7-4] AUTO_RTS_START UART_TCR[3-0] AUTO_RTS_HALT	0x-
Enable or disable special character function (0: Disable; 1: Enable)	UART_EFR[5] SPEC_CHAR	0x-
Set the software flow control mode	UART_EFR[3-0] SW_FLOW_CONTROL	0x-

### 12.1.6.5.7 IrDA Programming Model

#### 12.1.6.5.7.1 SIR mode

##### 12.1.6.5.7.1.1 Receive

The following programming model explains how to program the module to receive an IrDA frame with parity forced to 1, baud rate = 115.2 kbps, FIFOs disabled, 2 stop-bits, and 8-bit word length:

**Table 12-157. SIR Mode Receive Settings**

Step	Register/Bit Field/Programming Model	Value
Disable UART mode	UART_MDR1[2-0] MODE_SELECT	0x7
Grant access to the UART_DLL and UART_DLH registers	UART_LCR[7-0]	0x80
Load the baud rate(115.2 Kbps)	UART_DLL[7-0] CLOCK_LSB	0x1A
	UART_DLH[5-0] CLOCK_MSB	0x00
Set SIR mode	UART_MDR1[2-0] MODE_SELECT	0x1
Disable access to the UART_DLL and UART_DLH registers	UART_LCR[7-0]	0x00
Enable the UART_RHR interrupt	UART_IER_IRDA[0] RHR_IT	1

##### 12.1.6.5.7.1.2 Transmit

The following programming model explains how to program the module to transmit an IrDA 6-byte frame with no parity, baud rate = 115.2 kbps, FIFOs disabled, 3/16 encoding, 2 stop-bits, and 7-bit word length:

**Table 12-158. SIR Mode Transmit Settings**

Step	Register/Bit Field/Programming Model	Value
Disable UART mode	UART_MDR1[2-0] MODE_SELECT	0x7
Grant access to the UART_DLL and UART_DLH registers	UART_LCR[7-0]	0x80
Load the baud rate (115.2 Kbps)	UART_DLL[7-0] CLOCK_LSB	0x1A
	UART_DLH[5-0] CLOCK_MSB	0x00
Set SIR mode	UART_MDR1[2-0] MODE_SELECT	0x1
Disable access to the UART_DLL and UART_DLH registers	UART_LCR[7-0]	0x00
Force output DTR to active	UART_MCR[0] DTR	1
Enable the UART_THR interrupt	UART_IER_IRDA[1] THR_IT	0x1
Set transmit frame length to 6 bytes	UART_TXFLL[7-0] TXFLL	0x06
Set the seven starts of frame transmission	UART_EBLR[7-0] EBLR	0x08
Set SIR pulse width to be 1.6 $\mu$ s	UART_ACREG[7] PULSE_TYPE	1

#### 12.1.6.5.7.2 MIR mode

##### 12.1.6.5.7.2.1 Receive

The following programming model explains how to program the module to receive an IrDA frame with no parity, baud rate = 1.152 Mbps, and FIFOs disabled.

**Table 12-159. MIR Mode Receive Settings**

Step	Register/Bit Field/Programming Model	Value
Disable UART mode	UART_MDR1[2-0] MODE_SELECT	0x7
Grant access to the UART_DLL and UART_DLH registers	UART_LCR[7-0]	0x80
Load the baud rate (1.152 bps)	UART_DLL[7-0] CLOCK_LSB	0x01
	UART_DLH[5-0] CLOCK_MSB	0x00
Set MIR mode	UART_MDR1[2-0] MODE_SELECT	0x4
Disable access to the UART_DLL and UART_DLH registers	UART_LCR[7-0]	0x00

**Table 12-159. MIR Mode Receive Settings (continued)**

Step	Register/Bit Field/Programming Model	Value
Force outputs DTR and RTS to active	UART_MCR[1-0]	0x3
Enable the UART_RHR interrupt	UART_IER_IRDA[0] RHR_IT	1

#### 12.1.6.5.7.2.2 Transmit

The following programming model explains how to program the module to transmit an IrDA 60-byte frame with no parity, baud rate = 1.152 Mbps, and FIFOs disabled.

**Table 12-160. MIR Mode Transmit Settings**

Step	Register/Bit Field/Programming Model	Value
Disable UART mode	UART_MDR1[2-0] MODE_SELECT	0x7
Grant access to the UART_DLL and UART_DLH registers	UART_LCR[7-0]	0x80
Load the baud rate (115.2 kbps)	UART_DLL[7-0] CLOCK_LSB	0x01
	UART_DLH[5-0] CLOCK_MSB	0x00
Set SIR mode	UART_MDR1[2-0] MODE_SELECT	0x4
Disable access to the UART_DLL and UART_DLH registers	UART_LCR[7-0]	0x00
Force output DTR to active	UART_MCR[0] DTR	1
Enable the UART_THR interrupt	UART_IER_IRDA[1] THR_IT	0x1
Set transmit frame length to 60 bytes	UART_TXFLL[7-0] TXFLL	0x3C
Set the eight additional starts of frame transmission	UART_EBLR[7-0] EBLR	0x08
SIP is sent at the end of transmission	UART_ACREG[3] SEND_SIP	1

#### 12.1.6.5.7.3 FIR mode

##### 12.1.6.5.7.3.1 Receive

The following programming model explains how to program the module to receive the IrDA frame with no parity, baud rate = 4 Mbps, FIFOs enabled, 8-bit word length.

**Table 12-161. FIR Mode Receive Settings**

Step	Register/Bit Field/Programming Model	Value
Disable UART mode	UART_MDR1[2-0] MODE_SELECT	0x7
Grant access to the UART_DLL and UART_DLH registers	UART_LCR[7-0]	0x80
Enable access to change UART_FCR[0]	UART_DLL[7-0] CLOCK_LSB	0x0
	UART_DLH[7-0] CLOCK_MSB	
FIFO clear and enable	UART_FCR[2-0]	0x7
Set the FIFO trigger level	see <a href="#">Section 12.1.6.5.4</a> , <i>Load FIFO trigger and DMA mode settings</i>	
Set FIR mode	UART_MDR1[2-0] MODE_SELECT	0x5
Set frame length	UART_RXFLL[7-0] RXFLL	0xA
Disable access to the UART_DLL and UART_DLH registers	UART_LCR[7-0]	0x00
Enable the UART_RHR interrupt	UART_IER_IRDA[0] RHR_IT	1

##### 12.1.6.5.7.3.2 Transmit

The following programming model explains how to program the module to transmit an IrDA 4-byte frame with no parity, baud rate = 4 Mbps, FIFOs enabled, and 8-bit word length.

**Table 12-162. FIR Mode Transmit Settings**

Step	Register/Bit Field/Programming Model	Value
Disable UART mode	UART_MDR1[2-0] MODE_SELECT	0x7

**Table 12-162. FIR Mode Transmit Settings (continued)**

Step	Register/Bit Field/Programming Model	Value
Grant access to the UART_DLL and UART_DLH registers	UART_LCR[7-0]	0x80
Enable access to change UART_FCR[0]	UART_DLL[7-0] CLOCK_LSB UART_DLH[5-0] CLOCK_MSB	0x0
FIFO clear and enable	UART_FCR[2-0]	0x7
Set the FIFO trigger level	see <a href="#">Section 12.1.6.5.4, Load FIFO trigger and DMA mode settings</a>	
Set FIR mode	UART_MDR1[2-0] MODE_SELECT	0x1
Disable access to the UART_DLL and UART_DLH registers	UART_LCR[7-0]	0x00
Set FIR mode and enable auto-SIP mode	UART_MDR1[7-0]	0x45
Set frame length	UART_TXFLL[7-0] TXFLL UART_TXFLH[7-0] TXFLH	0x4 0x0
Force output DTR to active	UART_MCR[0] DTR	1
Enable the UART_THR interrupt	UART_IER_IRDA[1] THR_IT	1
Set the eight additional starts of frame transmission	UART_EBLR[7-0] EBLR	0x08
SIP is sent at the end of transmission	UART_ACREG[3] SEND_SIP	1

## 12.2 High-speed Serial Interfaces

This section describes the high-speed serial interfaces in the device.



## 12.2.1 Gigabit Ethernet MAC (MCU\_CPSW0)

This chapter describes the Gigabit Ethernet MAC (Media Access Controller). For conceptual purposes the below documentation refers to this MAC as being a two port CPSW with port 0 being the CPPI DMA host port and port 1 being the Ethernet port.

### 12.2.1.1 MCU\_CPSW0 Overview

The two-port Gigabit Ethernet MAC (MCU\_CPSW0) subsystem provides Ethernet packet communication for the device and is configured in a similar manner as an Ethernet switch.

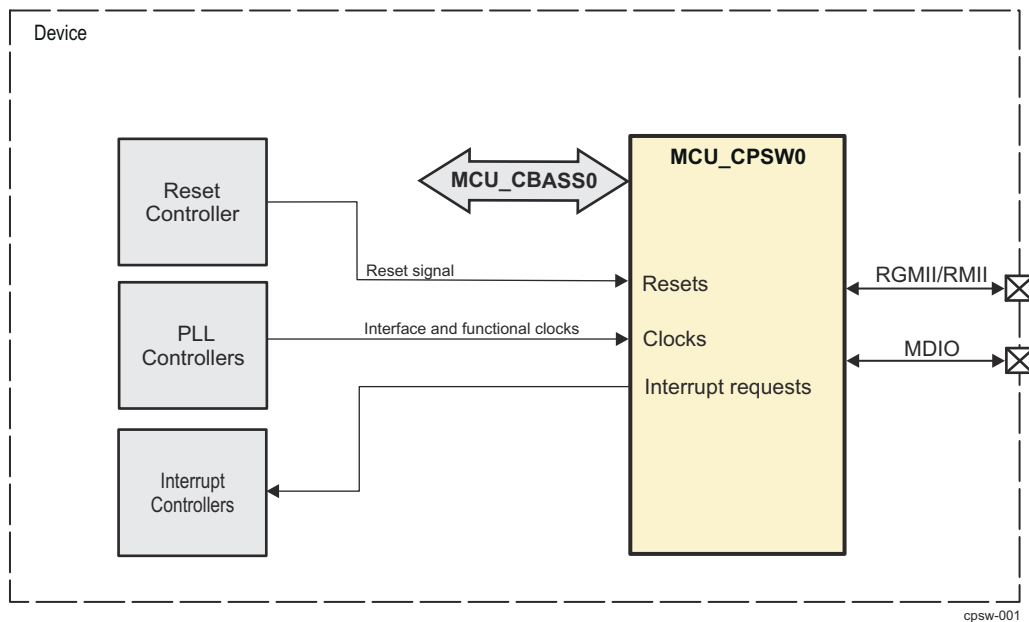
MCU\_CPSW0 features the Reduced Gigabit Media Independent Interface (RGMII), Reduced Media Independent Interface (RMII), and the Management Data Input/Output (MDIO) interface for physical layer device (PHY) management.

The device has integrated two-port Gigabit Ethernet Switch subsystem into device MCU domain named MCU\_CPSW0. [Table 12-163](#) shows the MCU\_CPSW0 module allocation within device domains.

**Table 12-163. MCU\_CPSW0 Module Allocation within Device Domains**

Module Instance	Domain		
	WKUP	MCU	MAIN
MCU_CPSW0	-	✓	-

[Figure 12-128](#) shows the MCU\_CPSW0 module overview.



**Figure 12-128. MCU\_CPSW0 Overview**

#### 12.2.1.1.1 MCU\_CPSW0 Features

The 2-port MCU\_CPSW0 subsystem provides the following features:

- One Ethernet port (port 1) with selectable RGMII and RMII interfaces and an internal Communications Port Programming Interface (CPPI) port (port 0)
- Synchronous 10/100/1000 Mbit operation
- Flexible logical FIFO-based packet buffer structure
- Eight priority level Quality Of Service (QOS) support (802.1p)
- Support for Audio/Video Bridging (P802.1Qav/D6.0)
- Support for IEEE 1588 Clock Synchronization (2008 Annex D, Annex E and Annex F)

- Timestamp module capable of time stamping external timesync events like Pulse-Per-Second and also generating Pulse-Per-Second outputs
- CPTS module that supports time stamping for IEEE1588 with support for 4 hardware push events and generation of compare output pulses
- DSCP Priority Mapping (IPv4 and IPv6)
- Energy Efficient Ethernet (EEE) support (802.3az)
- Flow Control Support (802.3x)
- Wire rate switching (802.1d)
- Non-Blocking switch fabric
- Time Sensitive Network Support
  - IEEE P802.3br/D2.0 Interspersing Express Traffic
  - IEEE 802.1Qbv/D2.2 Enhancements for Scheduled Traffic
- Address Lookup Engine (ALE)
  - 64 ALE table entries
  - Configurable number of addresses plus VLANs
  - Wire rate lookup
  - Host controlled time-based aging and/or auto-aging
  - Spanning tree support
  - L2 address lock and L2 filtering support
  - MAC authentication (802.1x)
  - Receive-based or destination-based Multicast and Broadcast rate limits
  - MAC address blocking
  - Source port locking
  - OUI (Vendor ID) host accept/deny feature
  - Configurable number of classifier/policers (8)
  - VLAN support
    - 802.1Q compliant
      - Auto add port VLAN for untagged frames on ingress
      - Auto VLAN removal on egress and auto pad to minimum frame size
- EtherStats and 802.3Stats Remote Network Monitoring (RMON) statistics gathering (per port statistics)
- Ethernet Mac transmit to Ethernet Mac receive Loopback mode (digital loopback) supported
- CPSGMII Loopback Modes (transmit to receive)
- Maximum frame size of 2024 bytes
- Management Data Input/Output (MDIO) module for PHY Management with Clause 45 support
- Programmable interrupt control with selected interrupt pacing
- Host port CPPI Streaming Packet Interface (CPPI\_GCLK)
- Digital loopback and FIFO loopback modes supported
- Emulation support
- Full duplex mode supported in 10/100/1000 Mbps. Half-duplex mode supported only in 10/100 Mbps modes only.
- RAM Error Detection and Correction (SECCDED)

#### 12.2.1.1.2 MCU\_CPSW0 Not Supported Features

The following features are not supported for 2-port MCU\_CPSW0 switch:

- Maximum frame size of 9600 bytes
- MII and GMII Mode
- SGMII Mode
- MACSEC
- Synchronous Ethernet
- InterVLAN routing
- RGMII Internal Delay Mode disabled

#### 12.2.1.1.3 Terminology

**Terminology:**

<b>AVB</b>	Audio Video Bridging
<b>AVBTP</b>	Audio Video Bridging Transport Protocol
<b>BMCA</b>	Best Master Clock Algorithm
<b>CFI</b>	Canonical Format Indicator
<b>CPPI</b>	Communications Port Programming Interface
<b>CPSW</b>	Common Platform Switch
<b>DLR</b>	Device Level Ring
<b>DSCP</b>	Differentiated Services Code Point
<b>EEE</b>	Energy Efficient Ethernet
<b>EMAC</b>	Ethernet Media Access Control
<b>EOP</b>	End of Packet
<b>EOQ</b>	End of Queue
<b>IPG</b>	Inter-Packet Gap
<b>LPI</b>	Low Power Indicator
<b>MDIO</b>	Management Data Input/Output
<b>MOF</b>	Middle of Frame
<b>OUI</b>	Organizationally Unique Identifier
<b>PFC</b>	Priority based Flow Control
<b>PTP</b>	Precision Time Protocol
<b>RMON</b>	Remote Monitoring
<b>RTCP</b>	RTP Control Protocol
<b>RTP</b>	Real-time Transport Protocol
<b>SCR</b>	Switched Central Resource
<b>SRP</b>	Stream Reservation Protocol
<b>TOS</b>	Type of Service
<b>VLAN</b>	Virtual Local Area Network

### 12.2.1.2 MCU\_CPSW0 Environment

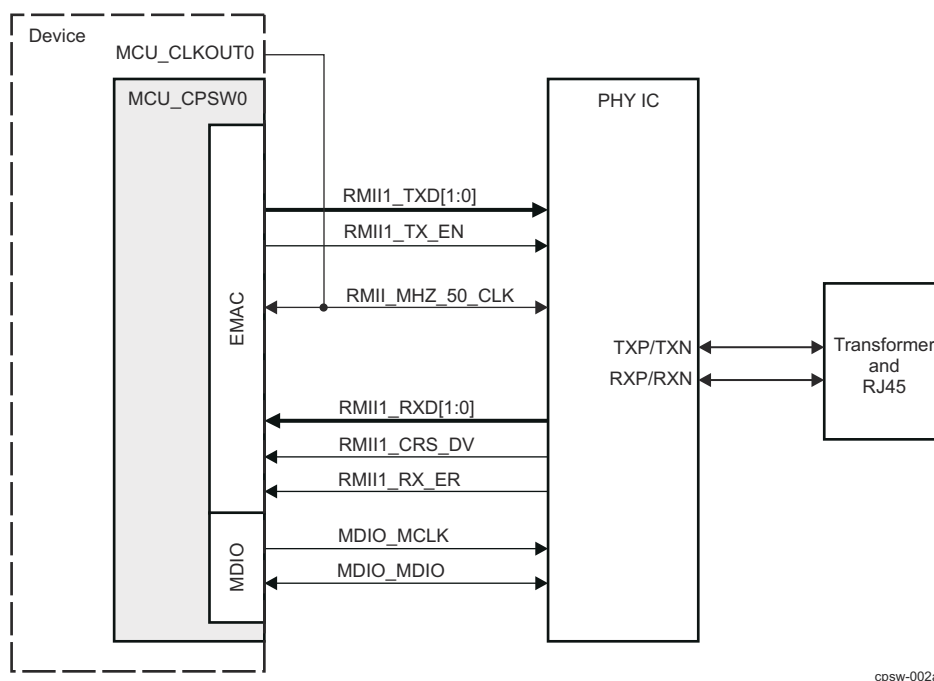
This section describes the MCU\_CPSW0 external connections (environment).

#### 12.2.1.2.1 MCU\_CPSW0 RMII Interface

Figure 12-129 shows a device with integrated EMAC and MDIO interfaced via a RMII connection in a typical system. The individual MCU\_CPSW0 and MDIO signals for the RMII interface are summarized in Table 12-164.

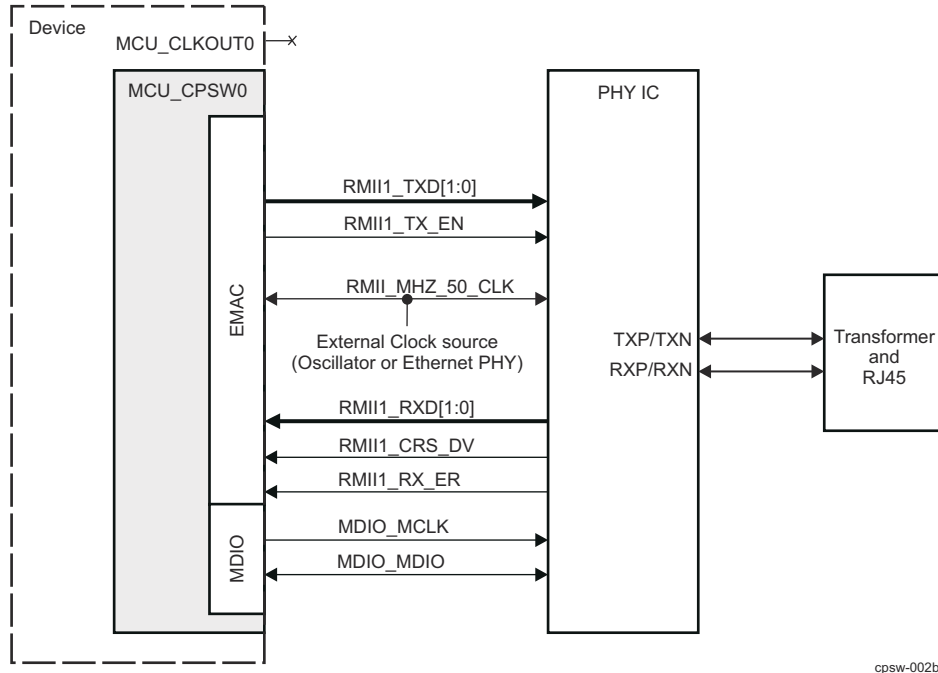
The MCU\_CPSW0 module integrated in the device supports internal and external clock sources in RMII mode. Figure 12-129 shows the internal clock source for RMII\_MHZ\_50\_CLK clock. It is 50 MHz clock source that is provided on the MCU\_CLKOUT0 device pin. This clock has to be routed on the PCB to the MCU\_RMII1\_REF\_CLK device pin and the external PHY, RMII clock input.

For more information, refer to either the IEEE 802.3 standard or ISO/IEC 8802-3:2000(E).



**Figure 12-129. RMII Interface Typical Application (Internal Clock Source)**

Figure 12-130 shows the external clock source for RMII\_MHZ\_50\_CLK clock. In this case a 50 MHz clock is available on the PCB and it can be sourced from an oscillator or from the Ethernet PHY. This externally generated clock should be routed to both MCU\_RMII1\_REF\_CLK device pin and the external PHY, RMII clock input.



cpsw-002b

**Figure 12-130. RMI1 Interface Typical Application (External Clock Source)**

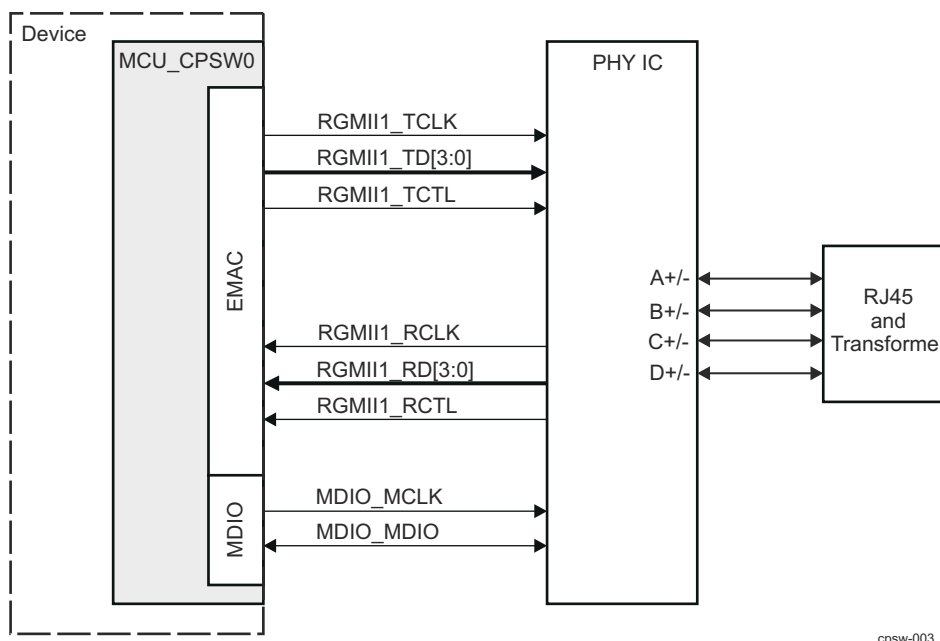
**Table 12-164. RMI1 I/O Description**

Signal	Device Pin(s)	I/O <sup>(1)</sup>	Description
RMI1_TXD[1:0]	MCU_RMI1_TXD[1:0]	O	Transmit data. The transmit data pins are a collection of 2 bits of data. TXD0 is the least-significant bit (LSB). The signals are synchronized by RMI1_MHZ_50_CLK and valid only when RMI1_TX_EN is asserted.
RMI1_TX_EN	MCU_RMI1_TX_EN	O	RMI1 transmit enable. The transmit enable signal indicates that the MCU_RMI1_TXD[1:0] pins are generating data for use by the PHY. RMI1_TX_EN is synchronous to RMI1_MHZ_50_CLK.
RMI1_MHZ_50_CLK	MCU_RMI1_REF_CLK	I	RMI1 50MHz reference clock.  The reference clock is used to synchronize all RMI1 signals. RMI1_MHZ_50_CLK must be continuous and fixed at 50 MHz.
RMI1_RXD[1:0]	MCU_RMI1_RXD[1:0]	I	Receive data. The receive data pins are a collection of 2 bits of data. RXD0 is the least-significant bit (LSB). The signals are synchronized by RMI1_MHZ_50_CLK and valid only when RMI1_CRD_DV is asserted and RMI1_RX_ER is de-asserted.
RMI1_CRD_DV	MCU_RMI1_CRD_DV	I	Carrier sense/receive data valid. Multiplexed signal between carrier sense and receive data valid.
RMI1_RX_ER	MCU_RMI1_RX_ER	I	Receive error. The receive error signal is asserted to indicate that an error was detected in the received frame.
MDIO_MCLK	MCU_MDIO0_MDC	O	Management data clock (MDIO_MCLK). The MDIO data clock is sourced by the MDIO module on the system. It is used to synchronize MDIO data access operations done on the MCU_MDIO0_MDIO pin.
MDIO_MDIO	MCU_MDIO0_MDIO	I/O	MDIO data pin drives PHY management data into and out of the PHY by way of an access frame consisting of start of frame, read/write indication, PHY address, register address, and data bit cycles. The MCU_MDIO0_MDIO pin acts as an output for all but the data bit cycles at which time it is an input for read operations.

(1) I = Input; O = Output

### 12.2.1.2.2 MCU\_CPSW0 RGMII Interface

Figure 12-131 shows a device with integrated EMAC and MDIO interfaced via a RGMII connection in a typical system. The individual MCU\_CPSW0 and MDIO signals for the RGMII interface are summarized in Table 12-165.



cpsw-003

**Figure 12-131. RGMII Interface Typical Application**

**Table 12-165. RGMII I/O Description**

Signal	Device Pin(s)	I/O <sup>(1)</sup>	Description
RGMII1_TD[3:0]	MCU_RGMII1_TD[3:0]	O	The transmit data pins are a collection of 4 bits of data. TD0 is the least-significant bit (LSB). The signals are valid only when RGMII1_TCTL is asserted.
RGMII1_TCTL	MCU_RGMII1_TX_CTL	O	Transmit Control/enable. The transmit enable signal indicates that the TD pins are generating data for use by the PHY.
RGMII1_TCLK	MCU_RGMII1_TXC	O	The transmit reference clock. The clock is 2.5 MHz at 10 Mbps operation, 25 MHz at 100 Mbps operation, and 125 MHz at 1000 Mbps of operation.
RGMII1_RD[3:0]	MCU_RGMII1_RD[3:0]	I	The receive data pins are a collection of 4 bits of data. RD0 is the least-significant bit (LSB). The signals are valid only when RGMII1_RCTL is asserted
RGMII1_RCTL	MCU_RGMII1_RX_CTL	I	The receive data valid/control signal indicates that the RD pins are nibble data for use by the EMAC.
RGMII1_RCLK	MCU_RGMII1_RXC	I	The receive clock is a continuous clock that provides the timing reference for receive operations. The clock is generated by the PHY and is 2.5 MHz at 10 Mbps operation, 25 MHz at 100 Mbps operation, 125 MHz at 1000 Mbps of operation.
MDIO_MCLK	MCU_MDIO0_MDC	O	Management data clock (MDIO_MCLK). The MDIO data clock is sourced by the MDIO module on the system. It is used to synchronize MDIO data access operations done on the MCU_MDIO0_MDIO pin.
MDIO_MDIO	MCU_MDIO0_MDIO	I/O	The MCU_MDIO0_MDIO pin drives PHY management data into and out of the PHY by way of an access frame consisting of start of frame, read/write indication, PHY address, register address, and data bit cycles. The MCU_MDIO0_MDIO pin acts as an output for all but the data bit cycles at which time it is an input for read operations.

(1) I = Input; O = Output

---

**Note**

The Control Module registers assign the specific function to the device pads. For more information on Control Module settings, see *Pad Configuration Registers* in *Control Module (CTRL\_MMR)* and the device-specific Datasheet.

---

Figure 12-132 shows the integration of the MCU CPSW0 module in the device.





The following MCU\_CPSW0 control registers are located in MCU\_CTRL\_MMR0 module: CTRLMMR\_MCU\_ENET\_CTRL, CTRLMMR\_MCU\_ENET\_CLKSEL, CTRLMMR\_MCU\_MAC\_ID0, CTRLMMR\_MCU\_MAC\_ID1.

Table 12-166 through Table 12-168 summarize the integration of the MCU\_CPSW0 module in the device.

**Table 12-166. MCU\_CPSW0 Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
MCU_CPSW0	WKUP_PSC0	PD0	LPSC0	MCU_CBASS0

**Table 12-167. MCU\_CPSW0 Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
MCU_CPSW0	CPPI_ICLK	MCU_SYSCCLK0/3	MCU_PLLCTRL	CPPI packet streaming interface clock (333-MHz). Main clock for MCU_CPSW0.
	GMII_RFT_CLK	MCU_PLL2_HSDIV0_CLK OUT/2	MCU_PLL2 (HSDIV0 of MCU_CPSW0 PLL)	125-MHz GMII Gigabit mode clock.
	RGMII_MHZ_5_CLK	MCU_PLL2_HSDIV0_CLK OUT/50	MCU_PLL2 (HSDIV0 of MCU_CPSW0 PLL)	5-MHz RGMII reference clock.
	RGMII_MHZ_50_CLK	MCU_PLL2_HSDIV0_CLK OUT/5	MCU_PLL2 (HSDIV0 of MCU_CPSW0 PLL)	50-MHz RGMII reference clock.
	RGMII_MHZ_250_CLK	MCU_PLL2_HSDIV0_CLK OUT	MCU_PLL2 (HSDIV0 of MCU_CPSW0 PLL)	250-MHz RGMII reference clock.
	RMII_MHZ_50_CLK	MCU_RMII1_REF_CLK	MCU_RMII1_REF_CLK pad	50-MHz RMII reference clock. This clock is derived from the MCU_RMII1_REF_CLK pad.
	CPTS_RFT_CLK	MAIN_PLL3_HSDIV1_CLK KOUT	HSDIV1 of CPSW0 PLL Controller, selected through CPTS Multiplexer (200 or 250-MHz clock)	CPTS IEEE 1588 clock. Selected through the CTRLMMR_MCU_ENET_CLKSEL register.
		MAIN_PLL0_HSDIV6_CLK KOUT	HSDIV6 of MAIN PLL0 Controller, selected through CPTS Multiplexer (200 or 250-MHz clock)	
		MCU_CPTS_RFT_CLK pad	MCU_CPTS_RFT_CLK pad, selected through CPTS Multiplexer (200-MHz clock)	
		CPTS_RFT_CLK pad	CPTS_RFT_CLK pad, selected through CPTS Multiplexer (200-MHz clock)	
		MCU_EXT_REFCLK0 pad	MCU_EXT_REFCLK0 pad, selected through CPTS Multiplexer (100-MHz clock)	
		EXT_REFCLK1 pad	EXT_REFCLK1 pad, selected through CPTS Multiplexer (100-MHz clock)	
		SERDES0_IP2_LN0_TXM_CLK	SERDES0 Lane0 (500-MHz clock)	
		SERDES0_IP2_LN1_TXM_CLK	SERDES0 Lane1 (500-MHz clock)	

**Table 12-167. MCU\_CPSW0 Clocks and Resets (continued)**

	SERDES1_IP2_LN0_TXM_CLK	SERDES1 Lane0 (500-MHz clock)
	SERDES1_IP2_LN1_TXM_CLK	SERDES1 Lane1 (500-MHz clock)
	SERDES2_IP2_LN0_TXM_CLK	SERDES2 Lane0 (500-MHz clock)
	SERDES2_IP2_LN1_TXM_CLK	SERDES2 Lane1 (500-MHz clock)
	SERDES3_IP2_LN0_TXM_CLK	SERDES3 Lane0 (500-MHz clock)
	SERDES3_IP2_LN1_TXM_CLK	SERDES3 Lane1 (500-MHz clock)
	MCU_PLL2_HSDIV1_CLK_OUT	HSDIV1 of MCU_CPSW0 PLL Controller, selected through CPTS Multiplexer (500-MHz clock)
	MCU_SYSCLOCK/2	MCU_PLLCTRL (500-MHz clock)
GMII1_MT_CLK	RGMII_MHZ_250_CLK/10	RGMII_MHZ_250_CLK
		<i>Note: GMII mode is not supported on this device. GMII1_MT_CLK transmit reference clock is needed to enable clock-stop protocol on this module.</i>
GMII1_MR_CLK	RGMII_MHZ_250_CLK/10	RGMII_MHZ_250_CLK
		<i>Note: GMII mode is not supported on this device. GMII1_MR_CLK receive reference clock is needed to enable clock-stop protocol on this module.</i>
RGMII1_RXC_I	RGMII1_RXC	MCU_RGMII1_RXC pad
		RGMII reference clock that provides the timing reference for receive operations.
RGMII1_TXC_O	RGMII1_TXC	MCU_RGMII1_TXC pad
		RGMII transmit reference clock.
MDIO_MCLK	MDIO_MCLK	MCU_MDIO0_MDC pad
		Management data clock (MDIO_MCLK). The MDIO data clock is sourced by the MDIO module on the system. It is used to synchronize MDIO data access operations done on the MDIO pin.

**Resets**

Module Instance	Module Reset Input	Source Reset Signal	Source	Description
MCU_CPSW0	MCU_CPSW0_RST	MOD_G_RST	LPSC0	Module Reset

**Table 12-168. MCU\_CPSW0 Hardware Requests**
**Interrupt Requests**

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCU_CPSW0	MCU_CPSW0_STAT_PE_ND_0	MCU_R5FSS0_CORE0_INT_R_IN_32	MCU_R5FSS0_CO_RE0	MCU_CPSW0 statistic pending interrupt 0	Level
		MCU_R5FSS0_CORE1_INT_R_IN_32	MCU_R5FSS0_CO_RE1	MCU_CPSW0 statistic pending interrupt 0	Level
		GIC500_SPI_IN_888	COMPUTE_CLUSTER0	MCU_CPSW0 statistic pending interrupt 0	Level

**Table 12-168. MCU\_CPSW0 Hardware Requests (continued)**

	C66SS0_INTRTR0_IN_279	C66SS0_INTRTR0	MCU_CPSW0 statistic pending interrupt 0	Level
	C66SS1_INTRTR0_IN_279	C66SS1_INTRTR0	MCU_CPSW0 statistic pending interrupt 0	Level
	R5FSS0_INTRTR0_IN_117	R5FSS0_INTRTR0	MCU_CPSW0 statistic pending interrupt 0	Level
	R5FSS1_INTRTR0_IN_117	R5FSS1_INTRTR0	MCU_CPSW0 statistic pending interrupt 0	Level
MCU_CPSW0_MDIO_PE ND_0	MCU_R5FSS0_CORE0_INT R_IN_35	MCU_R5FSS0_CO RE0	MCU_CPSW0 MDIO interrupt	Level
	MCU_R5FSS0_CORE1_INT R_IN_35	MCU_R5FSS0_CO RE1	MCU_CPSW0 MDIO interrupt	Level
	GIC500_SPI_IN_889	COMPUTE_CLUST ER0	MCU_CPSW0 MDIO interrupt	Level
	C66SS0_INTRTR0_IN_280	C66SS0_INTRTR0	MCU_CPSW0 MDIO interrupt	Level
	C66SS1_INTRTR0_IN_280	C66SS1_INTRTR0	MCU_CPSW0 MDIO interrupt	Level
	R5FSS0_INTRTR0_IN_118	R5FSS0_INTRTR0	MCU_CPSW0 MDIO interrupt	Level
	R5FSS1_INTRTR0_IN_118	R5FSS1_INTRTR0	MCU_CPSW0 MDIO interrupt	Level
MCU_CPSW0_EVNT_PE ND_0	MCU_R5FSS0_CORE0_INT R_IN_34	MCU_R5FSS0_CO RE0	MCU_CPSW0 event pending interrupt	Level
	MCU_R5FSS0_CORE1_INT R_IN_34	MCU_R5FSS0_CO RE1	MCU_CPSW0 event pending interrupt	Level
	GIC500_SPI_IN_890	COMPUTE_CLUST ER0	MCU_CPSW0 event pending interrupt	Level
	C66SS0_INTRTR0_IN_281	C66SS0_INTRTR0	MCU_CPSW0 event pending interrupt	Level
	C66SS1_INTRTR0_IN_281	C66SS1_INTRTR0	MCU_CPSW0 event pending interrupt	Level
	R5FSS0_INTRTR0_IN_119	R5FSS0_INTRTR0	MCU_CPSW0 event pending interrupt	Level
	R5FSS1_INTRTR0_IN_119	R5FSS1_INTRTR0	MCU_CPSW0 event pending interrupt	Level
MCU_CPSW0_ECC_SEC _PEND_0	MCU_ESM0_LVL_IN_14	MCU_ESM0	MCU_CPSW0 SEC ECC error interrupt	Level
MCU_CPSW0_ECC_DED _PEND_0	MCU_ESM0_LVL_IN_15	MCU_ESM0	MCU_CPSW0 DED ECC error interrupt	Level

Time Sync and Compare Events					
Module Instance	Module Event	Destination Event Input	Destination	Description	Type

**Table 12-168. MCU\_CPSW0 Hardware Requests (continued)**

MCU_CPSW0	MCU_CPSW0_CPTS_CO MP_0	CMPEVENT_INTRTR0_IN_1 0	CMPEVT_INTRTR0	MCU_CPSW0 compare event interrupt	Edge
	MCU_CPSW0_CPTS_GE NF0_0	TIMESYNC_INTRTR0_IN_16	TIMESYNC_INTRTR0 R0	MCU_CPSW0 CPTS generator function event interrupt 0	Edge
	MCU_CPSW0_CPTS_GE NF1_0	TIMESYNC_INTRTR0_IN_17	TIMESYNC_INTRTR0 R0	MCU_CPSW0 CPTS generator function event interrupt 1	Edge
	MCU_CPSW0_CPTS_SY NC_0	TIMESYNC_INTRTR0_IN_38	TIMESYNC_INTRTR0 R0	MCU_CPSW0 CPTS sync event interrupt	Edge

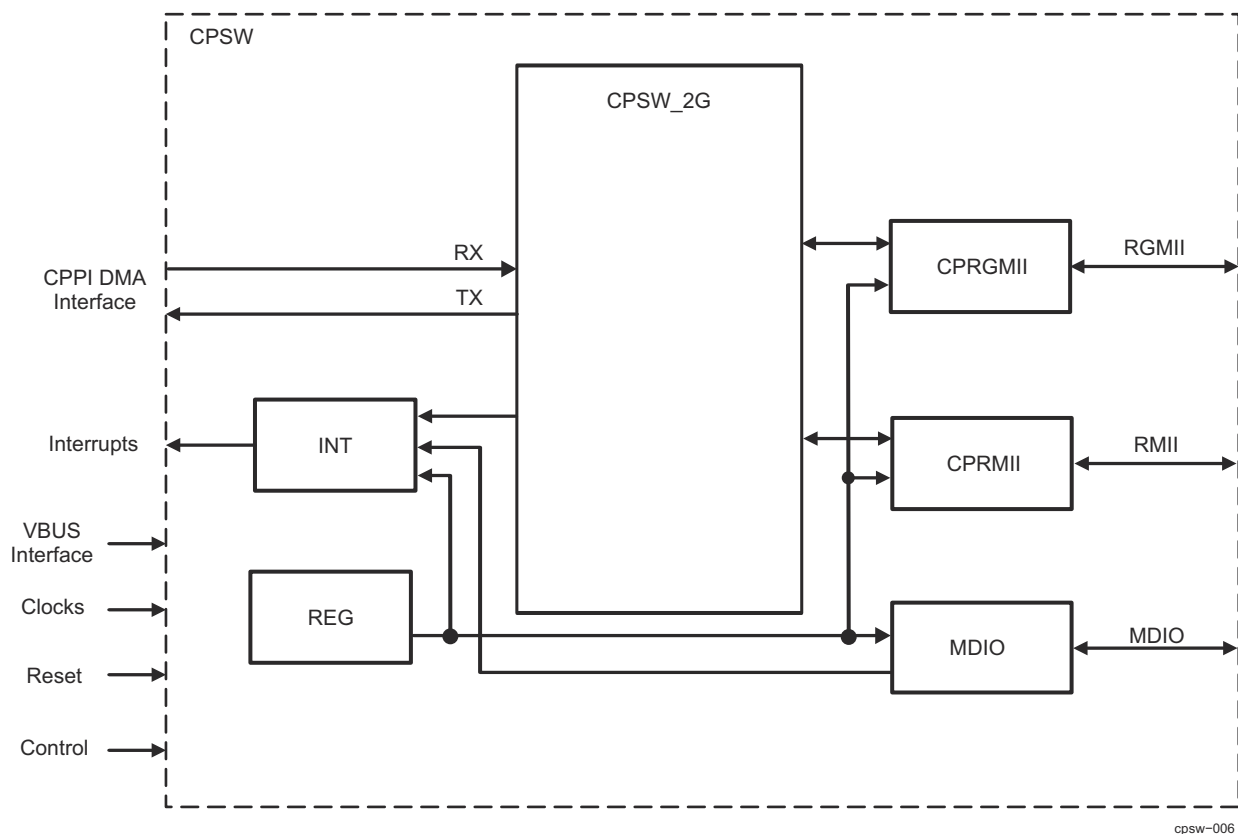
#### 12.2.1.4 MCU\_CPSW0 Functional Description

The two-port switch Ethernet subsystem module (CPSW) is compliant to the IEEE Std 802.3 Specification. CPSW top level functional block diagram is shown in [Figure 12-133](#).

##### 12.2.1.4.1 Functional Block Diagram

The two-port Ethernet subsystem consists of:

- CPSW\_2G
- One RGMII interface module
- One RMII interface module
- One Host Port 0 CPPI Packet Streaming Interface
- CPSW subsystem control registers (REG)
- One MDIO interface module
- One Interrupt Controller module



**Figure 12-133. CPSW Top Level Block Diagram**

##### 12.2.1.4.2 CPSW Ports

The Ethernet Subsystem has two ports. Port 0 is the Host port (internal to the Subsystem). Port 1 is the external port connected to RGMII, or RMII interfaces as per the interface selected.

Naming conventions followed in this chapter:

- Port0 is referred to the CPPI Host Port
- Port1 is referred to the interfaces RGMII/RMII

##### 12.2.1.4.2.1 Interface Mode Selection

The two-port switch (CPSW) Ethernet Subsystem has one 10/100/1000 Ethernet port with selectable RMII, and RGMII interfaces.

The interface mode is selected by configuring the Ethernet interface mode selection bitfield (MODE\_SEL) in the CTRLMMR\_MCU\_ENET\_CTRL register.

See the device-specific Datasheet for configuring the pin mux mode as per the interface selected.

### **12.2.1.4.3 Clocking**

#### **12.2.1.4.3.1 Subsystem Clocking**

CPSW clocking summary is shown in *CPSW Integration*.

#### **12.2.1.4.3.2 Interface Clocking**

Data is transmitted and received with respect to the reference clocks of the interface pins.

##### **12.2.1.4.3.2.1 RGMII Interface Clocking**

RGMII\_RXC, RGMII\_TXC frequencies are:

- 2.5 MHz at 10 Mbps
- 25 MHz at 100 Mbps
- 125 MHz at 1000 Mbps

##### **12.2.1.4.3.2.2 RMII Interface Clocking**

RMII interface clock RMII\_50MHZ\_CLK frequency is:

- 50 MHz at 10 Mbps
- 50 MHz at 100 Mbps

MCU\_RMII1\_REF\_CLK device pin or internal RGMII\_MHZ\_50\_CLK clock (default clock) can be selected through CTRLMMR\_MCU\_ENET\_CLKSEL[0] RMII\_CLK\_SEL and one of these clocks can be used as the clock source for RMII interface. For more details on RMII clocking, please see *MCU\_CPSW0 Integration*

CTRLMMR\_MCU\_CLKOUT0\_CTRL[4] CLK\_EN and CTRLMMR\_MCU\_CLKOUT0\_CTRL[0] CLK\_SEL bits are used to enable and select the clock source for MCU\_CLKOUT0 device pin.

##### **12.2.1.4.3.2.3 MDIO Clocking**

The MDIO clock is based on a divide-down of the interface (CPPI\_ICLK) clock. The application software or driver must control the divide-down value.

See the CPSW\_MDIO\_CONTROL\_REG register for configuring the Clock Divider ([15-0]CLKDIV) value.

#### **12.2.1.4.4 Software IDLE**

The submodule software idle register bits enable CPSW operation to be completely or partially suspended by software control. There are two CPSW submodules that contain software idle register bits. Each of the two submodules may be individually commanded to enter the idle state. The idle state is entered at packet boundaries, and no further packet operations will occur on an idled submodule until the idle command is removed. The CPSW module enters the idle state when all two submodules are commanded to enter and have entered the idle state. Idle status is determined by reading or polling the two submodule idle bits. The CPSW\_2G is in the idle state when all two submodules are in the idle state. The CPSW\_SOFT\_IDLE\_REG[0] SOFT\_IDLE bit may be set if desired after the submodules are in the idle state. The SOFT\_IDLE bit causes packets to not be transferred from one FIFO to another FIFO internal to the switch.

#### **12.2.1.4.5 Interrupt Functionality**

CPSW Ethernet Subsystem has six interrupt outputs:

- EVNT\_PEND - CPTS Event Pending Interrupt
- STAT\_PEND0 – Statistics Pending Interrupts
- ECC\_DED\_INT - ECC DED Level Interrupt
- ECC\_SEC\_INT - ECC SEC Level Interrupt
- MDIO\_INTR - MDIO Pending Interrupt (combined MDIO\_LINKINT and MDIO\_USERINT events)

#### 12.2.1.4.5.1 EVNT\_PEND Interrupt

See *Common Platform Time Sync (CPTS)* for more details on this interrupt.

#### 12.2.1.4.5.2 Statistics Interrupt (STAT\_PEND0)

The statistics level interrupt (STAT\_PEND0) will be asserted, if enabled when any statistics value is greater than or equal to 8000 0000h. The statistics interrupt is cleared by writing to decrement any statistics values greater than 8000 0000h (such that their new values are less than 8000 0000h). The statistics are mapped into internal memory space and are 32-bits wide. The raw and masked statistics interrupt status may be read by reading the CPSW\_CPTS\_INTSTAT\_RAW\_REG and CPSW\_CPTS\_INTSTAT\_MASKED\_REG registers, respectively.

The statistics interrupt is enabled by setting to 1h the TS\_PEND\_EN bit in the CPSW\_CPTS\_INT\_ENABLE\_REG register.

#### 12.2.1.4.5.3 ECC DED Level Interrupt (ECC\_DED\_INT)

Level interrupt (ECC\_DED\_INT) indicating an ECC double error has been detected.

#### 12.2.1.4.5.4 ECC SEC Level Interrupt (ECC\_SEC\_INT)

Level interrupt (ECC\_SEC\_INT) indicating an ECC single error has been detected and corrected.

#### 12.2.1.4.5.5 MDIO Interrupts

**Normal Mode:** (CPSW\_MDIO\_POLL\_REG[30] STATECHANGEMODE = 0h)

MDIO\_LINKINT event is set if there is a change in the link state of the PHY corresponding to the address in the PHYADR\_MON field of the CPSW\_MDIO\_USER\_PHY\_SEL\_REG\_k register and the corresponding LINKINT\_ENABLE bit is set. The MDIO\_LINKINT event is also captured in the MDIO CPSW\_MDIO\_LINK\_INT\_MASKED\_REG register. When the GO bit in the CPSW\_MDIO\_USER\_ACCESS\_REG\_k register transitions from 1 to 0, indicating the completion of a user access, and the corresponding USERINTMASKSET bit in the CPSW\_MDIO\_USER\_INT\_MASK\_SET\_REG register is set, the MDIO\_USERINT signal is asserted. The MDIO\_USERINT event is also captured in the CPSW\_MDIO\_USER\_INT\_MASKED\_REG register.

**State Change Mode:** (CPSW\_MDIO\_POLL\_REG[30] STATECHANGEMODE = 1h)

In State Change Mode, the MDIO will assert MDIO\_LINKINT[0] when any bit in the MDIO CPSW\_MDIO\_ALIVE\_REG or CPSW\_MDIO\_LINK\_REG registers changes due to MDIO operations. The MDIO\_LINKINT event is also captured in the CPSW\_MDIO\_LINK\_INT\_MASKED\_REG register. MDIO\_LINKINT[1] output is unused in State Change Mode. The CPSW\_MDIO\_USER\_PHY\_SEL\_REG\_k registers are unused in state change mode.

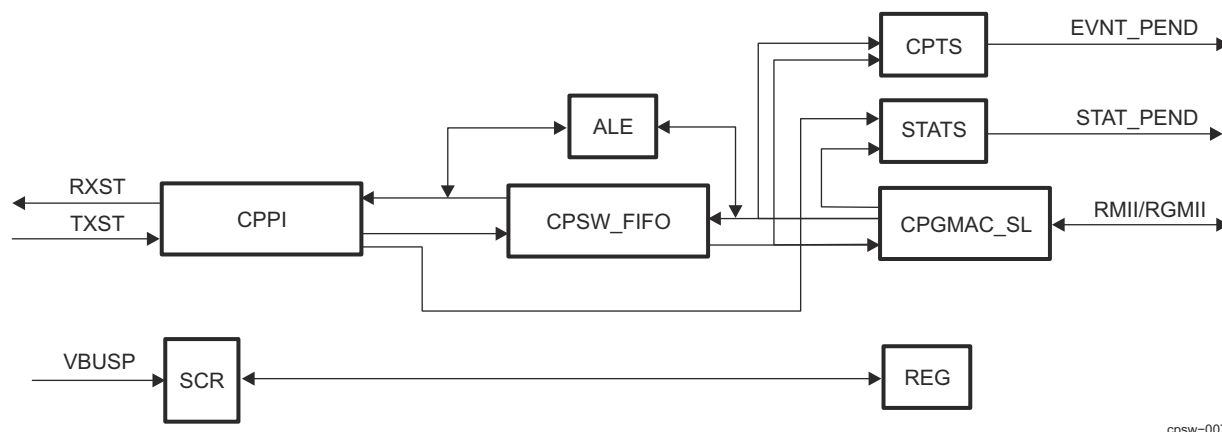
**Manual Mode:** (CPSW\_MDIO\_POLL\_REG[31] MANUALMODE = 1h)

Manual Mode allows software to directly control the MDIO serial clock output (CPSW\_MDIO\_MANUAL\_IF\_REG[2] MDIO\_MDCLK\_O) and the MDIO serial data output enable (CPSW\_MDIO\_MANUAL\_IF\_REG[1] MDIO\_OE). Manual Interface Mode is enabled when the MANUALMODE bit is set in the CPSW\_MDIO\_POLL\_REG register. Manual Mode is intended to be used by software for slow speed general purpose IO operations and not for MDIO PHY operations.

#### 12.2.1.4.6 CPSW\_2G

The CPSW\_2G RMII/ RGMII interface is compliant to the IEEE Std 802.3 Specification.

The CPSW\_2G contains one Ethernet port interface (Ethernet port 1), one CPPI packet streaming interface host port (port 0), Common Platform Time Sync (CPTS), ALE Engine and Statistics (STATS). A top-level block diagram of the CPSW\_2G is shown in [Figure 12-134](#).



**Figure 12-134. CPSW\_2G Block Diagram**

#### 12.2.1.4.6.1 Address Lookup Engine (ALE)

The Address Lookup Engine (ALE) is a sub-block of the CPSW Switch and it processes all received packets and determines to which port(s) the packet should be forwarded. The ALE uses the incoming packet received port number, destination address, source address, length/type, and VLAN information to determine how the packet should be forwarded. The ALE outputs the port mask to the switch fabric that indicates the port(s) the packet should be forwarded to. The ALE is enabled when the ENABLE\_ALE bit in the CPSW\_ALE\_CONTROL register is set. All packets are dropped when the ENABLE bit is cleared to 0.

##### 12.2.1.4.6.1.1 Error Handling

In normal operation, the Ethernet port modules are configured to issue an abort, instead of an end of packet, at the end of a packet that contains an error (runt, frag, oversize, jabber, crc, alignment, code etc.) or at the end of a MAC control packet. However, when the CPSW\_PN\_MAC\_CONTROL\_REG configuration bit(s) RX\_CEF\_EN, RX\_CSF\_EN, or RX\_CMF\_EN are set, error frames, short frames or MAC control frames have a normal end of packet instead of an abort at the end of the packet. When the ALE receives a packet that contains errors (due to a set header error bit), or a MAC control frame and does not receive an abort, the packet will be forwarded only to the host port (port 0). Packets with errors that are forwarded to the host have no VLAN untagging or drop due to rate limiting. No ALE learning occurs on packets with errors or mac control frames. Learning is based on source address and lookup is based on destination address. Directed packets from the host are not learned, updated, or touched.

##### 12.2.1.4.6.1.2 Bypass Operations

The ALE may be configured to operate in bypass mode by setting the ENABLE\_BYPASS bit in the CPSW\_ALE\_CONTROL register. When in bypass mode, all Ethernet port received packets are forwarded only to the host port (port 0). In bypass mode, the ALE processes host port transmit packets the same as in normal mode. In general, packets would be directed by the host in bypass mode.

##### 12.2.1.4.6.1.3 OUI Deny or Accept

The ALE may be configured to operate in OUI deny mode by setting the ENABLE\_OUI\_DENY bit in the CPSW\_ALE\_CONTROL register. When in OUI deny mode, a packet with a non-matching OUI source address will be dropped unless the destination address matches a multicast table entry with the SUPER bit set. When ENABLE\_OUI\_DENY bit is cleared, any packet source address matching an OUI address table entry will be dropped to the host unless the destination address matches with a supervisory address table entry. Broadcast packets will be dropped unless the broadcast address is entered into the table with the SUPER bit set. Unicast packets will be dropped unless the unicast address is in the table with BLOCK and SECURE both set (supervisory unicast packet).



#### 12.2.1.4.6.1.4 Statistics Counting

ALE sends many statistics along with the frame routing so the CPSW can count them on a per port basis. There are many reasons to drop frames the below drop events are individually counted in CPSW per port statistics counters. For more information on ALE statistics please refer to the *CPSW Network Statistics* section.

#### 12.2.1.4.6.1.5 Automotive Security Features

The ALE has many automotive security features that most enterprise switches do not require.

- VLANs can be configured to not allow fragmented IPv4 frames. That is a VLAN can be configured to not allow fragmented IPv4 traffic.
- VLANs can be configured to only allow up to four different IPv4 Protocols or IPv6. Next Header values, for example a VLAN can be configured to only allow TCP traffic in both IPv4 and IPv6 packets.
- Drop invalid Source Addresses, that is drop Source Addresses with bit 40 set (Multicast/Broadcast indicator on Destination Addresses)
- IEEE802.3 Length Check, drop frames that the IEEE802.3 Length is not contained within the frame. (Ether Types 0-1500)
- Any Source Address can be secured to a port dropping any attempts from other ports to masquerade as a service.
- Any source or destination address can be blocked.
- Per Port or Per VLAN ingress checking, dropping traffic from non-member ports.
- Classification, Policing on L2 and L3 information.

#### 12.2.1.4.6.1.6 CPSW Switching Solutions

The host port can operate in many different modes as well depending on the functionality of the host. It is important to understand the modes and configure them properly.

The ALE Table is designed to maximize the modes without compromise of the system functionality as well.

##### 12.2.1.4.6.1.6.1 Basics of 2-port Switch Type

The 2-port switch has a host port, and that port can operate in two fundamental modes. Bridge mode allows the host to extend the switched domain to another network like Wi-Fi or another multi-port switch. In this case the host must be able to see unknown unicast addresses so they can be broadcast to the other network. In Port mode the host need not see any unknown unicast traffic. The CPSW\_ALE\_CONTROL[8] EN\_HOST\_UNI\_FLOOD bit determines the host mode for unknown unicast traffic. This bit should only be set if you are bridging two or more networks together.

Essentially the two port switch is a MAC with all the switching features like VLAN insertion, removal, update, etc.

The 2-port switch would normally have a very small ALE table. Its primary use is for the Host filtering features. For example you would not ever need to auto learn or add addresses of external nodes to the ALE table, the table's primary purpose is to host unicast addresses, multicast addresses, VLANs, IP addresses etc.

#### 12.2.1.4.6.1.7 VLAN Routing and OAM Operations

##### 12.2.1.4.6.1.7.1 InterVLAN Routing

The CPSW module supports wire rate InterVLAN routing for a small number of routes, that is the host will setup an ALE classifier with an associated egress operation that will cause the CPSW to perform particular egress operations. The ALE can optionally check time to live validity as well.

The ALE uses the classifier along with an egress opcode, destination port mask and TTL check field to tell the CPSW to manipulate the packet on the egress. The CPSW will use the opcode along with a per port operation table to process the packet. By setting up the CPSW egress operation table you can replace the DA, SA and VLAN along with optionally updating the time to live IP header field. This allows the CPSW to perform the routing function for a small set of routes without getting the local host/CPU involved.

The Egress opcode will only be used for a classifier match and the packet would normally be sent only to the host. That is the host would have routed the packet but the CPSW has been configured to do the work instead. In the event that the time-to-live check feature is enabled and the time-to-live is either 0 or 1, the packet will not get

the egress opcode and instead be sent to the host as if the route is not setup. This allows the host to deal with invalid TTL fields.

#### 12.2.1.4.6.1.7.2 OAM Operations

The ALE supports OAM loopback on ports so that a remote link can be tested. That is a port placed in OAM loopback will echo packets received on a port back to the port with an egress op of 0xFF which will swap the SA and DA on egress in the CPSW.

Any supervisory packet will not be affected, so the spanning tree and other bridging functions are not affected.

Packets will only be echoed if the port is in OAM loopback mode, the received packet is not a supervisor packet. The port is in a forwarding state and the packet received DA!=SA and there are no errors in the packet.

When a port is in OAM loopback the port will not egress traffic from other ports, no address for loop backed traffic will be learned if enabled. Any packet received on the OAM loopback port with an error will be process as if the port is not in OAM. That is if the host has enabled copy errored frames the errored frames will be sent to the host instead.

#### 12.2.1.4.6.1.8 Supervisory packets

Multicast supervisory packets are designated by the SUPER bit in the table entry. Unicast supervisory packets are indicated when BLOCK and SECURE are both set. Supervisory packets are not dropped due to rate limiting, OUI, or VLAN processing. The purpose of supervisory packets is to allow packets that would be otherwise blocked to be forwarded for special purposes.

#### 12.2.1.4.6.1.9 Address Table Entry

The ALE table contains multiple table entry types. Each table entry represents a free entry, an address, a VLAN, an address/VLAN pair, or an OUI address. Software should ensure that there are not double address entries in the table. The double entry used would be indeterminate. Reserved table bits must be written with zeroes.

Source Address learning occurs for packets with a unicast, multicast or broadcast destination address and a unicast or multicast (including broadcast) source address. Multicast source addresses have the group bit (bit 40) cleared before ALE processing begins, changing the multicast source address to a unicast source address. A multicast address of all ones is the broadcast address which may be added to the table. A learned unicast source address is added to the table with the following control bits:

**Table 12-169. Learned Address Control Bits**

Bit(s)	Value
Ageable	1
Touch	1
BLOCK	0
SECURE	0

If a received packet has a source address that is equal to the destination address then the following occurs:

- The address is learned if the address is not found in the table.
- The address is updated if the address is found.
- The packet is dropped.

#### Table Entry Type

00 - Free Entry

01 - Address Entry : unicast or multicast determined by destination **address bit 40**.

10 - VLAN entry

11 - VLAN Address Entry : unicast or multicast determined by **address bit 40**.

#### 12.2.1.4.6.1.9.1 Free Table Entry

**Table 12-170. Free (Unused) Address Table Entry Bit Values**

70:62	61:60	59:0
Reserved	ENTRY_TYPE (00)	Reserved

#### 12.2.1.4.6.1.9.2 Multicast Address Table Entry

**Table 12-171. Multicast Address Table Entry Bit Values**

70:68	67:66	65	64	63:62	61:60	59:48	47:0
Reserved	PORT_MASK	SUPER	Reserved	MCAST_FWD_STATE	ENTRY_TYPE (01)	Reserved	MULTICAST_ADDRESS

#### Supervisory Packet (SUPER)

When set, this field indicates that the packet with a matching multicast destination address is a supervisory packet.

0: Non-supervisory packet

1: Supervisory packet

#### Port Mask(1:0) (PORT\_MASK)

This 2-bit field is the port bit mask that is returned with a found multicast destination address. There may be multiple bits set indicating that the multicast packet may be forwarded to multiple ports (but not the receiving port).

#### Multicast Forward State (MCAST\_FWD\_STATE)

Indicates the port state(s) required for the received port on a destination address lookup in order for the multicast packet to be forwarded to the transmit port(s). A transmit port must be in the Forwarding state in order to forward the packet. If the transmit PORT\_MASK has multiple set bits then each forward decision is independent of the other transmit port(s) forward decision.

00 - Forwarding

01 - Blocking/Forwarding/Learning

10 - Forwarding/Learning

11 - Forwarding

The forward state test returns a true value if both the RX and TX ports are in the required state.

#### Table Entry Type (ENTRY\_TYPE)

Address entry type. Unicast or multicast determined by address bit 40.

01: Address entry. Unicast or multicast determined by address bit 40.

#### Packet Address (MULTICAST\_ADDRESS)

This is the 48-bit packet MAC address. For an OUI address, only the upper 24-bits of the address are used in the source or destination address lookup. Otherwise, all 48-bits are used in the lookup.

#### 12.2.1.4.6.1.9.3 VLAN/Multicast Address Table Entry

**Table 12-172. VLAN/Multicast Address Table Entry Bit Values**

70:68	67:66	65	64	63:62	61:60	59:48	47:0
Reserved	PORT_MASK	SUPER	Reserved	MCAST_FWD_STATE	ENTRY_TYPE (11)	VLAN_ID	MULTICAST_ADDRESS

### Supervisory Packet (SUPER)

When set, this field indicates that the packet with a matching multicast destination address is a supervisory packet.

0: Non-supervisory packet

1: Supervisory packet

### Port Mask(1:0) (PORT\_MASK)

This 2-bit field is the port bit mask that is returned with a found multicast destination address. There may be multiple bits set indicating that the multicast packet may be forwarded to multiple ports (but not the receiving port).

### Multicast Forward State (MCAST\_FWD\_STATE)

Indicates the port state(s) required for the received port on a destination address lookup in order for the multicast packet to be forwarded to the transmit port(s). A transmit port must be in the Forwarding state in order to forward the packet. If the transmit PORT\_MASK has multiple set bits then each forward decision is independent of the other transmit port(s) forward decision.

00 - Forwarding

01 - Blocking/Forwarding/Learning

10 - Forwarding/Learning

11 - Forwarding

The forward state test returns a true value if both the RX and TX ports are in the required state.

### Table Entry Type (ENTRY\_TYPE)

Address entry type. Unicast or multicast determined by address bit 40.

11: VLAN address entry. Unicast or multicast determined by address bit 40.

### VLAN ID (VLAN\_ID)

The unique identifier for VLAN identification. This is the 12-bit VLAN ID.

### Packet Address (MULTICAST\_ADDRESS)

This is the 48-bit packet MAC address. For an OUI address, only the upper 24-bits of the address are used in the source or destination address lookup. Otherwise, all 48-bits are used in the lookup.

#### 12.2.1.4.6.1.9.4 Unicast Address Table Entry

**Table 12-173. Unicast Address Table Entry Bit Values**

70:68	67:66	65	64	63:62	61:60	59:48	47:0
Reserved	PORT_NUMBER	BLOCK	SECURE	UNICAST_TYPE (00) or (X1)	ENTRY_TYPE (01)	Reserved	UNICAST_ADDR ESS

### Port Number (PORT\_NUMBER)

This field indicates the port number (not port mask) that the packet with a unicast destination address may be forwarded to. Packets with unicast destination addresses are forwarded only to a single port (but not the receiving port).

### Block (BLOCK)

The block bit indicates that a packet with a matching source or destination address should be dropped (block the address).

0 - Address is not blocked.

1 - Drop a packet with a matching source or destination address (secure must be zero)

If block and secure are both set, then they no longer mean block and secure. When both are set, the block and secure bits indicate that the packet is a unicast supervisory (super) packet and they determine the unicast forward state test criteria. If both bits are set then the packet is forwarded if the receive port is in the Forwarding/Blocking/Learning state. If both bits are not set then the packet is forwarded if the receive port is in the Forwarding state.

### Secure (SECURE)

This bit indicates that a packet with a matching source address should be dropped if the received port number is not equal to the table entry port\_number.

0 - Received port number is a don't care.

1 - Drop the packet if the received port is not the secure port for the source address and do not update the address (block must be zero)

### Unicast Type (UNICAST\_TYPE)

This field indicates the type of unicast address the table entry contains.

00 - Unicast address that is not ageable.

01 - Ageable unicast address that has not been touched.

10 - OUI address - lower 24-bits are don't cares (not ageable).

11 - Ageable unicast address that has been touched.

### Table Entry Type (ENTRY\_TYPE)

Address entry. Unicast or multicast determined by address bit 40.

01: Address entry. Unicast or multicast determined by address bit 40.

### Packet Address (UNICAST\_ADDRESS)

This is the 48-bit packet MAC address. All 48-bits are used in the lookup.

#### 12.2.1.4.6.1.9.5 OUI Unicast Address Table Entry

**Table 12-174. OUI Unicast Address Table Entry Bit Values**

70:64	63:62	61:60	59:48	47:24	23:0
Reserved	UNICAST_TYPE (10)	ENTRY_TYPE (01)	Reserved	UNICAST_OUI	Reserved

### Unicast Type (UNICAST\_TYPE)

This field indicates the type of unicast address the table entry contains.

00 - Unicast address that is not ageable.

01 - Ageable unicast address that has not been touched.

10 - OUI address - lower 24-bits are don't cares (not ageable).

11 - Ageable unicast address that has been touched.

### Table Entry Type (ENTRY\_TYPE)

Address entry. Unicast or multicast determined by address bit 40.

01: Address entry. Unicast or multicast determined by address bit 40.

### Packet Address (UNICAST\_OUI)

For an OUI address, only the upper 24-bits of the address are used in the source or destination address lookup.

### 12.2.1.4.6.1.9.6 VLAN/Unicast Address Table Entry

**Table 12-175. Unicast Address Table Entry Bit Values**

70:68	67:66	65	64	63:62	61:60	59:48	47:0
Reserved	PORT_NUMBER	BLOCK	SECURE	UNICAST_TYPE (00) or (X1)	ENTRY_TYPE (11)	VLAN_ID	UNICAST_ADDRESS

#### Port Number (PORT\_NUMBER)

This field indicates the port number (not port mask) that the packet with a unicast destination address may be forwarded to. Packets with unicast destination addresses are forwarded only to a single port (but not the receiving port).]

#### Block (BLOCK)

The block bit indicates that a packet with a matching source or destination address should be dropped (block the address).

0 - Address is not blocked.

1 - Drop a packet with a matching source or destination address (secure must be zero)

If block and secure are both set, then they no longer mean block and secure. When both are set, the block and secure bits indicate that the packet is a unicast supervisory (super) packet and they determine the unicast forward state test criteria. If both bits are set then the packet is forwarded if the receive port is in the Forwarding/Blocking/Learning state. If both bits are not set then the packet is forwarded if the receive port is in the Forwarding state.

#### Secure (SECURE)

This bit indicates that a packet with a matching source address should be dropped if the received port number is not equal to the table entry PORT\_NUMBER.

0 - Received port number is a don't care.

1 - Drop the packet if the received port is not the secure port for the source address and do not update the address (block must be zero)

#### Unicast Type (UNICAST\_TYPE)

This field indicates the type of unicast address the table entry contains.

00 - Unicast address that is not ageable.

01 - Ageable unicast address that has not been touched.

10 - OUI address - lower 24-bits are don't cares (not ageable).

11 - Ageable unicast address that has been touched.

#### Table Entry Type (ENTRY\_TYPE)

Address entry. Unicast or multicast determined by address bit 40.

11: VLAN address entry. Unicast or multicast determined by address bit 40.

#### VLAN ID (VLAN\_ID)

The unique identifier for VLAN identification. This is the 12-bit VLAN ID.

#### Packet Address (UNICAST\_ADDRESS)

This is the 48-bit packet MAC address. All 48-bits are used in the lookup.

### 12.2.1.4.6.1.9.7 VLAN Table Entry

**Table 12-176. VLAN Table Entry**

70:68	67	66	65	64:62	61:60	59:48	47:46	45:44
Reserved	NO_LEARN_MASK	Reserved	VLAN_FORCE_INGRESS_CHECK	Reserved (000)	ENTRY_TYPE (10)	VLAN_ID	Reserved	REG_MCAST_FLOOD_INDEX
43:26	25:24	23:22	21:20	19:2	1:0			
Reserved	FORCE_UNTAGGED_EGRESS	Reserved	UNREG_MCAST_FLOOD_INDEX	Reserved	VLAN_MEMBER_LIST			

#### No Learn Mask (NO\_LEARN\_MASK)

When a bit is set in this mask, a packet with an unknown source address received on the associated port will not be learned (i.e. When a VLAN packet is received and the source address is not in the table, the source address will not be added to the table).

#### VLAN Force Ingress Check (VLAN\_FORCE\_INGRESS\_CHECK)

If the receive port is not a member of this VLAN then the packet is dropped. This is similar to the `ly_REG_Py_VID_INGRESS_CHECK` bit in the `CPSW_ly_ALE_PORTCTL0_y` registers except this check is for this VLAN only (not all VLANs).

#### Table Entry Type (ENTRY\_TYPE)

10: VLAN entry

#### VLAN ID (VLAN\_ID)

The unique identifier for VLAN identification. This is the 12-bit VLAN ID.

#### Registered Multicast Flood Index (REG\_MCAST\_FLOOD\_INDEX)

Index into `CPSW_ALE_MSK_MUX0` to `CPSW_lx_ALE_MSK_MUXx` register array that is used to create the registered multicast flood mask.

#### Force Untagged Packet Egress (FORCE\_UNTAGGED\_EGRESS)

This field causes the packet VLAN tag to be removed on egress (except on port 0).

#### Unregistered Multicast Flood Mask (UNREG\_MCAST\_FLOOD\_INDEX)

Index into `CPSW_ALE_MSK_MUX0` to `CPSW_lx_ALE_MSK_MUXx` register array that is used to create the unregistered multicast flood mask.

#### VLAN Member List (VLAN\_MEMBER\_LIST)

This field indicates which port(s) are members of the associated VLAN. One bit per port.

### 12.2.1.4.6.1.10 ALE Policing and Classification

The ALE has a number of configurable classifier engines (policers) that can be used for classification. Classification is a subset of the policing function and uses a policer without the color marking or rate limiting functions. A policer is a hardware engine that is used for policing. The `POLCNTDIV8` field in the `CPSW_ALE_STATUS` register indicates the number of policers available to be used for classification. Each policer can be enabled to match on one or more of any of the below packet fields for classification. All but Port and Priority are index references to the ALE table entries.

- Port Number
- Priority extracted from VLAN, mapped from DSCP if enabled, or Default Port Priority
- Organization Network Unique identifier - ONU
- Destination Address - DA
- Source Address - SA
- VLANID



- Ether Type
- IP Source Address - IPSA with full CIDR masking
- IP Destination Address - IPSA with full CIDR masking
- Support Host Thread/Flow ID mapping based on any packet classification above

#### 12.2.1.4.6.1.10.1 ALE Classification

When the policers are configured as classifiers, the color marking and policing functions of the policing/classifier engines are not used. One or multiple classifiers can be configured to match on a single packet. For example, a classifier can be enabled to match on priority while another classifier could match IP address.

##### 12.2.1.4.6.1.10.1.1 Classifier to CPPI Transmit Flow ID Mapping

The ALE can generate a 6-bit transmit CPPI Flow ID based on classifier matches that can be used instead of the switch default transmit Flow ID mapping. The switch default flow ID is the remapped received packet priority (0 to 7). Thread and flow ID are used interchangeably for this since there is a single hardware thread (TXST\_THREAD\_MREADY) but there are 6-bits of FLOW\_ID in the transmit CPPI INFO word 0. When enabled, the highest classifier match can map to a particular 6-bit flow ID value that is associated with the classifier. The ALE also supports an optional ALE default thread/flow ID value in the event that no classifiers match. Each thread/flow ID, including the ALE default thread/flow ID, has an enable such that the ALE default thread/Flow ID is used if enabled and if no matches occur (instead of the remapped received packet priority). If the ALE default is not enabled and no matches occur then the switch default value will be used. If multiple classifier matches occur, the highest match with a thread enable bit set will be used. The resultant flow ID has the CPSW\_P0\_FLOW\_ID\_OFFSET\_REG register value added to it to determine the actual value in the INFO 0 Flow ID field.

Three registers are used for ALE classification thread/flow ID mapping configuration (CPSW\_ALE\_THREADMAPDEF, CPSW\_ALE\_THREADMAPCTL and CPSW\_ALE\_THREADMAPVAL). The three thread mapping registers are used independently and are separate from the other ALE policing registers. The CPSW\_ALE\_THREADMAPCTL register allows the CPSW\_ALE\_THREADMAPVAL register contents to be written to the selected classifier. There is a single CPSW\_ALE\_THREADMAPDEF that is used for all classifiers. The thread mapping registers can be written or changed at any time but any packets that are already processed will not have their thread altered.

##### 12.2.1.4.6.1.11 DSCP

The ALE can map DSCP field to priority prior to classification matching. When enabled the DSCP is mapped via 64 priority entries such that any DSCP value can be mapped to any of the eight priorities. When a packet is received without a VLAN priority this remapped priority can be used instead of the default Port VLAN priority field. See CPSW\_P0\_RX\_DSCP\_MAP\_REG\_y and CPSW\_PN\_RX\_DSCP\_MAP\_REG\_y registers in the Register Manual section for DSCP mapping.

##### 12.2.1.4.6.1.12 Packet Forwarding Processes

There are four processes that an incoming received packet may go through to determine packet forwarding. The processes are *Ingress Filtering*, *VLAN\_Aware Lookup*, and *Egress*.

Packet processing begins in the Ingress Filtering process. Each port has an associated packet forwarding state that can be one of four values (Disabled, Blocked, Learning, or Forwarding). The default state for all ports is Disabled. The host sets the packet forwarding state for each port.

In the packet ingress process (receive packet process), there is a forward state test for unicast destination addresses and a forward state test for multicast addresses. The multicast forward state test indicates the port states required for the receiving port in order for the multicast packet to be forwarded to the transmit port(s). A transmit port must be in the Forwarding state for the packet to be forwarded for transmission. The MCAST\_FWD\_STATE indicates the required port state for the receiving port as indicated in the preceding table. The unicast forward state test indicates the port state required for the receiving port in order to forward the unicast packet. The transmit port must be in the Forwarding state in order to forward the packet. The BLOCK and SECURE bits determine the unicast forward state test criteria. If both bits are set then the packet is forwarded if the receive port is in the Forwarding/Blocking/Learning state. If both bits are not set then the packet



is forwarded if the receive port is in the Forwarding state. The transmit port must be in the Forwarding state regardless. The forward state test used in the ingress process is determined by the destination address packet type (multicast/unicast).

In general, packets received with errors are dropped by the address lookup engine without learning, updating, or touching the address. The error condition and the abort are indicated by the Ethernet port to the ALE. Packets with errors may be passed to the host (not aborted) by a Ethernet port, if the port has the RX\_CMF\_EN, RX\_CEF\_EN, or RX\_CSF\_EN bit(s) set in the CPSW\_PN\_MAC\_CONTROL\_REG register. Error packets that are passed to the host by the Ethernet port are considered to be bypass packets by the ALE and are sent only to the host. Error packets do not learn, update, or touch addresses regardless of whether they are aborted or sent to the host. Packets with long or short errors received by the host are dropped. Packets with errors received by the host are forwarded as normal.

The following control bits are in the CPSW\_PN\_MAC\_CONTROL\_REG register:

- [22] RX\_CEF\_EN - enables frames that are fragments, long, jabber, CRC, code, and alignment errors to be forwarded
- [23] RX\_CSF\_EN - enables short frames to be forwarded
- [24] RX\_CMF\_EN - enables MAC control frames to be forwarded.

#### 12.2.1.4.6.1.12.1 Ingress Filtering Process

Condition and action
If ((ALE BYPASS) and (host port is not the receive port)) then use host portmask and go to Egress process
if (directed packet) then use directed port number and go to Egress process
If (Rx Iy_REG_Py_PORTSTATE is Disabled) then discard the packet
if ((ALE BYPASS or error packet) and (host port is not the receive port)) then use host portmask and go to Egress process
if (((BLOCK) and (unicast source address found)) or ((BLOCK) and (unicast destination address found))) then discard the packet
if ((ENABLE_RATE_LIMIT) and (rate limit exceeded) and (not BCAST_MCAST_CTL)) then if (((Multicast/Broadcast destination address found) and (not SUPER)) or (Multicast/Broadcast destination address not found)) then discard the packet
if ((not forward state test valid) and (destination address found)) then discard the packet to any port not meeting the requirements <ul style="list-style-type: none"> <li>• Unicast destination addresses use the unicast forward state test and multicast destination addresses use the multicast forward state test.</li> </ul>
if ((destination address not found) and ((not transmit port forwarding) or (not receive port forwarding))) then discard the packet to any ports not meeting the above requirements
if (source address found) and (secure) and (not block) and (receive port number != port_number)) then discard the packet
if ((not super) and (drop_untagged) and ((non-tagged packet) or ((priority tagged) and not(en_vid0_mode))) then discard the packet

<p>If (VLAN_Unaware)</p> <p>CPSW_ALE_UVLAN_UNTAG = "000"</p> <p>CPSW_ALE_UVLAN_RMCAST = "111"</p> <p>CPSW_ALE_UVLAN_URCAST = "111"</p> <p>CPSW_ALE_UVLAN_MEMBER = "111"</p> <p>else if (VLAN not found)</p> <p>CPSW_ALE_UVLAN_UNTAG = CPSW_ALE_UVLAN_UNTAG</p> <p>CPSW_ALE_UVLAN_RMCAST = CPSW_ALE_UVLAN_RMCAST</p> <p>CPSW_ALE_UVLAN_URCAST = CPSW_ALE_UVLAN_URCAST</p> <p>CPSW_ALE_UVLAN_MEMBER = CPSW_ALE_UVLAN_MEMBER</p> <p>else</p> <p>CPSW_ALE_UVLAN_UNTAG = found CPSW_ALE_UVLAN_UNTAG</p> <p>CPSW_ALE_UVLAN_URCAST = found CPSW_ALE_UVLAN_URCAST</p> <p>CPSW_ALE_UVLAN_RMCAST = found CPSW_ALE_UVLAN_RMCAST</p> <p>CPSW_ALE_UVLAN_MEMBER = found CPSW_ALE_UVLAN_MEMBER</p>
<p>if ((not SUPER) and (ly_REG_Py_VID_INGRESS_CHECK) and (Rx port is not VLAN member))</p> <p>then discard the packet</p>
<p>if ((ENABLE_AUTH_MODE) and (source address not found) and not(destination address found and (SUPER)))</p> <p>then discard the packet</p>
<p>if (destination address equals source address)</p> <p>then discard the packet</p>
<p>if (VLAN_AWARE) goto VLAN_Aware_Lookup process</p> <p>else goto VLAN_Unaware_Lookup process</p>

#### 12.2.1.4.6.1.12.2 VLAN\_Aware Lookup Process

Condition and action
<p>if ((unicast packet) and (destination address found with or without VLAN) and (not SUPER))</p> <p>then portmask is the logical "AND" of the PORT_NUMBER and UVLAN_MEMBER_LIST less the host port</p> <p>and goto Egress process</p>
<p>if ((unicast packet) and (destination address found with or without VLAN) and (not SUPER))</p> <p>then portmask is the logical "AND" of the PORT_NUMBER and the UVLAN_MEMBER_LIST and goto Egress process</p>
<p>if ((unicast packet) and (destination address found with or without VLAN) and (SUPER))</p> <p>then portmask is the PORT_NUMBER and goto Egress process</p>
<p>if (Unicast packet) # destination address not found</p> <p>then portmask is VLAN member LIST less host port and goto Egress process</p>
<p>if ((Multicast packet) and (destination address found with or without VLAN) and (not SUPER))</p> <p>then portmask is the logical "AND" of CPSW_ALE_UVLAN_URCAST and found destination address/VLAN portmask (PORT_MASK) and UVLAN_MEMBER_LIST and goto Egress process</p>
<p>if ((Multicast packet) and (destination address found with or without VLAN) and (SUPER))</p> <p>then portmask is the PORT_MASK and goto Egress process</p>

if (Multicast packet) # destination address not found then portmask is the logical "AND" of CPSW_ALE_UVLAN_URCAST and UVLAN_MEMBER_LIST then goto Egress process
if (Broadcast packet) then use found UVLAN_MEMBER_LIST and goto Egress process

### Note

The UVLAN\_MEMBER\_LIST, UVLAN\_UNREG\_MCAST\_FLOOD\_MASK, UVLAN\_REG\_MCAST\_FLOOD\_MASK and UVLAN\_FORCE\_UNTAGGED\_EGRESS are set in the [Section 12.2.1.4.6.1.12.1 Ingress Filtering Process](#), based on VLAN\_Unaware, Unknown\_VLAN rules and VLAN table entries.

#### 12.2.1.4.6.1.12.3 Egress Process

Condition and action
Clear Rx port from portmask (don't send packet to Rx port).
Clear disabled ports from portmask.
if ((ENABLE_OUI_DENY) and (OUI source address not found) and (not ALE BYPASS) and (not error packet) and ((not mcast destination address) and (SUPER))) then Clear host port from portmask
if ((not ENABLE_OUI_DENY) and (OUI source address found) and (not ALE BYPASS) and (not error packet) and not ((mcast destination address) and (SUPER))) then Clear host port from portmask
if ((ENABLE_RATE_LIMIT) and (BCAST_MCAST_CTL)) then if (not SUPER) and (rate limit exceeded on any tx port) then clear rate limited tx port from portmask If address not found then SUPER cannot be set.
If portmask is zero then discard packet
Send packet to portmask ports.

#### 12.2.1.4.6.1.12.4 Learning/Updating/Touching Processes

The learning, updating, and touching processes are applied to each receive packet that is not aborted. The processes are concurrent with the packet forwarding process. In addition to the following, a packet must be received without error in order to learn/update/touch an address.

##### 12.2.1.4.6.1.12.4.1 Learning Process

The learning process is applied to each receive packet that is not aborted. The learning process is a concurrent process with the packet forwarding process.

Condition and action
If (directed) then do not learn, update, or set touched else continue
If (not (Learning or Forwarding) or (ENABLE_AUTH_MODE) or (packet error) or (ly_REG_Py_NO_LEARN)) then do not learn address
if ((Non-tagged packet) and (ly_REG_Py_DROP_UN_TAGGED)) then do not learn address

if ((VLAN_AWARE) and (VLAN not found) and (unknown UVLAN_MEMBER_LIST = "000")) then do not learn address
if ((ly_REG_Py_VID_INGRESS_CHECK) and (Rx port is not VLAN member) and (VLAN found)) then do not learn address
if ((source address found) and (receive port_number != PORT_NUMBER) and (SECURE or BLOCK)) then do not update address else continue
if ((source address found) and (receive port number != PORT_NUMBER)) then update address else continue
if ((source address not found) and (VLAN_AWARE) and not (LEARN_NO_VLANID)) then learn address with VLAN
if ((source address not found) and ((not VLAN_AWARE) or (VLAN_AWARE and LEARN_NO_VLANID))) then learn address without VLAN

#### 12.2.1.4.6.1.12.4.2 Updating Process

Condition and action
if (dlr_unicast) then do not update address
If (not(Learning or Forwarding) or (ENABLE_AUTH_MODE) or (packet error) or (ly_REG_Py_NO_SA_UPDATE)) then do not update address
if ((Non-tagged packet) and (ly_REG_Py_DROP_UN_TAGGED)) then do not update address
if ((VLAN_AWARE) and (VLAN not found) and (unknown UVLAN_MEMBER_LIST = "000")) then do not update address
if ((ly_REG_Py_VID_INGRESS_CHECK) and (Rx port is not VLAN member) and (VLAN found)) then do not update address
if ((source address found) and (receive port number != PORT_NUMBER) and (SECURE or BLOCK)) then do not update address
if ((source address found) and (receive port number != PORT_NUMBER)) then update address

#### 12.2.1.4.6.1.12.4.3 Touching Process

if ((source address found) and (ageable) and (not touched)) then set touched
---

#### 12.2.1.4.6.1.13 VLAN Aware Mode

The CPSW is in VLAN aware mode when the VLAN\_AWARE bit is set in the CPSW\_CONTROL\_REG register.

In VLAN aware mode, transmitted packet data is changed depending on the packet type (PKT\_TYPE), packet priority (PKT\_PRI), and VLAN information.

The VLAN\_LTYPE\_SEL value is selected by the S\_CN\_SWITCH bit in the CPSW\_CONTROL\_REG register and is either the VLAN\_LTYPE\_INNER (8100h default) or VLAN\_LTYPE\_OUTER (88A8h default) value.

#### 12.2.1.4.6.1.14 VLAN Unaware Mode

An egress port is operating in the VLAN unaware mode when the VLAN\_AWARE bit in the CPSW\_CONTROL\_REG register is cleared to 0h. In VLAN unaware mode, transmit (egress) packets are not modified on egress.

#### 12.2.1.4.6.2 Packet Priority Handling

Packets are received on two ports, one Ethernet port and one CPPI host port. Received packets have a received packet priority (0 to 7, with 7 being the highest priority).

The received packet priority is determined as follows:

1. If the first packet LTYPE = VLAN\_LTYPE\_SEL then the received packet priority is the packet priority (VLAN tagged and priority tagged packets).
2. Else if the first packet LTYPE = 0x0800 and byte 14 (following the LTYPE) is equal to 0x4X, and DSCP\_IPV4\_EN is set in CPSW\_P0\_CONTROL\_REG or CPSW\_PN\_CONTROL\_REG, then the received packet priority is the 6-bit TOS field in byte 15 (upper 6 bits) mapped through the port's DSCP priority mapping registers (IPv4 packet).
3. Else if the first packet LTYPE = 0x86DD and the most significant nibble of byte 14 (following the LTYPE) is equal to 0x6, and DSCP\_IPV6\_EN is set in CPSW\_P0\_CONTROL\_REG or CPSW\_PN\_CONTROL\_REG, then the received packet priority is the 6-bit priority (in the 6-bits following the upper nibble 0x6) mapped through the port's DSCP priority mapping registers (IPv6 packet).
4. Else the received packet priority is the source (ingress) port priority.

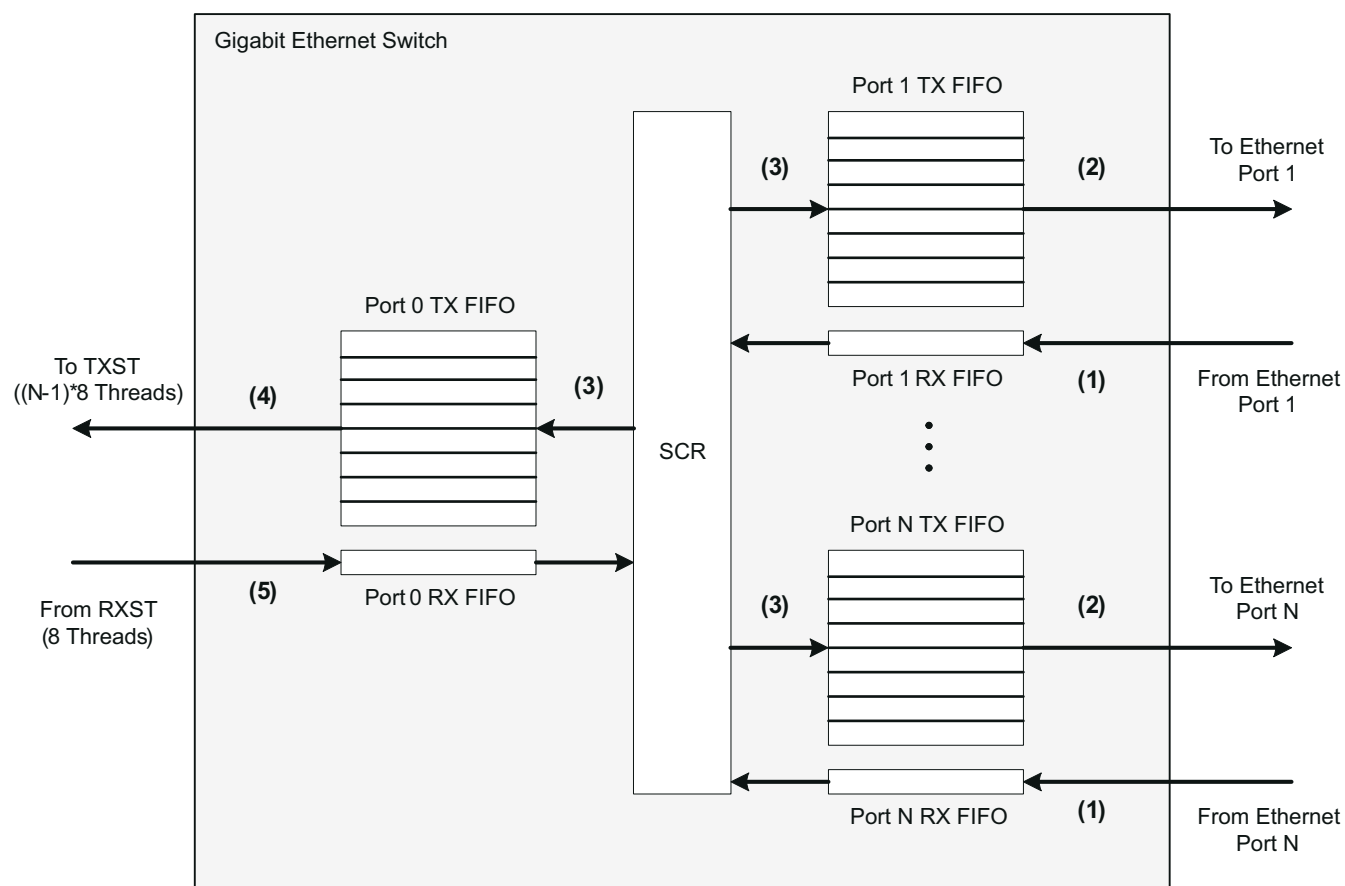
The received packet priority is mapped through the receive ports associated packet-priority-to-header-packet-priority-mapping register (CPSW\_PN\_RX\_PRI\_MAP\_REG) to obtain the header packet priority. The header packet priority is the hardware switch priority. The header packet priority is also used as the actual transmit packet priority if the VLAN information is to be sent on egress.

The header packet priority is mapped at each destination FIFO through the CPSW\_PN\_TX\_PRI\_MAP\_REG register (header priority to switch priority mapping register) to obtain the hardware switch priority (hardware queue 0 through 7).

##### 12.2.1.4.6.2.1 Priority Mapping and Transmit VLAN Priority

There are three priorities that are used inside the Gigabit Ethernet Switch: **packet priority**, **header packet priority**, and **switch priority**. The **header packet priority** is used as the outgoing VLAN priority if the packet is egressing from the switch with a VLAN tag. The **switch priority** determines which of the 8 FIFO priority queues the packet uses during egress.

[Figure 12-135](#) below, as well as the corresponding explanation that follows, explains each of the priorities, how they are determined, and how they are used. A number in parentheses in the figure indicates a process (Ethernet port ingress, host port egress, etc.). Each bullet in the text following the diagram explains one of the 5 processes pointed out in the figure.



**Figure 12-135. Gigabit Ethernet Switch Priority Mapping and Transmit VLAN Processing**

From [Figure 12-135](#) above:

- (1) is the ingress process that occurs at the external Ethernet ports
  - The incoming packet is assigned a **packet priority** based on either its VLAN priority, IPv4 or IPv6 DSCP value, or the ingress port's priority. This **packet priority** is then mapped to a **header packet priority** using the CPSW\_PN\_RX\_PRI\_MAP\_REG register where N is the port where the packet entered the switch. This process is explained in further detail in [Section 12.2.1.4.6.2](#).
- (2) is the egress process that occurs at the external Ethernet ports
  - If the switch is in VLAN Aware mode then the VLAN header may be added, replaced, or removed during the egress process. If the VLAN header is to be added or replaced, the VLAN priority will come from the **header packet priority** that was determined in process (1) or (5). Transmit VLAN processing is the same for both the host port and the external Ethernet ports and is described in [Section 12.2.1.4.6.4.1](#).
- (3) is the process by which it is decided which priority TX queue to place the packet on in the Port N TX FIFO during egress
  - Each Port's TX FIFO has 8 queues that each correspond to a priority that is used when determining which packet will egress from the switch next at that port. The **header packet priority** (Ethernet port ingress, process (1)) or the receive packet thread (host port 0 ingress, process (5)) gets mapped through the CPSW\_PN\_RX\_PRI\_MAP\_REG register (where N is the egress port number) to determine the **switch priority** of the packet. The **switch priority** determines which TX FIFO queue to place the packet in. The FIFO architecture is described in [Section 12.2.1.4.6.10.5](#). The header packet priority to switch priority mapping is discussed in [Section 12.2.1.4.6.2](#).
- (4) is the egress process that occurs at Host Port 0 toward the transmit streaming interface (TXST)
  - The TXST has 8 egress threads for each external Ethernet port (32 threads in a 5-port switch and 64 threads in a 9-port switch). The TX thread that is selected for a packet is determined by the Ethernet port of ingress and the **switch priority** of the packet that was determined in process (3).
  - If the switch is in VLAN Aware mode then the VLAN header may be added, replaced, or removed during the egress process. If the VLAN header is to be added or replaced, the VLAN priority will come from the **header packet priority** that was determined in process (1). Transmit VLAN processing is the same for both the host port and the external Ethernet ports and is described in [Section 12.2.1.4.6.4.1](#).
- (5) is the ingress process that occurs at Host Port 0
  - The incoming packet is assigned a **packet priority** based on either its VLAN priority, IPv4 or IPv6 DSCP value, or the host port's priority. This packet priority is then mapped to a **header packet priority** using the CPSW\_PN\_RX\_PRI\_MAP\_REG register.
  - Host port 0 also has a received packet thread. The receive packet thread is based on either the packet's VLAN priority, IPv4 or IPv6 DSCP value, or the streaming interface (RXST) thread that the packet entered host port 0 on.

#### 12.2.1.4.6.3 CPPI Port Ingress

Packets received on the CPPI host port have a received packet priority (0 to 7 with 7 being the highest priority).

The received packet priority is determined as follows:

1. If the first packet LTYPE = VLAN\_LTYPE\_SEL then the received packet priority is the packet priority (VLAN tagged and priority tagged packets).
2. Else if the first packet LTYPE = 0x0800 and byte 14 (following the LTYPE) is equal to 0x4X, and DSCP\_IPV4\_EN is set in CPSW\_P0\_CONTROL\_REG or CPSW\_PN\_CONTROL\_REG register, then the received packet priority is the 6-bit TOS field in byte 15 (upper 6 bits) mapped through the port's DSCP priority mapping registers (IPv4 packet).
3. Else if the first packet LTYPE = 0x86DD and the most significant nibble of byte 14 (following the LTYPE) is equal to 0x6, and DSCP\_IPV6\_EN is set in CPSW\_P0\_CONTROL\_REG or CPSW\_PN\_CONTROL\_REG register, then the received packet priority is the 6-bit priority (in the 6-bits following the upper nibble 0x6) mapped through the port's DSCP priority mapping registers (IPv6 packet).
4. Else the received packet priority is the source (ingress) port priority



The CPPI Port (port 0) also has a received packet thread. The received packet thread is determined exactly as the received packet priority except for untagged packets. For untagged packets, the received packet thread is the packet streaming interface thread that the packet was received on (instead of the port VLAN priority or DSCP priority). The received packet thread is the hardware switch priority. The received packet thread only determines which hardware switch priority the packet should be sent to, the egress VLAN rules are identical to packets that were received on Ethernet ports (as determined by the header packet priority).

For CPPI ingress packets, the destination port hardware switch priority is the below selected value remapped through CPSW\_PN\_RX\_PRI\_MAP\_REG:

1. If the ingress packet is priority tagged or vlan tagged:
  - If RX\_REMAP\_VLAN in CPSW\_P0\_CONTROL\_REG register is clear then the destination hardware switch priority is the CPPI receive thread number.
  - If RX\_REMAP\_VLAN in CPSW\_P0\_CONTROL\_REG register is set then the destination hardware switch priority is the packet priority value. Port transmit remapping (CPSW\_PN\_TX\_PRI\_MAP\_REG should remain the default value) is not compatible with this bit being set, but remapping can be configured on port 0 receive.
2. Else if the ingress packet has the first packet LTYPE = 0x0800 and byte 14 (following the LTYPE) is equal to 0x4X, and DSCP\_IPV4\_EN is set in CPSW\_P0\_CONTROL\_REG register:
  - If RX\_REMAP\_DSCP\_V4 bit in CPSW\_P0\_CONTROL\_REG register is clear then the destination hardware switch priority is the CPPI receive thread number.
  - If RX\_REMAP\_DSCP\_V4 bit in CPSW\_P0\_CONTROL\_REG register is set then the destination hardware switch priority is the 6-bit TOS field in byte 15 (upper 6-bits) mapped through the port's DSCP priority mapping registers (IPv4 packet). Port 1 transmit remapping (CPSW\_PN\_TX\_PRI\_MAP\_REG should remain the default value) is not compatible with this bit being set, but remapping can be configured on port 0 receive.
3. Else if the ingress packet has the first packet LTYPE = 0x86DD and the most significant nibble of byte 14 (following the LTYPE) is equal to 0x6, and DSCP\_IPV6\_EN is set in P0\_CONTROL\_REG register:
  - If RX\_REMAP\_DSCP\_V6 bit in CPSW\_P0\_CONTROL\_REG register is clear then the destination hardware switch priority is the CPPI receive thread number.
  - If RX\_REMAP\_DSCP\_V6 bit in CPSW\_P0\_CONTROL\_REG register is set then the destination hardware switch priority is the 6-bit priority (in the 6-bits following the upper nibble 0x6) mapped through the port's DSCP priority mapping registers (IPv6 packet). Port 1 transmit remapping (CPSW\_PN\_TX\_PRI\_MAP\_REG should remain the default value) is not compatible with this bit being set, but remapping can be configured on port 0 receive.
4. Else the ingress packet is non-tagged and the destination hardware switch priority is the CPPI receive thread number.

#### 12.2.1.4.6.4 Packet CRC Handling

The P0\_TX\_CRC\_REMOVE bit in the CPSW\_CONTROL\_REG register determines if host port egress packets have CRC included or not. If P0\_TX\_CRC\_REMOVE is set to 1h then all packets that are transmitted from port 0 do not contain CRC. If P0\_TX\_CRC\_REMOVE bit is cleared to 0h then all packets that are transmitted from port 0 contain CRC. The CRC type, if present, is determined by the CRC\_TYPE bit in the CPSW\_PN\_MAC\_CONTROL\_REG register. If the CRC\_TYPE bit is cleared to 0h then the CRC present in each packet after host port egress is Ethernet CRC. If the CRC\_TYPE bit is set to 1h then the CRC present in each packet after host port egress is Castagnoli CRC.

#### Note

The CRC type present in the packet after host port egress is determined solely by the CRC\_TYPE bit in the CPSW\_PN\_MAC\_CONTROL\_REG register regardless of the CRC type present in the packet during Ethernet port ingress.



#### 12.2.1.4.6.4.1 Transmit VLAN Processing

Transmit packets are NOT modified during switch egress when the VLAN\_AWARE bit in the CPSW\_CONTROL\_REG register is cleared to 0h. This means that the switch is not in VLAN-aware mode.

The next three sections cover transmit processing when the switch is in VLAN-aware mode for different packet types. The Gigabit Ethernet switch is in VLAN-aware mode when the VLAN\_AWARE bit is set in the CPSW\_CONTROL\_REG register. While in VLAN-aware mode, VLAN is added, removed, or replaced according to the type of packet as well as the CPSW\_ALE\_UVLAN\_UNTAG[1-0] UVLAN\_FORCE\_UNTAGGED\_EGRESS bit in the packet header as explained below.

##### 12.2.1.4.6.4.1.1 Untagged Packets (No VLAN or Priority Tag Header)

Untagged packets are all packets that are not a VLAN packet or a priority tagged packet. According to the CPSW\_ALE\_UVLAN\_UNTAG[1-0] UVLAN\_FORCE\_UNTAGGED\_EGRESS bit in the packet header the packet may exit the switch with a VLAN tag inserted or the packet may leave the switch unchanged. The two cases are discussed below.

- Insert VLAN Case:

Untagged input packets have the header packet VLAN inserted when the CPSW\_ALE\_UVLAN\_UNTAG[1-0] UVLAN\_FORCE\_UNTAGGED\_EGRESS bit in the transmit packet header is de-asserted. For untagged packets, the VLAN EtherType = 0x8100 is inserted after the source address followed by the two byte header packet VLAN. The header packet VLAN is composed of the header packet priority along with the PORT\_CFI and PORT\_VID values from the CPSW\_PN\_PORT\_VLAN\_REG register (where N is the port that the untagged packet entered the switch) through. The packet length/type field is output four bytes later than it is input and is not removed or replaced. If the CRC is present in the packet data (PASS\_CRC is asserted), then the packet CRC is replaced with a hardware generated CRC.

- No Change Case:

Untagged input packets are output unchanged when the CPSW\_ALE\_UVLAN\_UNTAG[1-0] UVLAN\_FORCE\_UNTAGGED\_EGRESS transmit packet header bit is asserted.

##### 12.2.1.4.6.4.1.2 Priority Tagged Packets (VLAN VID == 0 && EN\_VID0\_MODE == 0h)

Priority tagged packets are packets that contain a VLAN header with VID = 0. According to the CPSW\_ALE\_UVLAN\_UNTAG[1-0] UVLAN\_FORCE\_UNTAGGED\_EGRESS bit in the packet header, priority tagged packets may exit the switch with their VLAN ID and priority replaced or they may have their priority tag completely removed. The two cases are discussed below.

#### Note

In order for a priority tagged packet to fall into this category the ENABLE\_VID0\_MODE bit in the CPSW\_ALE\_CONTROL register must also be set to 0h. If the ENABLE\_VID0\_MODE bit in the CPSW\_ALE\_CONTROL register is set to 1h, then packets with a VLAN VID of 0 will fall into the VLAN Tagged Packets category in [Section 12.2.1.4.6.4.1.3](#) below.

- Replace Priority and VLAN ID Case:

Priority tagged input packets have the packet VLAN ID and the packet priority replaced with the header packet VLAN ID and the header packet priority when the transmit packet header CPSW\_FORCE\_UNTAGGED\_EGRESS\_REG[1-0] MASK bit is de-asserted. The header packet VLAN ID comes from the PORT\_VID bits in the CPSW\_PN\_PORT\_VLAN\_REG register (where N is the port where the packet entered the switch). The header packet priority is based on the packet priority to header packet priority mapping in the CPSW\_PN\_RX\_PRI\_MAP\_REG register (where N is the port where the packet entered the switch).

- Remove VLAN Header Case:

Priority tagged input packets have the 4-byte packet VLAN information removed when the transmit packet header CPSW\_ALE\_UVLAN\_UNTAG[1-0] UVLAN\_FORCE\_UNTAGGED\_EGRESS bit is asserted. The 0x8100 EtherType is removed as is the two byte packet VLAN. Input 64-67 byte priority tagged packets go out with the VLAN removed and padded to 64-bytes if the PASS\_CRC input bit is asserted. The input CRC bytes are used as the pad data. Input 64-byte priority-tagged packets use all four input CRC bytes as pad, input 65-byte priority-tagged packets use three of the input CRC bytes as pad, and so on. No pad is performed if the PASS\_CRC input bit is not asserted - input 64-67 byte (on the wire) priority-tagged packets go out as 60-63 byte packets. The output CRC is replaced with a generated CRC when the VLAN is removed.

#### **12.2.1.4.6.4.1.3 VLAN Tagged Packets (VLAN VID != 0 || (EN\_VID0\_MODE == 1h && VLAN VID == 0))**

VLAN tagged packets are packets that contain a VLAN header specifying the VLAN the packet belongs to (VID), the packet priority (PRI), and the drop eligibility indicator (CFI). According to the CPSW\_ALE\_UVLAN\_UNTAG[1-0] UVLAN\_FORCE\_UNTAGGED\_EGRESS bit in the packet header, VLAN tagged packets may exit the switch with their VLAN priority replaced or they may have their VLAN header completely removed. The two cases are discussed below.

- Replace Priority Case:

VLAN tagged input packets are output with the packet priority replaced with the header packet priority when the transmit packet header CPSW\_ALE\_UVLAN\_UNTAG[1-0] UVLAN\_FORCE\_UNTAGGED\_EGRESS bit is deasserted. The header packet priority is based on the packet priority to header packet priority mapping in the CPSW\_PN\_RX\_PRI\_MAP\_REG register (where N is the port where the packet entered the switch). If the CRC is present in the packet data (PASS\_CRC is asserted), then the packet CRC is replaced with a generated CRC.

- Remove VLAN Header Case:

VLAN tagged input packets have the 4-byte packet VLAN information removed when the transmit packet header CPSW\_ALE\_UVLAN\_UNTAG[1-0] UVLAN\_FORCE\_UNTAGGED\_EGRESS bit is asserted. The 0x8100 EtherType is removed as is the two byte packet VLAN. Input 64-67 byte priority tagged packets go out with the VLAN removed and padded to 64-bytes if the PASS\_CRC input bit is asserted. The input CRC bytes are used as the pad data. Input 64-byte priority tagged packets use all four input CRC bytes as pad, input 65-byte priority tagged packets use three of the input CRC bytes as pad, and so on. No pad is performed if the PASS\_CRC input bit is not asserted - input 64-67 byte (on the wire) priority tagged packets go out as 60-63 byte packets. The output CRC is replaced with a generated CRC when the VLAN is removed.

#### **12.2.1.4.6.4.2 Ethernet Port Ingress Packet CRC**

All Ethernet ports check the ingress packet CRC in all modes/speeds. The receive port can check either Ethernet CRC or Castagnoli CRC as determined by the CRC\_TYPE bit in the CPSW\_PN\_MAC\_CONTROL\_REG register.

#### **12.2.1.4.6.4.3 Ethernet Port Egress Packet CRC**

Ethernet ports transmit each egress packet with the CRC selected by the CRC\_TYPE bit in the CPSW\_PN\_MAC\_CONTROL\_REG register, regardless of the type of CRC that the packet had on ingress to the switch. At the egress port after passing through the switch, the packet CRC is checked for correctness and if the CRC is correct then the packet is output with the generated selected output CRC. If the packet CRC is incorrect, due either to a bit flip in a memory or an error CRC passed in on host ingress, then the generated egress CRC type is used with at least a single byte of the CRC inverted to indicate the error. If the packet length including CRC is divisible by 4 then all 4 CRC bytes will be inverted on error. If there are three bytes remainder after dividing the packet length by 4 then three bytes will be inverted (and so on down to one byte remainder).

#### **12.2.1.4.6.4.4 CPPI Port Ingress Packet CRC**

CPPI port ingress packets can be passed in with or without a CRC. The host port is Ethernet CRC only. CPPI ingress packets are not checked for CRC correctness on CPPI ingress, however they are checked for

correctness on Ethernet egress and are output with a CRC error if they came in with a CRC error. If a CPPI ingress packet does not have the INFO0 PASSED\_CRC bit set then a CRC will be generated for the packet on CPPI ingress. If the PASSED\_CRC bit is set then the packet is received and forwarded unchanged. The CRC type is input in the INFO0 word CRC\_TYPE bit and must be Ethernet CRC.

#### 12.2.1.4.6.4.5 CPPI Port Egress Packet CRC

The P0\_TX\_CRC\_REMOVE bit in the CPSW\_CONTROL\_REG register determines if CPPI egress packet have an Ethernet CRC included or not.

#### 12.2.1.4.6.5 FIFO Memory Control

Each of the two CPSW\_3G ports has an identical associated FIFO. Each FIFO contains a single logical receive queue and eight logical transmit queues (priority 0 through 7 with 7 the highest priority). Each FIFO memory contains 20,480 bytes (20k) total organized as 2560 by 64-bit words contained in a single memory instance. The FIFO memory is used for the associated port transmit and receive queues. The TX\_MAX\_BLKs field in the FIFOs associated CPSW\_PN\_MAX\_BLKs\_REG register determines the maximum number of 1k FIFO memory blocks to be allocated to the eight logical transmit queues (transmit total). The RX\_MAX\_BLKs field in the FIFO's associated CPSW\_PN\_MAX\_BLKs\_REG register determines the maximum number of 1k memory blocks to be allocated to the logical receive queue. The TX\_MAX\_BLKs value plus the RX\_MAX\_BLKs value must sum to 20 (the total number of blocks in the FIFO). If the sum were less than 20, then some memory blocks would be unused. The default is 17 (decimal) transmit blocks and three receive blocks. The FIFOs follow the naming convention of the Ethernet ports. Host Port is Port0 and External Ports is Port1.

Each transmit FIFO contains a configurable number of blocks (20 max) that are either 1k or 4k byte blocks that can be allocated to any priority.

#### 12.2.1.4.6.6 FIFO Transmit Queue Control

There are eight transmit queues in the Ethernet port transmit FIFO. Software has some flexibility in determining how packets are loaded into the queues and on how packet priorities are selected for transmission (how packets are removed and transmitted from queues).

##### 12.2.1.4.6.6.1 CPPI Port Receive Rate Limiting

Port 0 receive operations can be configured to rate limit the host ingress data for each receive thread (priority). CPPI Receive has 8 threads for QOS. There is a committed information rate (CPSW\_P0\_PRI\_CIR\_REG\_y, where y = 0 to 7) and an excess information rate for each thread (CPSW\_P0\_PRI\_EIR\_REG\_y, where y = 0 to 7). Rate limiting is enabled for a thread when the committed information rate for the thread is non-zero. The excess information rate for a priority is enabled when the excess information rate for the priority is non-zero. The committed information rate must be non-zero if the excess information rate is configured to be non-zero. That is, there must be a configured non-zero committed information rate for there to be a configured non-zero excess information rate. Bulk traffic on other non-rate limited priorities does not impact the committed information traffic on a priority. However, bulk traffic on other non-rate limited threads does impact the excess information rates. No bulk thread will be enabled to send unless there are CPSW\_PN\_PRI\_CTL\_REG[15-12] TX\_HOST\_BLKs\_REM number of unused blocks remaining in each of the Ethernet port transmit FIFOs. The "blocks remaining check" ensures that bulk traffic from the host will not block rate-limited traffic from the host. Rate limited channels must be the highest priority channels. For example, if two rate limited channels are required then threads 7 and 6 should be configured for committed information (and excess information if desired). When any channels are configured to be rate-limited, the priority type must be fixed for receive. Round-robin priority type is not allowed when rate-limiting is configured for any thread. The configured transfer rate includes the inter-packet gap (12 bytes) and the preamble (8 bytes). The rate in Mbits/second that each priority is configured to receive is controlled by the below equation. If the configured excess information rate is zero, then only the committed information rate is transferred:

Priority Transfer rate [Mbit/s] = (((Frequency in MHz) \* CPSW\_P0\_PRI\_CIR\_REG\_y) / 32768) + (((Frequency in MHz) \* CPSW\_P0\_PRI\_EIR\_REG\_y) / 32768))

Where the *frequency* is the CPPI\_ICLK frequency (in MHz) and priority 0 to 7.

For example, 10Mbps on priority 7 would give the below:

10Mbps =  $\sim ((350 * 936) / 32768)$ , at 350Mhz and CPSW\_P0\_PRI\_CIR\_REG\_y[27-0] PRI\_CIR value = 936 (no excess information rate)

#### 12.2.1.4.6.6.2 Ethernet Port Transmit Rate Limiting

Ethernet port transmit operations can be configured to rate limit egress data for each egress priority. There is a committed information rate (CPSW\_P0\_PRI\_CIR\_REG\_y, where y = 0 to 7) and an excess information rate for each priority (CPSW\_P0\_PRI\_EIR\_REG\_y, where y = 0 to 7). Rate limiting is enabled for a priority when the committed information rate for the priority is non-zero. The excess information rate for a priority is enabled when the excess information rate for the priority is non-zero. The committed information rate must be non-zero if the excess information rate is configured to be non-zero. That is, there must be a configured non-zero committed information rate for there to be a configured non-zero excess information rate. Bulk traffic on other non-rate limited priorities does not impact the committed information traffic on a priority. However, bulk traffic on other non-rate limited threads does impact the excess information rates. Rate limited channels must be the highest priority channels. For example, if two rate limited channels are required then priorities 7 and 6 should be configured for committed information (and excess information if desired). The configured transfer rate includes the inter-packet gap (12 bytes) and the preamble (8 bytes). The rate in Mbits/second that each priority is configured to send is controlled by the below equation. If the excess information rate is disabled then the committed information rate only is transferred:

Priority Transfer rate [Mbit/s] =  $(((((\text{Frequency in MHZ}) * \text{CPSW\_P0\_PRI\_CIR\_REG\_y}) / 32768) + (((\text{Frequency in MHZ}) * \text{CPSW\_P0\_PRI\_EIR\_REG\_y}) / 32768)))$

Where the *frequency* is the CPPI\_ICLK frequency (in MHz) and priority 0 to 7.

#### 12.2.1.4.6.7 Intersperced Express Traffic (IET – P802.3br/D2.0)

When IET is enabled through CPSW\_CONTROL\_REG[17] IET\_ENABLE = 1h and configured, IET allows preemptable traffic on selected transmit priorities to be preempted by express traffic on selected express priorities. Received traffic is separated onto express and preempt receive queues. When IET is disabled (IET\_ENABLE = 0h), all Ethernet traffic is express traffic and switch operation is as if IET does not exist. Traffic is only intended to be moved to the preempt queue after preemption is verified and enabled.

##### 12.2.1.4.6.7.1 IET Configuration

- Using the preempt receive queue requires more blocks to be allocated to the ports receive FIFO. Write a value of decimal 7 to the CPSW\_PN\_MAX\_BLKES\_REG[7-0] RX\_MAX\_BLKES bit field, and a value of decimal 13 to the CPSW\_PN\_MAX\_BLKES\_REG[15-8] TX\_MAX\_BLKES register for every port to be enabled for IET.
- Write the [23-0]MAC\_VERIFY\_CNT bit field in the Ethernet port CPSW\_PN\_IET\_VERIFY\_REG register to set the verify/response timeout count. The default is 10ms for Gigabit mode. For other time values or link speeds the verify count should be updated. If CPSW\_PN\_IET\_CONTROL\_REG[2] MAC\_DISABLEVERIFY bit is to be set (this is forced mode) then this step is unneeded.
- The receive FIFO block allocation is insufficient if CPSW\_STAT0\_RX\_BOTTOM\_OF\_FIFO\_DROP/ CPSW\_STAT1\_RX\_BOTTOM\_OF\_FIFO\_DROP[31-0] COUNT is nonzero, indicating that receive packets are being dropped due to FIFO block allocation.
- Write the CPSW\_PN\_IET\_CONTROL\_REG register in the Ethernet port as below:
  - Set the CPSW\_PN\_CONTROL\_REG[16] IET\_PORT\_EN bit. The port will not actually be enabled until bit IET\_ENABLE is set in the CPSW\_CONTROL\_REG.
  - The CPSW\_PN\_IET\_CONTROL\_REG[0] MAC\_PENABLE bit can be set as desired. No effect will occur until IET\_ENABLE is set. This bit enables preemptable packets to be preempted by express traffic but does not preclude packets from being sent to the preempt queue.
  - If verify/response is desired then CPSW\_PN\_IET\_CONTROL\_REG[3] MAC\_LINKFAIL should be cleared by software to enable verify and response packets. Otherwise, MAC\_DISABLEVERIFY bit should be set for forced mode. Verification and response will occur immediately after clearing this bit.
  - Configure the remaining CPSW\_PN\_IET\_CONTROL\_REG register bits as desired.
- Set the IET\_ENABLE bit in the CPSW\_CONTROL\_REG register to enable IET operations.

6. After preemption has been verified, the CPSW\_PN\_IET\_CONTROL\_REG[23-16] MAC\_PREMPT field is written to configure the FIFO priorities to be sent to the preempt queue (the other priorities with cleared bits go to the express queue). The hardware switch for each queue from express to preempt happens only when there are no packets queued on the priority.

#### **12.2.1.4.6.8 Enhanced Scheduled Traffic (EST – P802.1Qbv/D2.2)**

##### **12.2.1.4.6.8.1 Enhanced Scheduled Traffic Overview**

- When enabled and configured, EST allows express queue traffic to be scheduled (placed) on the wire at specific repeatable time intervals.
- EST operates on a repeating time interval generated by the CPTS EST function generator. For example, a 125us repeating time interval can be configured.
- Each Ethernet port has 128 EST fetch commands maximum in the global EST fetch RAM.
- Each 22-bit fetch command consists of a 14-bit fetch count (14 MSB's) and an 8-bit priority fetch allow (8 LSB's) that will be applied for the fetch count time in wireside clocks.
- The configured port fetch commands are executed in sequence, beginning at port address zero each time through the time interval beginning at cycle start.
- EST allows non-scheduled express and preempt queue traffic to be cleared from the wire to ensure that the scheduled traffic is transmitted at the proper time (zero allow).
- EST can be used with or without preemption. The CPSW\_PN\_IET\_CONTROL\_REG[23-16] MAC\_PREMPT value determines whether the priority is enabled on the express or preempt queue. Whether a priority is on the express or preempt queue only effects the wire clear time from an EST operation perspective.
- Software should not move priorities to the preempt queue unless preemption is configured, enabled, and verified allowing preemption to occur.
- Express packet time stamp events can be enabled to assist software in configuring and timing EST operations.

##### **12.2.1.4.6.8.2 Enhanced Scheduled Traffic Fetch RAM**

- The EST fetch RAM is read/writable in the CPSW configuration address space.
- The Ethernet transmit port has 128 locations in the global EST fetch RAM.
  - Ethernet port 1 has EST fetch RAM addresses 0x000-0x07F.
- **One buffer operation:** When CPSW\_PN\_EST\_CONTROL\_REG[0] EST\_ONEBUF is set to 1h, the 128 port locations operate as one buffer. The EST\_BUFACT bit in CPSW\_PN\_FIFO\_STATUS\_REG register is the upper address bit of the port's fetch RAM address indicating whether operation is currently in the upper or lower 64 locations of the port's fetch RAM.
- **Two buffer operation:** When CPSW\_PN\_EST\_CONTROL\_REG[0] EST\_ONEBUF is cleared there are two 64-location buffers with CPSW\_PN\_EST\_CONTROL\_REG[1] EST\_BUFSEL selecting the buffer to be used. When the buffer is switched by changing the CPSW\_PN\_EST\_CONTROL\_REG[1] EST\_BUFSEL value, the actual switch occurs on cycle start. The actual buffer being used is indicated by the EST\_BUFACT bit in CPSW\_PN\_FIFO\_STATUS\_REG. Software should avoid writing the switched out buffer fetch RAM locations until it detects that the actual switch has occurred.
- The first address location in the port's fetch RAM space (location zero) is read at the beginning of each EST time interval (cycle start). Addresses are then read in ascending order for the duration of the interval. The port address zero is then read again at the beginning of the next cycle repeating the time interval packet operations.

##### **12.2.1.4.6.8.3 Enhanced Scheduled Traffic Time Interval**

- Each Ethernet port has an Enhanced Scheduled Traffic Function (ESTF) generator in the CPTS submodule.
- The EST function generator generates the EST time interval as a configured number of CPTS reference clocks (CPTS\_RFT\_CLK).
- The EST function generator rising edge is the cycle start time and the cycle repeats (cycle start occurs) after every time interval.



- The first fetch allowed value is at the port base address zero in the EST fetch RAM and is actually applied 16 wireside clocks after cycle start. The 16 clock cycle delay allows the first fetch value time to be fetched from the EST fetch RAM (prefetch time at cycle start).
- Each successive fetch allow is applied for the associated fetch count thereafter. The minimum non-zero fetch count is 16. The minimum value of 16 guarantees that the next fetch value has time to be fetched before the current fetch count is over. There are 64 maximum fetch values when CPSW\_PN\_EST\_CONTROL\_REG[0] EST\_ONEBUF = 0h, and 128 maximum fetch values when CPSW\_PN\_EST\_CONTROL\_REG[0] EST\_ONEBUF = 1h.
- The next cycle start then causes the fetch to once again start at the port address zero.

#### 12.2.1.4.6.8.4 Enhanced Scheduled Traffic Fetch Values

- The 22-bit fetch value is made up of the 14-bit fetch count and the 8-bit fetch allow.
- The fetch time indicates the number of wireside clocks that the fetch allow will be active.
- The fetch count is in Ethernet wireside clocks which is bytes in Gigabit mode (CPSW\_PN\_MAC\_CONTROL\_REG[7] GIG = 1h) and nibbles in 10/100Mbps mode.
- When a fetch allow bit is set, the corresponding priority is enabled to begin packet transmission on an allowed priority subject to rate limiting. The actual packet transmission on the wire may carry over into the next fetch count and is the reason for the wire clear time in the fetch zero allow.
- When a fetch allow bit is cleared, the corresponding priority is not enabled to transmit for the fetch count time.
- A non-zero fetch allow value with a non-zero fetch count causes the fetch allow value to be applied for the fetch count number of wireside clocks.
- A zero fetch count causes the associated fetch allow to be held for the duration of the cycle (until the next cycle start).
- A zero fetch allow with a non-zero fetch count is intended to clear the wire for a scheduled (timed) express packet in the next fetch. A zero fetch allow indicates that no packet can be started for transmission for the associated fetch count. The associated fetch count must be sufficient to guarantee that the wire is cleared given that a packet on an allowed priority in the previous fetch could have been started on the previous clock and that there is hardware latency in the clear time. The timed packet should be sent on a priority that is enabled in the next fetch but disabled in the current zero allow fetch. The fetch allow previous to a zero allow should have only preempt priorities enabled or only express priorities enabled but not both.
- The number of clocks required to clear the wire varies depending Ethernet wire speed and on whether express or preempt priorities were allowed in the previous fetch command.

#### 12.2.1.4.6.8.5 Enhanced Scheduled Traffic Packet Fill

Packet fill can be enabled (CPSW\_PN\_EST\_CONTROL\_REG[8] EST\_FILL\_EN = 1h) to occur in the fetch count time associated with a fetched zero allow. The intention with fill is that a smaller packet on a non-timed priority might be able to be inserted on the wire during the wire clear time which would increase wire utilization. Fill is intended to be used with either only express priorities in the previous non-zero fetched allow, or only preempt priorities in the previous non-zero fetched allow but not both. Fill must be configured to ensure that any fill packet does not conflict with the timed express packet allowed in the next fetch.

- **Express fill:**
  1. CPSW\_PN\_EST\_CONTROL\_REG[8] EST\_FILL\_EN is set and
  2. A fetch contains a non-zero fetch count with a zero fetch allow, and
  3. The previous allow contained only express priorities.
  4. The fetch count loaded should be at least TBD clocks and the CPSW\_PN\_EST\_CONTROL\_REG[25-16] EST\_FILL\_MARGIN should be at least TBD clocks in Gigabit mode to ensure that the wire is cleared for the timed express packet allowed in the next fetch.
  5. The fetch count loaded should be at least TBD clocks and the CPSW\_PN\_EST\_CONTROL\_REG[25-16] EST\_FILL\_MARGIN should be at least TBD clocks in 10/100Mbps mode to ensure that the wire is cleared for the timed express packet allowed in the next fetch.
- **Preempt fill:**
  1. CPSW\_PN\_EST\_CONTROL\_REG[8] EST\_FILL\_EN is set and

2. A fetch contains a non-zero fetch count with a zero fetch allow, and
  3. The previous allow contained only preempt priorities
  4. The fetch count loaded should be at least TBD clocks in Gigabit mode to ensure that the wire is cleared for the timed express packet allowed in the next fetch.
  5. The fetch count loaded should be at least TBD clocks in 10/100Mbps mode to ensure that the wire is cleared for the timed express packet allowed in the next fetch.
- **Express non-fill:**
    1. CPSW\_PN\_EST\_CONTROL\_REG[8] EST\_FILL\_EN is clear and
    2. The previous allow contained only express priorities.
    3. The fetch count loaded should be at least TBD clocks in Gigabit mode to ensure that the wire is cleared for the timed express packet allowed in the next fetch.
    4. The fetch count loaded should be at least TBD clocks in 10/100Mbps mode to ensure that the wire is cleared for the timed express packet allowed in the next fetch.
  - **Preempt non-fill:**
    1. CPSW\_PN\_EST\_CONTROL\_REG[8] EST\_FILL\_EN is clear and
    2. The previous allow contained only preempt priorities.
    3. The fetch count loaded should be at least TBD clocks in Gigabit mode to ensure that the wire is cleared for the timed express packet allowed in the next fetch.
    4. The fetch count loaded should be at least TBD clocks in 10/100Mbps mode to ensure that the wire is cleared for the timed express packet allowed in the next fetch.

#### 12.2.1.4.6.8.6 Enhanced Scheduled Traffic Time Stamp

The EST can be configured to generate CPTS timestamp events for selected express traffic. The EST timestamp events use the CPTS host event type (CPSW\_CPTS\_EVENT\_1\_REG[23-20] EVENT\_TYPE = 7 decimal). The EST timestamps will not override host sent timestamps for packets that were sent from the host with an enabled host timestamp.

- EST Events (host events) contain the below information:
  - Time Stamp of the selected express packet.
  - The event CPSW\_CPTS\_EVENT\_1\_REG[28-24] PORT\_NUMBER indicates the transmit port number.
  - The event CPSW\_CPTS\_EVENT\_1\_REG[23-20] EVENT\_TYPE is decimal 7 (host event).
  - The event CPSW\_CPTS\_EVENT\_1\_REG[23-20] MESSAGE\_TYPE indicates the packet transmit hardware switch priority.
  - The event CPSW\_CPTS\_EVENT\_1\_REG[15-0] SEQUENCE\_ID upper nibble indicates the packet receive port number.
  - The event CPSW\_CPTS\_EVENT\_1\_REG[15-0] SEQUENCE\_ID lower byte indicates the sequence number of the express packet in numerical order. The first event is event one, the second is event two and so on. The sequence ID rolls over to zero after 0xFF (8-bits).
  - The event domain is the value from the CPSW\_EST\_TS\_DOMAIN\_REG[7-0] EST\_TS\_DOMAIN register.
- When CPSW\_PN\_EST\_CONTROL\_REG[2] EST\_TS\_EN is set, timestamp events will be generated on selected express traffic.
- When CPSW\_PN\_EST\_CONTROL\_REG[3] EST\_TS\_FIRST is also set, events will be generated only on the first express packet in each time interval. If CPSW\_PN\_EST\_CONTROL\_REG[4] EST\_TS\_ONEPRI is also set then the event will only be on the first CPSW\_PN\_EST\_CONTROL\_REG[7-5] EST\_TS\_PRI express packet in the time interval. If CPSW\_PN\_EST\_CONTROL\_REG[4] EST\_TS\_ONEPRI is clear then the event will be generated on the first express packet in the time interval on any priority.
- When CPSW\_PN\_EST\_CONTROL\_REG[3] EST\_TS\_FIRST is clear, events will be generated on every express packet. If CPSW\_PN\_EST\_CONTROL\_REG[4] EST\_TS\_ONEPRI is set then the event will be generated on every CPSW\_PN\_EST\_CONTROL\_REG[7-5] EST\_TS\_PRI express packet. If CPSW\_PN\_EST\_CONTROL\_REG[4] EST\_TS\_ONEPRI is clear then event will be generated on every express packet on any priority.

### 12.2.1.4.6.8.7 Enhanced Scheduled Traffic Packets Per Priority

The number of packets allowed in a transmit FIFO priority can be selected by writing a non-zero value to CPSW\_P0\_RX\_PKTS\_PRI\_REG register (packet priority 0 to 7). Then port 0 receive gap should then be enabled by setting the corresponding priority through CPSW\_P0\_RX\_GAP\_REG[7-0] RX\_GAP\_EN bit field. The receive gap allows a packet to land in the transmit FIFO before another packet is allowed in which guarantees that only the selected number (max) of packets is allowed in on the specified priority. If the receive gap is not enabled, then there might be one or two more packets allowed in on the priority/thread than the packets per priority value has selected (through CPSW\_P0\_RX\_PKTS\_PRI\_REG register).

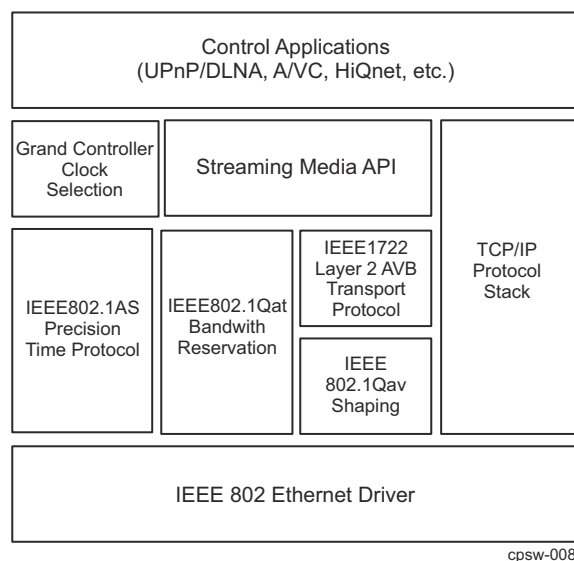
### 12.2.1.4.6.9 Audio Video Bridging

Audio Video Bridging is an ongoing project of IEEE 802.1 concerned with enabling low-latency streaming of time-sensitive audiovisual data over networks. Devices are designated as talkers (transmitters), bridges, or listeners (receivers). It is suggested that the maximum latency could be 2 ms over 7 hops for Class A devices and 20 ms over 7 hops for Class B devices. A hop is essentially a single local area network stage in the journey of a packet. Every time a bridge is encountered between one network section and another a hop is involved. One of the performance goals is that AVB streams will not use more than 75 percent of a link's bandwidth, leaving the remaining capacity for non-AVB streams.

The goal of developing AVB is simply--extend Ethernet's data-networking capabilities to the realm of reliable real-time audio/video networking.

An "Audio Video Bridging" network is one that implements a set of protocols being developed by the IEEE 802.1 Audio/Video Bridging Task Group. There are four primary differences between the proposed Audio Video Bridging architecture and existing 802 architectures (from now on the term "AVB" will be used instead of "Audio Video Bridging"):

1. Precise synchronization - IEEE 802.1AS: "*Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks*." a.k.a Precision Time Protocol (PTP).
2. Traffic shaping for media streams - IEEE 802.1Qav: "*Virtual Bridged Local Area Networks: Forwarding and Queuing for Time-Sensitive Streams*."
3. Admission controls - IEEE 802.1Qat: "*Virtual Bridged Local Area Networks - Amendment 9: Stream Reservation Protocol (SRP)*."
4. Identification of non-participating devices - IEEE 802.1BA: "Audio/Video Bridging (AVB) Systems"



**Figure 12-136. The Network Static with AVB**

The following sections describe the media transport protocols that work within the AVB framework.



**12.2.1.4.6.9.1 IEEE 802.1AS: Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks (Precision Time Protocol (PTP))**

The protocol defined by 802.1AS automatically selects a device to be the controller clock, and then distributes this clock throughout the bridged LAN / IP subnet to all other network devices using link-specific transmit/receive time-stamping. However, we only use a two-step solution only on transmit. That is, we do not modify a packet with the timestamp on the way out. The timestamp packet is sent out and then a separate message with the timestamp is sent by the host afterward. Receive can be one or two step.

---

**Note**

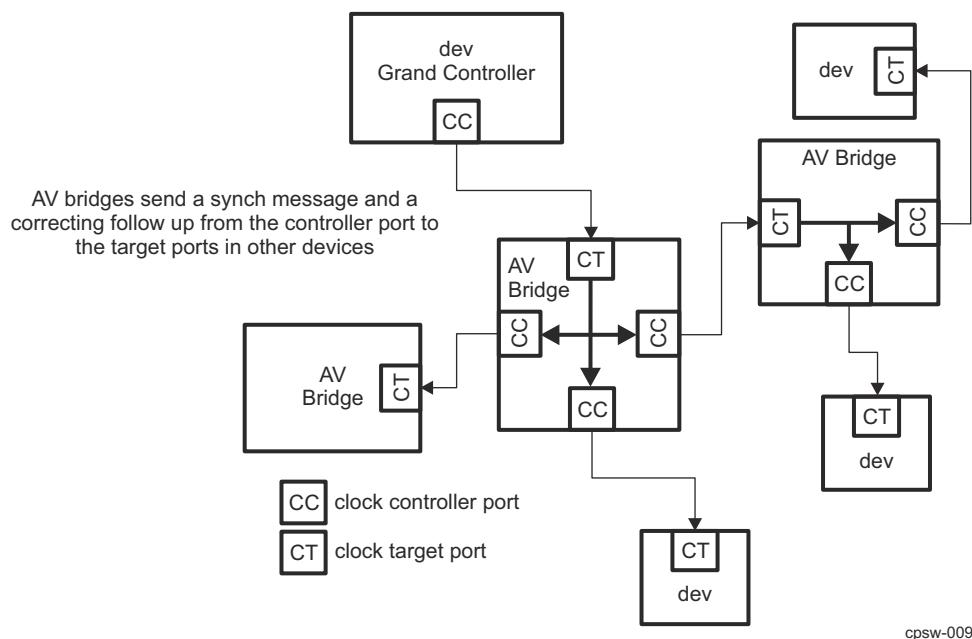
The 802.1AS-distributed clock is not used as a media clock. Rather, the shared 802.1AS clock reference is used to regenerate the media clock at the listener/renderer. Such a reference removes the need to force the latency of the network to be constant, or compute long running averages in order to estimate the actual media rate of the transmitter in the presence of substantial network jitter. IEEE 802.1AS is based on the ratified IEEE 1588 standard.

---

Based on IEEE 1588:2002, A PTP devices exchange standard Ethernet messages that synchronize network nodes to a common time reference by defining clock controller selection and negotiation algorithms, link delay measurement and compensation, and clock rate matching and adjustment mechanisms.

Designed as a simplified profile of IEEE 1588, a primary difference between 1588 and IEEE 802.1AS is that PTP is a layer 2-in other words, a non-IP routable protocol. Like IEEE 1588, PTP defines an automatic method for negotiating the network clock controller, the Best Controller Clock Algorithm (BCCA). PTP nodes can be assigned one of eight priority levels, presumably based on clock quality. BMCA defines the underlying negotiation and signaling mechanism whose purpose is to identify the AVB LAN Grandcontroller. Once a Grandcontroller has been selected, synchronization automatically begins.

At the core of 802.1AS synchronization is time-stamping. In short, during PTP message ingress/egress from the 802.1AS-capable MAC, the PTP Ether type triggers the sampling of the value of a local real-time counter (RTC). Target nodes compare the value of their RTC against the PTP Grandcontroller and, by use of link delay measurement and compensation techniques, match their RTC value to the time of the AVB LAN PTP domain. After network time throughout the AVB LAN has converged, periodic SYNC and FOLLOW\_UP messages provide the information that enables the PTP rate matching adjustment algorithms. The result is all PTP nodes are then synchronized to the same "Wall Clock" time. PTP assures 1- $\mu$ s accuracy over seven network hops.



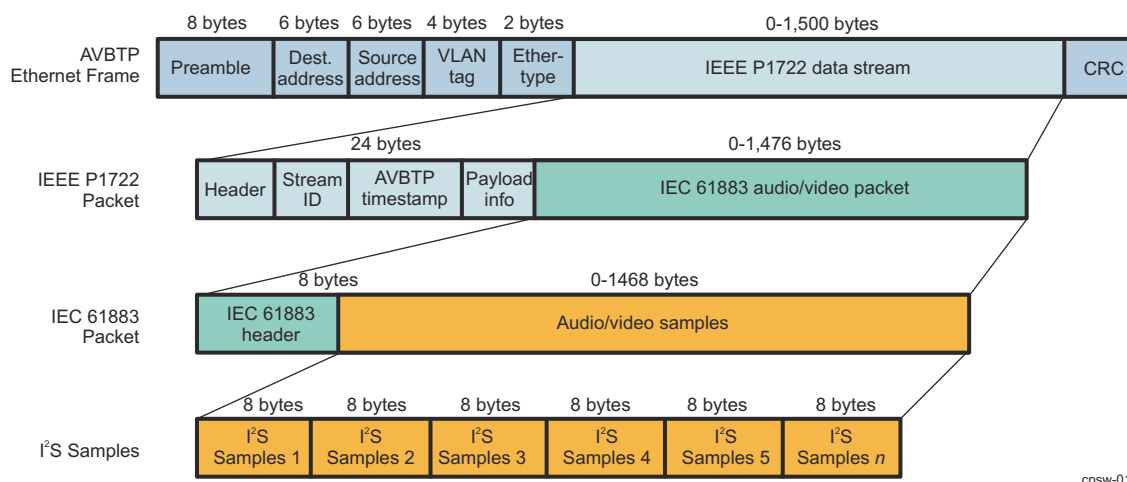
**Figure 12-137. AVB Network & PTP Clock Entities**

The media transport protocols that work within the AVB framework are:

#### 12.2.1.4.6.9.1.1 IEEE 1722: "Layer 2 Transport Protocol for Time-Sensitive Streams"

AVBTP or 1722 sits above the IEEE 802.1 AVB plumbing and below the application layer. It acts as the conduit between an Ethernet MAC and a streaming application. AVBTP abstracts the underlying network transmission channel to enable the virtual connection of distributed audio and video CODECs over reliable Ethernet networks. A complete AVBTP Ethernet packet is shown in Figure 12-138 and illustrates how IEC 61883-6 AM824 uncompressed audio samples are encapsulated in an Ethernet frame.

#### IEEE 1722 Packet Construction



**Figure 12-138. IEEE 1722 Packets**

1722 or AVBTP Presentation Time and Synchronization:

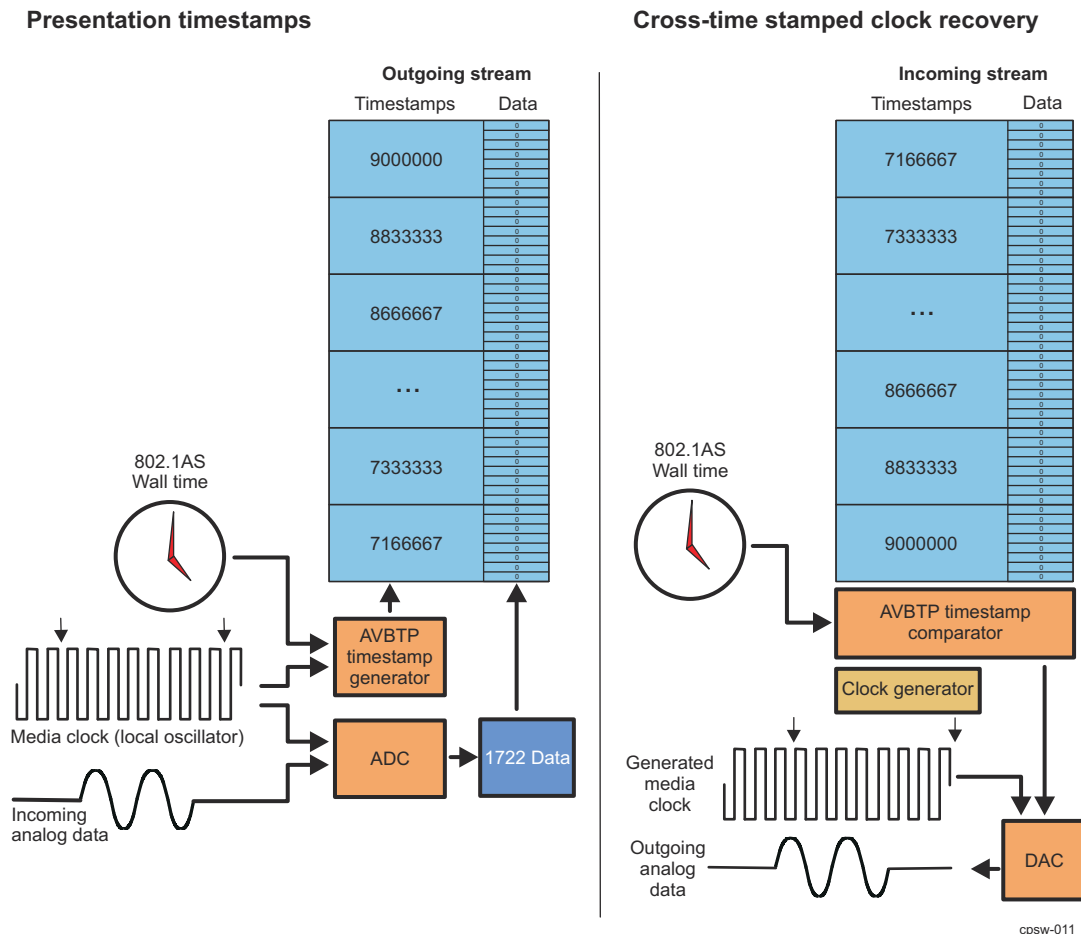
Synchronization in an AVB network starts with the Precision Time Protocol but ends with synchronized media clocks. PTP is responsible for synchronizing all nodes in an AVB network to identical wall clock time; not for

synchronizing media clocks. In other words, PTP does not actually transport synchronized media clocks but instead provides a low-level building block crucial for managing a distributed media synchronization system.

A crucial benefit of this approach is coexistence of multiple, independent media clock domains on an AVB network. Unrelated audio and video streams can simultaneously exist in the same LAN.

#### 12.2.1.4.6.9.1.1 Cross-timestamping and Presentation Timestamps

AVBTP assumes that AVB node media clocks are clocked by free-running oscillators. It is also assumed that the node's internal concept of wall clock time has been synchronized to the PTP Grandcontroller. AVBTP media clock sources embed "AVBTP Presentation Timestamps" in AVBTP streaming packets. [Figure 12-139](#) illustrates the relationship between PTP network time and AVBTP Presentation Timestamps.



**Figure 12-139. Cross Time Stamping and Presentation Timestamps**

#### 12.2.1.4.6.9.1.2 IEEE 1733: Extends RTCP for RTP Streaming over AVB-supported Networks

This standard specifies the protocol, data encapsulations, connection management and presentation time procedures used to ensure interoperability between audio and video based end stations that use standard networking services provided by all IEEE 802 networks meeting QoS requirements for time-sensitive applications by leveraging the Real-time Transport Protocol (RTP) family of protocols and IEEE 802.1 Audio/Video Bridging (AVB) protocols.

#### 12.2.1.4.6.9.2 IEEE 802.1Qav: "Virtual Bridged Local Area Networks: Forwarding and Queuing for Time-Sensitive Streams"

This standard allows bridges to provide guarantees for time-sensitive (that is, bounded latency and delivery variation), loss-sensitive real-time audio video (AV) data transmission (AV traffic). It specifies per priority ingress

metering, priority regeneration, and timing-aware queue draining algorithms. This standard uses the timing derived from IEEE 802.1AS. Virtual Local Area Network (VLAN) tag encoded priority values are allocated, in aggregate, to segregate frames among controlled and non-controlled queues, allowing simultaneous support of both AV traffic and other bridged traffic over and between wired and wireless Local Area Networks (LANs).

Such a guarantee in bandwidth is provided by two functional entities:

- A registration protocol, which registers the service and its maximum network utilization with a device or switch (IEEE 802.1Qat: "Virtual Bridged Local Area Networks - Amendment 9: Stream Reservation Protocol (SRP)")
- A hardware bandwidth management service.
  - Receive policing
  - Transmit rate control.

### End Station Behavior

In order for an end station to successfully participate in the transmission and reception of time-sensitive streams, it is necessary for their behavior to be compatible with the operation of the forwarding and queuing mechanisms employed in bridges.

The requirements for end stations that participate as "talkers" i.e., sources of time-sensitive streams are different from the requirements that apply to "listeners", the destination station(s) for the streams.

### Talker Behavior

In order for Talker-originated data streams to make use of the credit-based shaper behavior in Bridges, it is a requirement for a Talker to use the priorities that the Bridges in the network recognize as being associated with SR classes exclusively for transmitting stream data.

It is also necessary for the Talker and the Bridges in the path to the Listener(s), to have a common view of the bandwidth required in order to transmit the Talker's streams, and for that bandwidth to be reserved along the path to the Listener(s). This latter requirement can be met by means of stream reservation mechanisms, such as defined in SRP, or by other management means.

End stations that are Talkers shall exhibit transmission behavior for frames that are part of "time-sensitive streams" that is consistent with the operation of the credit-based shaper algorithm, both in terms of the way they transmit frames that are part of an individual data stream, and in terms of the way they transmit stream data frames from a Port.

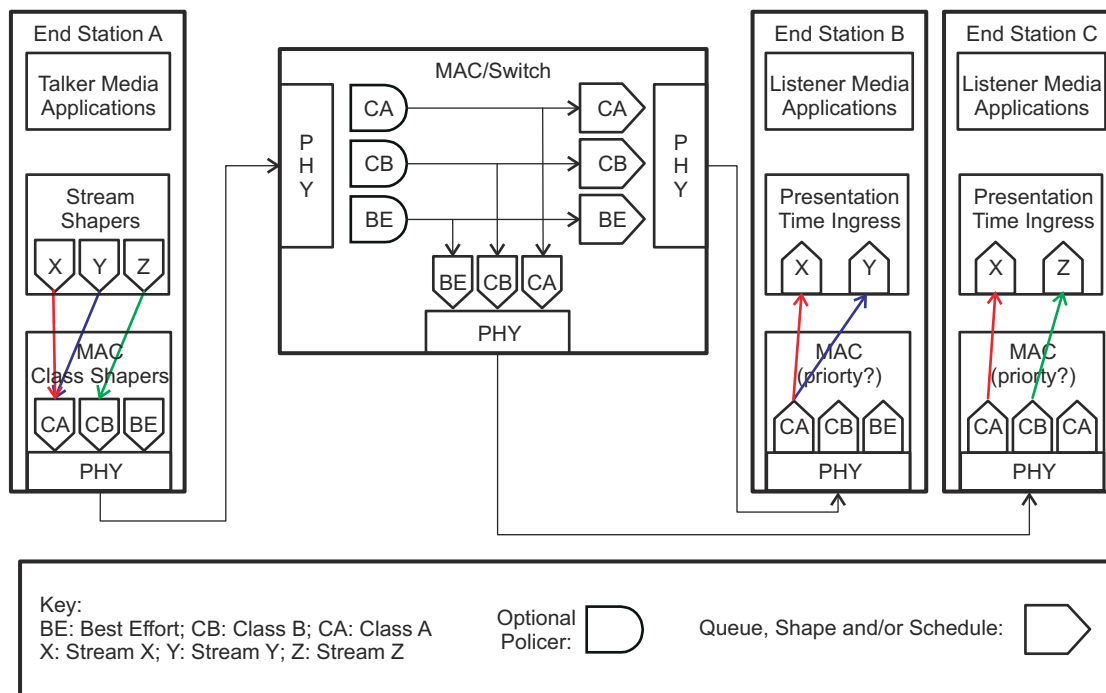
In effect, the queuing model for a Talker Port (and a Listener port), and for given priorities, can be considered to look like [Figure 12-140](#).

### Listener Behavior

The primary requirement for a listener station is that it is capable of buffering the amount of data that could be transmitted for a stream during a time period equivalent to the accumulated maximum jitter that could be experienced by stream data frames in transmission between Talker and Listener.

From the point of view of the specification of the forwarding and queuing requirements for time-sensitive streams, it is assumed that the listener will assess the buffering required for a stream as part of the stream bandwidth reservation mechanisms employed by the implementation.

The credit-based shaper's operation details are beyond the scope of this document.



cpsw-012

**Figure 12-140. AV Stream Queuing/Policing**

#### 12.2.1.4.6.9.2.1 Configuring the Device for 802.1Qav Operation

There is no dedicated register-set to be configured for the time-sensitive stream handling. The list of functional features of CPSW that will have to be configured are:

- DESCRIPTORS and CHANNEL CONFIGURATIONS:
  - CPPI TX and RX descriptors
  - VLAN and Priority tags

**Table 12-177. Example of TX Configuration**

TX DMA CHANNEL	Packet Priority	Switch Queue Priority
7	7	3
6	5	2
5	3	1
4	1	0

**Table 12-178. Example of RX Configuration**

RX DMA CHANNEL	Packet Priority	Switch Queue Priority
0	7	0
0	5	0
0	3	0
0	1	0

- ALE Configuration:
  - ALE in VLAN-ware mode, Non-ALE in bypass mode.

#### 12.2.1.4.6.10 Ethernet MAC Sliver

The Ethernet port peripheral is compliant to the IEEE Std 802.3 Specification. Half-duplex mode is supported in 10/100 Mbps mode, but not in 1000 Mbps (gigabit) mode.

## Features:

- Synchronous 10/100/1000 Mbit operation
- RMII/RGMII Interface
- Hardware Error handling including CRC
- Full-Duplex Gigabit operation (half-duplex gigabit is not supported)
- EtherStats and 802.3Stats RMON statistics gathering support for external statistics collection module
- Transmit CRC generation selectable on a per channel basis
- Emulation Support
- VLAN Aware Mode Support
- Hardware flow control
- Programmable Inter Packet Gap (IPG).

**12.2.1.4.6.10.1****12.2.1.4.6.10.1.1****12.2.1.4.6.10.1.1.1 CRC Insertion**

The MAC generates and appends a 32-bit Ethernet CRC onto the transmitted data, if the transmit packet header PASS\_CRC bit is 0h. For the Ethernet port generated CRC case, a CRC at the end of the input packet data is not allowed.

If the header word PASS\_CRC bit is set, then the last four bytes of the TX data are transmitted as the frame CRC. The four CRC data bytes should be the last four bytes of the frame and should be included in the packet byte count value. The MAC performs no error checking on the outgoing CRC when the PASS\_CRC bit is set.

**12.2.1.4.6.10.1.1.2 MTXER**

The MTXER signal is only used for EEE. If an underflow condition occurs on a transmitted frame, the frame CRC will be inverted to indicate the error to the network. Underflow is a hardware error.

**12.2.1.4.6.10.1.1.3 Adaptive Performance Optimization (APO)**

The Ethernet MAC port incorporates Adaptive Performance Optimization (APO) logic that may be enabled by setting the TX\_PACE bit in the CPSW\_PN\_MAC\_CONTROL\_REG register. Transmission pacing to enhance performance is enabled when set. Adaptive performance pacing introduces delays into the normal transmission of frames, delaying transmission attempts between stations, reducing the probability of collisions occurring during heavy traffic (as indicated by frame deferrals and collisions) thereby increasing the chance of successful transmission.

When a frame is deferred, suffers a single collision, multiple collisions or excessive collisions, the pacing counter is loaded with an initial value of 31. When a frame is transmitted successfully (without experiencing a deferral, single collision, multiple collision or excessive collision) the pacing counter is decremented by one, down to zero.

With pacing enabled, a new frame is permitted to immediately (after one IPG) attempt transmission only if the pacing counter is zero. If the pacing counter is non zero, the frame is delayed by the pacing delay, a delay of approximately four inter-packet gap delays. APO only affects the IPG preceding the first attempt at transmitting a frame. It does not affect the back-off algorithm for re-transmitted frames.

**12.2.1.4.6.10.1.1.4 Inter-Packet-Gap Enforcement**

The measurement reference for the IPG of 96-bit times is changed depending on frame traffic conditions. If a frame is successfully transmitted without collision, and MCRS is de-asserted within approximately 48-bit times of MTXEN being de-asserted, then 96-bit times is measured from MTXEN. If the frame suffered a collision, or if MCRS is not de-asserted until more than approximately 48-bit times after MTXEN is de-asserted, then 96-bit times (approximately, but not less) is measured from MCRS.

The Ethernet port transmit inter-packet gap (IPG) may be shortened by eight bit times when short gap is enabled and triggered. Setting the [10] TX\_SHORT\_GAP\_ENABLE bit in the CPSW\_PN\_MAC\_CONTROL\_REG register enables the gap to be shortened when triggered. The condition is triggered when the ports associated transmit packet FIFO has a user defined number of FIFO blocks used. The associated transmit FIFO blocks used value

determines if the gap is shortened, and so on. The CPSW\_GAP\_THRESH\_REG register value determines the short gap threshold. If the FIFO blocks used is greater than or equal to the GAP\_THRESH value then short gap is triggered.

#### **12.2.1.4.6.10.1.1.5 Back Off**

The Gigabit Ethernet Mac Sliver implements the 802.3 binary exponential back-off algorithm.

#### **12.2.1.4.6.10.1.1.6 Programmable Transmit Inter-Packet Gap**

The transmit inter-packet gap (IPG) is programmable through the CPSW\_PN\_MAC\_CONTROL\_REG register. The default value is decimal 12. The transmit IPG may be increased to the maximum value of 1FFh. Increasing the IPG is not compatible with transmit pacing. The short gap feature will override the increased gap value, so the short gap feature may not be compatible with an increased IPG.

#### **12.2.1.4.6.10.1.1.7 Speed, Duplex and Pause Frame Support Negotiation**

The Ethernet port can operate in half duplex or full duplex in 10/100 Mbit modes, and can operate in full duplex only in 1000 Mbit mode. Pause frame support is included in 10/100/1000 Mbit modes as configured by the host.

#### **12.2.1.4.6.10.2 RMII Interface**

The CPRMII peripheral is compliant to the RMII specification document.

##### **12.2.1.4.6.10.2.1 Features**

- Source Synchronous 10/100 Mbit operation
- Full and Half Duplex support

##### **12.2.1.4.6.10.2.2 RMII Receive (RX)**

The CPRMII receive (RX) interface converts the input data from the external RMII PHY (or switch) into the required MII (CPGMAC) signals. The carrier sense and collision signals are determined from the RMII input data stream and transmit inputs as defined in the RMII specification.

An asserted RMII\_RXER on any di-bit in the received packet will cause an MII\_RXER assertion to the CPGMAC during the packet. In 10Mbps mode, the error is not required to be duplicated on 10 successive clocks. Any di-bit which has an asserted RMII\_RXER during any of the 10 replications of the data will cause the error to be propagated.

Any received packet that ends with an improper nibble boundary aligned RMII\_CRS\_DV toggle will issue an MII\_RXER during the packet to the CPGMAC. Also, a change in speed or duplex mode during packet operations will cause packet corruption.

The CPRMII can accept receive packets with shortened preambles, but 0x55 followed by a 0x5D is the shortest preamble that will be recognized (1 preamble byte with the start of frame byte). At least one byte of preamble with the start of frame indicator is required to begin a packet. An asserted RMII\_CRS\_DV without at least a single correct preamble byte followed by the start of frame indicator will be ignored.

##### **12.2.1.4.6.10.2.3 RMII Transmit (TX)**

The CPRMII transmit (TX) interface converts the CPGMAC MII input data into the RMII transmit format. The data is then output to the external RMII PHY.

The CPGMAC does not source the transmit error (MII\_TXERR) signal. Any transmit frame from the CPGMAC with an error (underrun) will be indicated as an error by an error CRC. Transmit error is assumed to be de-asserted at all times and is not an input into the CPRMII module. Zeroes are output on RMII\_TXD[1:0] for each clock that RMII\_TXEN is de-asserted.

#### **12.2.1.4.6.10.3 RGMII Interface**

The CPRGMII peripheral is compliant to the RGMII specification document.

##### **12.2.1.4.6.10.3.1 Features**

- Supports 1000/100/10 Mbps speed



- Full and Half Duplex support (CPGMAC supports only Full duplex in Gigabit mode).
- MII mode support
- Energy Efficient Ethernet Support

#### 12.2.1.4.6.10.3.2 RGMII Receive (RX)

The CPRGMII receive (RX) interface converts the source synchronous DDR input data from the external RGMII PHY into the required G/MII (CPGMAC) signals.

#### 12.2.1.4.6.10.3.3 In-Band Mode of Operation

The CPRGMII is operating in the in-band mode of operation when the RGMII\_RX\_INBAND input is asserted. RGMII\_RX\_INPUT is asserted by configuring the EXT\_EN bit to 1h of the CPSW\_PN\_MAC\_CONTROL\_REG register. The link status, duplexity, and speed are determined from the RGMII input data stream RXD[3:0] when RX\_CTL is deasserted, as defined in the RGMII specification. The PHY might need to be configured beforehand to output in-band data. The in-band data is indicated as shown in [Table 12-179](#).

**Table 12-179. In-Band Data**

RXD3	RXD[2:1]		RXD0
Duplex status:	Link Speed:	RXC_CLK Speed:	Link Status:
0h: half-duplex	0h: 10-Mbps mode	2.5 MHz	0h: Link is down
1h: full-duplex	1h: 100-Mbps mode	25 MHz	1h: Link is up
	2h: 1000-Mbps mode	125 MHz	
	3h: Reserved	Reserved	

#### 12.2.1.4.6.10.3.4 Forced Mode of Operation

The CPRGMII is operating in the forced mode of operation when the RGMII\_RX\_INBAND input is deasserted by setting to 0h bit EXT\_EN of the CPSW\_PN\_MAC\_CONTROL\_REG register. In the forced mode of operation, the in-band data is ignored if present. The link status is forced high, and the duplexity and speed are determined from the CPSW\_PN\_MAC\_CONTROL\_REG[0] FULLDUPLEX and CPSW\_PN\_MAC\_CONTROL\_REG[7] GIG bits. If bit [7] GIG = 1h, then CPRGMII is operating in Gigabit mode. If bit [7] GIG is cleared (0h), then CPRGMII is operating in 100 Mbps mode.

#### 12.2.1.4.6.10.3.5 RGMII Transmit (TX)

The CPRGMII transmit (TX) interface converts the CPMAC G/MII input data into the DDR RGMII format. The DDR data is then output to the external PHY.

The CPMAC does not source the transmit error (TXERR) signal. Any transmit frame from the CPMAC with an error (underrun) will be indicated as an error by an error CRC. Transmit error is assumed to be deasserted at all times and is not an input into the CPRGMII module.

In 10/100 Mbps mode, the TXD[7:0] data bus uses only the lower nibble. The CPRGMII will output the lower nibble twice in 10/100 Mbps mode to avoid unnecessary signal switching.

Packets will be precluded from transmission through the CPRGMII module for 4096 transmit clocks after the rising edge of RGMII\_LINK. Packet transmission will begin on the first TX\_CTL rising edge after the 4096 transmit clock count has expired.

#### 12.2.1.4.6.10.4 Frame Classification

Received frames are proper (good) frames if they are between 64 and CPSW\_P0\_RX\_MAXLEN\_REG[13:0] RX\_MAXLEN in length (inclusive) and contain no errors (code/align/CRC).

Received frames are long frames if their frame count exceeds the value in the CPSW\_P0\_RX\_MAXLEN\_REG/ CPSW\_PN\_RX\_MAXLEN\_REG register. The register reset (default) value is 1518 (decimal). Long received frames are either oversized or jabber frames. Long frames with no errors are oversized frames. Long frames with CRC, code, or alignment errors are jabber frames.



Received frames are short frames if their frame count is less than 64 bytes. Short frames that contain no errors are undersized frames. Short frames with CRC, code, or alignment errors are fragment frames. If RX\_CSF\_EN bit in CPSW\_PN\_MAC\_CONTROL\_REG is set to 1h, undersized frames from 33 to 63 bytes will be forwarded only to the host on a best effort basis (meaning that the ALE may or may not be able to keep up with the packet rate and the short packet may be dropped due to bandwidth limitations). If RX\_CSF\_EN and RX\_CEF\_EN in CPSW\_PN\_MAC\_CONTROL\_REG are set, fragment frames from 33 to 63 bytes will also be forwarded only to the host on a best effort basis. Ethernet port received frames shorter than 33 bytes are dropped in all cases.

A received long packet will always contain RX\_MAXLEN number of bytes transferred to memory (if CPSW\_PN\_MAC\_CONTROL\_REG[22]RX\_CEF\_EN = 1h). An example with RX\_MAXLEN = 1518 is:

- If the frame length is 1518, then the packet is not a long packet and there will be 1518 bytes transferred to memory.
- If the frame length is 1519, there will be 1518 bytes transferred to memory. The last three bytes will be the first three CRC bytes.
- If the frame length is 1520, there will be 1518 bytes transferred to memory. The last two bytes will be the first two CRC bytes.
- If the frame length is 1521, there will be 1518 bytes transferred to memory. The last byte will be the first CRC byte.

If the frame length is 1522, there will be 1518 bytes transferred to memory. The last byte will be the last data byte.

#### 12.2.1.4.6.10.5 Receive FIFO Architecture

This section describes the architecture of the Ethernet port's receive FIFOs. Internal to the Gigabit Ethernet switch, all Ethernet ports have an identical associated packet FIFO. Each transmit packet FIFO contains eight logical transmit queues (priority 0 through 7 with 7 the highest priority). Each transmit FIFO memory contains 81,920 bytes total organized as 2560 by 256-bit words. Each FIFO also contains a single memory for the receive queue. Each receive FIFO memory contains a total of 32768 bytes total organized as 1024 by 256-bit words.

#### 12.2.1.4.6.11 Embedded Memories

**Table 12-180. Embedded Memories**

Memory Type Description	Number of Instances	
Single-port 3072-word × 64 RAM	1	(Combined FIFO RAM)
Single-port 128-word × 28-bit RAM	1	1 (EST)

#### 12.2.1.4.6.12 Memory Error Detection and Correction

The CPSW error detection and correction logic uses the ECC Aggregator Module.

The ECC CPSW\_ECC\_VECTOR register is used to select which ECC RAM's status and control registers are currently being read or written as shown in [Table 12-181](#). The CPSW FIFO RAMs implement ECC only on packet headers. The packet data is protected by Ethernet CRC. The ALE and EST RAMs have complete ECC as normal.

**Table 12-181. ECC RAM to CPSW RAM Mapping**

ECC RAM Number	CPSW RAM
0	ALE RAM
1	Port 0 FIFO RAM

##### 12.2.1.4.6.12.1 Packet Header ECC

Only packet headers bits are protected by ECC in the FIFO RAMs. The ECC\_ERR\_CTRL1[31-0] ECC\_ROW bit is used to activate a row address where force single-bit error or force double-bit error needs to be applied. ECC\_ERR\_CTRL2 [15-0] ECC\_BIT1 is implemented to determine which bit of the header is flipped for an SEC error when the ECC\_CRC\_MODE bit is cleared in the CPSW\_CONTROL\_REG register. The ECC status registers return the RAM row address where the single or double-bit error has occurred (ECC\_ERR\_STAT2[31-0] ECC\_ROW) along with the bit position in the RAM data that is in error (ECC\_ERR\_STAT1[31-16] ECC\_BIT1 value). Forcing double-bit errors in testing can cause indeterminate operation if multiple used packet header bits are flipped given that only single-bit errors are fixed by the ECC logic. Header bits 207 down to 200 are not currently used in the CPSW and may be used to test double bit errors without the possibility of requiring a reset for the switch to recover from the double bit error. No header bits are flipped when ECC\_CRC\_MODE is set to 1h. Either the RX\_ECC\_ERR\_EN (enable receive ECC error operations) or the TX\_ECC\_ERR\_EN (enable transmit ECC error operations) bits must be set in the CPSW\_P0\_CONTROL\_REG register to test ECC header errors.

The header ECC code is stored in bits 255 down to 208. If any bit is flipped in the ECC code, the flipped bit will be corrected, but the index of the flipped bit will be reported as bit zero. This implies that when the aggregator reports that there is a SEC on bit 0, it can mean two things: either SEC on data bit 0 or SEC somewhere inside the ECC code. Any packet header with ECC error issues a pulse interrupt (ECC\_PULSE\_INTR) as does an ALE RAM ECC error.

##### 12.2.1.4.6.12.2 Packet Protect CRC

Each packet received without error is passed through the CPSW\_3G memories with a generated Ethernet protect CRC. The protect CRC is checked on egress for correctness and removed. If the CRC is correct (no RAM bit errors), then the packet is output with the selected port CRC type. If a protect CRC error is detected on host egress then the TXST\_DROP signal will be asserted so that the packet is dropped to the host. If a protect CRC error is detected on Ethernet egress then the egress CRC will be generated on the packet and at least one byte of the CRC will be inverted on output. CRC memory protect errors do not assert the ECC\_PULSE\_INTR signal. CRC memory protect errors are counted in the associated port statistics registers and issue an interrupt on STAT\_PEND\_INTR if any CRC memory protect error occurs (and the statistics for that port are enabled). When the ECC\_CRC\_MODE bit in the CPSW\_CONTROL\_REG register is set, the ECC\_ERR\_CTRL2 [15-0] ECC\_BIT1 bit field will flip the associated column bit in any FIFO memory read operation, inducing a CRC protect error when the protect CRC is checked. No header bits are flipped when ECC\_CRC\_MODE is set. Either the RX\_ECC\_ERR\_EN or the TX\_ECC\_ERR\_EN bits must be set in the CPSW\_P0\_CONTROL\_REG register to test packet CRC errors.

##### 12.2.1.4.6.12.3 Aggregator RAM Control

The ECC logic for each FIFO RAM (receive and transmit) is divided into eight separate ECC encoders/decoders that encode/decode 26-bits of data each. Each of the 8 encoders (0 to 7) generates 6-bits of ECC code (48 code bits total), and each of the eight decoders (0 to 7) checks 6-bits of ECC code across the 26-bits of data (208 data bits total). The 48-bits of ECC code are passed through the RAM in the upper 48 unused bits in the header word.

The header data bits and ECC code bits are shown in [Table 12-182](#). The [15-0] ECC\_BIT1 value returned on error is a 16-bit value that is the concatenation of 5 bits of zero, 3 bits of the encoder/decoder number (0 to 7), 3 bits of zero, and 5 bits of index into the indicated 26-bit encoder/decoder.

For example, an ECC\_BIT1 value of 0x0308 is bit 8 of encoder/decoder 3, which is header bit 86 (that is,  $(26 \times 3) + 8$ ).

**Table 12-182. ECC Submodule Header Data Bit to Encoder/Decoder Mapping**

Header Data Bits	Encoder/Decoder
25:0	Encoder/Decoder 0 Data
51:26	Encoder/Decoder 1 Data
77:52	Encoder/Decoder 2 Data
103:78	Encoder/Decoder 3 Data
129:104	Encoder/Decoder 4 Data
155:130	Encoder/Decoder 5 Data
181:156	Encoder/Decoder 6 Data
207:182	Encoder/Decoder 7 Data
213:208	Encoder/Decoder 0 ECC
219:214	Encoder/Decoder 1 ECC
225:220	Encoder/Decoder 2 ECC
231:226	Encoder/Decoder 3 ECC
237:232	Encoder/Decoder 4 ECC
243:238	Encoder/Decoder 5 ECC
249:244	Encoder/Decoder 6 ECC
255:250	Encoder/Decoder 7 ECC

#### 12.2.1.4.6.13 Ethernet Port Flow Control

The Ethernet port have flow control available for transmit and receive. Transmit flow control stops the Ethernet port from transmitting packets to the wire (switch egress) in response to a received pause frame. Transmit flow control does not depend on FIFO usage.

The Ethernet port have flow control available for receive operations (packet ingress). Ethernet port receive flow control is initiated when enabled and triggered. Packets received on an Ethernet port can be sent to the CPPI port. The destination port can trigger the receive Ethernet port flow control. An Ethernet destination port triggers another Ethernet receive flow control when the destination port is full.

When a packet is received on an Ethernet port interface with enabled flow control the below occurs:

- The packet will be sent to all ports that currently have room to take the entire packet.
- The packet will be retried until successful to all ports that indicate they don't have room for the packet.

The flow control trigger to the Ethernet port will be asserted until the packet has been sent, and there is room in the logical receive FIFO for packet runout from another flow control trigger (RX\_BLK\_CNT = 0h). Ethernet port receive flow control is disabled by default on reset. Ethernet port receive flow control requires that the RX\_FLOW\_EN bit in CPSW\_PN\_MAC\_CONTROL\_REG be set to 1h. When receive flow control is enabled on a port, the port's associated FIFO block allocation must be adjusted. The port RX allocation must increase from the default three blocks to accommodate the flow control runout. A corresponding decrease in the TX block allocation is required. If a sending port ignores a pause frame then packets may overrun on receive (and be dropped) but will not be dropped on transmit.

### 12.2.1.4.6.13.1 Ethernet Receive Flow Control

For every Ethernet port to be configured for full-duplex receive flow control, write a value of decimal 7 to the CPSW\_PN\_MAX\_BLKs\_REG[7-0] RX\_MAX\_BLKs bit field, and a value of decimal 13 to the CPSW\_PN\_MAX\_BLKs\_REG[15-8] TX\_MAX\_BLKs register. This re-allocation allows for flow control runout on the receive FIFO at the expense of FIFO memory on the Ethernet transmit side. 10/100Mbps half-duplex collision based receive flow control does not need this re-allocation. Receive flow control is enabled by the RX\_FLOW\_EN bit in the CPSW\_PN\_MAC\_CONTROL\_REG register.

#### 12.2.1.4.6.13.1.1 Collision Based Receive Buffer Flow Control

Collision-based receive buffer flow control provides a means of preventing frame reception when the port is operating in half-duplex mode (FULLDUPLEX is cleared in CPSW\_PN\_MAC\_CONTROL\_REG). When receive flow control is enabled and triggered, the port will generate collisions for received frames. The jam sequence transmitted will be the twelve byte sequence C3.C3.C3.C3.C3.C3.C3.C3.C3.C3.C3.C3 (hex). The jam sequence will begin no later than approximately as the source address starts to be received. Note that these forced collisions will not be limited to a maximum of 16 consecutive collisions, and are independent of the normal back-off algorithm. Receive flow control does not depend on the value of the incoming frame destination address. A collision will be generated for any incoming packet, regardless of the destination address.

#### 12.2.1.4.6.13.1.2 IEEE 802.3X Based Receive Flow Control

IEEE 802.3x based receive flow control provides a means of preventing frame reception when the port is operating in full-duplex mode (FULLDUPLEX bit is set in the CPSW\_PN\_MAC\_CONTROL\_REG register). When receive flow control is enabled and triggered, the port will transmit a pause frame to request that the sending station stop transmitting for the period indicated within the transmitted pause frame.

The Ethernet port will transmit a pause frame to the reserved multicast address at the first available opportunity (immediately if currently idle, or following the completion of the frame currently being transmitted). The pause frame will contain the maximum possible value for the pause time (FFFFh). The MAC will count the receive pause frame time (decrements FF00h down to 0) and retransmit an outgoing pause frame if the count reaches zero. When the flow control request is removed, the MAC will transmit a pause frame with a zero pause time to cancel the pause request.

Note that transmitted pause frames are only a request to the other end station to stop transmitting. Frames that are received during the pause interval will be received normally (provided the RX FIFO is not full at which time the receive FIFO will overrun and CPSW\_STAT0\_RX\_BOTTOM\_OF\_FIFO\_DROP/ CPSW\_STAT1\_RX\_BOTTOM\_OF\_FIFO\_DROP[31-0] COUNT value will increment).

Pause frames will be transmitted if enabled and triggered regardless of whether or not the port is observing the pause time period from an incoming pause frame.

The Ethernet port will transmit pause frames as described below:

- The 48-bit reserved multicast destination address 01.80.C2.00.00.01.
- The 48-bit source address - from SL\_SA[47-0] input.
- The 16-bit length/type field containing the value 88.08
- The 16-bit pause opcode equal to 00.01
- The 16-bit pause time value FF.FF. A pause-quantum is 512 bit-times. Pause frames sent to cancel a pause request will have a pause time value of 00.00.
- Zero padding to 64-byte data length (The Ethernet port will transmit only 64 byte pause frames).
- The 32-bit frame-check sequence (CRC word).

All quantities above are hexadecimal and are transmitted most-significant byte first. The least-significant bit is transferred first in each byte.

If CPSW\_PN\_MAC\_CONTROL\_REG[3] RX\_FLOW\_EN is cleared to 0h while the pause time is nonzero, then the pause time will be cleared to 0h and a 0 count pause frame will be sent.

#### 12.2.1.4.6.13.2 Flow Control Trigger

Receive flow control is triggered (when enabled), when the number of words in the receive FIFO is greater than or equal to the value written in the CPSW\_PN\_RX\_FLOW\_THRESH\_REG[8-0] COUNT bit field. The flow control packet runout is then contained in the remainder of the receive FIFO.

#### 12.2.1.4.6.13.3 Ethernet Transmit Flow Control

Incoming pause frames are acted upon, when enabled, to prevent the Ethernet port from transmitting any further frames. Incoming pause frames are only acted upon when the [0] FULLDUPLEX and [4] TX\_FLOW\_EN bits in the CPSW\_PN\_MAC\_CONTROL\_REG register are set. Pause frames are not acted upon in half-duplex mode. Pause frame action will be taken if enabled, but normally the frame will be filtered and not transferred to memory. MAC control frames will be transferred to memory if the [24] RX\_CMF\_EN (RX Copy MAC Control Frames Enable) bit in the CPSW\_PN\_MAC\_CONTROL\_REG register is set. The [4] TX\_FLOW\_EN and [0] FULLDUPLEX bits effect whether or not MAC control frames are acted upon, but they have no effect upon whether or not MAC control frames are transferred to memory or filtered.

Pause frames are a subset of MAC Control Frames with an opcode field = 0001h. Incoming pause frames will only be acted upon by the port if:

- [4] TX\_FLOW\_EN is set in CPSW\_PN\_MAC\_CONTROL\_REG register, and
- the RX maximum frame length is 64 bytes inclusive (CPSW\_PN\_RX\_MAXLEN\_REG[13-0] RX\_MAXLEN), and
- the frame contains no CRC error or align/code errors.

The pause time value from valid frames will be extracted from the two bytes following the opcode. The pause time will be loaded into the port's transmit pause timer and the transmit pause time period will begin.

If a valid pause frame is received during the transmit pause time period of a previous transmit pause frame then:

- if the destination address is not equal to the reserved multicast address or any enabled or disabled unicast address, then the transmit pause timer will immediately expire, or
- if the new pause time value is zero then the transmit pause timer will immediately expire, else the port transmit pause timer will immediately be set to the new pause frame pause time value. (Any remaining pause time from the previous pause frame will be discarded).

If [4] TX\_FLOW\_EN in CPSW\_PN\_MAC\_CONTROL\_REG register is cleared, then the pause-timer will immediately expire.

The port will not start the transmission of a new data frame any sooner than 512-bit times after a pause frame with a non-zero pause time has finished being received (MRXDV going inactive). No transmission will begin until the pause timer has expired (the port may transmit pause frames in order to initiate outgoing flow control). Any frame already in transmission when a pause frame is received will be completed and unaffected.

Incoming pause frames consist of the below:

- A 48-bit destination address equal to:
  - The reserved multicast destination address 01.80.C2.00.00.01, or the Ethernet port SL\_SA [47:0] input.
- The 48-bit source address of the transmitting device.
- The 16-bit length/type field containing the value 88.08
- The 16-bit pause opcode equal to 00.01
- The 16-bit CPSW\_PN\_MAC\_TX\_PAUSETIMER\_REG[15-0] TX\_PAUSETIMER. A pause-quantum is 512 bit-times.
- Padding to 64-byte data length.
- The 32-bit frame-check sequence (CRC word).

All quantities above are hexadecimal and are transmitted most-significant byte first. The least-significant bit is transferred first in each byte.

The padding is required to make up the frame to a minimum of 64 Bytes. The standard allows pause frames longer than 64 Bytes to be discarded or interpreted as valid pause frames. The Ethernet port will recognize any pause frame between 64 Bytes and CPSW\_PN\_RX\_MAXLEN\_REG[13-0] RX\_MAXLEN bytes in length.

#### 12.2.1.4.6.14 Energy Efficient Ethernet Support (802.3az)

Energy Efficient Ethernet (EEE) allows the LPSC to turn off the module clock during inactive periods as determined by network and host traffic. The module can then be awakened by host queued transmit packet(s) or by a port's external Ethernet PHY. The module EEE clock stop interface is used by the external controller to control module EEE operations. EEE operations are configured as shown below:

1. The 12-bit EEE clock pre-scale value is written to the CPSW\_EEE\_PRESCALE\_REG register. The pre-scaler is used to clock all EEE-related counters
2. The port Idle to LPI count values (CPSW\_PN\_IDLE2LPI\_REG[23-0] COUNT) are written with the desired values
3. The port LPI to Wake count values (CPSW\_PN\_LPI2WAKE\_REG[23-0] COUNT) are written with the desired values
4. The [0] EEE\_EN bit is set in the switch CPSW\_SS\_CONTROL\_REG register

EEE operation can begin after configuration. The host allows (through LPSC) the CPSW to enter a low power state by asserting the EEE\_CLKSTOP\_REQ signal. There are no requirements on host queues or traffic in order for the host to assert or de-assert EEE\_CLKSTOP\_REQ to the CPSW.

Each Ethernet port has a transmit and a receive LPI (low power indicate) state. The PHY indicates LPI by asserting MRXER with a MRXD[7:0] value of 0x01 while MRXDV is deasserted (inter-packet gap). The Ethernet transmit port indicates LPI after the CPSW\_PN\_IDLE2LPI\_REG value has been counted (the transmit port has gone idle for the configured amount of time). If another packet is received for transmit during the count then the count is restarted. When the transmit port has been idle for the Idle to LPI time, the transmit port enters the LPI state and indicates LPI to the associated PHY. The LPI is indicated to the external PHY by an asserted MTXER with a MTXD[7:0] while MTXEN is deasserted (inter-packet gap). The CPPI (port 0) LPI state includes transmit and receive. The CPPI LPI state is entered when the CPPI transmit and receive have both been idle for the Idle to LPI time (CPSW\_P0\_IDLE2LPI\_REG). The Idle to LPI time value for all ports must be large relative to the switch latency to ensure that the count is not able to complete between successive packets.

#### Note

External PHY signaling has the following conditions:

- RGMII is a DDR interface. TXEN and TXER are the sampled values of TX\_CTL at the rising and the falling TXC edges, respectively. RXDV and RXER are the sampled values of RX\_CTL at the rising and the falling RXC clock edges, respectively
- In RMII mode, EEE is not supported.

When all transmit and receive ports are in the LPI state (CPSW LPI state), the EEE\_CLKSTOP\_ACK signal is asserted, and the LPSC is allowed to stop the module clock. When EEE\_CLKSTOP\_ACK is asserted, the clock may be turned on and off as desired by the host. The host is allowed to restart the clock, perform target read/write operations to the CPSW memory address space, and then turn off the clock again while EEE\_CLKSTOP\_ACK is asserted.

The software can remove and disable from re-entering the CPSW LPI state by restarting the module clock and then de-asserting EEE\_CLKSTOP\_REQ. There must be at least one rising edge of the clock before EEE\_CLKSTOP\_REQ is de-asserted. The module EEE\_CLKSTOP\_ACK output signal will be deasserted on the clock after the de-assertion of EEE\_CLKSTOP\_REQ. The host may queue CPPI receive packets at any time without regard to the CPSW module LPI state. The Host must deassert EEE\_CLKSTOP\_REQ on wakeup for a minimum of two clock periods. If EEE\_CLKSTOP\_REQ is deasserted for less than 5 clock periods for a wakeup event from the host to a particular Ethernet port (or visa versa), then the wakeup event will not cause the other Ethernet port to awaken.



The external Ethernet PHY's can also wakeup the LPSC by removing the Ethernet receive LPI indication. If the CPSW module is in Idle state with `EEE_CLKSTOP_ACK` asserted and the receive LPI indication is removed, the `EEE_CLKSTOP_WAKEUP` signal will be asynchronously asserted. On wakeup, the LPSC restarts the clock and de-assert the `EEE_CLKSTOP_REQ` signal. The `EEE_CLKSTOP_WAKEUP` signal will be synchronously deasserted with `EEE_CLKSTOP_ACK`. Upon the deassertion of `EEE_CLKSTOP_REQ`, the Ethernet ports will count the `CPSW_PN_LPI2WAKE_REG` time for each port at which time the port is available for transmit.

#### 12.2.1.4.6.15 Ethernet Switch Latency

When CPSW is configured as a store and forward switch, the switch latency is defined as the amount of time between the end of packet reception of the received packet to the start of the output packet transmit.

The store and forward latency is shown in [Table 12-183](#):

**Table 12-183. Switch Latency**

Mode	Latency
Gig (1000)	880 ns
100	1.3 $\mu$ s
10	6.5 $\mu$ s

#### 12.2.1.4.6.16 MAC Emulation Control

The emulation control input (EMUSUSP) and submodule emulation control registers allow CPSW operation to be completely or partially suspended. Each Ethernet port has associated emulation control registers (`CPSW_EM_CONTROL_REG` and `CPSW_PN_MAC_EMCONTROL_REG`). The submodule emulation control registers must be accessed to facilitate CPSW emulation control. The CPSW module enters the emulation suspend state if all three submodules are configured for emulation suspend and the emulation suspend input is asserted. A partial emulation suspend state is entered if one or two submodules is configured for emulation suspend and the emulation suspend input is asserted. Emulation suspend occurs at packet boundaries. The emulation control feature is implemented for compatibility with other peripherals.

#### Ethernet port Emulation Control

The emulation control input (TBEMUSUP) and register bits (SOFT and FREE bits in the `CPSW_PN_MAC_EMCONTROL_REG` register) allow Ethernet port operation to be suspended. When the emulation suspend state is entered, the Ethernet port will stop processing receive and transmit frames at the next frame boundary. Any frame currently in reception or transmission will be completed normally without suspension. For receive, frames that are detected by the Ethernet port after the suspend state is entered are ignored.

[Table 12-184](#) shows the operations of the emulation control input and register bits.

**Table 12-184. Emulation Control Input**

EMUSUSP	SOFT	FREE	Description
0	X	X	Normal Operation
1	0	0	Normal Operation
1	1	0	Emulation Suspend
1	X	1	Normal Operation

#### 12.2.1.4.6.17 MAC Command IDLE

The `CMD_IDLE` bit in the `CPSW_PN_MAC_CONTROL_REG` register allows MAC operation to be suspended. When the idle state is commanded, the MAC will stop processing receive and transmit frames at the next frame boundary. Any frame currently in reception or transmission will be completed normally without suspension. For transmission, any complete or partial frame in the TX cell FIFO will be transmitted. For receive, frames that are detected by the MAC after the suspend state is entered are ignored. No statistics will be kept for ignored frames. Commanded idle is similar in operation to emulation control and clock stop.

### 12.2.1.4.6.18 CPSW Network Statistics

The CPSW has a set of statistics that record events associated with frame traffic on selected switch ports. The statistics values are cleared to zero 38 clocks after the rising edge of MCU\_CPSW0\_RST. When one or more port enable (Pn\_STAT\_EN) bits in the CPSW\_STAT\_PORT\_EN\_REG register are set, all statistics registers are write to decrement. The value written will be subtracted from the register value with the result being stored in the register. If a value greater than the statistics value is written, then zero will be written to the register (writing 0xFFFF FFFF clears a statistics location). When all port enable bits are cleared to zero, all statistics registers are read/write (normal write direct, so writing 0x0000 0000 clears a statistics location). All write accesses must be 32-bit accesses.

The statistics interrupt (STAT\_PEND0) will be issued if enabled when any statistics value is greater than or equal to 0x8000 0000. The statistics interrupt is removed by writing to decrement any statistics value greater than 0x8000 0000. The statistics are mapped into internal memory space and are 32-bits wide. All statistics rollover from 0xFFFF FFFF to 0x0000 0000.

[Section 12.2.1.4.6.18.1](#) and [Section 12.2.1.4.6.18.8](#) summarize network statistics.

#### 12.2.1.4.6.18.1 Rx-only Statistics Descriptions

##### 12.2.1.4.6.18.1.1 Good Rx Frames (Offset = 3A000h - Port 0 or Offset = 3A200h - Port 1)

#### All ports

The total number of good frames received on the port. A good frame is defined to be:

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Had a length of 64 to RX\_MAXLEN bytes inclusive
- Had no CRC error, alignment error or code error.

See the *Rx Align/Code Errors* and *Rx CRC errors* statistic descriptions for definitions of alignment, code and CRC errors.

Overruns have no effect upon this statistic.

##### 12.2.1.4.6.18.1.2 Broadcast Rx Frames (Offset = 3A004h - Port 0 or Offset = 3A204h - Port 1)

#### All ports

The total number of good broadcast frames received on the port. A good broadcast frame is defined to be:

- Any data or MAC control frame which was destined for address FF.FF.FF.FF.FF.FF
- Had a length of 64 to RX\_MAXLEN bytes inclusive
- Had no CRC error, alignment error or code error.

See the *Rx Align/Code Errors* and *Rx CRC errors* statistic descriptions for definitions of alignment, code and CRC errors.

Overruns have no effect upon this statistic.

##### 12.2.1.4.6.18.1.3 Multicast Rx Frames (Offset = 3A008h - Port 0 or Offset = 3A208h - Port 1)

#### All ports

The total number of good multicast frames received on the port. A good multicast frame is defined to be:

- Any data or MAC control frame which was destined for any multicast address other than FF.FF.FF.FF.FF.FF
- Had a length of 64 to RX\_MAXLEN bytes inclusive
- Had no CRC error, alignment error or code error

See the [Section 12.2.1.4.6.18.1.6](#), *Rx Align/Code Errors* and [Section 12.2.1.4.6.18.1.5](#), *Rx CRC errors* statistic descriptions for definitions of alignment, code and CRC errors.

Overruns have no effect upon this statistic.



#### 12.2.1.4.6.18.1.4 Pause Rx Frames (Offset = 3A20Ch - Port 1)

##### Ethernet port 1

The total number of IEEE 802.3X pause frames received by the port (whether acted upon or not). Such a frame:

- Contained any unicast, broadcast, or multicast address
- Contained the length/type field value 88.08 (hex) and the opcode 0x0001
- Was of length 64 to RX\_MAXLEN bytes inclusive
- Had no CRC error, alignment error or code error
- Pause-frames had been enabled on the port (TX\_FLOW\_EN = 1h).

The port could have been in either half or full-duplex mode.

See the [Section 12.2.1.4.6.18.1.6, Rx Align/Code Errors](#) and [Section 12.2.1.4.6.18.1.5, Rx CRC errors](#) statistic descriptions for definitions of alignment, code and CRC errors.

Overruns have no effect upon this statistic.

#### 12.2.1.4.6.18.1.5 Rx CRC Errors (Offset = 3A010h - Port 0 or Offset = 3A210h - Port 1)

##### All ports

The total number of frames received on the port that experienced a CRC error. Such a frame:

- Was any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Was of length 64 to RX\_MAXLEN bytes inclusive
- Had no code/align error
- Had a CRC error

Overruns have no effect upon this statistic.

A CRC error is defined to be:

- A frame containing an even number of nibbles, and
- Failing the Frame Check Sequence test.

#### 12.2.1.4.6.18.1.6 Rx Align/Code Errors (Offset = 3A214h - Port 1)

##### Ethernet port 1

The total number of frames received on the port that experienced an alignment error or code error. Such a frame:

- Was any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Was of length 64 to RX\_MAXLEN bytes inclusive
- Had either an alignment error, or a code error.

Overruns have no effect upon this statistic.

An alignment error is defined to be:

- A frame containing an odd number of nibbles
- Failing the Frame Check Sequence test if the final nibble is ignored

A code error is defined to be a frame which has been discarded because the port's MRXER pin driven with a one for at least one bit-time's duration at any point during the frame's reception.

#### Note

RFC 1757 etherStatsCRCAAlignErrors Ref. 1.5 can be calculated by summing Rx Align/Code Errors and Rx CRC errors.

**12.2.1.4.6.18.1.7 Oversize Rx Frames (Offset = 3A018h - Port 0 or Offset = 3A218h - Port 1)****All ports**

The total number of oversized frames received on the port. An oversized frame is defined to be:

- Was any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Was greater than RX\_MAXLEN in bytes
- Had no CRC error, alignment error, or code error.

See the [Section 12.2.1.4.6.18.1.6, Rx Align/Code Errors](#) and [Section 12.2.1.4.6.18.1.5, Rx CRC errors](#) statistic descriptions for definitions of alignment, code and CRC errors.

Overruns have no effect upon this statistic.

**12.2.1.4.6.18.1.8 Rx Jabbers (Offset = 3A21Ch - Port 1)****All ports**

The total number of jabber frames received on the port. A jabber frame:

- Was any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Was greater than RX\_MAXLEN in bytes
- Had no CRC error, alignment error or code error

See the [Section 12.2.1.4.6.18.1.6, Rx Align/Code Errors](#) and [Section 12.2.1.4.6.18.1.5, Rx CRC errors](#) statistic descriptions for definitions of alignment, code and CRC errors.

Overruns have no effect upon this statistic.

**12.2.1.4.6.18.1.9 Undersize (Short) Rx Frames (Offset = 3A020h - Port 0 or Offset = 3A220h - Port 1)****All ports**

The total number of undersized frames received on the port. An undersized frame is defined to be:

- Was any data frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Was less than 64 bytes
- Had no CRC error, alignment error, or code error

See the [Section 12.2.1.4.6.18.1.6, Rx Align/Code Errors](#) and [Section 12.2.1.4.6.18.1.5, Rx CRC errors](#) statistic descriptions for definitions of alignment, code and CRC errors.

Overruns have no effect upon this statistic.

**12.2.1.4.6.18.1.10 Rx Fragments (Offset = 3A024h - Port 0 or Offset = 3A224h - Port 1)****Ethernet port 1**

The total number of frame fragments received on the port. A frame fragment is defined to be:

- Any data frame (address matching does not matter)
- Less than 64 bytes long
- Having a CRC error, an alignment error, or a code error
- Not the result of a collision caused by half-duplex, collision-based flow control

See the [Section 12.2.1.4.6.18.1.6, Rx Align/Code Errors](#) and [Section 12.2.1.4.6.18.1.5, Rx CRC errors](#) statistic descriptions for definitions of alignment, code and CRC errors.

Overruns have no effect upon this statistic.

#### 12.2.1.4.6.18.1.11 RX IPG Error (Offset = 3A25Ch - Port 1)

The total number of 10G frames received on a port that had a correct preamble but did not have at least five bytes of IDLE preceding the frame. This does not indicate if the frame with the IPG error was kept or ignored.

#### 12.2.1.4.6.18.1.12 ALE Drop (Offset = 3A028h - Port 0 or Offset = 3A228h - Port 1)

##### All ports

The total number of frames received on a port such that the destination address was not equal to the source address and the packet was not destined to the port it was received on, but the frame was not forwarded to any port (the PORT\_MASK was zero).

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Any length (including less than 64 bytes and greater than RX\_MAXLEN bytes)
- had no CRC error, alignment error, or code error
- the destination address was not equal to the source address
- the packet was not destined for the port it was receive on
- had a zero PORT\_MASK

#### 12.2.1.4.6.18.1.13 ALE Overrun Drop (Offset = 3A02Ch - Port 0 or Offset = 3A22Ch - Port 1)

##### All ports

The total number of frames received on a port that were dropped (zero PORT\_MASK) due to exceeding the maximum ALE lookup rate (Port 0 should not have ALE Overrun Drops because the ingress rate is controlled to prevent it). This statistic should be zero and when non-zero indicates a system clock issue or indicates that short packets were sent with RX\_CSF\_EN at a rate that exceeded the maximum lookup rate.

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Any length (including less than 64 bytes and greater than RX\_MAXLEN bytes)
- the maximum ALE lookup rate was exceeded so the lookup was aborted and the packet was dropped.

#### 12.2.1.4.6.18.1.14 Rx Octets (Offset = 3A030h - Port 0 or Offset = 3A230h - Port 1)

##### All ports

The total number of bytes in all good frames received on the port. A good frame is defined to be:

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Of length 64 to RX\_MAXLEN bytes inclusive
- Had no CRC error, alignment error or code error

See the [Section 12.2.1.4.6.18.1.6, Rx Align/Code Errors](#) and [Section 12.2.1.4.6.18.1.5, Rx CRC errors](#) statistic descriptions for definitions of alignment, code and CRC errors.

Overruns have no effect upon this statistic.

#### 12.2.1.4.6.18.1.15 Rx Bottom of FIFO Drop (Offset = 3A084h - Port 0 or Offset = 3A284h - Port 1)

##### Ethernet port 1

The total number of frames received on a port that overran the port's receive FIFO and were dropped (bottom of receive FIFO). Port 0 (CPPI receive port) should not drop packets on receive because port 0 receive flow control should be enabled. The Ethernet ports will only drop packets in the receive FIFO when receive flow control is enabled and the sending port ignores sent pause frame and then overruns the receive FIFO. An overrun frame is defined to be:

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Any length (including less than 64 bytes and greater than RX\_MAXLEN bytes)

- Was dropped on port 0 due to a lack of memory space in the receive FIFO.

---

**Note**

This statistic should be zero if proper flow control is being followed.

---

**Host port 0**

This statistic also counts frames dropped on port 0 that were 17 to 63 bytes (only for port 0). For Ethernet ports, the drop count for frames shorter than 33 bytes is included in the undersized or fragment count. Port 0 simply gives an indication that a packet was dropped. No other statistics are counted for frames shorter than 33 bytes.

**12.2.1.4.6.18.1.16 Portmask Drop (Offset = 3A088h - Port 0 or Offset = 3A288h - Port 1)**
**All ports**

The total number of frames received on a port that were dropped by the ALE (the ALE did not forward the packet to any port). Port mask drop frame is defined to be:

- Any data or MAC control frame
- Any length greater than 32 bytes
- Was dropped by the ALE due to PORT\_MASK=0 (was not sent to any destination port)
- The frame could have been dropped due to error or other counted reason, so it could be counted elsewhere also.

---

**Note**

This statistic does not count in the overall total as it includes every packet received greater than 32 bytes that had a zero PORT\_MASK.

---

**12.2.1.4.6.18.1.17 Rx Top of FIFO Drop (Offset = 3A08Ch - Port 0 or Offset = 3A28Ch - Port 1)**
**All ports**

The total number of frames received on a port that had a start-of-frame (SOF) overrun on any destination port egress (when attempting to load the packet from the top of the ingress port receive FIFO into any other port's transmit FIFO). If a multicast/broadcast packet is dropped by multiple destination ports then this statistic will increment by the number of ports that dropped the packet. Rx Top Of FIFO Drop is defined to be:

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Any length (including less than 64 bytes and greater than RX\_MAXLEN bytes)
- had no CRC error, alignment error or code error
- had a SOF of frame overrun on another port egress.

**12.2.1.4.6.18.1.18 ALE Rate Limit Drop (Offset = 3A090h - Port 0 or Offset = 3A290h - Port 1)**
**All ports**

The total number of frames received on a port that were dropped (zero PORT\_MASK) due to receive rate limiting on this port or due to transmit rate limiting on any destination port (not sent to all expected destination ports if transmit rate limiting).

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Any length (including less than 64 bytes and greater than RX\_MAXLEN bytes)
- had no CRC error, alignment error, or code error
- the receive rate was exceeded and the packet was dropped, or the transmit rate was exceeded to any destination port and the packet was dropped to one or more expected destination ports (indicates that the destinations were reduced due to rate limiting).

#### **12.2.1.4.6.18.1.19 ALE VLAN Ingress Check Drop (Offset = 3A094h - Port 0 or Offset = 3A294h - Port 1)**

##### **All ports**

The total number of frames received on a port that were dropped (zero PORT\_MASK) due to VLAN ingress check failure.

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Any length (including less than 64 bytes and greater than RX\_MAXLEN bytes)
- had no CRC error, alignment error, or code error
- the VLAN ID ingress check failed (the receive port was not in the group)
- The address lookup did not return a match with the SUPER bit set.

#### **12.2.1.4.6.18.1.19.1 ALE DA=SA Drop (Offset = 3A098h - Port 0 or Offset = 3A298h - Port 1)**

##### **All ports**

The total number of frames received on a port that were dropped (zero PORT\_MASK) due to destination address equal to source address.

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Any length (including less than 64 bytes and greater than RX\_MAXLEN bytes)
- had no CRC error, alignment error, or code error
- the destination address was equal to the source address
- the source address was not an entry in the table.

#### **12.2.1.4.6.18.1.19.2 Block Address Drop (Offset = 3A09Ch - Port 0 or Offset = 3A29Ch - Port 1)**

The total number of frames received on a port that were dropped (zero PORT\_MASK) due to the destination or source address being blocked.

- was any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode, and
- Any length (including less than 64 bytes and greater than RX\_MAXLEN bytes)
- had no CRC error, alignment error, or code error, and
- the source or destination address matched a table entry with the block bit set.

#### **12.2.1.4.6.18.1.19.3 ALE Secure Drop (Offset = 3A0A0h - Port 0 or Offset = 3A2A0h - Port 1)**

The total number of frames received on a port that were dropped (zero port\_mask) due to a secure violation (the source address is owned by a different receive port).

- was any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode, and
- Any length (including less than 64 bytes and greater than RX\_MAXLEN bytes)
- had no CRC error, alignment error, or code error, and
- the source address is an entry in the table with the SECURE bit set and a port number for a different receive port.

#### **12.2.1.4.6.18.1.19.4 ALE Authentication Drop (Offset = 3A0A4h - Port 0 or Offset = 3A2A4h - Port 1)**

The total number of frames received on a port that were dropped (zero port\_mask) due to authentication failure.

- was any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode, and
- Any length (including less than 64 bytes and greater than RX\_MAXLEN bytes)
- had no CRC error, alignment error, or code error, and
- CPSW\_ALE\_CONTROL[1] ENABLE\_AUTH\_MODE is set to 1h, and
- the source address is not equal to the destination address, and
- the source address is not a table entry, and
- the destination address is not a table entry with the SUPER bit set.

#### **12.2.1.4.6.18.1.19.5 ALE Unknown Unicast (Offset = 3A0A8h - Port 0 or Offset = 3A2A8h - Port 1)**

##### **All ports**

The total number of frames received on a port that had a unicast destination address with an unknown source address.

- was any data frame with a unicast destination address
- the source address was not a table entry
- was of length 64 to RX\_MAXLEN bytes inclusive
- had no CRC error, alignment error, or code error

Note: The ALE Unknown Unicast Bytecount statistic is the number of bytes contained in the ALE Unknown Unicast frames.

#### **12.2.1.4.6.18.1.19.6 ALE Unknown Unicast Bytecount (Offset = 3A0ACh - Port 0 or Offset = 3A2ACh - Port 1)**

The total number of bytes received on a port that had a unicast destination address with an unknown source address.

#### **12.2.1.4.6.18.1.19.7 ALE Unknown Multicast (Offset = 3A0B0h - Port 0 or Offset = 3A2B0h - Port 1)**

The total number of frames received on a port that had a multicast destination address with an unknown source address. The frame is defined to be:

- was any data frame with a unicast destination address
- the source address was not a table entry, and
- was of length 64 to RX\_MAXLEN bytes inclusive
- had no CRC error, alignment error or code error

Note: The ALE Unknown Multicast Bytecount statistic is the number of bytes contained in the ALE Unknown Multicast frames.

#### **12.2.1.4.6.18.1.19.8 ALE Unknown Multicast Bytecount (Offset = 3A0B4h - Port 0 or Offset = 3A2B4h - Port 1)**

The total number of bytes received on a port that had a multicast destination address with an unknown source address.

#### **12.2.1.4.6.18.1.19.9 ALE Unknown Broadcast (Offset = 3A0B8h - Port 0 or Offset = 3A2B8h - Port 1)**

The total number of frames received on a port that had a broadcast destination address with an unknown source address. The frame is defined to be:

- was any data frame with a unicast destination address
- the source address was not a table entry, and
- was of length 64 to RX\_MAXLEN bytes inclusive
- had no CRC error, alignment error or code error

Note: The ALE Unknown Broadcast Bytecount statistic is the number of bytes contained in the ALE Unknown Broadcast frames.

#### **12.2.1.4.6.18.1.19.10 ALE Unknown Broadcast Bytecount (Offset = 3A0BCh - Port 0 or Offset = 3A2BCh - Port 1)**

The total number of bytes received on a port that had a broadcast destination address with an unknown source address.

#### **12.2.1.4.6.18.1.19.11 ALE Policier/Classifier Match (Offset = 3A0C0h - Port 0 or Offset = 3A2C0h - Port 1)**

##### **All ports**

The total number of frames received on a port that matched a policier. The frame is defined to be:

- was any data frame
- matched a condition on a policier,
- was of length 64 to RX\_MAXLEN bytes inclusive
- had no CRC error, alignment error, or code error

#### **12.2.1.4.6.18.2 ALE Policer Match Red (Offset = 3A0C4h - Port 0 or Offset = 3A2C4h - Port 1)**

The total number of frames received on a port that had matched a policer and the condition was red. The frame is defined to be:

- was any data frame
- matched a condition on a policer,
- was of length 64 to RX\_MAXLEN bytes inclusive
- had no CRC error, alignment error, or code error

#### **12.2.1.4.6.18.3 ALE Policer Match Yellow (Offset = 3A0C8h - Port 0 or Offset = 3A2C8h - Port 1)**

The total number of frames received on a port that had matched a policer and the condition was red. The frame is defined to be:

- was any data frame
- matched a condition on a policer,
- was of length 64 to RX\_MAXLEN bytes inclusive
- had no CRC error, alignment error, or code error

#### **12.2.1.4.6.18.4 IET Receive Assembly Error (Offset = 3A140h - Port 0 or Offset = 3A340h - Port 1)**

The total number of preemptable received frames with IET assembly errors.

- any frame received
- was any size, and
- was a non-initial fragment that mismatched the frame count or fragment count (went to the assembly error state in the IET receive state machine)

#### **12.2.1.4.6.18.5 IET Receive Assembly OK (Offset = 3A144h - Port 0 or Offset = 3A344h - Port 1)**

The total number of correctly received and re-assembled preemptable frames.

- any preemptable frame received
- was any size, and
- was correctly received and re-assembled without error

#### **12.2.1.4.6.18.6 IET Receive SMD Error (Offset = 3A148h - Port 0 or Offset = 3A348h - Port 1)**

The total number of received frames rejected due to an unknown SMD value or received frames rejected with an SMD-C when no frame is in progress.

- any frame received
- was any size, and
- was rejected because of an unknown SMD value or SMD-C with no frame in progress.

Note: If IET\_ENABLE is not set, this statistic counts any received frame with any non express SMD.

#### **12.2.1.4.6.18.7 IET Receive Merge Fragment Count (Offset = 3A14Ch - Port 0 or Offset = 3A34Ch - Port 1)**

The total number of received non-initial fragments that did not have an assembly error. The IET stat CPSW\_STAT0\_RXFRAGMENTS/ CPSW\_STAT1\_RXFRAGMENTS is derived by adding the Receive Assembly Error count to this value.

- any frame received
- was any size, and
- was a non-initial fragment that did not contain an assembly error

#### **12.2.1.4.6.18.8 Tx-only Statistics Descriptions**

The maximum and minimum transmit frame size is software controllable.

#### **12.2.1.4.6.18.8.1 Good Tx Frames (Offset = 3A034h - Port 0 or Offset = 3A234h - Port 1)**

**All ports**



The total number of good frames received on the port. A good frame is defined to be:

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Any length
- Had no late or excessive collisions, no carrier loss and no underrun

#### **12.2.1.4.6.18.8.2 Broadcast Tx Frames (Offset = 3A038h - Port 0 or Offset = 3A238h - Port 1)**

##### **All ports**

The total number of good broadcast frames transmitted on the port. A good broadcast frame is defined to be:

- Any data or MAC control frame which was destined for only address FF.FF.FF.FF.FF.FF
- Any length
- Had no late or excessive collisions, no carrier loss and no underrun

#### **12.2.1.4.6.18.8.3 Multicast Tx Frames (Offset = 3A03Ch - Port 0 or Offset = 3A23Ch - Port 1)**

##### **All ports**

The total number of good multicast frames transmitted on the port. A good multicast frame is defined to be:

- Any data or MAC control frame which was destined for any multicast address other than FF.FF.FF.FF.FF.FF
- Any length
- Had no late or excessive collisions, no carrier loss and no underrun

#### **12.2.1.4.6.18.8.4 Pause Tx Frames (Offset = 3A240h - Port 1)**

##### **Ethernet port 1**

This statistic indicates the number of IEEE 802.3X pause frames transmitted by the port.

Pause frames cannot contain a CRC error because they are created in the transmitting MAC, so these error conditions have no effect upon the statistic. Pause frames sent by software will not be included in this count.

Since pause frames are only transmitted in full duplex, carrier loss and collisions have no effect upon this statistic.

Transmitted pause frames are always 64-byte multicast frames so will appear in the *Tx Multicast Frames* and *64octet Frames* statistics.

#### **12.2.1.4.6.18.8.5 Deferred Tx Frames (Offset = 3A244h - Port 1)**

##### **Ethernet port 1**

The total number of frames transmitted on the port that first experienced deferment. Such a frame:

- Was any data or MAC control frame destined for any unicast, broadcast or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced no collisions before being successfully transmitted
- Found the medium busy when transmission was first attempted, so had to wait.

CRC errors have no effect upon this statistic.

#### **12.2.1.4.6.18.8.6 Collisions (Offset = 3A248h - Port 1)**

##### **Ethernet port 1**

This statistic records the total number of times that the port experienced a collision. Collisions occur under two circumstances.

1. When a transmit data or MAC control frame:
  - Was destined for any unicast, broadcast or multicast address
  - Was any size



- Had no carrier loss and no underrun
- Experienced a collision. A jam sequence is sent for every non-late collision, so this statistic will increment on each occasion if a frame experiences multiple collisions (and increments on late collisions)

CRC errors have no effect upon this statistic.

2. When the port is in half-duplex mode, flow control is active, and a frame reception begins.

#### **12.2.1.4.6.18.8.7 Single Collision Tx Frames (Offset = 3A24Ch - Port 1)**

##### **Ethernet port 1**

The total number of frames transmitted on the port that experienced exactly one collision. Such a frame:

- Was any data or MAC control frame destined for any unicast, broadcast or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced one collision before successful transmission. The collision was not late.

CRC errors have no effect upon this statistic.

#### **12.2.1.4.6.18.8.8 Multiple Collision Tx Frames (Offset = 3A250h - Port 1)**

##### **Ethernet port 1**

The total number of frames transmitted on the port that experienced multiple collisions. Such a frame:

- Was any data or MAC control frame destined for any unicast, broadcast or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced 2 to 15 collisions before being successfully transmitted. None of the collisions were late.

CRC errors have no effect upon this statistic.

#### **12.2.1.4.6.18.8.9 Excessive Collisions (Offset = 3A254h - Port 1)**

##### **Ethernet port 1**

The total number of frames for which transmission was abandoned due to excessive collisions. Such a frame:

- Was any data or MAC control frame destined for any unicast, broadcast or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced 16 collisions before abandoning all attempts at transmitting the frame. None of the collisions were late.

CRC errors have no effect upon this statistic.

#### **12.2.1.4.6.18.8.10 Late Collisions (Offset = 3A258h - Port 1)**

##### **Ethernet port 1**

The total number of frames on the port for which transmission was abandoned because they experienced a late collision. Such a frame:

- Was any data or MAC control frame destined for any unicast, broadcast or multicast address
- Was any size
- Experienced a collision later than 512 bit-times into the transmission. There may have been up to 15 previous (non-late) collisions which had previously required the transmission to be re-attempted. The *Late Collisions* statistic dominates over the single-, multiple-, and excessive- collision statistics. If a late collision occurs, the frame will not be counted in any of these other three statistics.

CRC errors have no effect upon this statistic.

#### **12.2.1.4.6.18.8.11 Carrier Sense Errors (Offset = 3A260h - Port 1)**

##### **Ethernet port 1**

The total number of frames on the port that experienced carrier loss. Such a frame:

- Was any data or MAC control frame destined for any unicast, broadcast or multicast address
- Was any size
- The carrier sense condition was lost or never asserted when transmitting the frame (the frame is not retransmitted). This is a transmit only statistic. Carrier Sense is a don't care for received frames. Transmit frames with carrier sense errors are sent until completion and are not aborted.

CRC errors have no effect upon this statistic.

#### **12.2.1.4.6.18.8.12 Tx Octets (Offset = 3A064h - Port 0 or Offset = 3A264h - Port 1)**

##### **All ports**

The total number of bytes in all good frames transmitted on the port. A good frame is defined to be:

- Any data or MAC control frame which was destined for any unicast, broadcast or multicast address
- Was any size
- Had no late or excessive collisions, no carrier loss and no underrun.

#### **12.2.1.4.6.18.8.13 Transmit Priority 0-7 (Offset = 3A380h to 3A3A8h - Port 1)**

The total number of frames transmitted on the port from transmit FIFO priority 0-7. Collision retries do not affect this statistic. Pause frames do not affect this statistic.

- Any frame transmitted from priority 0-7, and
- Was less than or equal to CPSW\_TX\_PRI0\_MAXLEN\_REG to CPSW\_TX\_PRI7\_MAXLEN\_REG
- Collision retries are not counted in this statistic.
- Pause frames are not counted in this statistic.
- Carrier sense errors do not affect this statistic.

Note: The Transmit Priority 0-7 Bytecount statistic is the number of bytes contained in the frames of the Transmit Priority 0-7 statistic.

#### **12.2.1.4.6.18.8.14 Transmit Priority 0-7 Drop (Offset = 3A3C0h to 3A3E8h - Port 1)**

The total number of transmit frames on the port that overran the transmit FIFO priority 0-7 and were dropped. This count includes frames dropped due to CPSW\_TX\_PRI0\_MAXLEN\_REG to CPSW\_TX\_PRI7\_MAXLEN\_REG.

- Any frame destined to be transmitted from priority 0-7, and
- Was any size, and
- Was dropped due to priority 0-7 FIFO overrun (Start of packet overrun).
- Was dropped due to frame size larger than CPSW\_TX\_PRI0\_MAXLEN\_REG to CPSW\_TX\_PRI7\_MAXLEN\_REG.

Note: The Transmit Priority 0-7 Drop Bytecount statistic is the number of bytes contained in the frames of the Transmit Priority 0-7 Drop statistic.

#### **12.2.1.4.6.18.8.15 Tx Memory Protect Errors (Offset = 3A17Ch - Port 0 or Offset = 3A37Ch - Port 1)**

##### **All ports**

The total number of transmit frames on the port that had a memory protect CRC error on egress:

- Any frame destined to be transmitted,
- Was any size
- Had a memory protect CRC error on egress.

### Note

1. Frames to the host with memory protect errors are dropped via TXST\_DROP. Ethernet frames will have at least one byte of the generated port type CRC inverted on egress.
2. This statistic is 8-bits wide only and will not rollover but will limit at 0xFF.
3. A non-zero value in this statistic will issue a STAT\_PEND0 interrupt for the associated port.

#### 12.2.1.4.6.18.8.16 IET Transmit Merge Hold Count (Offset = 3A350h - Port 1)

The total number of preemptable frames that were preempted and reassembled by the assertion of MAC\_HOLD in the CPSW\_PN\_IET\_CONTROL\_REG register or were preempted by Enhanced Scheduled Traffic (EST). The IET statistic can be derived and maintained by software.

- Any frame destined to be transmitted on the preemptable port, and
- was any size, and
- was preempted by the assertion of MAC\_HOLD.
- was preempted by Enhanced Scheduled Traffic (EST).

#### 12.2.1.4.6.18.8.17 IET Transmit Merge Fragment Count (Offset = 3A154h - Port 0 or Offset = 3A354h - Port 1)

The total number of non-initial preemptable transmit fragments on preemptable transmit.

- Any frame destined to be transmitted on the preemptable port, and
- Was any size, and
- was a non-initial fragment.

#### 12.2.1.4.6.18.9 Rx- and Tx (Shared) Statistics Descriptions

**Table 12-185. Rx Statistics Summary**

Rx Statistic	Frame/ Oct	Rx/ Tx	Frame Type					Frame Size (bytes)								Event				
			MAC control		Data <sup>(5)</sup>			<64	64	65-127	128-255	256-511	512-1023	1024- rx_max len	>rx_max len	Flow Coll. (8)	CRC Error	Align/ Code	Overrun	Add. Disc.
			Pause frame	Non-pause <sup>(4)</sup>	Multicast	Broadcast	Unicast													
Good Rx Frames	F	Rx	(y  <sup>(1)</sup>	y	y	y	y	n	(y	y	y	y	y	y)	n	.(2)	n	n	-	n
Broadcast Rx Frames	F	Rx	(%  (6)	%	n	y)	n	n	(y	y	y	y	y	y)	n	-	n	n	-	n
Multicast Rx Frames	F	Rx	(%	%	y)	n	n	n	(y	y	y	y	y	y)	n	-	n	n	-	n
Pause Rx Frames	F	Rx	y	n	n	n	n	n	(y	y	y	y	y	y)	n	-	n	n	-	-
Rx CRC Errors	F	Rx	(y	y	y	y	y)	n	(y	y	y	y	y	y)	n	-	y	n	-	n
Rx Align/Code Errors	F	Rx	(y	y	y	y	y)	n	(y	y	y	y	y	y)	n	-	-	y	-	n
Oversized Rx Frames	F	Rx	(y	y	y	y	y)	n	n	n	n	n	n	n	y	-	n	n	-	n
Rx Jabbers	F	Rx	(y	y	y	y	y)	n	n	n	n	n	n	n	y	-	(y	y)	-	n
Undersized Rx Frames	F	Rx	n	n	(y	y	y)	y	n	n	n	n	n	n	n	-	n	n	-	n
Rx Fragments	F	Rx	n	n	(y	y	y)	y <sup>(7)</sup>	n	n	n	n	n	n	n	-	(y	y)	-	-
Rx Overruns <sup>(9)</sup>	F	Rx	(y	y	y	y	y)	(y	y	y	y	y	y	y)	y)	-	-	-	y	n
64octet Frames	F	Rx+ Tx <sup>(3)</sup>	(y	y	y	y	y)	n	y	n	n	n	n	n	n	-	-	-	-	n

**Table 12-185. Rx Statistics Summary (continued)**

Rx Statistic	Frame/ Oct	Rx/ Rx+ Tx	Frame Type					Frame Size (bytes)										Event				
			MAC control		Data <sup>(5)</sup>			<64	64	65-127	128-255	256-511	512-1023	1024-rx_max len	>rx_max len	Flow Coll. <sup>(8)</sup>	CRC Error	Align/ Code	Overrun	Add. Disc.		
			Pause frame	Non-pause <sup>(4)</sup>	Multicast	Broadcast	Unicast															
65-127octet Frames	F	Rx+Tx	(y	y	y	y	y)	n	n	y	n	n	n	n	n	-	-	-	-	n		
128-255octet Frames	F	Rx+Tx	(y	y	y	y	y)	n	n	n	y	n	n	n	n	-	-	-	-	n		
256-511octet Frames	F	Rx+Tx	(y	y	y	y	y)	n	n	n	n	y	n	n	n	-	-	-	-	n		
512-1023octet Frames	F	Rx+Tx	(y	y	y	y	y)	n	n	n	n	n	y	n	n	-	-	-	-	n		
1024-UPoctet Frames	F	Rx+Tx	(y	y	y	y	y)	n	n	n	n	n	n	y	n	-	-	-	-	n		
Rx Octets	O	Rx	(y	y	y	y	y)	n	(y	y	y	y	y	y)	n	-	n	n	-	n		
Net Octets	O	Rx+Tx	(y	y	y	y	y)	(y	y	y	y	y	y	y	y)	y)	-	-	-	-		

- (1) "AND" is assumed horizontally across the table between all conditions which form the statistic (marked y or n) except where (y|y), meaning "OR" is indicated. Parentheses are significant.
- (2) "-" indicates conditions which are ignored in the formations of the statistic.
- (3) Statistics marked "Rx+Tx" are formed by summing the Rx and Tx statistics, each of which is formed independently.
- (4) The non-pause column refers to all MAC control frames (for example, frames with length/type=88.08) with opcodes other than 0x0001. The pauseframe column refers to MAC frames with the opcode=0x0001.
- (5) The multicast, broadcast and unicast columns in the table refer to non-MAC Control/non-pause frames (i.e. data frames).
- (6) "%" If either a MAC control frame or pause frame has a multicast or broadcast destination address then the appropriate statistics will be updated.
- (7) "y^" Frame fragments are not counted if less than 8 bytes.
- (8) Flow coll. are half-duplex collisions forced by the MAC to achieve flow-control. A collision will be forced during the first 8 bytes so should not show in frame fragments. Some of the '-'s in this column might in reality be 'n's.
- (9) The rx\_overruns stat is for RX\_MOF\_OVERRUNS and RX\_SOF\_OVERRUNS added together.

**Table 12-186. Tx Statistics Summary**

Tx Statistic <sup>(9)</sup>	Fra me/ Oct	Tx/ Rx +Tx	Frame Type					Frame Size (bytes)							Event										
			MAC control (4)		Data			64	65- 127	128- 255	256- 511	512- 1023	1024- 1535	>1535	CRC Error	Collision Type					No Carrier	Queue	Deferred	Under run	
			Pause- MAC	Any- CPU	Multi- cast	Broad- cast	Uni- cast									Flow w <sup>(8)</sup>	1	2-15	16	Late					
Good Tx Frames	F	Tx	(y  (1)	y	y	y	y)	(y	y	y	y	y	y	y)	-(2)	-	-	-	n	n	n	-	-	n	
Broadcast Tx Frames	F	Tx	n	(%  (5)	n	y)	n	(y	y	y	y	y	y	y)	-	-	-	-	n	n	n	-	-	n	
Multicast Tx Frames	F	Tx	(y	%	y)	n	n	(y	y	y	y	y	y	y)	-	-	-	-	n	n	n	-	-	n	
Pause Tx Frames	F	Tx	y	n	n	n	n	y	n	n	n	n	n	n	-	-	-	-	-	-	-	-	-	-	
Collisions	F	Tx	n	(y	y	y	y)	(y	y	y	y	y	y	y)	-	(+ (6)	+	+	+	+	n	-	-	-	
Single Collision Tx Frames	F	Tx	n	(y	y	y	y)	(y	y	y	y	y	y	y)	-	-	y	n	n	n	n	-	-	-	

**Table 12-186. Tx Statistics Summary (continued)**

Tx Statistic <sup>(9)</sup>	Fra me/ Oct	Tx/ Rx +Tx	Frame Type					Frame Size (bytes)										Event									
			MAC control (4)		Data			64	65- 127	128- 255	256- 511	512- 1023	1024- 1535	>1535	CR C Error	Collision Type					No Carrier	Queue d	Def erred	Under run			
			Pause- MAC	Any- CPU	Multi- cast	Broadcast	Uni- cast									Flow <sup>(8)</sup>	1	2-15	16	Late							
Multiple Collision Tx Frames	F	Tx	n	(y	y	y	y)	(y	y	y	y	y	y	y)	-	-	n	y	n	n	n	-	-	-			
Excessive Collisions	F	Tx	n	(y	y	y	y)	(y	y	y	y	y	y	y)	-	-	n	n	y	n	n	-	-	-			
Late Collisions	F	Tx	n	(y	y	y	y)	n	(y	y	y	y	y	y)	-	-	-	-	-	y	-	-	-	-			
Deferred Tx Frames	F	Tx	n	(y	y	y	y)	(y	y	y	y	y	y	y)	-	-	n	n	n	n	n	-	y	n			
Carrier Sense Errors	F	Tx	(y	y	y	y)	(y	y	y	y	y	y	y	y)	-	-	-	-	-	-	y	-	-	-			
64octet Frames	F	Rx+ Tx <sup>(3)</sup>	(y	y	y	y)	y	n	n	n	n	n	n	n	-	-	-	-	n	n	n	-	-	-			
65-127octet Frames	F	Rx+ Tx	(y	y	y	y)	n	y	n	n	n	n	n	n	-	-	-	-	n	n	n	-	-	-			
128-255octet Frames	F	Rx+ Tx	(y	y	y	y)	n	n	y	n	n	n	n	n	-	-	-	-	n	n	n	-	-	-			
256-511octet Frames	F	Rx+ Tx	(y	y	y	y)	n	n	n	y	n	n	n	n	-	-	-	-	n	n	n	-	-	-			
512-1023octet Frames	F	Rx+ Tx	(y	y	y	y)	n	n	n	n	y	n	n	n	-	-	-	-	n	n	n	-	-	-			
1024-UPoctet Frames	F	Rx+ Tx	(y	y	y	y)	n	n	n	n	n	y	y	y	-	-	-	-	n	n	n	-	-	-			
Tx Octets	O	Tx	(y	y	y	y)	(y	y	y	y	y	y	y	y)	-	-	-	-	n	n	n	-	-	n			
Net Octets	O	Rx+ Tx	(y	y	y	y)	(y	y	y	y	y	y	y	y)	-	-	\$ <sup>(7)</sup>	\$	\$	\$	\$	-	-	-			

- (1) "AND" is assumed horizontally across the table between all conditions which form the statistic (marked y or n) except where (y|y), meaning "OR" is indicated. Parentheses are significant.
- (2) "-" indicates conditions which are ignored in the formations of the statistic.
- (3) Statistics marked "Rx+Tx" are formed by summing the Rx and Tx statistics, each of which is formed independently.
- (4) Pause (MAC) frames are issued in the MAC as perfect (no CRC error) 64 byte frames in full duplex only, so they cannot collide.
- (5) "%" If a CPU sourced MAC control frame has a multicast or broadcast destination address then the appropriate statistics will be updated.
- (6) "+" indicates collisions which are "summed" (i.e. every collision is counted in the Collisions statistic). Jam sequences used for halfduplex flow control are also counted.
- (7) "\$" Every byte written on the wire during each retry attempt is also counted in addition to frames which experience no collisions or carrier loss.
- (8) The flow collision type is for half-duplex collisions forced by the MAC to achieve flow control. Some of the '-'s in this column might in reality be 'n's. To prevent double-counting, Net Octets are unaffected by the jam sequence – the 'received' bytes, however, are counted. (See Rx Statistics Summary.)
- (9) When the transmit Tx FIFO is drained due to the MAC being disabled or link being lost, then the frames being purged will not appear in the Tx statistics.

#### 12.2.1.4.6.18.9.1 Rx + Tx 64 Octet Frames (Offset = 3A068h - Port 0 or Offset = 3A268h - Port 1)

#### All ports

The total number of 64-byte frames received and transmitted on the port. Such a frame is defined to be:

- Any data or MAC control frame which was destined for any unicast, broadcast or multicast address
- Did not experience late collisions, excessive collisions, or carrier sense error
- Was exactly 64 bytes long. (If the frame was being transmitted and experienced carrier loss that resulted in a frame of this size being transmitted, then the frame will be recorded in this statistic).

CRC errors, code/align errors and overruns do not affect the recording of frames in this statistic.

#### **12.2.1.4.6.18.9.2 Rx + Tx 65–127 Octet Frames (Offset = 3A06Ch - Port 0 or Offset = 3A26Ch - Port 1)**

##### **All ports**

The total number of frames of size 65 to 127 bytes received and transmitted on the port. Such a frame is defined to be:

- Any data or MAC control frame which was destined for any unicast, broadcast or multicast address
- Did not experience late collisions, excessive collisions, or carrier sense error
- Was 65 to 127 bytes long

CRC errors, code/align errors, underruns and overruns do not affect the recording of frames in this statistic.

#### **12.2.1.4.6.18.9.3 Rx + Tx 128–255 Octet Frames (Offset = 3A070h - Port 0 or Offset = 3A270h - Port 1)**

##### **All ports**

The total number of frames of size 128 to 255 bytes received and transmitted on the port. Such a frame is defined to be:

- Any data or MAC control frame which was destined for any unicast, broadcast or multicast address
- Did not experience late collisions, excessive collisions, or carrier sense error
- Was 128 to 255 bytes long

CRC errors, code/align errors, underruns and overruns do not affect the recording of frames in this statistic.

#### **12.2.1.4.6.18.9.4 Rx + Tx 256–511 Octet Frames (Offset = 3A074h - Port 0 or Offset = 3A274h - Port 1)**

##### **All ports**

The total number of frames of size 256 to 511 bytes received and transmitted on the port. Such a frame is defined to be:

- Any data or MAC control frame which was destined for any unicast, broadcast or multicast address
- Did not experience late collisions, excessive collisions, or carrier sense error
- Was 256 to 511 bytes long

CRC errors, code/align errors, underruns and overruns do not affect the recording of frames in this statistic.

#### **12.2.1.4.6.18.9.5 Rx + Tx 512–1023 Octet Frames (Offset = 3A078h - Port 0 or Offset = 3A278h - Port 1)**

##### **All ports**

The total number of frames of size 512 to 1023 bytes received and transmitted on the port. Such a frame is defined to be:

- Any data or MAC control frame which was destined for any unicast, broadcast or multicast address
- Did not experience late collisions, excessive collisions, or carrier sense error
- Was 512 to 1023 bytes long

CRC errors, code/align errors, underruns and overruns do not affect the recording of frames in this statistic.

#### **12.2.1.4.6.18.9.6 Rx + Tx 1024\_Up Octet Frames (Offset = 3A07Ch - Port 0 or Offset = 3A27Ch - Port 1)**

##### **All ports**

The total number of frames of size 1024 to RX\_MAXLEN bytes for receive or 1024 up for transmit on the port. Such a frame is defined to be:

- Any data or MAC control frame which was destined for any unicast, broadcast or multicast address
- Did not experience late collisions, excessive collisions, or carrier sense error
- Was 1024 to RX\_MAXLEN bytes long on receive, or any size on transmit

CRC errors, code/align errors, underruns and overruns do not affect the recording of frames in this statistic.

#### 12.2.1.4.6.18.9.7 Net Octets (Offset = 3A080h - Port 0 or Offset = 3A280h - Port 1)

#### All ports

The total number of bytes of frame data received and transmitted on the port. Each frame counted:

- was any data or MAC control frame destined for any unicast, broadcast or multicast address (address match does not matter)
- Any length (including less than 64 bytes and greater than RX\_MAXLEN bytes)

Also counted in this statistic is:

- Every byte transmitted before a carrier-loss was experienced
- Every byte transmitted before each collision was experienced, (that is, multiple retries are counted each time)
- Every byte received if the port is in half-duplex mode until a jam sequence was transmitted to initiate flow control. (The jam sequence was not counted to prevent double-counting)

Error conditions such as alignment errors, CRC errors, code errors, overruns and underruns do not affect the recording of bytes by this statistic.

The objective of this statistic is to give a reasonable indication of Ethernet utilization.

#### 12.2.1.4.6.18.10

**Table 12-187. Rx Statistics Summary**

Rx Statistic	Frame/ Oct	Rx/ Rx+ Tx	Frame Type					Frame Size (bytes)								Event				
			MAC control		Data <sup>(5)</sup>			<64	64	65-127	128-255	256-511	512-1023	1024-rx_maxlen	>rx_maxlen	Flow Coll. (8)	CRC Error	Align/ Code	Overrun	Add. Disc.
			Pause frame	Non-pause <sup>(4)</sup>	Multicast	Broadcast	Unicast													
Good Rx Frames	F	Rx	(y  <sup>(1)</sup> )	y	y	y	y	n	(y	y	y	y	y	y)	n	.(2)	n	n	-	n
Broadcast Rx Frames	F	Rx	(%  (6)	%	n	y)	n	n	(y	y	y	y	y	y)	n	-	n	n	-	n
Multicast Rx Frames	F	Rx	(%	%	y)	n	n	n	(y	y	y	y	y	y)	n	-	n	n	-	n
Pause Rx Frames	F	Rx	y	n	n	n	n	n	(y	y	y	y	y	y)	n	-	n	n	-	-
Rx CRC Errors	F	Rx	(y	y	y	y	y)	n	(y	y	y	y	y	y)	n	-	y	n	-	n
Rx Align/Code Errors	F	Rx	(y	y	y	y	y)	n	(y	y	y	y	y	y)	n	-	-	y	-	n
Oversized Rx Frames	F	Rx	(y	y	y	y	y)	n	n	n	n	n	n	n	y	-	n	n	-	n
Rx Jabbers	F	Rx	(y	y	y	y	y)	n	n	n	n	n	n	n	y	-	(y	y)	-	n
Undersized Rx Frames	F	Rx	n	n	(y	y	y)	y	n	n	n	n	n	n	n	-	n	n	-	n
Rx Fragments	F	Rx	n	n	(y	y	y)	y <sup>(7)</sup>	n	n	n	n	n	n	n	-	(y	y)	-	-
Rx Overruns <sup>(9)</sup>	F	Rx	(y	y	y	y	y)	(y	y	y	y	y	y	y)	y)	-	-	-	y	n
64octet Frames	F	Rx+Tx <sup>(3)</sup>	(y	y	y	y	y)	n	y	n	n	n	n	n	n	-	-	-	-	n

**Table 12-187. Rx Statistics Summary (continued)**

Rx Statistic	Frame/ Oct	Rx/ Rx+ Tx	Frame Type					Frame Size (bytes)										Event				
			MAC control		Data <sup>(5)</sup>			<64	64	65-127	128-255	256-511	512-1023	1024-rx_max len	>rx_max len	Flow Coll. <sup>(8)</sup>	CRC Error	Align/ Code	Overrun	Add. Disc.		
			Pause frame	Non-pause <sup>(4)</sup>	Multicast	Broadcast	Unicast															
65-127octet Frames	F	Rx+Tx	(y	y	y	y	y)	n	n	y	n	n	n	n	n	-	-	-	-	n		
128-255octet Frames	F	Rx+Tx	(y	y	y	y	y)	n	n	n	y	n	n	n	n	-	-	-	-	n		
256-511octet Frames	F	Rx+Tx	(y	y	y	y	y)	n	n	n	n	y	n	n	n	-	-	-	-	n		
512-1023octet Frames	F	Rx+Tx	(y	y	y	y	y)	n	n	n	n	n	y	n	n	-	-	-	-	n		
1024-UPoctet Frames	F	Rx+Tx	(y	y	y	y	y)	n	n	n	n	n	n	y	n	-	-	-	-	n		
Rx Octets	O	Rx	(y	y	y	y	y)	n	(y	y	y	y	y	y)	n	-	n	n	-	n		
Net Octets	O	Rx+Tx	(y	y	y	y	y)	(y	y	y	y	y	y	y	y)	y)	-	-	-	-		

- (1) "AND" is assumed horizontally across the table between all conditions which form the statistic (marked y or n) except where (y|y), meaning "OR" is indicated. Parentheses are significant.
- (2) "-" indicates conditions which are ignored in the formations of the statistic.
- (3) Statistics marked "Rx+Tx" are formed by summing the Rx and Tx statistics, each of which is formed independently.
- (4) The non-pause column refers to all MAC control frames (for example, frames with length/type=88.08) with opcodes other than 0x0001. The pauseframe column refers to MAC frames with the opcode=0x0001.
- (5) The multicast, broadcast and unicast columns in the table refer to non-MAC Control/non-pause frames (i.e. data frames).
- (6) "%" If either a MAC control frame or pause frame has a multicast or broadcast destination address then the appropriate statistics will be updated.
- (7) "y^" Frame fragments are not counted if less than 8 bytes.
- (8) Flow coll. are half-duplex collisions forced by the MAC to achieve flow-control. A collision will be forced during the first 8 bytes so should not show in frame fragments. Some of the '-'s in this column might in reality be 'n's.
- (9) The rx\_overruns stat is for RX\_MOF\_OVERRUNS and RX\_SOF\_OVERRUNS added together.

**Table 12-188. Tx Statistics Summary**

Tx Statistic <sup>(9)</sup>	Fra me/ Oct	Tx/ Rx +Tx	Frame Type					Frame Size (bytes)							Event											
			MAC control (4)		Data			64	65- 127	128- 255	256- 511	512- 1023	1024- 1535	>1535	CRC Error	Collision Type					No Carrier	Queue	Deferred	Under run		
			Pause- MAC	Any- CPU	Multi- cast	Broad- cast	Uni- cast									Flow w <sup>(8)</sup>	1	2-15	16	Late						
Good Tx Frames	F	Tx	(y  (1)	y	y	y	y)	(y	y	y	y	y	y	y)	-(2)	-	-	-	n	n	n	-	-	n		
Broadcast Tx Frames	F	Tx	n	(%  (5)	n	y)	n	(y	y	y	y	y	y	y)	-	-	-	-	n	n	n	-	-	n		
Multicast Tx Frames	F	Tx	(y	%	y)	n	n	(y	y	y	y	y	y	y)	-	-	-	-	n	n	n	-	-	n		
Pause Tx Frames	F	Tx	y	n	n	n	n	y	n	n	n	n	n	n	-	-	-	-	-	-	-	-	-	-		
Collisions	F	Tx	n	(y	y	y	y)	(y	y	y	y	y	y	y)	-	(+ (6)	+	+	+	+	n	-	-	-		
Single Collision Tx Frames	F	Tx	n	(y	y	y	y)	(y	y	y	y	y	y	y)	-	-	y	n	n	n	n	-	-	-		



**Table 12-188. Tx Statistics Summary (continued)**

Tx Statistic <sup>(9)</sup>	Fra me/ Oct	Tx/ Rx +Tx	Frame Type					Frame Size (bytes)										Event									
			MAC control (4)		Data			64	65- 127	128- 255	256- 511	512- 1023	1024- 1535	>1535	CR C Error	Collision Type					No Carrier	Queue d	Def erred	Under run			
			Pause- MAC	Any- CPU	Multi- cast	Broadcast	Unicast									Flow <sup>(8)</sup>	1	2-15	16	Late							
Multiple Collision Tx Frames	F	Tx	n	(y	y	y	y)	(y	y	y	y	y	y	y)	-	-	n	y	n	n	n	-	-	-			
Excessive Collisions	F	Tx	n	(y	y	y	y)	(y	y	y	y	y	y	y)	-	-	n	n	y	n	n	-	-	-			
Late Collisions	F	Tx	n	(y	y	y	y)	n	(y	y	y	y	y	y)	-	-	-	-	-	y	-	-	-	-			
Deferred Tx Frames	F	Tx	n	(y	y	y	y)	(y	y	y	y	y	y	y)	-	-	n	n	n	n	n	-	y	n			
Carrier Sense Errors	F	Tx	(y	y	y	y	y)	(y	y	y	y	y	y	y)	-	-	-	-	-	-	y	-	-	-			
64octet Frames	F	Rx+ Tx <sup>(3)</sup>	(y	y	y	y	y)	y	n	n	n	n	n	n	-	-	-	-	n	n	n	-	-	-			
65-127octet Frames	F	Rx+ Tx	(y	y	y	y	y)	n	y	n	n	n	n	n	-	-	-	-	n	n	n	-	-	-			
128-255octet Frames	F	Rx+ Tx	(y	y	y	y	y)	n	n	y	n	n	n	n	-	-	-	-	n	n	n	-	-	-			
256-511octet Frames	F	Rx+ Tx	(y	y	y	y	y)	n	n	n	y	n	n	n	-	-	-	-	n	n	n	-	-	-			
512-1023octet Frames	F	Rx+ Tx	(y	y	y	y	y)	n	n	n	n	y	n	n	-	-	-	-	n	n	n	-	-	-			
1024-UPoctet Frames	F	Rx+ Tx	(y	y	y	y	y)	n	n	n	n	n	y	y	-	-	-	-	n	n	n	-	-	-			
Tx Octets	O	Tx	(y	y	y	y	y)	(y	y	y	y	y	y	y)	-	-	-	-	n	n	n	-	-	n			
Net Octets	O	Rx+ Tx	(y	y	y	y	y)	(y	y	y	y	y	y	y)	-	-	\$ <sup>(7)</sup>	\$	\$	\$	\$	-	-	-			

- (1) "AND" is assumed horizontally across the table between all conditions which form the statistic (marked y or n) except where (y|y), meaning "OR" is indicated. Parentheses are significant.
- (2) "-" indicates conditions which are ignored in the formations of the statistic.
- (3) Statistics marked "Rx+Tx" are formed by summing the Rx and Tx statistics, each of which is formed independently.
- (4) Pause (MAC) frames are issued in the MAC as perfect (no CRC error) 64 byte frames in full duplex only, so they cannot collide.
- (5) "%" If a CPU sourced MAC control frame has a multicast or broadcast destination address then the appropriate statistics will be updated.
- (6) "+" indicates collisions which are "summed" (i.e. every collision is counted in the Collisions statistic). Jam sequences used for halfduplex flow control are also counted.
- (7) "\$" Every byte written on the wire during each retry attempt is also counted in addition to frames which experience no collisions or carrier loss.
- (8) The flow collision type is for half-duplex collisions forced by the MAC to achieve flow control. Some of the '-'s in this column might in reality be 'n's. To prevent double-counting, Net Octets are unaffected by the jam sequence – the 'received' bytes, however, are counted. (See [Table 12-187](#).)
- (9) When the transmit Tx FIFO is drained due to the MAC being disabled or link being lost, then the frames being purged will not appear in the Tx statistics.

### 12.2.1.4.7 Common Platform Time Sync (CPTS)

The Common Platform Time Sync (CPTS) module is used to facilitate host control of time sync operations. It enables compliance with the IEEE 1588-2008 standard for a precision clock synchronization protocol.

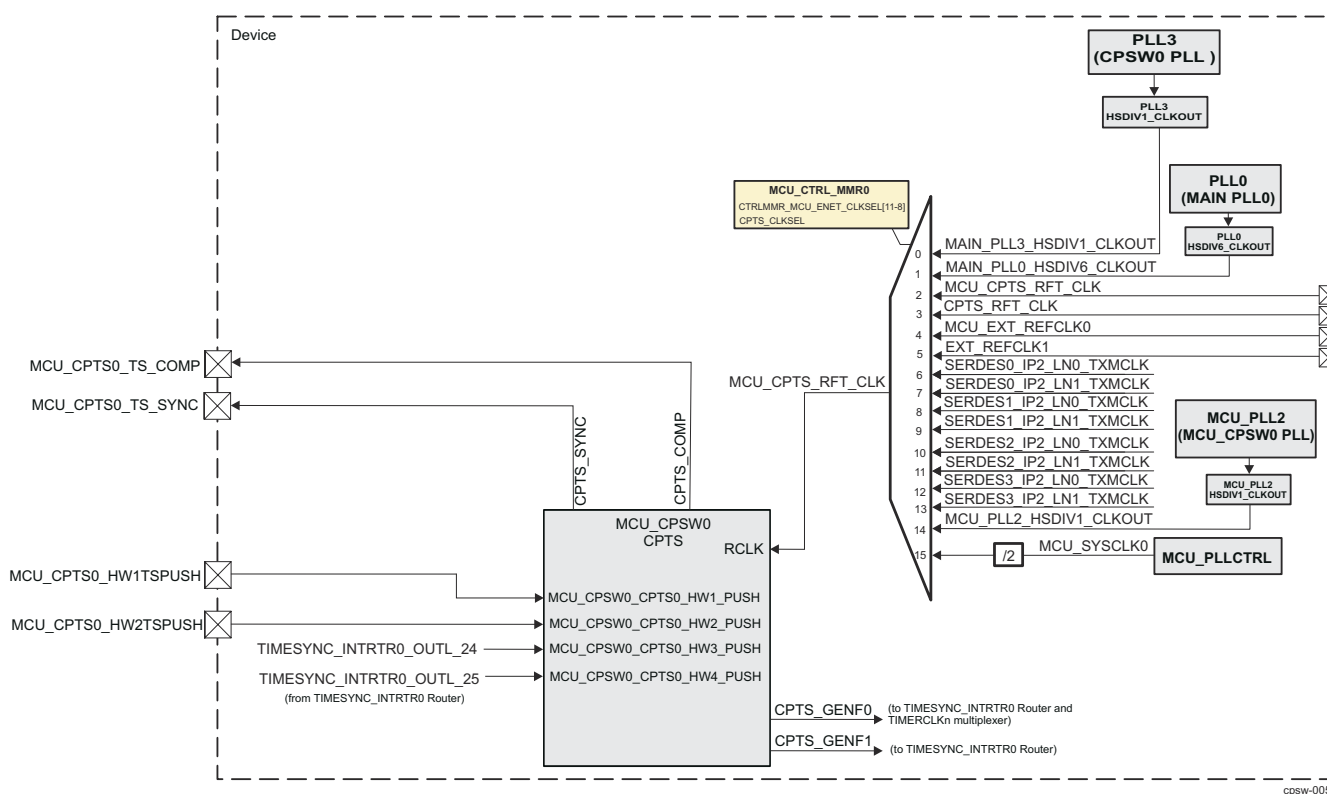
Main features of CPTS module are:

- Supports the selection of multiple external clock sources
- Software control of time sync events via interrupt or polling
- Supports up to 4 hardware timestamp push inputs
- Supports timestamp counter compare output (CPTS\_COMP)
- Supports timestamp counter bit output (CPTS\_SYNC)
- Supports a configurable number of timestamp Generator bit outputs (CPTS\_GENFn).
- Supports Ethernet Enhanced Scheduled Traffic Operations (CPTS\_ESTFn).
- 32-bit and 64-bit timestamp modes with PPM and nudge adjustment.

#### 12.2.1.4.7.1 MCU\_CPSW0 CPTS Integration

This section describes CPTS module integration in the MCU\_CPSW0 module, including information about clocks, resets, and hardware requests.

Figure 12-141 shows CPTS integration in the device MCU\_CPSW0 module.



**Figure 12-141. MCU\_CPSW0 CPTS Integration**

CPTS IEEE 1588 clock (RCLK) is selected through the CTRLMMR\_MCU\_ENET\_CLKSEL register.

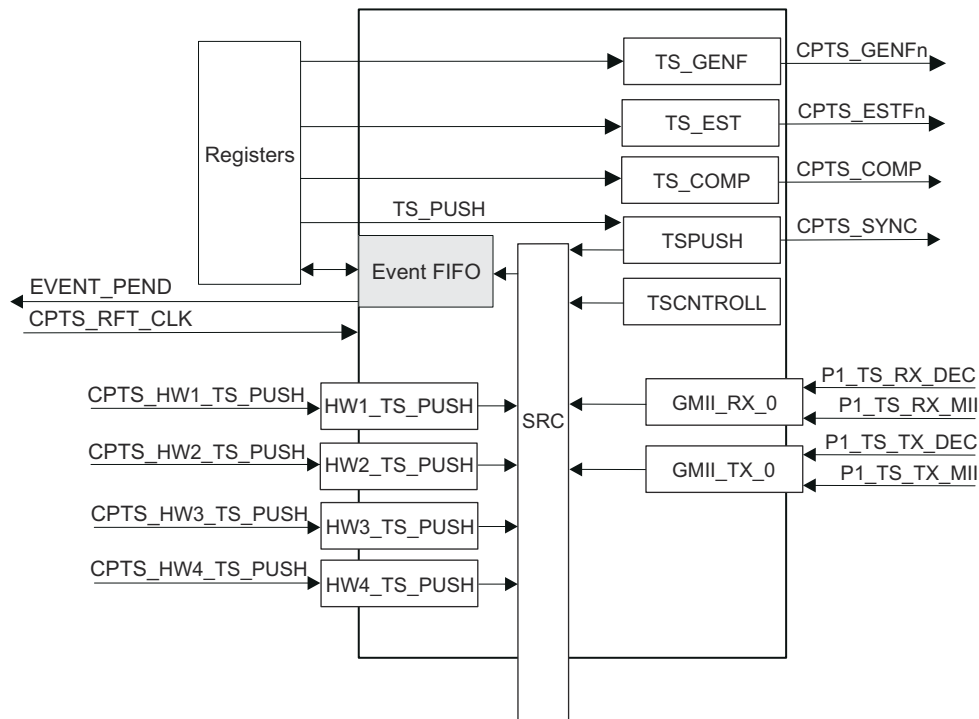
#### Note

For more information about CPTS clocks and resets, see *MCU\_CPSW0 Integration*.

#### 12.2.1.4.7.2 CPTS Architecture

Figure 12-142 shows the architecture of the CPTS module inside the CPSW Ethernet Subsystem. Time stamp values for every packet transmitted or received on external port of the CPSW are recorded. At the same time, each packet is decoded to determine if it is a valid time sync event. If so, an event is loaded into the Event FIFO for processing containing the recorded time stamp value when the packet was transmitted or received.

In addition, both hardware (HWN\_TS\_PUSH) and software (TS\_PUSH) can be used to read the current time stamp value through the Event FIFO. The reference clock used for the time stamp (CPTS\_RFT\_CLK) can be derived from several sources.



Note: CPTS\_HW1\_TS\_PUSH to CPTS\_HW4\_TS\_PUSH events are used for CPTS implemented in MCU\_CPSW0 module.

cpsw-013

**Figure 12-142. CPTS Block Diagram**

#### Note

See *MCU\_CPSW0 CPTS Integration* for CPTS integration in the device MCU\_CPSW0 module.

#### 12.2.1.4.7.3 CPTS Initialization

The CPTS module should be configured as follows:

1. Reset the CPTS module.
2. Write the CPTS\_CLKSEL value in the CTRLMMR\_MCU\_ENET\_CLKSEL register with the desired reference clock selection. This value is allowed to be written only when the CPTS\_EN bit in the CPSW\_CPTS\_CONTROL\_REG register is cleared to zero.
3. Set the CPTS\_EN bit in the CPSW\_CPTS\_CONTROL\_REG register.
4. If using interrupts and not polling, enable the interrupt by setting the TS\_PEND\_EN bit in the CPSW\_CPTS\_INT\_ENABLE\_REG register.

#### 12.2.1.4.7.4 32-bit Time Stamp Value

The time stamp value is a 32-bit value that increments on each CPTS\_RFT\_CLK rising edge when CPTS\_EN bit is set to 1h. When CPTS\_EN bit is cleared to 0h, the time stamp value is reset to 0h.

If more than 32-bits of time stamp are required by the application, the host software must maintain the necessary number of upper bits. The upper time stamp value should be incremented by the host when the rollover event is detected.

For test purposes, the time stamp can be written via the time stamp load function (CPSW\_CPTS\_TS\_LOAD\_VAL\_REG / CPSW\_CPTS\_TS\_LOAD\_HIGH\_VAL\_REG and CPSW\_CPTS\_TS\_LOAD\_EN\_REG registers).

#### 12.2.1.4.7.5 64-bit Time Stamp Value

The time stamp value is a 64-bit value that increments on each CPTS\_RFT\_CLK rising edge when CPTS\_EN bit is set to 1h. When CPTS\_EN bit is cleared to 0h, the time stamp value is reset to 0h.

64-bit mode is selected when CPSW\_CPTS\_CONTROL\_REG[5] MODE bit set to 1h.

For test purposes, the time stamp value can be written via the time stamp load function (CPSW\_CPTS\_TS\_LOAD\_EN\_REG, CPSW\_CPTS\_TS\_LOAD\_VAL\_REG, and CPSW\_CPTS\_TS\_LOAD\_HIGH\_VAL\_REG registers). The CPSW\_CPTS\_TS\_ADD\_VAL\_REG feature is included to allow 1ns timestamp operations with an CPTS\_RFT\_CLK rate less than 1Ghz. [Table 12-189](#) shows the CPTS\_RFT\_CLK and CPSW\_CPTS\_TS\_ADD\_VAL\_REG values for 1ns operations.

**Table 12-189. ADD\_VAL feature**

CPTS_RFT_CLK (MHz)	CPSW_CPTS_TS_ADD_VAL_REG[2-0] ADD_VAL
1 GHz	0
500 MHz	1
333.33 MHz	2
250 MHz	3
200 MHz	4
166.66 MHz	5
142.85714 MHz	6
125 MHz	7

#### 12.2.1.4.7.6 64-Bit Timestamp Nudge

The 64-bit TIME\_STAMP value can be adjusted by writing the CPSW\_CPTS\_TS\_NUDGE\_VAL\_REG[7-0] TS\_NUDGE\_VAL bit field value which is a two's complement value. A value of FFh will subtract 1 clock cycle from the next incremented 64-bit time stamp value (CPSW\_CPTS\_EVENT\_0\_REG[31-0] TIME\_STAMP and CPSW\_CPTS\_EVENT\_3\_REG[31-0] TIME\_STAMP value). A nudge value of 1h will add 1 clock cycle to the next incremented TIME\_STAMP[63-0] value. For example, if the current TIME\_STAMP value is F06h, and CPSW\_CPTS\_TS\_ADD\_VAL\_REG[2-0] ADD\_VAL = 3h, the next incremented timestamp value would be F0Ah without a nudge and F0Ah +/- [7-0] TS\_NUDGE\_VAL with a nudge. The [7-0] TS\_NUDGE\_VAL value is cleared to zero when the nudge has occurred.

#### 12.2.1.4.7.7 64-bit Timestamp PPM

The 64-bit TIME\_STAMP can be adjusted by parts per million or by parts per hour. Writing a non-zero value to the CPSW\_CPTS\_TS\_PPM\_LOW\_VAL\_REG[31-0] TS\_PPM\_LOW\_VAL (Time stamp PPM Low value) and CPSW\_CPTS\_TS\_PPM\_HIGH\_VAL\_REG[9-0] TS\_PPM\_HIGH\_VAL (Time stamp PPM High value) enables PPM operations. The adjustment is up or down depending on the [7] TS\_PPM\_DIR bit in the CPSW\_CPTS\_CONTROL\_REG register. The TIME\_STAMP value is increased by the PPM value when [7] TS\_PPM\_DIR bit is cleared. The TIME\_STAMP value is decreased by the PPM value when [7] TS\_PPM\_DIR bit is set.

#### Parts Per Million example:

To adjust for 100 parts per million the configured value for TS\_PPM[41-0] (through CPSW\_CPTS\_TS\_PPM\_LOW\_VAL\_REG[31-0] TS\_PPM\_LOW\_VAL and CPSW\_CPTS\_TS\_PPM\_HIGH\_VAL\_REG[9-0] TS\_PPM\_HIGH\_VAL) is:  
 $1,000,000/100 = 10,000(\text{decimal})$

#### Parts Per Hour example:

To adjust for 1 part per hour at 1 GHz CPTS\_RFT\_CLK the configured value for TS\_PPM[41-0] (through CPSW\_CPTS\_TS\_PPM\_LOW\_VAL\_REG[31-0] TS\_PPM\_LOW\_VAL and CPSW\_CPTS\_TS\_PPM\_HIGH\_VAL\_REG[9-0] TS\_PPM\_HIGH\_VAL) is:  
 $(1,000,000,000\text{Hz}/1\text{pph}) * (3600 \text{ seconds/hour}) = 34630\text{B8A000} (\text{hex})$

#### 12.2.1.4.7.8 Event FIFO

All time sync events are pushed onto the Event FIFO. There are 32 locations in the event FIFO with no overrun indication supported. Software must service the event FIFO in a timely manner to prevent FIFO overrun.

#### 12.2.1.4.7.9 Timestamp Compare Output

CPTS features one Time Stamp Compare (CPTS\_COMP) output. The CPTS\_COMP function is a software oriented feature that is intended to be replaced going forward by the hardware oriented GENF function. CPTS\_COMP is not compatible with timestamp PPM or a non-zero CPSW\_CPTS\_TS\_ADD\_VAL\_REG[2-0] ADD\_VAL value.

##### 12.2.1.4.7.9.1 Non-Toggle Mode: 32-bit

The CPTS\_COMP output is asserted for CPSW\_CPTS\_TS\_COMP\_LEN\_REG[31-0] TS\_COMP\_LENGTH periods when the CPSW\_CPTS\_EVENT\_0\_REG[31-0] TIME\_STAMP value (lower 32-bits) compares with the CPSW\_CPTS\_TS\_COMP\_VAL\_REG[31-0] TS\_COMP\_VAL and the length value is non-zero. The CPTS\_COMP rising edge occurs three CPTS\_RFT\_CLK clock periods after the values compare. A timestamp compare event is pushed into the event FIFO when CPTS\_COMP is asserted. The polarity of the CPTS\_COMP output is determined by the CPSW\_CPTS\_CONTROL\_REG[2] TS\_COMP\_POLARITY bit. The output is asserted low when the polarity bit is 0h.

##### 12.2.1.4.7.9.2 Non-Toggle Mode: 64-bit

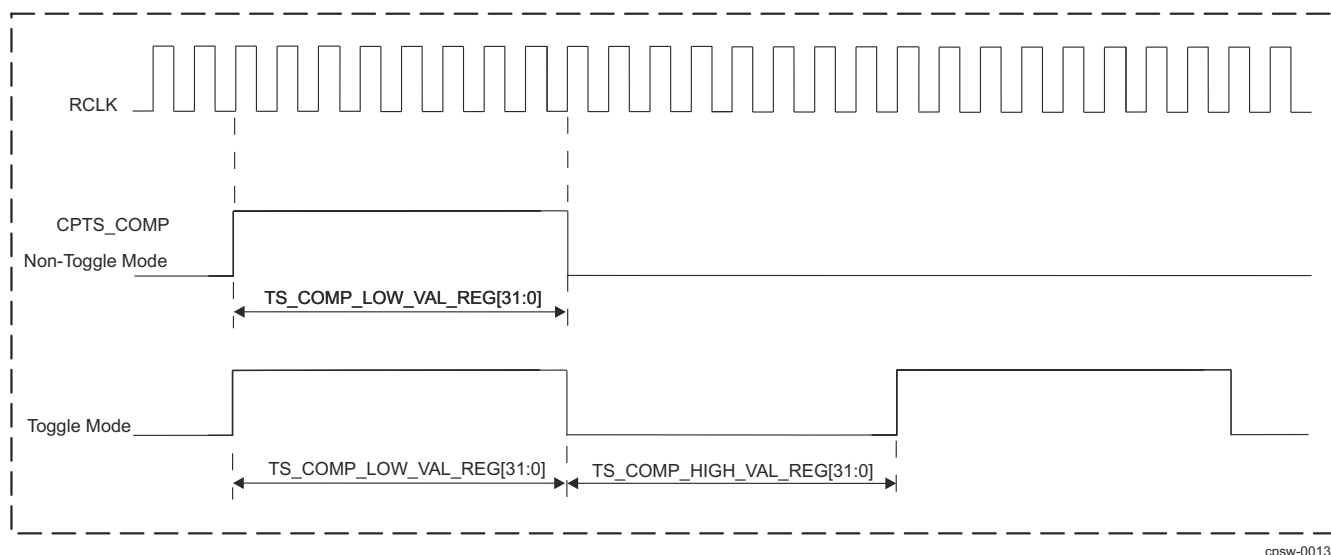
64-bit mode operation is identical to 32-bit mode except that all 64-bits of the TIME\_STAMP are used (CPSW\_CPTS\_EVENT\_0\_REG and CPSW\_CPTS\_EVENT\_3\_REG). In 32-bit mode only the lower 32-bits (CPSW\_CPTS\_EVENT\_0\_REG) are used.

##### 12.2.1.4.7.9.3 Toggle Mode: 32-bit

The CPTS\_COMP output is asserted (CPSW\_CPTS\_TS\_COMP\_LEN\_REG[31-0] TS\_COMP\_LENGTH) for CPTS\_RFT\_CLK clock periods when the TIME\_STAMP[31:0] value compares with the CPSW\_CPTS\_TS\_COMP\_VAL\_REG and the length value is non-zero. The CPTS\_COMP toggles thereafter on CPSW\_CPTS\_TS\_COMP\_VAL\_REG[31-0] TS\_COMP\_LENGTH for CPTS\_RFT\_CLK periods. The length high or low can be adjusted by writing the CPSW\_CPTS\_TS\_COMP\_NUDGE\_REG[7-0] NUDGE bit field value which is a two's complement value. A value of FFh will subtract one CPTS\_RFT\_CLK period from the CPSW\_CPTS\_TS\_COMP\_VAL\_REG[31-0] TS\_COMP\_LENGTH value. A value of 0x01h will add one CPTS\_RFT\_CLK period to the CPSW\_CPTS\_TS\_COMP\_LEN\_REG[31-0] TS\_COMP\_LENGTH value. Only a single high or low time is adjusted (nudged) and the CPSW\_CPTS\_TS\_COMP\_NUDGE\_REG[7-0] NUDGE value is cleared to zero when the nudge has occurred. The CPTS\_COMP output is asserted low when the CPSW\_CPTS\_CONTROL\_REG[2] TS\_COMP\_POLARITY bit is 0h. No compare events and no CPTS\_EVNT interrupts are generated in toggle mode. The CPSW\_CPTS\_CONTROL\_REG[6] TS\_COMP\_TOG bit must be set for toggle mode (value 1h). Note this bit must be set before writing a non-zero value to CPSW\_CPTS\_TS\_COMP\_VAL\_REG register.

#### 12.2.1.4.7.9.4 Toggle Mode: 64-bit

64-bit mode operation is identical to 32-bit mode except that all 64-bits of the **TIME\_STAMP** are used (**CPSW\_CPTS\_EVENT\_0\_REG** and **CPSW\_CPTS\_EVENT\_3\_REG**). In 32-bit mode only the lower 32-bits (**CPSW\_CPTS\_EVENT\_0\_REG**) are used.



**Figure 12-143. CPTS\_COMP Output in Toggle and Non-Toggle Mode**

#### 12.2.1.4.7.10 Timestamp Sync Output

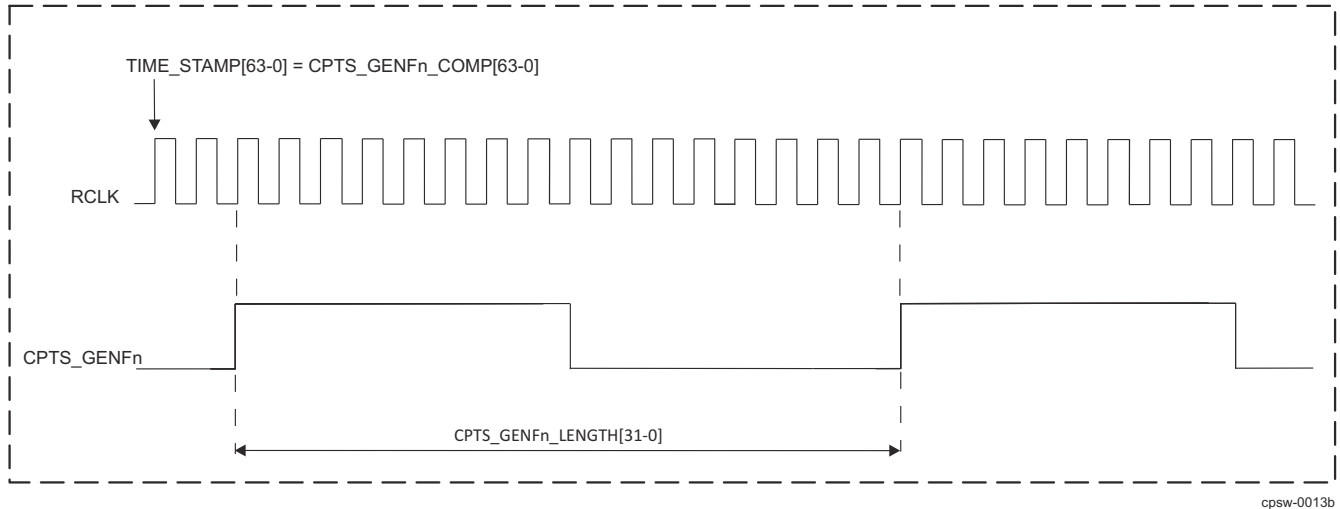
The **CPTS\_SYNC** output is a selected bit of the [31:0]**TIME\_STAMP** counter value. One of bits 17-31 can be selected in **CPSW\_CPTS\_CONTROL\_REG[31:28]** **TS\_SYNC\_SEL**. The **CPTS\_SYNC** output is disabled when **CPSW\_CPTS\_CONTROL\_REG[31:28]** **TS\_SYNC\_SEL** is zero.

If the selected counter bit is 1 at the time when **TS\_SYNC\_SEL** value is written then a rising edge will not occur on the **CPTS\_SYNC** output. A rising edge will occur on the **CPTS\_SYNC** output upon the next transition to 1 of the selected counter bit. The **TS\_SYNC\_SEL** value must be written to zero before changing to a different non-zero value. No events are generated due to the **CPTS\_SYNC** operation. The **CPTS\_SYNC** output is two **CPTS\_RFT\_CLK** periods after the actual count value.

#### 12.2.1.4.7.11 Timestamp GENFn Output

The **CPTS\_GENFn** outputs have a programmable cycle (frequency) with a PPM feature and software nudge feature. The **CPTS\_GENFn** output cycle is **CPSW\_GENF0\_LENGTH\_REG\_L[31:0]** **CPTS\_RFT\_CLK** periods (which is different than **CPTS\_COMP** operation). [Figure 12-144](#) represents the **CPTS\_GENFn** output signal.

The **CPTS\_GENFn** output cycle is **CPSW\_GENF0\_LENGTH\_REG\_L[31:0]** **CPTS\_RFT\_CLK** periods beginning when the 64-bit **TIME\_STAMP** value compares with the 64-bit **GENFn\_COMP** value (**CPSW\_GENF0\_COMP\_LOW\_REG\_L** and **CPSW\_GENF0\_COMP\_HIGH\_REG\_L** registers) and the length value is non-zero. The **CPTS\_GENFn** output cycle repeats thereafter every **CPSW\_GENF0\_LENGTH\_REG\_L[31:0]** **CPTS\_RFT\_CLK** periods. The upper 32-bit word should be written first for 64-bit values. The length should be zero while the comparison value and other configuration parameters are being configured. The length should be written non-zero to enable operations last. The first cycle after comparison is active high when the **CPSW\_CPTS\_CONTROL\_REG[2]** **TS\_COMP\_POLARITY** bit is low. No compare events and no **CPTS\_EVT** interrupts are generated.



**Figure 12-144. CPTS\_GENFn Output Signal Diagram**

#### 12.2.1.4.7.11.1 GENFn Nudge

The cycle length can be adjusted by writing the CPSW\_CPTS\_TS\_COMP\_NUDGE\_REG[7-0] NUDGE register value which is a two's complement value. A value of FFh will subtract 1 CPTS\_RFT\_CLK from the CPSW\_GENF0\_LENGTH\_REG\_L[31-0] value. A value of 1h will add 1 CPTS\_RFT\_CLK to the CPSW\_GENF0\_LENGTH\_REG\_L[31-0] value. The CPSW\_CPTS\_TS\_COMP\_NUDGE\_REG[7-0] NUDGE value is cleared to zero when the nudge has occurred.

#### 12.2.1.4.7.11.2 GENFn PPM

The CPTS\_GENFn output cycle can be adjusted by parts per million or by parts per hour. Writing a non-zero value to CPSW\_GENF0\_PPM\_LOW\_REG\_L/ CPSW\_GENF0\_PPM\_HIGH\_REG\_L enables PPM operations. The PPM counter continually loads and decrements to zero and then loads again. A single CPTS\_RFT\_CLK adjustment is made when the PPM counter decrements to zero. The adjustment is up or down depending on the CPSW\_GENF0\_TS\_GENF\_CONTROL\_REG[0] PPM\_DIR bit. When PPM\_DIR bit is set a single CPTS\_RFT\_CLK time is subtracted from the generate function counter which has the effect of increasing the generate function frequency by the PPM amount. When PPM\_DIR bit is cleared a single CPTS\_RFT\_CLK time is added to the generate function counter which has the effect of decreasing the generate function frequency by the PPM amount.

#### Parts Per Million example:

To adjust for 100 parts per million the configured value for GENF\_PPM[41-0] (through CPSW\_GENF0\_PPM\_LOW\_REG\_L and CPSW\_GENF0\_PPM\_HIGH\_REG\_L) is:  
 $1,000,000/100 = 10,000$ (decimal)

#### Parts Per Hour example:

To adjust for 1 part per hour at 1 GHz CPTS\_RFT\_CLK the configured value for GENF\_PPM[41-0] (through CPSW\_GENF0\_PPM\_LOW\_REG\_L and CPSW\_GENF0\_PPM\_HIGH\_REG\_L) is:  
 $(1,000,000,000\text{Hz}/1\text{pph}) * (3600 \text{ seconds/hour}) = 34630\text{B8A000}$  (hex)

#### 12.2.1.4.7.12 Timestamp ESTFn

Each Ethernet port has a dedicated ESTFn generator which operates identically to the GENFn function.

#### 12.2.1.4.7.13 Time Sync Events

Time Sync events are 96-bit values that are pushed onto the event FIFO and read by software in 32-bit reads. Four 32-bit registers, CPSW\_CPTS\_EVENT\_0\_REG through CPSW\_CPTS\_EVENT\_3\_REG hold the data of a time sync event. There are eight types of sync events:

- Time Stamp Push Event



- Time Stamp Counter Rollover Event (32-bit mode only)
- Time Stamp Counter Half-rollover Event (32-bit mode only)
- Hardware Time Stamp Push Event
- Ethernet Receive Event
- Ethernet Transmit Event
- Time Stamp Compare Event
- Host Transmit Event

#### **12.2.1.4.7.13.1 Time Stamp Push Event**

Software can obtain the current time stamp value (at the time of the write) by initiating a time stamp push event. The push event is initiated by setting the [0]TS\_PUSH bit of the CPSW\_CPTS\_TS\_PUSH\_REG register. The time stamp value is returned in the event, along with a time stamp push event code. The upper 32-bits (CPSW\_CPTS\_EVENT\_3\_REG register) of the timestamp are zero in 32-bit mode.

#### **12.2.1.4.7.13.2 Time Stamp Counter Rollover Event (32-bit mode only)**

The CPTS module contains a 32-bit time stamp value (CPSW\_CPTS\_EVENT\_0\_REG). The counter upper bits are maintained by host software. The rollover event indicates to software that the time stamp counter has rolled over from 0xFFFF FFFF to 0x0000 0000 and the software-maintained upper count value should be incremented. This event occurs only in 32-bit mode.

#### **12.2.1.4.7.13.3 Time Stamp Counter Half-rollover Event (32-bit mode only)**

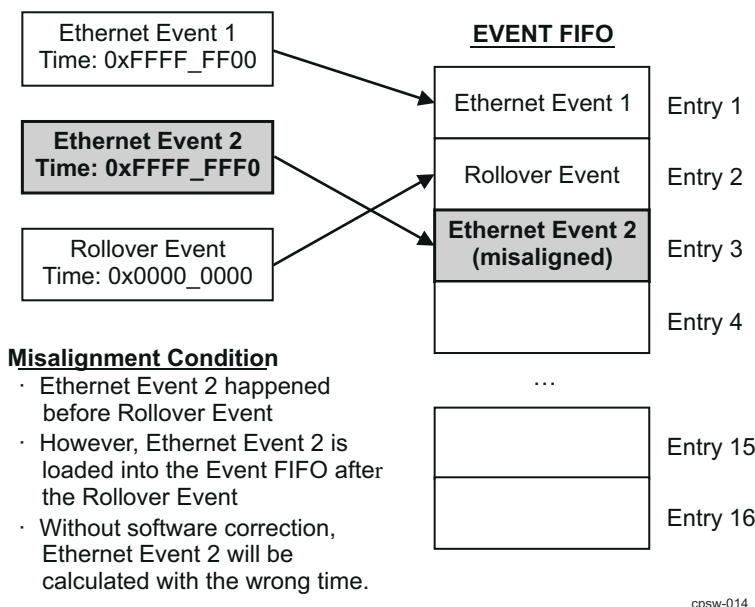
The CPTS includes a time stamp counter half-rollover event. The half-rollover event indicates to software that the time stamp value (CPSW\_CPTS\_EVENT\_0\_REG[31:0] TIME\_STAMP) has incremented from 0x7FFF FFFF to 0x8000 0000. The half-rollover event is included to enable software to correct a misaligned event condition. This event occurs only in 32-bit mode.

The half-rollover event is included to enable software to determine the correct time for each event that contains a valid time stamp value, such as an Ethernet event. If an Ethernet event occurs around a counter rollover (full rollover), the rollover event could possibly be loaded into the event FIFO before the Ethernet event, even though the Ethernet event time was actually taken before the rollover. [Figure 12-145](#) shows a misalignment condition. This misaligned event condition arises because an Ethernet event time stamp occurs at the beginning of a packet and time passes before the packet is determined to be a valid synchronization packet. The misaligned event condition occurs if the rollover occurs in the middle, after the packet time stamp has been taken, but before the packet has been determined to be a valid time sync packet.

Host software must detect and correct for misaligned event conditions. For every event time stamp after a rollover and before a half-rollover, software must examine the time stamp most significant bit. If bit 31 of the time stamp value is low (0x0000 0000 through 0x7FFF FFFF), then the event time stamp was taken after the rollover and no correction is required. If the value is high (0x8000 0000 through 0xFFFF FFFF), the time stamp value was taken before the rollover and a misalignment is detected. The misaligned case indicates to software that it must subtract one from the upper count value stored in software to calculate the correct time for the misaligned event. The misaligned event occurs only on the rollover boundary and not on the half-rollover boundary. Software only needs to check for misalignment from a rollover event to a half-rollover event.

When a rollover occurs, software increments the software time stamp upper value. The misaligned case indicates to software that the misaligned event time stamp has a valid upper value that is pre-increment, so one must be subtracted from the upper value to allow software to calculate the correct time for the misaligned event.





**Figure 12-145. Event FIFO Misalignment Condition**

#### 12.2.1.4.7.13.4 Hardware Time Stamp Push Event

There are four hardware time stamp inputs ( CPTS\_HW[1:4]\_TS\_PUSH events) that can cause hardware time stamp push events to be loaded into the Event FIFO. Each time stamp input is mapped in the device as shown in *CPSW0 CPTS Integration*. The event is loaded into the event FIFO on the rising edge of the timer, and the PORT\_NUMBER field in the CPSW\_CPTS\_EVENT\_1\_REG register indicates the hardware push input that caused the event (encoded).

The hardware time stamp inputs are asynchronous and are low frequency signals. The CPTS logic synchronizes and performs a rising edge detect on the incoming asynchronous input.

Each hardware time stamp input must be asserted for at least 10 periods of the selected CPTS\_RFT\_CLK clock. Each input can be enabled or disabled by setting the respective bits in the CPSW\_CPTS\_CONTROL\_REG register.

Hardware time stamps are intended to be an extremely low frequency signals, such that the event FIFO does not overrun. Software must keep up with the event FIFO and ensure that there is no overrun, or events will be lost.

#### 12.2.1.4.7.13.5 Ethernet Port Events

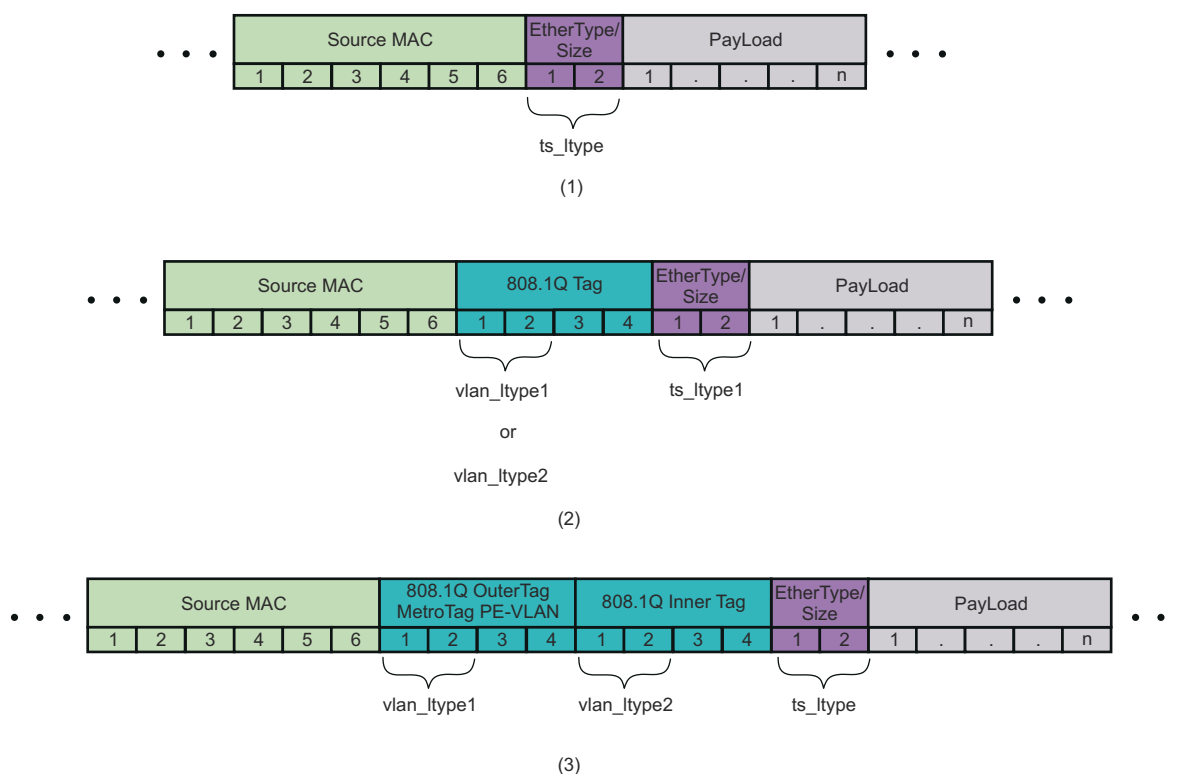
Packets transmitted or received on each Ethernet port can generate Ethernet Transmit Events or Ethernet Receive Events, respectively. The CPTS hardware will decode each packet to determine if it is a valid CPTS time sync event.

According to the IEEE 802.3 Ethernet standard, each Ethernet frame contains a 2-octet EtherType field to indicate which protocol is encapsulated in the PayLoad field, as shown in [Figure 12-146](#). For standard time sync packets, this will contain the EtherType for the Precision Time Protocol (IEEE 1588), which is defined as 0x88F7. The CPTS hardware will compare this field to the TS\_LTYPE1 field in the CPSW\_PN\_TS\_SEQ\_LTYPE\_REG or the TS\_LTYPE2 field in CPSW\_PN\_TS\_CTL\_LTYPE2\_REG register (depending on which enable bit was set) , which should also be programmed to 88F7h.

When a virtual LAN is used, an additional 4-octet 802.1Q tag is inserted in the Ethernet frame before the EtherType field, as shown in [Figure 12-146](#). To indicate to the CPTS hardware that a virtual LAN is in use, the TS\_TX\_VLAN\_LTYPE1\_EN (or TS\_TX\_VLAN\_LTYPE2\_EN) enable bit must be set in the

CPSW\_PN\_TS\_CTL\_REG register. The EtherType for the 802.1Q tag is defined as 0x8100, and the CPTS hardware will compare this value to the TS\_VLAN\_LTYPE1 (or TS\_VLAN\_LTYPE2 depending on which enable bit was set) field in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG register, which should also be programmed to 0x8100.

When two stacked VLANs are used, two additional 4-octet 801.Q tags are inserted in the Ethernet frame before the EtherType field, as shown in Figure 12-146. In this case, both TS\_VLAN\_LTYPE1 and TS\_VLAN\_LTYPE2 must be enabled. The outer tag must match the value of the TS\_VLAN\_LTYPE1 field, and the inner tag must match the value of the TS\_VLAN\_LTYPE2 field.



cpsw-015

**Figure 12-146. Partial Ethernet-II Frames Showing Register Mapping of EtherTypes for a Simple Frame (1), a Single 1Q Tag Added (2), and Two 1Q Tags Added (3)**

#### 12.2.1.4.7.13.5.1 Ethernet Port Receive Event

This section describes Ethernet port receive events. Ethernet port generates time synchronization events for valid received time sync packets. For every packet received on the Ethernet port, a timestamp will be captured by the receive module inside the CPTS for the corresponding port. The time stamp will be captured by the receive module regardless of whether or not the packet is a time synchronization packet to make sure that the time stamp is captured as soon as possible. The packet is sampled on both the rising and falling edges of the CPTS\_RFT\_CLK, and the time stamp will be captured once the start of frame delimiter for the receive packet is detected.

After the time stamp has been captured, the receive interface will begin parsing the packet to determine if it is a valid Ethernet time synchronization packet. The CPSW decoder determines if the packet is a valid Ethernet receive time synchronization event. The receive interface for the port will use the following criteria to determine if the packet is a valid Annex D, Annex E, or Annex F time synchronization Ethernet receive event:

#### Annex D (IPv4)

1. Receive annex D time sync is enabled (TS\_RX\_ANNEX\_D\_EN is set in the CPSW\_PN\_TS\_CTL\_REG register).

2. One of the sequences below is true.
  - a. The first packet LTYPE matches 0x0800
  - b. The first packet LTYPE matches TS\_VLAN\_LTYPE1 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG register and TS\_RX\_VLAN\_LTYPE1\_EN is set in the CPSW\_PN\_TS\_CTL\_REG register and the second packet LTYPE matches 0x0800
  - c. The first packet LTYPE matches TS\_VLAN\_LTYPE2 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG register and TS\_RX\_VLAN\_LTYPE2\_EN is set in the CPSW\_PN\_TS\_CTL\_REG register and the second packet LTYPE matches 0x0800
  - d. The first packet LTYPE matches TS\_VLAN\_LTYPE1 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG register and TS\_RX\_VLAN\_LTYPE1\_EN is set in the CPSW\_PN\_TS\_CTL\_REG register and the second packet LTYPE matches TS\_VLAN\_LTYPE2 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG register and TS\_RX\_VLAN\_LTYPE2\_EN is set in the CPSW\_PN\_TS\_CTL\_REG register and the third packet LTYPE matches 0x0800
3. Byte 14 (the byte after the LTYPE) contains 0x45 (IPv4).

#### Note

The byte numbering assumes that there are no VLANs. The byte number is intended to show the relative order of the bytes.

4. Byte 20 contains 0bXXX00000 (5 lower bits zero) and Byte 21 contains 0x00 (fragment offset zero)
5. Byte 22 contains 0x01 (HOP Limit = 1) if the TS\_TTL\_NONZERO bit in the switch CPSW\_PN\_TS\_CTL\_LTYPE2\_REG register is cleared to 0h, or byte 22 contains any value if CPSW\_PN\_TS\_CTL\_LTYPE2\_REG is set to 1h. Byte 22 is the TTL/HOP field.
6. Byte 23 contains 0x11 (Next Header UDP Fixed).
7. The TS\_UNI\_EN bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG register is cleared to 0h and Bytes 30 through 33 contain:
  - a. Decimal 224.0.1.129 and the TS\_129 bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG register is set, or
  - b. Decimal 224.0.1.130 and the TS\_130 bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG register is set, or
  - c. Decimal 224.0.1.131 and the TS\_131 bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG register is set, or
  - d. Decimal 224.0.1.132 and the TS\_132 bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG register is set, or
  - e. Decimal 224.0.0.107 and the TS\_107 bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG register is set
- OR-
- The TS\_UNI\_EN bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG register is set and Bytes 30 through 33 contain any values.
8. Bytes 36 and 37 contain:
  - a. Decimal 0x01 and 0x3F respectively and the TS\_319 bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG register is set -OR-
  - b. Decimal 0x01 and 0x40 respectively and the TS\_320 bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG register is set.
9. The PTP message begins in byte 42.
10. The packet message type is enabled in the TS\_MSG\_TYPE\_EN field in the CPSW\_PN\_TS\_CTL\_REG register.
11. The packet was received without error (not long/short/mac\_ctl/CRC/code/align).

#### Annex E (IPv6)

1. Receive annex E time sync is enabled (TS\_RX\_ANNEX\_E\_EN bit is set in the switch CPSW\_PN\_TS\_CTL\_REG register).
2. One of the sequences below is true.
  - a. The first packet LTYPE matches 0x86dd.
  - b. The first packet LTYPE matches TS\_VLAN\_LTYPE1 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG register and TS\_RX\_VLAN\_LTYPE1\_EN is set in the CPSW\_PN\_TS\_CTL\_REG register and the second packet LTYPE matches 0x86dd



- LTYPE matches TS\_LTYPE2 in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG register and TS\_LTYPE2\_EN is set in the CPSW\_PN\_TS\_CTL\_REG register.
- e. The first packet LTYPE matches TS\_VLAN\_LTYPE2 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG register and TS\_RX\_VLAN\_LTYPE2\_EN is set in the CPSW\_PN\_TS\_CTL\_REG register and the second packet LTYPE matches TS\_LTYPE1 in the CPSW\_PN\_TS\_SEQ\_LTYPE\_REG register.
  - f. The first packet LTYPE matches TS\_VLAN\_LTYPE2 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG register and TS\_RX\_VLAN\_LTYPE2\_EN is set in the CPSW\_PN\_TS\_CTL\_REG register and the second packet LTYPE matches TS\_LTYPE2 in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG register and TS\_LTYPE2\_EN is set in the CPSW\_PN\_TS\_CTL\_REG register.
  - g. The first packet LTYPE matches TS\_VLAN\_LTYPE1 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG register and TS\_RX\_VLAN\_LTYPE1\_EN is set in the CPSW\_PN\_TS\_CTL\_REG register and the second packet LTYPE matches TS\_VLAN\_LTYPE2 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG register and TS\_RX\_VLAN\_LTYPE2\_EN is set in the CPSW\_PN\_TS\_CTL\_REG register and the third packet LTYPE matches TS\_LTYPE1 in the CPSW\_PN\_TS\_SEQ\_LTYPE\_REG register.
  - h. The first packet LTYPE matches TS\_VLAN\_LTYPE1 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG register and TS\_RX\_VLAN\_LTYPE1\_EN is set in the CPSW\_PN\_TS\_CTL\_REG register and the second packet LTYPE matches TS\_VLAN\_LTYPE2 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG register and TS\_RX\_VLAN\_LTYPE2\_EN is set in the CPSW\_PN\_TS\_CTL\_REG register and the third packet LTYPE matches TS\_LTYPE2 in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG register and TS\_LTYPE2\_EN is set in the CPSW\_PN\_TS\_CTL\_REG register
3. The PTP message begins in the byte after the LTYPE.
  4. The packet message type is enabled in the TS\_MSG\_TYPE\_EN field in the CPSW\_PN\_TS\_CTL\_REG register.
  5. The packet was received without error (not long/short/mac\_ctl/CRC/code/align).

If all of the criteria described above are met for either Annex D, Annex E, or Annex F, and the packet is determined to be a valid time synchronization packet, then the RX interface will push an Ethernet receive event into the event FIFO.

#### 12.2.1.4.7.13.5.2 Ethernet Port Transmit Event

This section describes Ethernet port transmit events. For every packet transmitted on the Ethernet ports, the port transmit interface will begin parsing the packet to determine if it is a valid Ethernet time synchronization packet. The CPTS transmit interface for the port will use the following criteria to determine if the packet is a valid time synchronization Ethernet transmit event. The CPSW decoder determines if the packet is a valid ethernet receive time synchronization event. To be a valid Ethernet transmit time synchronization event, the conditions listed below must be true for either Annex D, Annex E, or Annex F:

##### **Annex D (IPv4)**

1. Transmit time sync is enabled (TS\_TX\_ANNEX\_D\_EN is set in the CPSW\_PN\_TS\_CTL\_REG register).
2. One of the sequences below is true.
  - a. The first packet LTYPE matches 0x0800
  - b. The first packet LTYPE matches TS\_VLAN\_LTYPE1 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG register and TS\_TX\_VLAN\_LTYPE1\_EN is set in the CPSW\_PN\_TS\_CTL\_REG register and the second packet LTYPE matches 0x0800
  - c. The first packet LTYPE matches TS\_VLAN\_LTYPE2 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG register and TS\_TX\_VLAN\_LTYPE2\_EN is set in the CPSW\_PN\_TS\_CTL\_REG register and the second packet LTYPE matches 0x0800
  - d. The first packet LTYPE matches TS\_VLAN\_LTYPE1 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG register and TS\_TX\_VLAN\_LTYPE1\_EN is set in the CPSW\_PN\_TS\_CTL\_REG register and the second packet LTYPE matches TS\_VLAN\_LTYPE2 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG register and TS\_TX\_VLAN\_LTYPE2\_EN is set in the CPSW\_PN\_TS\_CTL\_REG register and the third packet LTYPE matches 0x0800
3. Byte 14 (the byte after the LTYPE) contains 0x45 (IPv4).

The byte numbering assumes that there are no VLANs. The byte number is intended to show the relative order of the bytes.

4. Byte 20 contains 0bXXX00000 (5 lower bits zero) and Byte 21 contains 0x00 (fragment offset zero)
5. Byte 22 contains 0x01 (HOP Limit = 1) if the TS\_TTL\_NONZERO bit in the switch CPSW\_PN\_TS\_CTL\_LTYPE2\_REG register is cleared to 0h, or byte 22 contains any value if TS\_TTL\_NONZERO is set to 1h. Byte 22 is the TTL/HOP field.
6. Byte 23 contains 0x11 (Next Header UDP Fixed).
7. The TS\_UNI\_EN bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG register is cleared to 0h and Bytes 30 through 33 contain:
  - a. Decimal 224.0.1.129 and the TS\_129 bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG register is set, or
  - b. Decimal 224.0.1.130 and the TS\_130 bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG register is set, or
  - c. Decimal 224.0.1.131 and the TS\_131 bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG register is set, or
  - d. Decimal 224.0.1.132 and the TS\_132 bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG register is set, or
  - e. Decimal 224.0.0.107 and the TS\_107 bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG register is set, or
  - f. The TS\_UNI\_EN bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG register is set and Bytes 30 through 33 contain any values.
8. Bytes 36 and 37 contain:
  - a. Decimal 0x01 and 0x3F respectively and the TS\_319 bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG register is set, or
  - b. Decimal 0x01 and 0x40 respectively and the TS\_320 bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG register is set.
9. The PTP message begins in byte 42.
10. The packet message type is enabled in the TS\_MSG\_TYPE\_EN field in the CPSW\_PN\_TS\_CTL\_REG register.
11. The packet was sent by host port 0.

1. Transmit annex E time sync is enabled (TS\_TX\_ANNEX\_E\_EN bit is set in the switch CPSW\_PN\_TS\_CTL\_REG register).
2. One of the sequences below is true.
  - a. The first packet LTYPE matches 0x86dd.
  - b. The first packet LTYPE matches TS\_VLAN\_LTYPE1 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG register and TS\_TX\_VLAN\_LTYPE1\_EN is set in the CPSW\_PN\_TS\_CTL\_REG register and the second packet LTYPE matches 0x86dd
  - c. The first packet LTYPE matches TS\_VLAN\_LTYPE2 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG register and TS\_TX\_VLAN\_LTYPE2\_EN is set in the CPSW\_PN\_TS\_CTL\_REG register and the second packet LTYPE matches 0x86dd
  - d. The first packet LTYPE matches TS\_VLAN\_LTYPE1 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG register and TS\_TX\_VLAN\_LTYPE1\_EN is set in the CPSW\_PN\_TS\_CTL\_REG register and the second packet LTYPE matches TS\_VLAN\_LTYPE2 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG register and TS\_TX\_VLAN\_LTYPE2\_EN is set in the CPSW\_PN\_TS\_CTL\_REG register and the third packet LTYPE matches 0x86dd
3. Byte 14 (the byte after the LTYPE) contains 0x6X (IPv6).
4. Byte 20 contains 0x11 (UDP Fixed Next Header).
5. Byte 21 contains 0x01 (Hop Limit = 1) if the TS\_TTL\_NONZERO bit in the switch CPSW\_PN\_TS\_CTL\_LTYPE2\_REG register is cleared to 0h, or byte 21 contains any value if TS\_TTL\_NONZERO is set to 1h. Byte 21 is the TTL/HOP field..
6. The TS\_UNI\_EN bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG register is cleared to 0 and Bytes 38 through 53 contain:
  - a. FF0M:0:0:0:0:0:0:0181 and the TS\_129 bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG register is set, or
  - b. FF0M:0:0:0:0:0:0:0182 and the TS\_130 bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG register is set, or
  - c. FF0M:0:0:0:0:0:0:0183 and the TS\_131 bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG register is set, or





3. The packet message type is enabled in the TS\_MSG\_TYPE\_EN field in the CPSW\_PN\_TS\_CTL\_REG register.
4. The packet was sent by host port 0.

If all of the criteria described above are met, and the packet is determined to be a valid time synchronization packet, then the time stamp for the transmit event will not be generated until the start of frame delimiter of the packet is actually transmitted. The start of frame delimiter will be sampled on every rising and falling edge of the CPTS\_RFT\_CLK. Once the packet is transmitted, then the TX interface will push an Ethernet transmit event into the event FIFO.

**Table 12-190. Values of Message Type Field**

Message Type	Value (hex)
Sync	0
Delay_Req	1
Pdelay_Req	2
Pdelay_Resp	3
Reserved	4:7
Follow_Up	8
Delay_Resp	9
Pdelay_Resp_Follow_Up	A
Announce	B
Signaling	C
Management	D
Reserved	E:F

Once a transmitted or received packet is determined to be a valid time sync packet, the Ethernet Transmit Event or Ethernet Receive Event is loaded onto the Event FIFO.

The CPSW\_CPTS\_EVENT\_1\_REG register contains the Message Type and Sequence ID values from the original time sync packet. The CPSW\_CPTS\_EVENT\_0\_REG (and CPSW\_CPTS\_EVENT\_3\_REG) register contains the time stamp value when the packet arrived at the corresponding port.

**Table 12-191. Values of Message Type Field**

Message Type	Value (hex)
Sync	0
Delay_Req	1
Pdelay_Req	2
Pdelay_Resp	3
Reserved	4:7
Follow_Up	8
Delay_Resp	9
Pdelay_Resp_Follow_Up	A
Announce	B
Signaling	C
Management	D
Reserved	E:F

Once a transmitted or received packet is determined to be a valid time sync packet, the Ethernet Transmit Event or Ethernet Receive Event is loaded onto the Event FIFO.



The CPSW\_CPTS\_EVENT\_1\_REG register contains the Message Type and Sequence ID values from the original time sync packet. The CPSW\_CPTS\_EVENT\_0\_REG (and CPSW\_CPTS\_EVENT\_3\_REG) register contains the time stamp value when the packet arrived at the corresponding port.

#### 12.2.1.4.7.13.5.3

**Table 12-192. Values of Message Type Field**

Message Type	Value (hex)
Sync	0
Delay_Req	1
Pdelay_Req	2
Pdelay_Resp	3
Reserved	4:7
Follow_Up	8
Delay_Resp	9
Pdelay_Resp_Follow_Up	A
Announce	B
Signaling	C
Management	D
Reserved	E:F

Once a transmitted or received packet is determined to be a valid time sync packet, the Ethernet Transmit Event or Ethernet Receive Event is loaded onto the Event FIFO.

The CPSW\_CPTS\_EVENT\_1\_REG register contains the Message Type and Sequence ID values from the original time sync packet. The CPSW\_CPTS\_EVENT\_0\_REG (and CPSW\_CPTS\_EVENT\_3\_REG) register contains the time stamp value when the packet arrived at the corresponding port.

#### 12.2.1.4.7.14 Timestamp Compare Event

##### Note

Timestamp compare events are generated for non-toggle mode only.

The CPTS can generate an event for a time stamp comparison in 32-bit or 64-bit mode.

##### 12.2.1.4.7.14.1 32-Bit Mode

The CPTS\_COMP output is also asserted when the event is generated. The event is generated when the 32-bit time stamp value (CPSW\_CPTS\_EVENT\_0\_REG) compares with the CPSW\_CPTS\_TS\_COMP\_VAL\_REG register and the CPSW\_CPTS\_TS\_COMP\_LEN\_REG value is non-zero. The CPSW\_CPTS\_TS\_COMP\_LEN\_REG value should be written by software after the CPSW\_CPTS\_TS\_COMP\_VAL\_REG register is written and should be zero when the comparison value is written.

##### 12.2.1.4.7.14.2 64-Bit Mode

The CPTS\_COMP output is also asserted when the event is generated. The event is generated when the 64-bit time stamp value (CPSW\_CPTS\_EVENT\_0\_REG and CPSW\_CPTS\_EVENT\_3\_REG) compares with the CPSW\_CPTS\_TS\_COMP\_VAL\_REG and CPSW\_CPTS\_TS\_COMP\_HIGH\_VAL\_REG registers and the CPSW\_CPTS\_TS\_COMP\_LEN\_REG value is non-zero. The CPSW\_CPTS\_TS\_COMP\_LEN\_REG value should be written by software after the CPSW\_CPTS\_TS\_COMP\_VAL\_REG register is written and should be zero when the comparison value is written.

#### 12.2.1.4.7.15 Host Transmit Event

The host can send a packet to be transmitted on an Ethernet port that will generate a time synchronization event. The host sets the TSTAMP\_EN bit and sends the DOMAIN, MESSAGE\_TYPE, and SEQUENCE\_ID in the additional control information that resides in the protocol specific section of the descriptor that is transmitted to the CPSW\_3G. An event is then generated and placed on the event FIFO once the packet is transmitted. Host events allow the user to timestamp exactly when a software generated packet exits the device.

#### 12.2.1.4.7.16 CPTS Interrupt Handling

When an event is push onto the Event FIFO, an interrupt can be generated to indicate to software that a time sync event occurred. The following steps should be taken to process time sync events using interrupts:

1. Enable the TS\_PEND interrupt by setting the TS\_PEND\_EN bit of the CPSW\_CPTS\_INT\_ENABLE\_REG register.
2. Upon interrupt, read the CPSW\_CPTS\_EVENT\_0\_REG through CPSW\_CPTS\_EVENT\_3\_REG registers values.
3. Set the CPSW\_CPTS\_EVENT\_POP\_REG[0] EVENT\_POP bit to 1h to pop the previously read value off of the event FIFO.
4. Process the interrupt as required by the application software.

Software has the option of processing more than a single event from the event FIFO in the interrupt service routine in the following way:

1. Enable the TS\_PEND interrupt by setting the TS\_PEND\_EN bit of the CPSW\_CPTS\_INT\_ENABLE\_REG
2. Upon interrupt, enter the CPTS service routine.
3. Read the CPSW\_CPTS\_EVENT\_0\_REG through CPSW\_CPTS\_EVENT\_3\_REG registers values.
4. Set the CPSW\_CPTS\_EVENT\_POP\_REG[0] EVENT\_POP bit to 1h to pop the previously read value off of the event FIFO.
5. Wait for an amount of time greater than four CPTS\_RFT\_CLK periods plus four CPPI\_ICLK periods.
6. Read the TS\_PEND\_RAW bit in the CPSW\_CPTS\_INTSTAT\_RAW\_REG register to determine if another valid event is in the event FIFO. If bit TS\_PEND\_RAW is asserted, go to step 3. If bit TS\_PEND\_RAW is not asserted proceed with step 7.
7. Process the interrupt(s) as required by the application software.

Software also has the option of disabling the interrupt and polling the TS\_PEND\_RAW bit of the CPSW\_CPTS\_INTSTAT\_RAW\_REG register to determine if a valid event is on the event FIFO.

#### 12.2.1.4.8 CPPI Streaming Packet Interface

The receive streaming interface on port 0 of the CPSW is responsible for receiving packet for Ethernet egress data from the packet streaming switch in the NAVSS. The CPPI receive port is equivalent to an Ethernet port with the difference being that the data is provided to the CPSW in the 128-bit streaming interface data format instead RGMII data format.

In addition to the packet data, the receive streaming interface also can provide additional control information that resides in the information words of the descriptor that was transmitted to the CPSW.

The tables below show the information that may be passed along with which descriptor information word to put it in.

##### 12.2.1.4.8.1 Port 0 CPPI Transmit Packet Streaming Interface (CPSW\_2G Egress)

The CPSW2G has a single transmit packet streaming interface. All Ethernet packet data destined for the host (Port 0) is transferred on the transmit packet streaming interface. The transmit packet streaming interface is equivalent to an Ethernet MAC output with the difference being that 128-bit streaming interface data is output. Egress packet data is packed on the 128-bit data bus with all words having 16-bytes except possibly the last packet data word which is the word previous to the EOP word. INFO Word 0–3 is transferred on SOP. The EOP word contains the packet status 0–3 (data type 24) which includes the timestamp and checksum data. Packets are not dropped on the transmit streaming interface due to pushback, but packets may be dropped in the associated priority FIFO.

INFO Word 0–3 and Status Data Word 0–3 (on EOP) are the only non-payload data word types that are transferred. Long packets are truncated at the CPSW\_PN\_RX\_MAXLEN\_REG[13-0] RX\_MAXLEN byte value of the ingress port (only the CPSW\_PN\_RX\_MAXLEN\_REG number of bytes are kept if long packets are transferred due to CPSW\_PN\_MAC\_CONTROL\_REG register copy error frames set - RX\_CEF\_EN). MAC control frames are only transferred if the receiving Ethernet port has the CPSW\_PN\_MAC\_CONTROL\_REG[24] RX\_CMF\_EN bit set.

The error encoding on the TXST\_PKT\_ERR[3:0] output is shown in [Table 12-193](#).

**Table 12-193. Error Encoding on the TXST\_PKT\_ERR[3:0] Output**

TXST_PKT_ERR[3:0]	Description
0000	No Error
0001	CRC Error
0010	Code or alignment error
0011	Short (no code/align/crc error)
0100	FragCRC (short with crc error)
0101	Frag Code/Align (short with code/align error)
0110	Long
0111	Jabber CRC (long with crc error)
1000	Jabber Code/Align (long with code/align error)
1001	Mac ontl packet (CPSW_PN_MAC_CONTROL_REG[24] RX_CMF_EN set on ingress Ethernet port)
1010	Mac control CRC
1011	Mac control Code/Align
1100	Mac control short/frag (short MAC control frame with CRC/Code/Align)
1101	Mac control long/jabber (long MAC control frame with CRC/Code/Align)
1110	Reserved
1111	Reserved

The CPPI Port 0 transmit packet streaming interface has a single output thread. If a packet transmission has begun then the entire packet will be transmitted before the next packet is sent (packet data from multiple packets are not interleaved on the transmit streaming interface). The default egress flow is the port number minus 1, concatenated with the 3-bit Port 0 transmit FIFO hardware switch priority. For example, a packet that was received on port 4 with a Port 0 transmit FIFO hardware switch priority of 5 would be sent on flow 29 (decimal) or flow 0b0011101 (binary). Priority remapping on ingress and Port 0 transmit egress effects the output flow.

INFO Words () are a contiguous block of four 32-bit data words aligned on a 32-bit word boundary.

**Figure 12-147. TX INFO Word 0 Format**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PKT_TYPE					RESERVED			PASS_CRC	CRC_T YPE	RESERVED					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								FLOW_ID							

Bit	Field	Description
31-27	PKT_TYPE	Always set to 0b00111. Host PD (Packet Descriptor) Word 2. Packet Type: bits[31-27].
26-24	RESERVED	Reserved.

Bit	Field	Description
23	PASS_CRC	This bit is cleared to zero (no CRC passed) when the P0_TX_CRC_REMOVE bit in the CPSW_CONTROL_REG register is set (and the egress packet has no errors). When the remove bit is cleared to zero then this bit is cleared and no CRC is passed with the output packet. The packet length includes the CRC if it is present.
22	CRC_TYPE	The packet CRC type. The type of CRC passed is determined by CRC_TYPE field in the CPSW_PN_MAC_CONTROL_REG register (not by the type of CRC the packet had on Ethernet port ingress). Host PD Word 1. Protocol Specific Flags: bits[27-24]. 0h: Ethernet CRC 1h: Castagnoli CRC
21-8	RESERVED	Reserved.
7-0	FLOW_ID	This is the packet output transmit streaming interface flow. The default flow ID can be overridden by ALE classification (Thread mapping). The switch default flow is the 3-bit "From Port" value concatenated with the 3-bit "Switch Priority" {From_Port[2:0], Switch_Priority[2:0]} as shown below: Host PD (Packet Descriptor) Word 1. Flow ID: bits[13-0]. 0h: The packet was received on Ethernet port 1 Switch Priority – The actual hardware switch priority that the packet was stored in on the CPPI transmit FIFO.

**Figure 12-148. TX INFO Word 1 Format**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x4 (fixed_ps_size)												0			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	PKT_LENGTH														

Bit	Field	Description
31-20	FIXED_PS_SIZE	Fixed packet size: 0x4
19-14	RESERVED	Reserved.
13-0 (Host PD (Packet Descriptor) Word 1. Packet Length: bits[ 21-0])	PKT_LENGTH	Specifies the number of bytes in the entire packet. Offset bytes are not included. Valid only on SOP. The packet length must be greater than zero. The packet data will be truncated to the packet length if the packet length is shorter than the sum of the packet buffer descriptor buffer lengths. A host error occurs if the packet length is greater than the sum of the packet buffer descriptor buffer lengths.

**Figure 12-149. TX INFO Word 2 Format**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFFFF															

**Figure 12-150. TX INFO Word 3 Format**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								SRC_ID							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															

Bit	Field	Description
31-24	RESERVED	Reserved.
23-16	SRC_ID	The packet SRC_ID value comes from the PORT1 field in the CPSW_P0_SRC_ID_A_REG register. (src_tag) PD (Packet Descriptor) Word 3. Source Tag Low bits[23-16] if RFLOW[a]_RFC.rx_src_tag_lo_sel = 0x4 or (src_tag) PD (Packet Descriptor) Word 3. Source Tag High bits[31-24] if RFLOW[a]_RFC.rx_src_tag_hi_sel = 0x4
15-0	RESERVED	Reserved.

### Note

TX Status Data Word [0..3] are mapped to Host Packet Descriptor Protocol Specific Words if RFLOW[a]\_RFA.rx\_psinfo\_present = 1 and RFLOW[a]\_RFA.rx\_ps\_location = 0

**Figure 12-151. TX Status Data Word 0 Format**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMESTAMP[31:0]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMESTAMP[31:0]															

Bit	Field	Description
31-0	TIMESTAMP[31:0]	Contains the lower 32-bits of the time stamp value.

**Figure 12-152. TX Status Data Word 1 Format**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMESTAMP[63:32]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMESTAMP[63:32]															

Bit	Field	Description
31-0	TIMESTAMP[63:32]	Contains the upper 32-bits of the time stamp value.

**Figure 12-153. TX Status Data Word 2 Format**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED											IPV4_VALID	IPV6_VALID	TCP_UDP_N	FRAGMENT	CHECKSUM_ERROR
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHECKSUM_ADD															

Bit	Field	Description
31-21	RESERVED	Reserved.
20	IPV4_VALID	An IPV4 TCP or UDP Packet was detected.
19	IPV6_VALID	An IPV6 TCP or UDP Packet was detected.
18	TCP_UDP_N	Valid only when either the IPV4_VALID or IPV6_VALID bits are set. 0h: Indicates UDP packet was detected. 1h: Indicates TCP packet was detected.

Bit	Field	Description
17	FRAGMENT	Indicates that an IP fragment was detected. Valid only when either the IPV4_VALID or IPV6_VALID bits are set.
16	CHECKSUM_ERROR	Valid only when either the IPV4_VALID or IPV6_VALID bits are set.
15-0	CHECKSUM_ADD	This is the value that was summed during the checksum computation. This value is FFFFh for IPV4/6 UDP/TCP packets with no checksum error.

**Figure 12-154. TX Status Data Word 3 Format**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															

Bit	Field	Description
31-0	RESERVED	Reserved.

#### 12.2.1.4.8.2 Port 0 CPPI Receive Packet Streaming Interface (CPSW\_2G Ingress)

Info Word 0/1/2/3 (INFO1 and INFO3 are ignored) are transferred on SOP. If a timestamp word (Extended Packet Info Word 0/1/2/3) is to be transferred it must be after SOP and before any packet data is transferred. Any following non-data type words will be dropped. The EOP word must be payload data.

Input receive packets cannot be aborted by the host. The INFO Word bit descriptions and Extended Packet INFO Word bit descriptions are shown below. The PASS\_CRC bit indicates that the CRC is passed with the packet data. Packets that have a passed CRC that is an error CRC will be output on the Ethernet port with at least one CRC byte inverted to indicate the error if P0\_RX\_PASS\_CRC\_ERR bit is set, otherwise they are dropped. The packet is a directed packet when any of the TO\_PORT bits are nonzero. A packet may be directed only to a single port. The packet will be sent to the port number indicated. For directed packets the lookup process is skipped to determine the destination. However, in vlan aware mode (when VLAN\_AWARE bit in the CPSW\_CONTROL\_REG register is set to 1h) the lookup is performed to determine untagged egress. Packets longer than the value in CPSW\_P0\_RX\_MAXLEN\_REG[13-0] RX\_MAXLEN bit field are dropped. Packets shorter than 60-Bytes are padded to 64-Bytes (after adding pad and CRC) if P0\_RX\_PAD bit in the CPSW\_CONTROL\_REG register is set and if PASS\_CRC is clear, otherwise they are dropped. This means that packets shorter than 64-Bytes are dropped if the PASS\_CRC info bit is set regardless of P0\_RX\_PAD bit (packets are padded only if they are short and do not have CRC).

A RX INFO word () is a contiguous block of four 32-bit data words aligned on a 32-bit word boundary.

#### Note

RX Control Data Words [0..2] are mapped to Host Packet Descriptor Protocol Specific Words if (TCHAN[a]\_TCFG.tx\_filt\_pswords = 0) and (Host PD Word 1.Protocol Specific Region Location.bit[28] = 0h) and (Host PD Word 1.Protocol Specific Valid Word Count.bits[22-27] = 4h)

**Figure 12-155. RX INFO Word 0 Format**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED								PASS_CRC	CRC_T YPE	RESERVED					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															

Bit	Field	Description
31-24	RESERVED	Reserved.

Bit	Field	Description
23	PASS_CRC	The PASS_CRC bit indicates that the CRC is passed with the packet data. 0h: CRC is not passed with packet (CRC_TYPE is don't care) 1h: CRC of type CRC_TYPE is passed with the packet.
22 (Host PD (Packet Descriptor) Word 1. Protocol Specific Flags: bits[27-24])	CRC_TYPE	CRC Type 0h: Ethernet CRC 1h: Castagnoli CRC
21-0	RESERVED	Reserved.

**Figure 12-156. RX INFO Word 2 Format**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED											TO_PORT				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															

Bit	Field	Description
31-21	RESERVED	Reserved.
20-16 (Host PD (Packet Descriptor) Word 3. Dest Tag Low bits[8-0])	TO_PORT	Port number to send the directed packet to. This field is set by the host. This field is valid on SOP. Directed packets go to the directed port, but an ALE lookup is performed to determine untagged egress in VLAN_AWARE mode. 0h: Not directed 1h: Send the packet to port 1.
15-0	RESERVED	Reserved.

**Figure 12-157. RX Control Data Word 1 Format**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMES TAMP_ EN	RESERVED			DOMAIN								MSG_TYPE			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEQUENCE_ID															

Bit	Field	Description
31	TIMESTAMP_EN	When set, this bit indicates that the packet will generate a timesync event on Ethernet egress (if the CPTS is configured properly) with the associated DOMAIN, MSG_TYPE, and SEQUENCE_ID.
30-28	RESERVED	Reserved.
27-20	DOMAIN	Timesync domain.
19-16	MSG_TYPE	Timesync message type.
15-0	SEQUENCE_ID	Timesync sequence ID.

**Figure 12-158. RX Control Data Word 2 Format**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----



**Figure 12-158. RX Control Data Word 2 Format (continued)**

CHECKSUM_RESULT								CHECKSUM_START_BYTE							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHECKSUM_INV	RESERVED	CHECKSUM_BYTECOUNT													

Bit	Field	Description
31-24	CHECKSUM_RESULT	This is the packet byte number where the checksum result will be placed in the egress packet. The first packet byte which is the first byte of the destination address is Byte 1 (not byte zero).
23-16	CHECKSUM_START_BYTE	This is the packet byte number to start the checksum calculation on. The first packet byte is Byte 1.
15	CHECKSUM_INV	When set, a zero checksum value will be inverted and sent as FFFFh.
14	RESERVED	Reserved.
13-0	CHECKSUM_BYTECOUNT	This is the number of bytes to calculate the checksum on. The outgoing Ethernet packet will have a checksum inserted when this value is non-zero.

Other INFO words are not taken into account.

#### 12.2.1.4.8.3 CPPI Checksum Offload

The CPPI host port can be enabled to perform checksum offload on host port packet ingress and egress. UDP (User Datagram Protocol) and TCP (Transmission Control Protocol) over IPV4 and IPV6 are supported. For the purposes of checksum description, the first packet byte (the first byte of the destination address) is byte 1 (not byte 0). That is, a 64 byte packet goes from byte 1 to byte 64. For all packet types, the S\_CN\_SWITCH bit in the CPSW\_CONTROL\_REG register must be set for the Outer VLAN L type to be supported.

##### 12.2.1.4.8.3.1 CPPI Transmit Checksum Offload

IPV4 and IPV6 UDP and TCP packets that are received on any Ethernet port and destined for port 0 egress are checked for correct checksum as described below. The byte counts below are shown for packets with no VLAN's. The byte counts vary with one or two packet VLAN's. Packets received on an Ethernet port with errors are not checked for a correct checksum if they are passed to the host.

##### 12.2.1.4.8.3.1.1 IPV4 UDP

- Byte 15 Upper Nibble = 4 for IPV4
- Byte 15 Lower Nibble = IHL - Nibble with number of 32-bit words in IPV4 header (5 to 15 supported).
- Bytes 20-21 = fragment[15-0] – Bit 13 is the MF bit and bits [12-0] are the Fragment offset. A packet is a fragment if the MF bit is set or if the fragment offset is non-zero. The first packet fragment has MF=1 with a zero offset. Middle fragments have MF=1 with a nonzero offset. The last packet fragment has MF=0 with a nonzero offset. Non-fragmented packets have MF=0 and a zero offset. A count is output for packet fragments but no errors are reported. First fragments have the UDP header included in the count. Middle and last fragments have only data included in the count (there is no UDP header).
- Byte 24 = 0x11 for UDP protocol.
- Received packet UDP checksum of zero means that there is no IPV4 checksum sent with the packet so no error will be issued.
- Received packet UDP checksum of 0xFFFF means that the checksum was calculated to be 0xFFFF or 0x0000 but was sent in the transmitted packet as 0xFFFF by the sending originating entity.

##### 12.2.1.4.8.3.1.2 IPV4 TCP

- Byte 15 Upper Nibble = 4 for IPV4
- Byte 15 Lower Nibble = IHL - Nibble with number of 32-bit words in IPV4 header (5 to 15 supported).
- Bytes 20-21 = fragment[15-0] – Bit 13 is the MF bit and bits [12-0] are the Fragment offset. A packet is a fragment if the MF bit is set or if the fragment offset is non-zero. The first packet fragment has MF=1 with a



zero offset. Middle fragments have MF=1 with a nonzero offset. The last packet fragment has MF=0 with a nonzero offset. Non-fragmented packets have MF=0 and a zero offset. A count is output for packet fragments but no errors are reported. First fragments have the UDP header included in the count. Middle and last fragments have only data included in the count (there is no TCP header).

- Byte 24 = 0x06 for TCP protocol.

#### **12.2.1.4.8.3.1.3 IPV6 UDP**

- Byte 15 upper nibble = 6 for IPV6.
- Byte 21 = 0x11 for UDP protocol as next header.
- Fragment extension headers are supported. First fragments have a fragment extension header (byte 21 = 0x2C) followed by a UDP header (byte 55 = 0x11). Middle and last fragments have a fragment extension header followed by data only (no UDP header). The first packet fragment has MF=1 with a zero offset. Middle fragments have MF=1 with a nonzero offset. The last packet fragment has MF=0 with a nonzero offset. Non-fragmented packets do not have a fragment extension header. A count is output for packet fragments but no errors are reported.
- Received packet UDP checksum of zero means that there is no IPV6 checksum sent with the packet so no error will be issued.
- Received packet UDP checksum of 0xFFFF means that the checksum was calculated to be 0xFFFF or 0x0000 but was sent in the transmitted packet as 0xFFFF by the sending originating entity.

#### **12.2.1.4.8.3.1.4 IPV6 TCP**

- Byte 15 upper nibble = 6 for IPV6.
- Byte 21 = 0x06 for TCP protocol as next header.
- Fragment extension headers are supported. First fragments have a fragment extension header (byte 21 = 0x2C) followed by a UDP header (byte 55 = 0x06). Middle and last fragments have a fragment extension header followed by data only (no TCP header). The first packet fragment has MF=1 with a zero offset. Middle fragments have MF=1 with a nonzero offset. The last packet fragment has MF=0 with a nonzero offset. Non-fragmented packets do not have a fragment extension header. A count is output for packet fragments but no errors are reported.

#### **12.2.1.4.8.4 CPPI Receive Checksum Offload**

Packets sent from host port 0 (switch ingress) to any Ethernet port can have a checksum calculated and inserted into the Ethernet egress packet. The RX\_CHECKSUM\_EN bit in the CPSW\_P0\_CONTROL\_REG register must be set for receive checksum operation to be enabled. When bit RX\_CHECKSUM\_EN is enabled, Control Data Word 2 input on CPPI receive PSI interface determines how the checksum is calculated. The CHECKSUM\_RESULT field in Control Data Word 2 determines where the checksum is inserted. The checksum result location is adjusted by the egress port if a VLAN is to be inserted or removed on Ethernet port egress.

#### **12.2.1.4.8.5 Egress Packet Operations**

Each CPSW egress port (Ethernet and Host) is capable of performing egress packet processing operations (CPSW\_ALE\_EGRESSOP). IntraVLAN processing either adds, removes, or replaces VLAN information or does nothing. InterVLAN routing allows hardware routing between a limited number of VLANs - thereby allowing high-bandwidth or other routing operations to be offloaded from software to the CPSW (hardware). IntraVLAN processing and InterVLAN routing operations are mutually exclusive. In addition, the packet source and destination addresses can be swapped on egress to facilitate OAM or generic testing operations.

#### 12.2.1.4.9 MII Management Interface (MDIO)

The MII Management interface module implements the 802.3 serial management interface to interrogate and control external Ethernet PHY using a two-wire bus.

##### 12.2.1.4.9.1 MDIO Frame Formats

[Table 12-194](#) shows the address, [Table 12-195](#) shows the read format and [Table 12-196](#) shows the write format of the supported Clause 45 MII Management interface frames. Post-increment accesses are not supported.

**Table 12-194. MDIO Clause 45 Address Frame Format**

Pre-amble	Start Delimiter	Operation Code	PHY Address	MMD Number	Turnaround	Data
FFFF FFFFh	00	00	AAAAA	RRRRR	10	AAAA.AAAA.AAAA.AAAA

**Table 12-195. MDIO Clause 45 Read Frame Format**

Pre-amble	Start Delimiter	Operation Code	PHY Address	MMD Number	Turnaround	Data
FFFF FFFFh	00	11	AAAAA	RRRRR	Z0	DDDD.DDDD.DDDD.DDDD

**Table 12-196. MDIO Clause 45 Write Frame Format**

Pre-amble	Start Delimiter	Operation Code	PHY Address	MMD Number	Turnaround	Data
FFFF FFFFh	00	01	AAAAA	RRRRR	10	DDDD.DDDD.DDDD.DDDD

The default or idle state of the two wire serial interface is a logic one. All tri-state drivers should be disabled and the PHY's pull-up resistor will pull the MDIO line to a logic 1. Prior to initiating any other transaction, the station management entity shall send a preamble sequence of 32 contiguous logic 1 bits on the MDIO line with 32 corresponding cycles on MDCLK to provide the PHY with a pattern that it can use to establish synchronization. A PHY shall observe a sequence of 32 contiguous logic one bits on MDIO with 32 corresponding MDCLK cycles before it responds to any other transaction. The MDIO CPSW\_MDIO\_USER\_ADDR0\_REG register must be written before a read or write operation is performed to set the address used in the operation. Each read or write operation has a preceeding address frame.

#### Preamble

The start of a frame is indicated by a preamble, which consists of a sequence of 32 contiguous bits all of which are a 1. This sequence provides the PHY a pattern to use to establish synchronization. The preamble is required in clause 45 operation.

#### Start Delimiter

The preamble is followed by the start delimiter which is indicated by a 00 pattern.

#### Operation Code

The operation code for an address transaction is 00. The operation code for a read is 11, while the operation code for a write is a 01.

#### PHY Address

The PHY address is 5 bits allowing 32 unique values. The first bit transmitted is the MSB of the PHY address.

#### MMD Number

The MMD number is the 5 bits allowing 32 unique values. The first bit transmitted is the MSB.

#### Turnaround

An idle bit time during which no device actively drives the MDIO signal shall be inserted between the register address field and the data field of a read frame in order to avoid contention. During a read frame, the PHY shall

drive a zero bit onto MDIO for the first bit time following the idle bit and preceding the Data field. During a write frame, this field shall consist of a one bit followed by a zero bit.

### Address

The address field is 16 bits on address operations. The first bit transmitted is the MSB of the address word. Each read/write operation initiated has an automatic address operation initiated first that uses the MDIO CPSW\_MDIO\_USER\_ADDR0\_REG/ CPSW\_MDIO\_USER\_ADDR1\_REG register values as the 16-bit address.

### Data

The Data field is 16 bits on read and write operations. The first bit transmitted and received is the MSB of the data word.

#### 12.2.1.4.9.2 MDIO Functional Description

The MII Management I/F will remain idle until enabled by setting the ENABLE bit in the CPSW\_MDIO\_CONTROL\_REG register. The MII Management I/F will then continuously poll the link status from within the Generic Status Register of all possible 32 PHY addresses in turn recording the results in the MDIO CPSW\_MDIO\_LINK\_REG register. Individual PHY's can be enabled or disabled for polling the associated bit in the CPSW\_MDIO\_POLL\_EN\_REG register. The CPSW\_MDIO\_LINK\_REG and CPSW\_MDIO\_ALIVE\_REG register bit values are updated on the poll of each PHY. The LINKSEL bit in the CPSW\_MDIO\_USER\_PHY\_SEL\_REG\_k register determines the status input that is used. A change in the link status of the two PHYs being monitored will set the appropriate bit in the MDIO CPSW\_MDIO\_LINK\_INT\_RAW\_REG register and the MDIO CPSW\_MDIO\_LINK\_INT\_MASKED\_REG register, if enabled by the LINKINT\_ENABLE bit in the CPSW\_MDIO\_USER\_PHY\_SEL\_REG\_k register.

The MDIO CPSW\_MDIO\_ALIVE\_REG register is updated by the MII Management I/F module if the PHY acknowledged the read of the generic status register. In addition, any PHY register read transactions initiated by the host also cause the MDIO CPSW\_MDIO\_ALIVE\_REG register to be updated.

At any time, the host can define a transaction for the MII Management interface module to undertake using the DATA, PHYADR, REGADR, and WRITE fields in a CPSW\_MDIO\_USER\_ACCESS\_REG\_k register. When the host sets the GO bit in this register, the MII Management interface module will begin the transaction without any further intervention from the host. Upon completion, the MII Management interface will clear the GO bit and set the USERINTRAW field in the CPSW\_MDIO\_USER\_INT\_RAW\_REG register corresponding to the CPSW\_MDIO\_USER\_ACCESS\_REG\_k register being used. The corresponding bit in the CPSW\_MDIO\_USER\_INT\_MASKED\_REG register may also be set depending on the mask setting in the MDIO CPSW\_MDIO\_USER\_INT\_MASK\_SET\_REG and CPSW\_MDIO\_USER\_INT\_MASK\_CLEAR\_REG registers. A round-robin arbitration scheme is used to schedule transactions that may be queued by the host in different CPSW\_MDIO\_USER\_ACCESS\_REG\_k registers. The host should check the status of the GO bit in the MDIO CPSW\_MDIO\_USER\_ACCESS\_REG\_k register before initiating a new transaction to ensure that the previous transaction has completed. The host can use the ACK bit in the MDIO CPSW\_MDIO\_USER\_ACCESS\_REG\_k register to determine the status of a read transaction.

It is necessary for software to use the MII Management interface module to setup the auto-negotiation parameters of each PHY attached to a MAC port, retrieve the negotiation results, and setup the CPSW\_PN\_MAC\_CONTROL\_REG register in the corresponding MAC.

#### 12.2.1.5 MCU\_CPSW0 Programming Guide

##### 12.2.1.5.1 Initialization and Configuration of CPSW Subsystem

To configure the CPSW Ethernet Subsystem for operation, the host must perform the following:

1. Select the Interface (RMII, or RGMII ) Mode. See the CTRLMMR\_MCU\_ENET\_CTRL[1-0] MODE\_SEL register.
2. Configure pads (pin muxing), as per the interface selected. Refer to *Pad Configuration Registers* and the device-specific Datasheet.
3. Enable the CPSW Ethernet Subsystem clocks. See *CPSW Integration*
4. Ensure that at least 2000 CPPI\_ICLK periods are run after reset is de-asserted.

5. Configure the CPSW\_CONTROL\_REG register
6. Configure the Ethernet Port Source Address registers (CPSW\_PN\_SA\_L\_REG and CPSW\_PN\_SA\_H\_REG)
7. Configure the CPSW statistic port enable register CPSW\_STAT\_PORT\_EN\_REG
8. Configure the ALE (*Address Lookup Engine*)
9. Configure the MDIO (*Initializing the MDIO Module*)
10. Configure Ethernet port, as per the desired mode of operations

#### 12.2.1.5.2 CPSW Reset

To reset the Ethernet port, the host must perform the following:

1. Set CMD\_IDLE bit to 1h in the Ethernet port control register: CPSW\_PN\_MAC\_CONTROL\_REG.
2. Wait for IDLE bit to be set to 1h, which is indicated in the Ethernet port status register: CPSW\_PN\_MAC\_STATUS\_REG.
3. Set SOFT\_RESET bit to 1h in the Ethernet port software reset register: CPSW\_PN\_MAC\_SOFT\_RESET\_REG.
4. Wait for SOFT\_RESET bit in the CPSW\_PN\_MAC\_SOFT\_RESET\_REG registers to be cleared to confirm reset completion.
5. Configure the Ethernet ports.
6. Re-configure registers reset to default value by CPSW\_PN\_MAC\_SOFT\_RESET\_REG.

#### 12.2.1.5.3 MDIO Software Interface

##### 12.2.1.5.3.1 Initializing the MDIO Module

The following steps are performed by the application software or device driver to initialize the MDIO device:

1. Configure the PREAMBLE and CLKDIV bits in the MDIO Control register (CPSW\_MDIO\_CONTROL\_REG).
2. Enable the MDIO module by setting the ENABLE bit in CPSW\_MDIO\_CONTROL\_REG.
3. The MDIO PHY alive status register (MDIO CPSW\_MDIO\_ALIVE\_REG) can be read in polling fashion until a PHY connected to the system responded, and the MDIO PHY link status register (MDIO CPSW\_MDIO\_LINK\_REG) can determine whether this PHY already has a link.
4. Set the appropriate PHY addresses in the MDIO user PHY select register (CPSW\_MDIO\_USER\_PHY\_SEL\_REG\_k, where k = 0 or 1), and set the LINKINT\_ENABLE bit to enable a link change event interrupt if desirable.
5. Set the appropriate LINKSEL bit in the CPSW\_MDIO\_USER\_PHY\_SEL\_REG\_k register (where k = 0 or 1).
6. Set the appropriate USERINTMASKSET bit field in the CPSW\_MDIO\_USER\_INT\_MASK\_SET\_REG register.
7. If an interrupt on general MDIO register access is desired, set the corresponding bit in the MDIO user command complete interrupt mask set register (MDIO CPSW\_MDIO\_USER\_INT\_MASK\_SET\_REG) to use the MDIO user access register (MDIO CPSW\_MDIO\_USER\_ACCESS\_REG\_k, where k = 0 or 1).

##### 12.2.1.5.3.2 Writing Data To a PHY Register

The MDIO module includes a user access register (MDIO CPSW\_MDIO\_USER\_ACCESS\_REG\_k, where k = 0 or 1) to directly access a specified PHY device. To write a PHY register, perform the following:

1. Check to ensure that the GO bit in the MDIO user access register (MDIO CPSW\_MDIO\_USER\_ACCESS\_REG\_k) is cleared.
2. Write to the GO, WRITE, REGADR, PHYADR, and DATA bits in MDIO CPSW\_MDIO\_USER\_ACCESS\_REG\_k corresponding to the PHY and PHY register SW wants to write.
3. The write operation to the PHY is scheduled and completed by the MDIO module. Completion of the write operation can be determined by polling the GO bit in MDIO CPSW\_MDIO\_USER\_ACCESS\_REG\_k for a 0.
4. Completion of the operation sets the corresponding USERINTRAW bit (0 or 1) in the MDIO user command complete interrupt register (CPSW\_MDIO\_USER\_INT\_RAW\_REG) corresponding to MDIO CPSW\_MDIO\_USER\_ACCESS\_REG\_k used. If interrupts have been enabled on this bit using the MDIO user command complete interrupt mask set register (CPSW\_MDIO\_USER\_INT\_MASK\_SET\_REG), then the bit is also set in the MDIO user command complete interrupt register (CPSW\_MDIO\_USER\_INT\_MASKED\_REG) and an interrupt is triggered on the host processor.

#### 12.2.1.5.3.3 Reading Data From a PHY Register

The MDIO module includes a user access register (MDIO CPSW\_MDIO\_USER\_ACCESS\_REG\_k, where k = 0 or 1) to directly access a specified PHY device. To read a PHY register, perform the following:

1. Check to ensure that the GO bit in the MDIO user access register (CPSW\_MDIO\_USER\_ACCESS\_REG\_k, where k = 0 or 1) is cleared.
2. Write to the GO, REGADR, and PHYADR bits in the CPSW\_MDIO\_USER\_ACCESS\_REG\_k register corresponding to the PHY and PHY register SW wants to read.
3. The read data value is available in the DATA bit field in MDIO CPSW\_MDIO\_USER\_ACCESS\_REG\_k register after the module completes the read operation on the serial bus. Completion of the read operation can be determined by polling the GO and ACK bits in CPSW\_MDIO\_USER\_ACCESS\_REG\_k register. After the GO bit has cleared, the ACK bit is set on a successful read.
4. Completion of the operation sets the corresponding USERINTRAW bit (0 or 1) in the MDIO user command complete interrupt register (CPSW\_MDIO\_USER\_INT\_RAW\_REG) corresponding to MDIO CPSW\_MDIO\_USER\_ACCESS\_REG\_k used. If interrupts have been enabled on this bit using the MDIO user command complete interrupt mask set register (CPSW\_MDIO\_USER\_INT\_MASK\_SET\_REG), then the bit is also set in the MDIO user command complete interrupt register (CPSW\_MDIO\_USER\_INT\_MASKED\_REG) and an interrupt is triggered on the host processor.

## 12.2.2 Gigabit Ethernet Switch (CPSW0)

This chapter describes the Gigabit Ethernet Switch (CPSW0) subsystem in the device.

### 12.2.2.1 CPSW0 Overview

The 9-port Gigabit Ethernet Switch (CPSW0) subsystem provides Ethernet packet communication for the device and can be configured as an Ethernet switch.

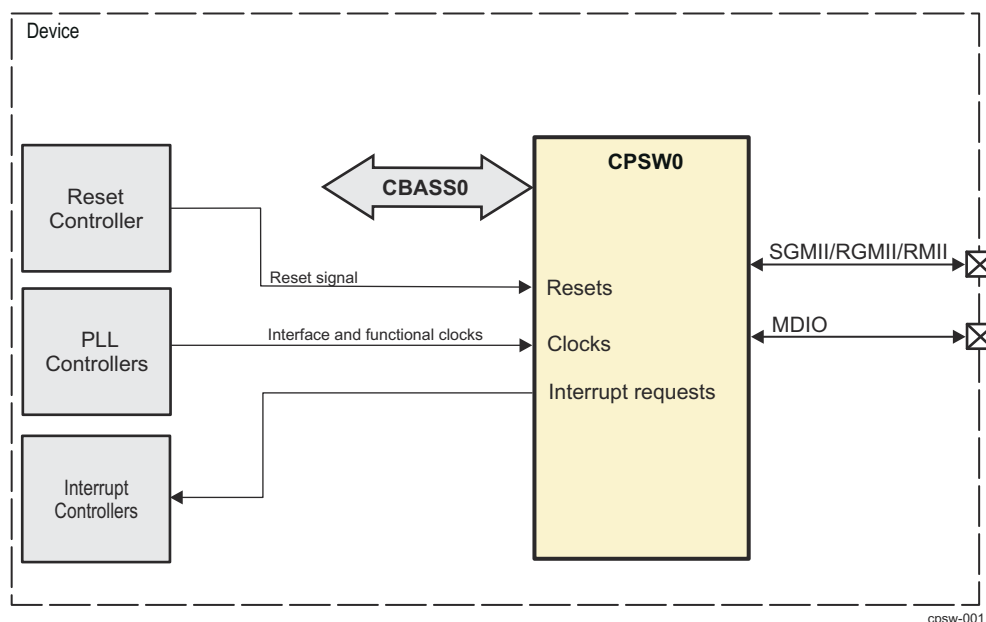
CPSW0 features the Serial Gigabit Media Independent Interface (SGMII), Reduced Gigabit Media Independent Interface (RGMII), Reduced Media Independent Interface (RMII) and the Management Data Input/Output (MDIO) interface for physical layer device (PHY) management.

The device has integrated one 9-port Gigabit Ethernet Switch subsystem into device MAIN domain named CPSW0. [Table 12-197](#) shows the CPSW0 module allocation within device domains.

**Table 12-197. CPSW0 Module Allocation within Device Domains**

Module Instance	Domain		
	WKUP	MCU	MAIN
CPSW0	-	-	✓

[Figure 12-159](#) shows the CPSW0 module overview.



**Figure 12-159. CPSW0 Overview**

#### 12.2.2.1.1 CPSW0 Features

The 9-port CPSW0 subsystem provides the following features:

- Eight Ethernet ports with selectable SGMII, RGMII, and RMII interfaces and an internal Communications Port Programming Interface (CPPI) port (port 0)
- Synchronous 10/100/1000 Mbit operation
- Flexible logical FIFO-based packet buffer structure
- Eight priority level Quality Of Service (QOS) support (802.1p)
- Support for Audio/Video Bridging (P802.1Qav/D6.0 and 802.1Qaz)
- Ethernet port reset isolation
- Support for IEEE 1588 Clock Synchronization (2008 Annex D, Annex E and Annex F)



- Timestamp module capable of time stamping external timesync events like Pulse-Per-Second and also generating Pulse-Per-Second outputs
- CPTS module that supports time stamping for IEEE1588 with support for 8 hardware push events and generation of compare output pulses
- DSCP Priority Mapping (IPv4 and IPv6)
- IPV4/IPV6 UDP/TCP checksum offload
- Priority Based Flow Control (802.1QBB) and Flow Control (802.3x) Support
- Wire rate switching (802.1d)
- Store and Forward Switching
- Non-Blocking switch fabric
- Time Sensitive Network Support
  - IEEE P802.3br/D2.0 Interspersing Express Traffic
  - IEEE 802.1Qbv/D2.2 Enhancements for Scheduled Traffic
- Address Lookup Engine (ALE)
  - 1024 ALE table entries
  - Parameterized number of addresses plus VLANs (total)
  - Wire rate lookup
  - LAN support
  - Host controlled time-based aging and/or auto-aging
  - Spanning tree support
  - L2 address lock and L2 filtering support
  - MAC authentication (802.1x)
  - Receive-based or destination-based Multicast and Broadcast rate limits
  - MAC address blocking
  - Source port locking
  - OUI (Vendor ID) host accept/deny feature
  - Configurable number of classifier/policers (96)
  - Port and VLAN, OUI, or address traffic mirroring
  - Port trunking for up to four trunks across any port combination
  - Policing of ingress data flows
  - Castagnoli or Ethernet CRC selectable per port
- EtherStats and 802.3 Stats Remote Network Monitoring (RMON) statistics gathering (per port statistics)
- Maximum frame size of 2020 bytes
- Management Data Input/Output (MDIO) module for PHY Management
- Host port CPPI Streaming Packet Interface (CPPI\_GCLK)
- Digital loopback and FIFO loopback modes supported
- Emulation support
- Full duplex mode supported in 10/100/1000 Mbps. Half-duplex mode supported only in 10/100 Mbps modes only.
- RAM Error Detection and Correction (SECCDED)

#### **12.2.2.1.2 CPSW0 Not Supported Features**

The following features are not supported for 9-port CPSW0 switch:

- MII and GMII Mode
- Gigabit half-duplex in 1Gbit or 10Gbit modes.
- Software reset on CPRGMII, CPRMII, PCSR submodules
- Interspersed Express Traffic (IET) in RMII mode
- Jumbo sized packets (9600 bytes)
- Rate limiting in half-duplex mode
- Dual VLAN switch operations
- MACSEC
- Synchronous Ethernet
- Cut Through Switching
- RGMII Internal Delay Mode disabled

### 12.2.2.1.3 Terminology



**Terminology:**

<b>AVB</b>	Audio Video Bridging
<b>AVBTP</b>	Audio Video Bridging Transport Protocol
<b>BMCA</b>	Best Master Clock Algorithm
<b>CFI</b>	Canonical Format Indicator
<b>CPPI</b>	Communications Port Programming Interface
<b>CPSW</b>	Common Platform Switch
<b>DLR</b>	Device Level Ring
<b>DSCP</b>	Differentiated Services Code Point
<b>EEE</b>	Energy Efficient Ethernet
<b>EMAC</b>	Ethernet Media Access Control
<b>EOP</b>	End of Packet
<b>EOQ</b>	End of Queue
<b>IPG</b>	Inter-Packet Gap
<b>LPI</b>	Low Power Indicator
<b>MDIO</b>	Management Data Input/Output
<b>MOF</b>	Middle of Frame
<b>OUI</b>	Organizationally Unique Identifier
<b>PFC</b>	Priority based Flow Control
<b>PTP</b>	Precision Time Protocol
<b>RMON</b>	Remote Monitoring
<b>RTCP</b>	RTP Control Protocol
<b>RTP</b>	Real-time Transport Protocol
<b>SCR</b>	Switched Central Resource
<b>SRP</b>	Stream Reservation Protocol
<b>TOS</b>	Type of Service
<b>VLAN</b>	Virtual Local Area Network

### 12.2.2.2 CPSW0 Environment

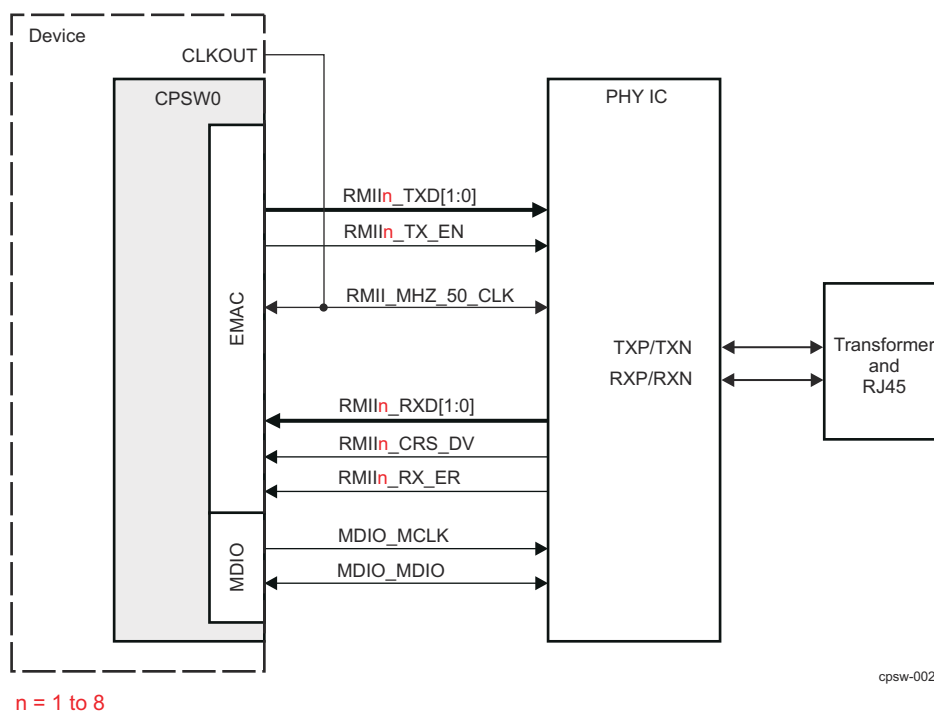
This section describes the CPSW0 external connections (environment).

#### 12.2.2.2.1 CPSW0 RMII Interface

Figure 12-160 shows a device with integrated EMAC and MDIO interfaced via a RMII connection in a typical system. The individual CPSW0 and MDIO signals for the RMII interface are summarized in Table 12-198.

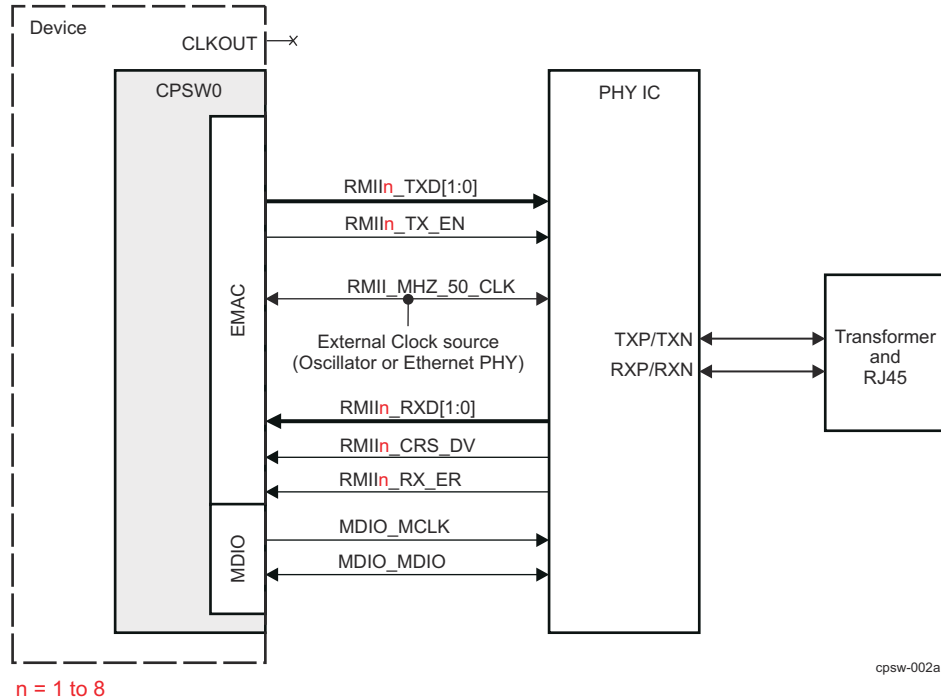
The CPSW0 module integrated in the device supports internal and external clock sources in RMII mode. Figure 12-160 shows the internal clock source for RMII\_MHZ\_50\_CLK clock. It is 50 MHz clock source that is provided on the CLKOUT device pin. For more details see *CPSW0 Integration*. This clock has to be routed on the PCB to the RMII\_REF\_CLK device pin and the external PHY, RMII clock input (shared by all RMII ports).

For more information, refer to either the IEEE 802.3 standard or ISO/IEC 8802-3:2000(E).



**Figure 12-160. RMII Interface Typical Application (Internal Clock Source)**

Figure 12-161 shows the external clock source for RMII\_MHZ\_50\_CLK clock. In this case a 50 MHz clock is available on the PCB and it can be sourced from an oscillator or from the Ethernet PHY. This externally generated clock has to be routed to both RMII\_REF\_CLK device pin and the external PHY, RMII clock input.



**Figure 12-161. RMII Interface Typical Application (External Clock Source)**

**Table 12-198. RMII I/O Description**

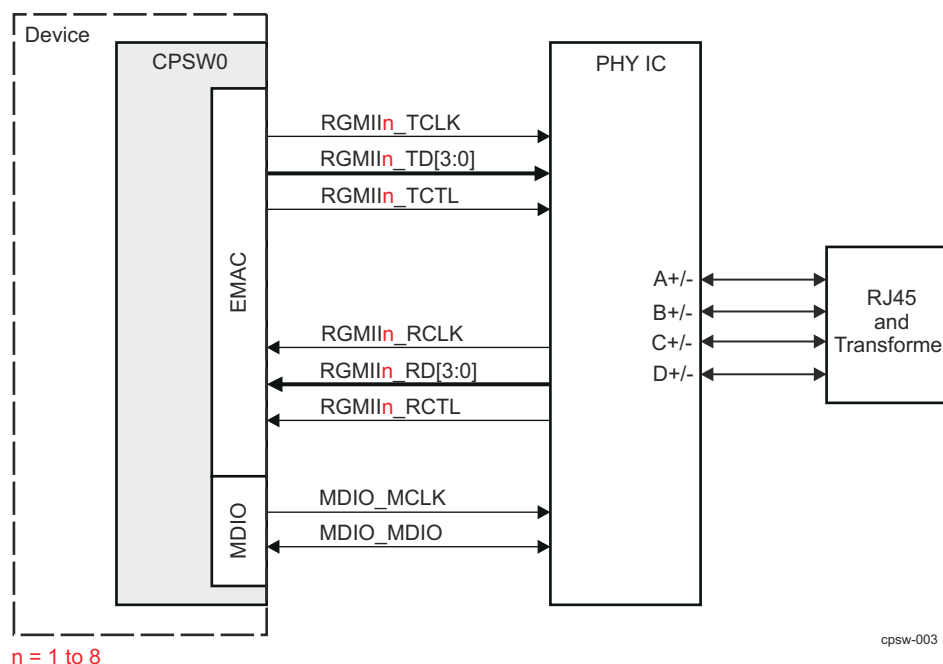
Signal <sup>(2)</sup>	Device Pin(s)	I/O <sup>(1)</sup>	Description
RMIIIn_TXD[1:0]	RMIIIn_TXD[1:0]	O	Transmit data. The transmit data pins are a collection of 2 bits of data. TXD0 is the least-significant bit (LSB). The signals are synchronized by RMII_MHZ_50_CLK and valid only when RMIIIn_TX_EN is asserted.
RMIIIn_TX_EN	RMIIIn_TX_EN	O	RMII transmit enable. The transmit enable signal indicates that the RMIIIn_TXD[1:0] pins are generating data for use by the PHY. RMIIIn_TX_EN is synchronous to RMII_MHZ_50_CLK.
RMII_MHZ_50_CLK	RMII_REF_CLK	I	RMII 50MHz reference clock.  The reference clock is used to synchronize all RMII signals. RMII_MHZ_50_CLK must be continuous and fixed at 50 MHz.
RMIIIn_RXD[1:0]	RMIIIn_RXD[1:0]	I	Receive data. The receive data pins are a collection of 2 bits of data. RXD0 is the least-significant bit (LSB). The signals are synchronized by RMII_MHZ_50_CLK and valid only when RMIIIn_CRS_DV is asserted and RMIIIn_RX_ER is de-asserted.
RMIIIn_CRS_DV	RMIIIn_CRS_DV	I	Carrier sense/receive data valid. Multiplexed signal between carrier sense and receive data valid.
RMIIIn_RX_ER	RMIIIn_RX_ER	I	Receive error. The receive error signal is asserted to indicate that an error was detected in the received frame.
MDIO_MCLK	MDIO0_MDC	O	Management data clock (MDIO_MCLK). The MDIO data clock is sourced by the MDIO module on the system. It is used to synchronize MDIO data access operations done on the MDIO0_MDIO data pin.
MDIO_MDIO	MDIO0_MDIO	I/O	MDIO data pin drives PHY management data into and out of the PHY by way of an access frame consisting of start of frame, read/write indication, PHY address, register address, and data bit cycles. The MDIO0_MDIO pin acts as an output for all but the data bit cycles at which time it is an input for read operations.

(1) I = Input; O = Output

(2) n 1 to 8

### 12.2.2.2.2 CPSW0 RGMII Interface

Figure 12-162 shows a device with integrated EMAC and MDIO interfaced via a RGMII connection in a typical system. The individual CPSW0 and MDIO signals for the RGMII interface are summarized in Table 12-199.



**Figure 12-162. RGMII Interface Typical Application**

**Table 12-199. RGMII I/O Description**

Signal <sup>(2)</sup>	Device Pin(s)	I/O <sup>(1)</sup>	Description
RGMII_n_TD[3:0]	RGMII_n_TD[3:0]	O	The transmit data pins are a collection of 4 bits of data. TD0 is the least-significant bit (LSB). The signals are valid only when RGMII_n_TCTL is asserted.
RGMII_n_TCTL	RGMII_n_TX_CTL	O	Transmit Control/enable. The transmit enable signal indicates that the TD pins are generating data for use by the PHY.
RGMII_n_TCLK	RGMII_n_TXC	O	The transmit reference clock. The clock is 2.5 MHz at 10 Mbps operation, 25 MHz at 100 Mbps operation, and 125 MHz at 1000 Mbps of operation.
RGMII_n_RD[3:0]	RGMII_n_RD[3:0]	I	The receive data pins are a collection of 4 bits of data. RD0 is the least-significant bit (LSB). The signals are valid only when RGMII_n_RX_CTL is asserted
RGMII_n_RCTL	RGMII_n_RX_CTL	I	The receive data valid/control signal indicates that the RD pins are nibble data for use by the EMAC.
RGMII_n_RCLK	RGMII_n_RXC	I	The receive clock is a continuous clock that provides the timing reference for receive operations. The clock is generated by the PHY and is 2.5 MHz at 10 Mbps operation, 25 MHz at 100 Mbps operation, 125 MHz at 1000 Mbps of operation.
MDIO_MCLK	MDIO0_MDC	O	Management data clock (MDIO_MCLK). The MDIO data clock is sourced by the MDIO module on the system. It is used to synchronize MDIO data access operations done on the MDIO0_MDIO pin.
MDIO_MDIO	MDIO0_MDIO	I/O	The MDIO0_MDIO pin drives PHY management data into and out of the PHY by way of an access frame consisting of start of frame, read/write indication, PHY address, register address, and data bit cycles. The MDIO0_MDIO pin acts as an output for all but the data bit cycles at which time it is an input for read operations.

(1) I = Input; O = Output

(2) n 1 to 8

---

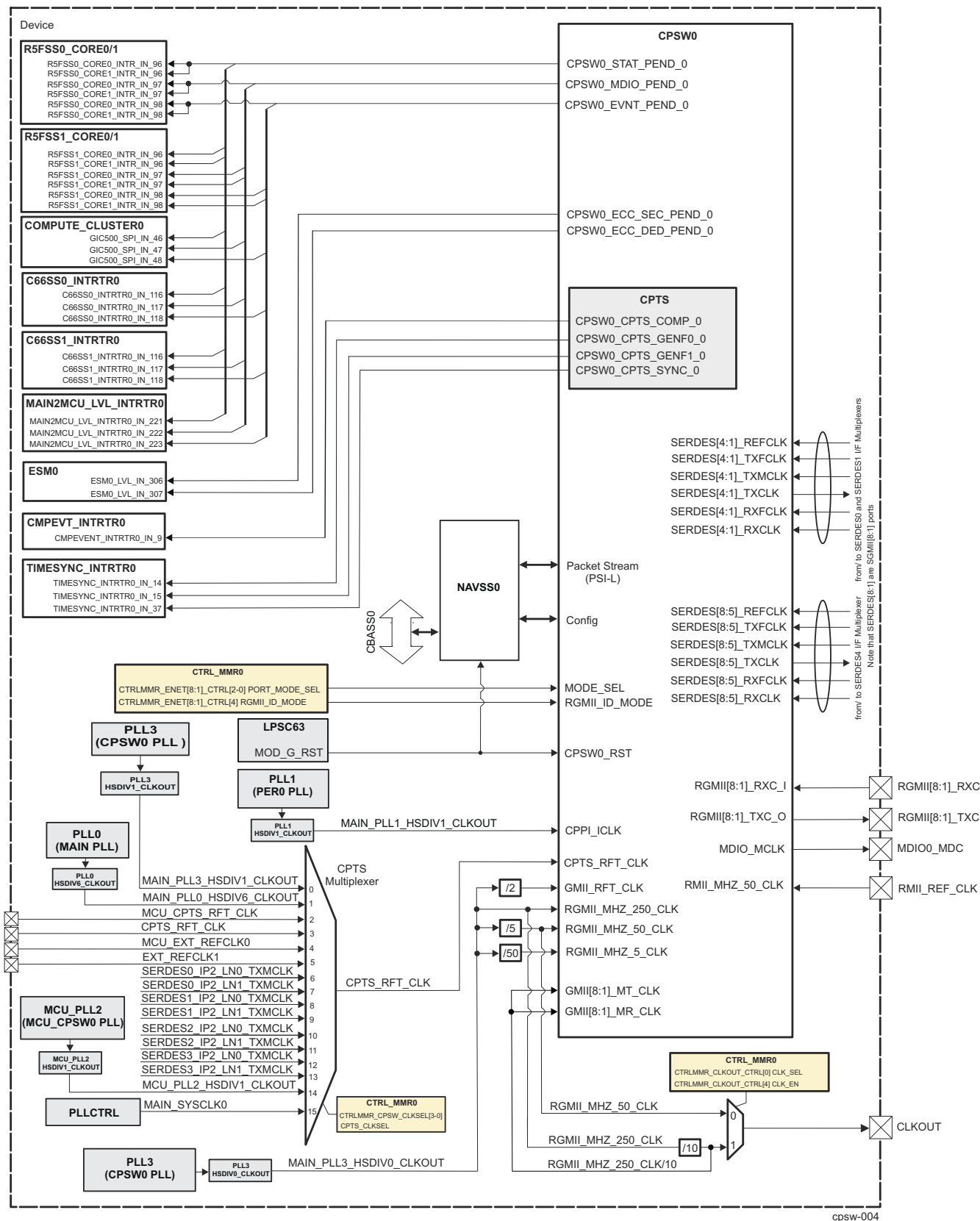
**Note**

The Control Module registers assign the specific function to the device pads. For more information on Control Module settings, see Pad Configuration Registers in *Control Module (CTRL\_MMR)* and the device-specific Datasheet.

---

### 12.2.2.3 CPSW0 Integration

[Figure 12-163](#) shows the integration of the CPSW0 module in the device.



### Figure 12-163. CPSW0 Integration

The following CPSW0 control registers are located in CTRL\_MMR0 module: CTRLMMR\_ENET1\_CTRL to CTRLMMR\_ENET8\_CTRL, CTRLMMR\_CPSW\_CLKSEL.

Table 12-200 through Table 12-202 summarize the integration of the CPSW0 module in the device.

**Table 12-200. CPSW0 Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
CPSW0	PSC0	PD4	LPSC63	CBASS0

**Table 12-201. CPSW0 Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
CPSW0	CPPI_ICLK	MAIN_PLL1_HSDIV1_CL KOUT	PLL1 (HSDIV1 of PER0 PLL)	CPPI packet streaming interface clock (320-MHz). Main clock for CPSW0.
	GMII_RFT_CLK	MAIN_PLL3_HSDIV0_CL KOUT/2	PLL3 (HSDIV0 of CPSW0 PLL)	125-MHz GMII Gigabit mode clock.
	RGMII_MHZ_5_CLK	MAIN_PLL3_HSDIV0_CL KOUT/50	PLL3 (HSDIV0 of CPSW0 PLL)	5-MHz RGMII reference clock.
	RGMII_MHZ_50_CLK	MAIN_PLL3_HSDIV0_CL KOUT/5	PLL3 (HSDIV0 of CPSW0 PLL)	50-MHz RGMII reference clock.
	RGMII_MHZ_250_CLK	MAIN_PLL3_HSDIV0_CL KOUT	PLL3 (HSDIV0 of CPSW0 PLL)	250-MHz RGMII reference clock.
	RMII_MHZ_50_CLK	RMII_REF_CLK	RMII_REF_CLK pad	50-MHz RMII reference clock. This clock is derived from the RMII_REF_CLK pad.
	CPTS_RFT_CLK	MAIN_PLL3_HSDIV1_CL KOUT	HSDIV1 of CPSW0 PLL Controller, selected through CPTS Multiplexer (200 or 250-MHz clock)	CPTS IEEE 1588 clock. Selected through the CTRLMMR_CPSW_CLKSEL register.
		MAIN_PLL0_HSDIV6_CL KOUT	HSDIV6 of MAIN PLL Controller, selected through CPTS Multiplexer (200 or 250-MHz clock)	
		MCU_CPTS_RFT_CLK pad	MCU_CPTS_RFT_CLK pad, selected through CPTS Multiplexer (200-MHz clock)	
		CPTS_RFT_CLK pad	CPTS_RFT_CLK pad, selected through CPTS Multiplexer (200-MHz clock)	
		MCU_EXT_REFCLK0 pad	MCU_EXT_REFCLK0 pad, selected through CPTS Multiplexer (100-MHz clock)	
		EXT_REFCLK1 pad	EXT_REFCLK1 pad, selected through CPTS Multiplexer (100-MHz clock)	
		SERDES0_IP2_LN0_TXM_CLK	SERDES0 Lane0 (500-MHz clock)	
		SERDES0_IP2_LN1_TXM_CLK	SERDES0 Lane1 (500-MHz clock)	
		SERDES1_IP2_LN0_TXM_CLK	SERDES1 Lane0 (500-MHz clock)	



**Table 12-201. CPSW0 Clocks and Resets (continued)**

	SERDES1_IP2_LN1_TXM_CLK	SERDES1 Lane1 (500-MHz clock)	
	SERDES2_IP2_LN0_TXM_CLK	SERDES2 Lane0 (500-MHz clock)	
	SERDES2_IP2_LN1_TXM_CLK	SERDES2 Lane1 (500-MHz clock)	
	SERDES3_IP2_LN0_TXM_CLK	SERDES3 Lane0 (500-MHz clock)	
	SERDES3_IP2_LN1_TXM_CLK	SERDES3 Lane1 (500-MHz clock)	
	MCU_PLL2_HSDIV1_CLK_OUT	HSDIV1 of MCU_CPSW0 PLL Controller, selected through CPTS Multiplexer (500-MHz clock)	
	MAIN_SYSCCLK0	PLLCTRL (500-MHz clock)	
GMII1_MT_CLK	RGMII_MHZ_250_CLK/10	RGMII_MHZ_250_CLK	<del>Note: GMII mode is not supported on this device. GMII<sub>n</sub>_MT_CLK (where n = 1 to 8) transmit reference clocks are needed to enable clock-stop protocol on this module.</del>
GMII2_MT_CLK			
GMII3_MT_CLK			
GMII4_MT_CLK			
GMII5_MT_CLK			
GMII6_MT_CLK			
GMII7_MT_CLK			
GMII8_MT_CLK			
GMII1_MR_CLK	RGMII_MHZ_250_CLK/10	RGMII_MHZ_250_CLK	<del>Note: GMII mode is not supported on this device. GMII<sub>n</sub>_MR_CLK (where n = 1 to 8) receive reference clocks are needed to enable clock-stop protocol on this module.</del>
GMII2_MR_CLK			
GMII3_MR_CLK			
GMII4_MR_CLK			
GMII5_MR_CLK			
GMII6_MR_CLK			
GMII7_MR_CLK			
GMII8_MR_CLK			
RGMII1_RXC_I	RGMII1_RXC	RGMII1_RXC pad	RGMII1 reference clock that provides the timing reference for receive operations.
RGMII2_RXC_I	RGMII2_RXC	RGMII2_RXC pad	RGMII2 reference clock that provides the timing reference for receive operations.
RGMII3_RXC_I	RGMII3_RXC	RGMII3_RXC pad	RGMII3 reference clock that provides the timing reference for receive operations.
RGMII4_RXC_I	RGMII4_RXC	RGMII4_RXC pad	RGMII4 reference clock that provides the timing reference for receive operations.
RGMII5_RXC_I	RGMII5_RXC	RGMII5_RXC pad	RGMII5 reference clock that provides the timing reference for receive operations.

**Table 12-201. CPSW0 Clocks and Resets (continued)**

RGMII6_RXC_I	RGMII6_RXC	RGMII6_RXC pad	RGMII6 reference clock that provides the timing reference for receive operations.
RGMII7_RXC_I	RGMII7_RXC	RGMII7_RXC pad	RGMII7 reference clock that provides the timing reference for receive operations.
RGMII8_RXC_I	RGMII8_RXC	RGMII8_RXC pad	RGMII8 reference clock that provides the timing reference for receive operations.
RGMII1_TXC_O	RGMII1_TXC	RGMII1_TXC pad	RGMII1 transmit reference clock.
RGMII2_TXC_O	RGMII2_TXC	RGMII2_TXC pad	RGMII2 transmit reference clock.
RGMII3_TXC_O	RGMII3_TXC	RGMII3_TXC pad	RGMII3 transmit reference clock.
RGMII4_TXC_O	RGMII4_TXC	RGMII4_TXC pad	RGMII4 transmit reference clock.
RGMII5_TXC_O	RGMII5_TXC	RGMII5_TXC pad	RGMII5 transmit reference clock.
RGMII6_TXC_O	RGMII6_TXC	RGMII6_TXC pad	RGMII6 transmit reference clock.
RGMII7_TXC_O	RGMII7_TXC	RGMII7_TXC pad	RGMII7 transmit reference clock.
RGMII8_TXC_O	RGMII8_TXC	RGMII8_TXC pad	RGMII8 transmit reference clock.
MDIO_MCLK	MDIO_MCLK	MDIO0_MDC pad	Management data clock (MDIO_MCLK). The MDIO data clock is sourced by the MDIO module on the system. It is used to synchronize MDIO data access operations done on the MDIO pin.

Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
CPSW0	CPSW0_RST	MOD_G_RST	LPSC63	Module Reset

**Table 12-202. CPSW0 Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
CPSW0	CPSW0_STAT_PEND_0	R5FSS0_CORE0_INTR_IN_9_6	R5FSS0_CORE0	CPSW0 statistic pending interrupt 0	Level
		R5FSS0_CORE1_INTR_IN_9_6	R5FSS0_CORE1	CPSW0 statistic pending interrupt 0	Level
		R5FSS1_CORE0_INTR_IN_9_6	R5FSS1_CORE0	CPSW0 statistic pending interrupt 0	Level
		R5FSS1_CORE1_INTR_IN_9_6	R5FSS1_CORE1	CPSW0 statistic pending interrupt 0	Level
		GIC500_SPI_IN_46	COMPUTE_CLUSTER0	CPSW0 statistic pending interrupt 0	Level
		C66SS0_INTRTR0_IN_116	C66SS0_INTRTR0	CPSW0 statistic pending interrupt 0	Level
		C66SS1_INTRTR0_IN_116	C66SS1_INTRTR0	CPSW0 statistic pending interrupt 0	Level
		MAIN2MCU_LVL_INTRTR0_IN_221	MAIN2MCU_LVL_INTRTR0	CPSW0 statistic pending interrupt 0	Level

**Table 12-202. CPSW0 Hardware Requests (continued)**

CPSW0_MDIO_PEND_0	R5FSS0_CORE0_INTR_IN_9	R5FSS0_CORE0	CPSW0 MDIO interrupt	Level
	R5FSS0_CORE1_INTR_IN_9	R5FSS0_CORE1	CPSW0 MDIO interrupt	Level
	R5FSS1_CORE0_INTR_IN_9	R5FSS1_CORE0	CPSW0 MDIO interrupt	Level
	R5FSS1_CORE1_INTR_IN_9	R5FSS1_CORE1	CPSW0 MDIO interrupt	Level
	GIC500_SPI_IN_47	COMPUTE_CLUSTER0	CPSW0 MDIO interrupt	Level
	C66SS0_INTRTR0_IN_117	C66SS0_INTRTR0	CPSW0 MDIO interrupt	Level
	C66SS1_INTRTR0_IN_117	C66SS1_INTRTR0	CPSW0 MDIO interrupt	Level
CPSW0_EVNT_PEND_0	MAIN2MCU_LVL_INTRTR0_IN_222	MAIN2MCU_LVL_INTRTR0	CPSW0 MDIO interrupt	Level
	R5FSS0_CORE0_INTR_IN_8	R5FSS0_CORE0	CPSW0 event pending interrupt	Level
	R5FSS0_CORE1_INTR_IN_8	R5FSS0_CORE1	CPSW0 event pending interrupt	Level
	R5FSS1_CORE0_INTR_IN_8	R5FSS1_CORE0	CPSW0 event pending interrupt	Level
	R5FSS1_CORE1_INTR_IN_8	R5FSS1_CORE1	CPSW0 event pending interrupt	Level
	GIC500_SPI_IN_48	COMPUTE_CLUSTER0	CPSW0 event pending interrupt	Level
	C66SS0_INTRTR0_IN_118	C66SS0_INTRTR0	CPSW0 event pending interrupt	Level
CPSW0_ECC_SEC_PEN D_0	C66SS1_INTRTR0_IN_118	C66SS1_INTRTR0	CPSW0 event pending interrupt	Level
	MAIN2MCU_LVL_INTRTR0_IN_223	MAIN2MCU_LVL_INTRTR0	CPSW0 event pending interrupt	Level
	ESM0_LVL_IN_306	ESM0	CPSW0 SEC ECC error interrupt	Level
CPSW0_ECC_DED_PEN D_0	ESM0_LVL_IN_307	ESM0	CPSW0 DED ECC error interrupt	Level

Time Sync and Compare Events					
Module Instance	Module Event	Destination Event Input	Destination	Description	Type
CPSW0	CPSW0_CPTS_COMP_0	CMPEVENT_INTRTR0_IN_9	CMPEVT_INTRTR0	CPSW0 compare event interrupt	Edge
	CPSW0_CPTS_GENF0_0	TIMESYNC_INTRTR0_IN_14	TIMESYNC_INTRTR0	CPSW0 CPTS generator function event interrupt 0	Edge
	CPSW0_CPTS_GENF1_0	TIMESYNC_INTRTR0_IN_15	TIMESYNC_INTRTR0	CPSW0 CPTS generator function event interrupt 1	Edge
	CPSW0_CPTS_SYNC_0	TIMESYNC_INTRTR0_IN_37	TIMESYNC_INTRTR0	CPSW0 CPTS sync event interrupt	Edge

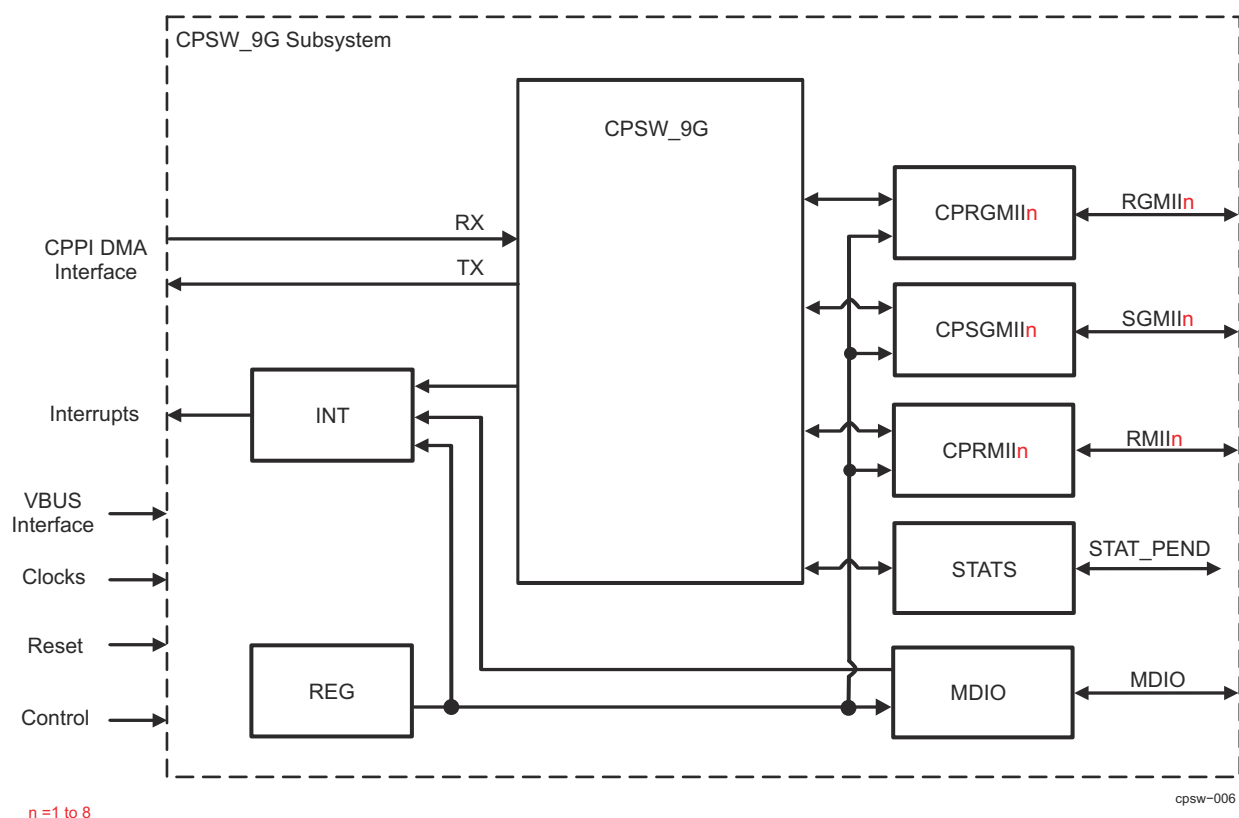
### 12.2.2.4 CPSW0 Functional Description

The 9-port switch Ethernet subsystem module (CPSW) is compliant to the IEEE Std 802.3 Specification. CPSW top level functional block diagram is shown in [Figure 12-164](#).

#### 12.2.2.4.1 Functional Block Diagram

The 9-port Ethernet subsystem consists of:

- CPSW\_9G
- One RGMII<sub>n</sub> (where n = 1 to 8) interface module
- One SGMII<sub>n</sub> (where n = 1 to 8) interface modules
- One RMII<sub>n</sub> (where n = 1 to 8) interface module
- One Host Port 0 CPPI Packet Streaming Interface
- CPSW subsystem control registers (REG)
- One MDIO interface module
- One Interrupt Controller module



**Figure 12-164. CPSW Top Level Block Diagram**

#### 12.2.2.4.2 CPSW Ports

The Ethernet Subsystem has 9 ports. Port 0 is the Host port (internal to the Subsystem). Port 1 to Port 8 are the external ports connected to RGMII, SGMII or RMII interfaces as per the interface selected.

Naming conventions followed in this chapter:

- Port0 is referred to the CPPI Host Port
- Port1 to Port8 is referred to the interfaces RGMII/SGMII/RMII

##### 12.2.2.4.2.1 Interface Mode Selection

The 9-port switch (CPSW) Ethernet Subsystem has eight 10/100/1000 Ethernet ports with selectable RMII, RGMII and SGMII interfaces.

The interface modes for all 8 Ethernet ports are selected by configuring the Ethernet interface mode selection bitfield (PORT\_MODE\_SEL) in the CTRLMMR\_ENET1\_CTRL to CTRLMMR\_ENET8\_CTRL registers.

See the device-specific Datasheet for configuring the pin mux mode as per the interface selected.

#### **12.2.2.4.3 Clocking**

##### **12.2.2.4.3.1 Subsystem Clocking**

CPSW clocking summary is shown in *CPSW0 Integration*.

##### **12.2.2.4.3.2 Interface Clocking**

Data is transmitted and received with respect to the reference clocks of the interface pins.

##### **12.2.2.4.3.2.1 RGMII Interface Clocking**

RGMII\_RXC, RGMII\_TXC frequencies are:

- 2.5 MHz at 10 Mbps
- 25 MHz at 100 Mbps
- 125 MHz at 1000 Mbps

##### **12.2.2.4.3.2.2 RMII Interface Clocking**

RMII interface clock RMII\_50MHZ\_CLK frequency is:

- 50 MHz at 10 Mbps
- 50 MHz at 100 Mbps

RMII\_REF\_CLK from device pin or internal RGMII\_MHZ\_50\_CLK clock (default clock) can be used as the clock source for RMII interface. For more details on RMII clocking, please see *CPSW0 Integration*

CTRLMMR\_CLKOUT\_CTRL[4]CLK\_EN and CTRLMMR\_CLKOUT\_CTRL[0]CLK\_SEL bits are used to enable and select the clock source for CLKOUT device pin.

##### **12.2.2.4.3.2.3 MDIO Clocking**

The MDIO clock is based on a divide-down of the interface (CPPI\_ICLK) clock. The application software or driver must control the divide-down value.

See the CPSW\_MDIO\_CONTROL\_REG register for configuring the Clock Divider ([15-0]CLKDIV) value.

##### **12.2.2.4.4 Software IDLE**

The submodule software idle register bits enable CPSW operation to be completely or partially suspended by software control. There are two CPSW submodules that contain software idle register bits. Each of the two submodules may be individually commanded to enter the idle state. The idle state is entered at packet boundaries, and no further packet operations will occur on an idled submodule until the idle command is removed. The CPSW module enters the idle state when all two submodules are commanded to enter and have entered the idle state. Idle status is determined by reading or polling the two submodule idle bits. The CPSW\_9G is in the idle state when all two submodules are in the idle state. The CPSW\_SOFT\_IDLE\_REG[0] SOFT\_IDLE bit may be set if desired after the submodules are in the idle state. The SOFT\_IDLE bit causes packets to not be transferred from one FIFO to another FIFO internal to the switch.

##### **12.2.2.4.5 Interrupt Functionality**

CPSW Ethernet Subsystem has six interrupt outputs:

- EVNT\_PEND - CPTS Event Pending Interrupt
- STAT\_PEND0 – Statistics Pending Interrupts
- ECC\_DED\_INT - ECC DED Level Interrupt
- ECC\_SEC\_INT - ECC SEC Level Interrupt
- MDIO\_INTR - MDIO Pending Interrupt (combined MDIO\_LINKINT and MDIO\_USERINT events)

#### 12.2.2.4.5.1 EVNT\_PEND Interrupt

See [Section 12.2.2.4.7](#), *Common Platform Time Sync (CPTS)* for more details on this interrupt.

#### 12.2.2.4.5.2 Statistics Interrupt (STAT\_PEND0)

The statistics level interrupt (STAT\_PEND0) will be asserted, if enabled when any statistics value is greater than or equal to 8000 0000h. The statistics interrupt is cleared by writing to decrement any statistics values greater than 8000 0000h (such that their new values are less than 8000 0000h). The statistics are mapped into internal memory space and are 32-bits wide. The raw and masked statistics interrupt status may be read by reading the CPSW\_CPTS\_INTSTAT\_RAW\_REG and CPSW\_CPTS\_INTSTAT\_MASKED\_REG registers, respectively.

The statistics interrupt is enabled by setting to 1h the TS\_PEND\_EN bit in the CPSW\_CPTS\_INT\_ENABLE\_REG register.

#### 12.2.2.4.5.3 ECC DED Level Interrupt (ECC\_DED\_INT)

Level interrupt (ECC\_DED\_INT) indicating an ECC double error has been detected.

#### 12.2.2.4.5.4 ECC SEC Level Interrupt (ECC\_SEC\_INT)

Level interrupt (ECC\_SEC\_INT) indicating an ECC single error has been detected and corrected.

#### 12.2.2.4.5.5 MDIO Interrupts

**Normal Mode:** (CPSW\_MDIO\_POLL\_REG[30] STATECHANGEMODE = 0h)

MDIO\_LINKINT event is set if there is a change in the link state of the PHY corresponding to the address in the PHYADR\_MON field of the CPSW\_MDIO\_USER\_PHY\_SEL\_REG\_k register and the corresponding LINKINT\_ENABLE bit is set. The MDIO\_LINKINT event is also captured in the MDIO CPSW\_MDIO\_LINK\_INT\_MASKED\_REG register. When the GO bit in the CPSW\_MDIO\_USER\_ACCESS\_REG\_k register transitions from 1 to 0, indicating the completion of a user access, and the corresponding USERINTMASKSET bit in the CPSW\_MDIO\_USER\_INT\_MASK\_SET\_REG register is set, the MDIO\_USERINT signal is asserted. The MDIO\_USERINT event is also captured in the CPSW\_MDIO\_USER\_INT\_MASKED\_REG register.

**State Change Mode:** (CPSW\_MDIO\_POLL\_REG[30] STATECHANGEMODE = 1h)

In State Change Mode, the MDIO will assert MDIO\_LINKINT[0] when any bit in the MDIO CPSW\_MDIO\_ALIVE\_REG or CPSW\_MDIO\_LINK\_REG registers changes due to MDIO operations. The MDIO\_LINKINT event is also captured in the CPSW\_MDIO\_LINK\_INT\_MASKED\_REG register. MDIO\_LINKINT[1] output is unused in State Change Mode. The CPSW\_MDIO\_USER\_PHY\_SEL\_REG\_k registers are unused in state change mode.

**Manual Mode:** (CPSW\_MDIO\_POLL\_REG[31] MANUALMODE = 1h)

Manual Mode allows software to directly control the MDIO serial clock output (CPSW\_MDIO\_MANUAL\_IF\_REG[2] MDIO\_MDCLK\_O) and the MDIO serial data output enable (CPSW\_MDIO\_MANUAL\_IF\_REG[1] MDIO\_OE). Manual Interface Mode is enabled when the MANUALMODE bit is set in the CPSW\_MDIO\_POLL\_REG register. Manual Mode is intended to be used by software for slow speed general purpose IO operations and not for MDIO PHY operations.

#### 12.2.2.4.6 CPSW\_9G

The CPSW\_9G RMII/ RGMII interface is compliant to the IEEE Std 802.3 Specification.

The CPSW\_9G contains one Ethernet port interface (Ethernet port 1), one CPPI packet streaming interface host port (port 0), Common Platform Time Sync (CPTS), ALE Engine and Statistics (STATS). A top-level block diagram of the CPSW\_9G is shown in [Figure 12-165](#).

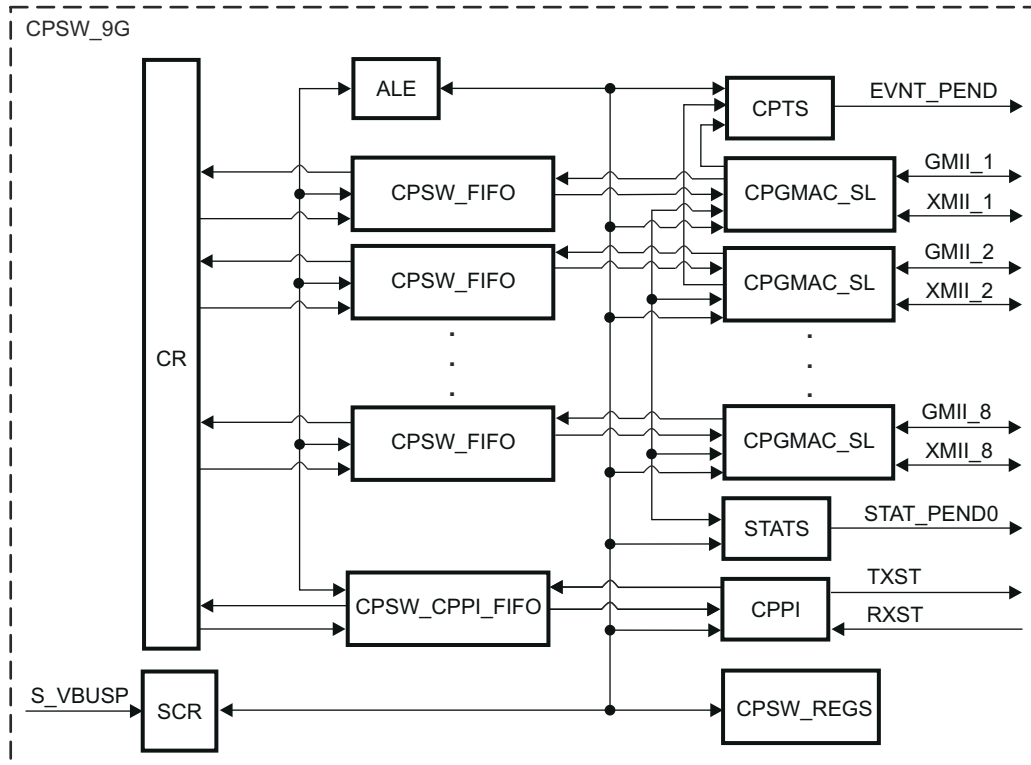


Figure 12-165. CPSW\_9G Block Diagram

#### 12.2.2.4.6.1 Address Lookup Engine (ALE)

The Address Lookup Engine (ALE) is a sub-block of the CPSW Switch and it processes all received packets and determines the packet destination port(s). The ALE uses the incoming packet data: received port number, destination address, source address, length/type, and VLAN information to determine how the packet should be forwarded. The ALE outputs the port mask to the switch fabric that indicates the port(s) the packet should be forwarded to. The ALE is enabled when the `ENABLE_ALE` bit in the `CPSW_ALE_CONTROL` register is set. All packets are dropped when the `ENABLE_ALE` bit is cleared to 0h.

##### 12.2.2.4.6.1.1 Error Handling

In normal operation, the Ethernet port modules are configured to issue an abort, instead of an end of packet, at the end of a packet that contains an error (runt, frag, oversize, jabber, crc, alignment, code etc.) or at the end of a MAC control packet. However, when the `CPSW_PN_MAC_CONTROL_REG_k` configuration bit(s) `RX_CEF_EN`, `RX_CSF_EN`, or `RX_CMF_EN` are set, error frames, short frames or MAC control frames have a normal end of packet instead of an abort at the end of the packet. When the ALE receives a packet that contains errors (due to a set header error bit), or a MAC control frame and does not receive an abort, the packet will be forwarded only to the host port (port 0). Packets with errors that are forwarded to the host have no VLAN untagging or drop due to rate limiting. No ALE learning occurs on packets with errors or mac control frames. Learning is based on source address and lookup is based on destination address. Directed packets from the host are not learned, updated, or touched.

##### 12.2.2.4.6.1.2 Bypass Operations

The ALE may be configured to operate in bypass mode by setting the `ENABLE_BYPASS` bit in the `CPSW_ALE_CONTROL` register. When in bypass mode, all Ethernet port received packets are forwarded only to the host port (port 0). In bypass mode, the ALE processes host port transmit packets the same as in normal mode. In general, packets would be directed by the host in bypass mode.



### 12.2.2.4.6.1.3 OUI Deny or Accept

The ALE may be configured to operate in OUI deny mode by setting the ENABLE\_OUI\_DENY bit in the CPSW\_ALE\_CONTROL register. When in OUI deny mode, a packet with a non-matching OUI source address will be dropped unless the destination address matches a multicast table entry with the SUPER bit set. When ENABLE\_OUI\_DENY bit is cleared, any packet source address matching an OUI address table entry will be dropped to the host unless the destination address matches with a supervisory address table entry. Broadcast packets will be dropped unless the broadcast address is entered into the table with the SUPER bit set. Unicast packets will be dropped unless the unicast address is in the table with BLOCK and SECURE both set (supervisory unicast packet).

### 12.2.2.4.6.1.4 Statistics Counting

ALE sends many statistics along with the frame routing so the CPSW can count them on a per port basis. There are many reasons to drop frames the below drop events are individually counted in CPSW per port statistics counters.

**Table 12-203. Learned Address Control Bits**

Abbreviation	Description
vddc	Dual VLAN or Double VLAN drop counter.
frdc	IPv4 fragmented drop counter.
nldc	IPv4 Protocol or IPv6 Next Header drops due to next header limit drop counter.
etdc	Ether Type field is <= 1500 and the frame size is too small drop counter.
smdc	The Source Address has bit 40 set drop counter.
badc	The Source or Destination Address blocked drop counter.
svdc	The Source Address is locked to a different port drop counter.
des	The Source and Destination are equal and the Source Address is not in the table drop counter.
adc	The Source Address is not in the table in authentication mode drop counter.
vicdc	The VLAN ingress check failed drop counter.
rlcd	Rate Limit Drop Counter.
over	Lookup Overflow, the rate of the ALE clock is not sufficient to handle the incoming frame rate (Should never happen).
drop	Packet is dropped and not forwarded.

### 12.2.2.4.6.1.5 Automotive Security Features

The ALE has many automotive security features that most enterprise switches do not require.

- VLANs can be configured to not allow fragmented IPv4 frames. That is a VLAN can be configured to not allow fragmented IPv4 traffic.
- VLANs can be configured to only allow up to four different IPv4 Protocols or IPv6. Next Header values, for example a VLAN can be configured to only allow TCP traffic in both IPv4 and IPv6 packets.
- Drop invalid Source Addresses, that is drop Source Addresses with bit 40 set (Multicast/Broadcast indicator on Destination Addresses)
- IEEE802.3 Length Check, drop frames that the IEEE802.3 Length is not contained within the frame. (Ether Types 0-1500)
- Any Source Address can be secured to a port dropping any attempts from other ports to masquerade as a service.



- Any source or destination address can be blocked.
- Per Port or Per VLAN ingress checking, dropping traffic from non-member ports.
- Classification, Policing on L2 and L3 information.

#### **12.2.2.4.6.1.6 CPSW Switching Solutions**

The host port can operate in many different modes as well depending on the functionality of the host. It is important to understand the modes and configure them properly.

The ALE Table is designed to maximize the modes without compromise of the system functionality as well.

##### **12.2.2.4.6.1.6.1 Basics of 5-port Switch Type**

The 5-port switch has a host port, and that port can operate in two fundamental modes. Bridge mode allows the host to extend the switched domain to another network like Wi-Fi or another multi-port switch. In this case the host must be able to see unknown unicast addresses so they can be broadcast to the other network. In Port mode the host need not see any unknown unicast traffic. The CPSW\_ALE\_CONTROL[8] EN\_HOST\_UNI\_FLOOD bit determines the host mode for unknown unicast traffic. This bit should only be set if you are bridging two or more networks together.

In 5-port switch, the ALE table is shared between the host port and the network.

##### **12.2.2.4.6.1.7 VLAN Routing and OAM Operations**

###### **12.2.2.4.6.1.7.1 InterVLAN Routing**

The CPSW module supports wire rate InterVLAN routing for a small number of routes, that is the host will setup an ALE classifier with and associated egress operation that will cause the CPSW to perform particular egress operations. The ALE can optionally check time to live validity as well.

The ALE uses the classifier along with an egress opcode, destination port mask and TTL check field to tell the CPSW to manipulate the packet on the egress. The CPSW will use the opcode along with a per port operation table to process the packet. By setting up the CPSW egress operation table you can replace the DA, SA and VLAN along with optionally updating the time to live IP header field. This allows the CPSW to perform the routing function for a small set of routes without getting the local host/CPU involved.

The Egress opcode will only be use for a classifier match and the packet would normally be sent only to the host. That is the host would have routed the packet but the CPSW has been configured to do the work instead. In the event that the time-to-live check feature is enabled and the time-to-live is either 0 or 1, the packet will not get the egress opcode and instead be sent to the host as if the route is not setup. This allows the host to deal with invalid TTL fields.

###### **12.2.2.4.6.1.7.2 OAM Operations**

The ALE supports OAM loopback on ports so that a remote link can be tested. That is a port placed in OAM loopback will echo packets received on a port back to the port with an egress op of 0xFF which will swap the SA and DA on egress in the CPSW.

Any supervisory packet will not be affected, so the spanning tree and other bridging functions are not affected.

Packets will only be echoed if the port is in OAM loopback mode, the received packet is not a supervisor packet. The port is in a forwarding state and the packet received DA!=SA and there are no errors in the packet.

When a port is in OAM loopback the port will not egress traffic from other ports, no address for loop backed traffic will be learned if enabled. Any packet received on the OAM loopback port with an error will be process as if the port is not in OAM. That is if the host has enabled copy errored frames the errored frames will be sent to the host instead.

###### **12.2.2.4.6.1.8 Supervisory packets**

Multicast supervisory packets are designated by the SUPER bit in the table entry. Unicast supervisory packets are indicated when BLOCK and SECURE are both set. Supervisory packets are not dropped due to rate limiting, OUI, or VLAN processing. The purpose of supervisory packets is to allow packets that would be otherwise

blocked to be forwarded for special purposes. For example the BPDU packets that sent during link start up must also be seen on ports in a blocking state. This allows the bridge software to remove loops in the fabric.

#### 12.2.2.4.6.1.9 Address Table Entry

The ALE table contains multiple table entry types. Each table entry represents a free entry, an address, a VLAN, an address/VLAN pair, or an OUI address. Software should ensure that there are not double address entries in the table. The double entry used would be indeterminate. Reserved table bits must be written with zeroes.

Source Address learning occurs for packets with a unicast, multicast or broadcast destination address and a unicast or multicast (including broadcast) source address. Multicast source addresses have the group bit (bit 40) cleared before ALE processing begins, changing the multicast source address to a unicast source address. A multicast address of all ones is the broadcast address which may be added to the table. A learned unicast source address is added to the table with the following control bits:

**Table 12-204. Learned Address Control Bits**

Bit(s)	Value
AGEABLE	1
Touch	1
BLOCK	0
SECURE	0

If a received packet has a source address that is equal to the destination address then the following occurs:

- The address is learned if the address is not found in the table.
- The address is updated if the address is found.
- The packet is dropped.

#### Table Entry Type

00 - Free Entry

01 - Address Entry : unicast or multicast determined by destination **address bit 40**.

10 - VLAN entry

11 - VLAN Address Entry : unicast or multicast determined by **address bit 40**.

#### 12.2.2.4.6.1.9.1 Free Table Entry

**Table 12-205. Free (Unused) Address Table Entry Bit Values**

74:62	61:60	59:0
RESERVED	ENTRY_TYPE (00)	RESERVED

#### 12.2.2.4.6.1.9.2 Multicast Address Table Entry (Bit 40 == 0)

**Table 12-206. Multicast Address Table Entry Bit Values**

74	73:70	69:66	65	64	63	62	61:60	59:48	47:0
TRUNK	RESERVED	PORT_NUMBER	BLOCK	SECURE	TOUCH	AGEABLE	ENTRY_T YPE (01)	RESERVE D	MULTICA ST_ADDR ESS

#### Trunk Indicator (TRUNK)

0h = The port bits in the entry are the port number

1h = The port bits in the entry are the trunk numbe

#### Port Number (PORT\_NUMBER)

This field indicates the port number (not port mask) that the packet with a unicast destination address may be forwarded to. Packets with unicast destination addresses are forwarded only to a single port (but not the receiving port).

### Block (BLOCK)

The block bit indicates that a packet with a matching source or destination address should be dropped (block the address).

0h = Address is not blocked.

1h = Drop a packet with a matching source or destination address (secure must be zero)

If block and secure are both set, then they no longer mean block and secure. When both are set, the block and secure bits indicate that the packet is a unicast supervisory (super) packet and they determine the unicast forward state test criteria. If both bits are set then the packet is forwarded if the receive port is in the Forwarding/Blocking/Learning state. If both bits are not set then the packet is forwarded if the receive port is in the Forwarding state.

### Secure (SECURE)

This bit indicates that a packet with a matching source address should be dropped if the received port number is not equal to the table entry port\_number.

0h = Received port number is a don't care.

1h = Drop the packet if the received port is not the secure port for the source address and do not update the address (block must be zero)

### Touch Indicator (TOUCH)

Only valid when AGEABLE it a 1h.

0h = Ageable unicast address has not been touched

1h = Ageable unicast address that has been touched

### Ageable (AGEABLE)

This bit indicates that the address is ageable.

0h = Unicast address that is not ageable

1h = Unicast address that is ageable

### Table Entry Type (ENTRY\_TYPE)

Address entry type. Unicast or multicast determined by address bit 40.

01: Address entry. Unicast or multicast determined by address bit 40.

### Packet Address (MULTICAST\_ADDRESS)

This is the 48-bit packet MAC address. For an OUI address, only the upper 24-bits of the address are used in the source or destination address lookup. Otherwise, all 48-bits are used in the lookup.

#### 12.2.2.4.6.1.9.3 Multicast Address Table Entry (Bit 40 == 1)

**Table 12-207. VLAN/Multicast Address Table Entry Bit Values**

74:66	65	64	63:62	61:60	59:48	47:0
PORT_MASK	SUPER	IGNMBITS	FWDSTLVL	ENTRY_TYP E (1h)	RESERVED	MULTICAST_AD DRESS

### Port Mask(1:0) (PORT\_MASK)

This 2-bit field is the port bit mask that is returned with a found multicast destination address. There may be multiple bits set indicating that the multicast packet may be forwarded to multiple ports (but not the receiving port).

### Supervisory Packet (SUPER)

When set, this field indicates that the packet with a matching multicast destination address is a supervisory packet.

0h = Non-supervisory packet

1h = Supervisory packet

### Ignore Multicast Bits (IGNMBITS)

Indication that the Multicast Address has ignored bits.

### Forward State Level (FWDSTLVL)

Indicates the port state(s) required for the received port on a destination address lookup in order for the multicast packet to be forwarded to the transmit port(s).

A transmit port must be in the Forwarding state in order to forward the packet. If the transmit port\_mask has multiple set bits then each forward decision is independent of the other transmit port(s) forward decision.

0h = Forwarding

1h = Blocking/Forwarding/Learning

2h = Forwarding/Learning

3h = Forwarding

The forward state test returns a true value if both the RX and TX ports are in the required state.

### Table Entry Type (ENTRY\_TYPE)

Address entry type. Unicast or multicast determined by address bit 40.

3h = VLAN address entry. Unicast or multicast determined by address bit 40.

### Packet Address (MULTICAST\_ADDRESS)

This is the 48-bit packet MAC address. For an OUI address, only the upper 24-bits of the address are used in the source or destination address lookup. Otherwise, all 48-bits are used in the lookup.

#### 12.2.2.4.6.1.9.4 VLAN Unicast Address Table Entry (Bit 40 == 0)

**Table 12-208. Unicast Address Table Entry Bit Values**

74	73:70	69:66	65	64	63	62	61:60	59:48	47:0
TRUNK	RESERVED	PORT_NUM BER	BLOCK	SECURE	TOUCH	AGEABLE	ENTRY_TY PE (11)	VLAN_ID	UNICAST_A DDRESS

### Trunk Indicator (TRUNK)

0h = The port bits in the entry are the port number

1h = The port bits in the entry are the trunk numbe

### Port Number (PORT\_NUMBER)

This field indicates the port number (not port mask) that the packet with a unicast destination address may be forwarded to. Packets with unicast destination addresses are forwarded only to a single port (but not the receiving port).

### Block (BLOCK)

The block bit indicates that a packet with a matching source or destination address should be dropped (block the address).

0h = Address is not blocked.

1h = Drop a packet with a matching source or destination address (secure must be zero)

If block and secure are both set, then they no longer mean block and secure. When both are set, the block and secure bits indicate that the packet is a unicast supervisory (super) packet and they determine the unicast forward state test criteria. If both bits are set then the packet is forwarded if the receive port is in the Forwarding/Blocking/Learning state. If both bits are not set then the packet is forwarded if the receive port is in the Forwarding state.

### Secure (SECURE)

This bit indicates that a packet with a matching source address should be dropped if the received port number is not equal to the table entry port\_number.

0h = Received port number is a don't care.

1h = Drop the packet if the received port is not the secure port for the source address and do not update the address (block must be zero)

### Touch Indicator (TOUCH)

Only valid when AGEABLE is a 1h.

0h = Ageable unicast address has not been touched

1h = Ageable unicast address that has been touched

### Ageable (AGEABLE)

This bit indicates that the address is ageable.

0h = Unicast address that is not ageable

1h = Unicast address that is ageable

### Table Entry Type (ENTRY\_TYPE)

Address entry. Unicast or multicast determined by address bit 40.

1h = Address entry. Unicast or multicast determined by address bit 40.

### Packet Address (UNICAST\_ADDRESS)

This is the 48-bit packet MAC address. All 48-bits are used in the lookup.

#### 12.2.2.4.6.1.9.5 OUI Unicast Address Table Entry

**Table 12-209. OUI Unicast Address Table Entry Bit Values**

74:64	63:62	61:60	59:48	47:24	23:0
RESERVED	UNICAST_TYPE (2h)	ENTRY_TYPE (1h)	RESERVED	UNICAST_OUI	RESERVED

### Unicast Type (UNICAST\_TYPE)

This field indicates the type of unicast address the table entry contains.

0h = Unicast address that is not ageable.

1h = Ageable unicast address that has not been touched.

2h = OUI address - lower 24-bits are don't cares (not ageable).

3h = Ageable unicast address that has been touched.

### Table Entry Type (ENTRY\_TYPE)

Address entry. Unicast or multicast determined by address bit 40.

1h: Address entry. Unicast or multicast determined by address bit 40.

### Packet Address (UNICAST\_OUI)

For an OUI address, only the upper 24-bits of the address are used in the source or destination address lookup.

#### 12.2.2.4.6.1.9.6 VLAN/Unicast Address Table Entry (Bit 40 == 0)

**Table 12-210. Unicast Address Table Entry Bit Values**

74	73:70	69:66	65	64	63	62	61:60	59:48	47:0
TRUNK	RESERVED	PORT_NUMBER	BLOCK	SECURE	TOUCH	AGEABLE	ENTRY_TYPE (3h)	VLAN_ID	UNICAST_ADDRESS

### Trunk Indicator (TRUNK)

0h = The port bits in the entry are the port number

1h = The port bits in the entry are the trunk number

### Port Number (PORT\_NUMBER)

This field indicates the port number (not port mask) that the packet with a unicast destination address may be forwarded to. Packets with unicast destination addresses are forwarded only to a single port (but not the receiving port).]

### Block (BLOCK)

The block bit indicates that a packet with a matching source or destination address should be dropped (block the address).

0h = Address is not blocked.

1h = Drop a packet with a matching source or destination address (secure must be zero)

If block and secure are both set, then they no longer mean block and secure. When both are set, the block and secure bits indicate that the packet is a unicast supervisory (super) packet and they determine the unicast forward state test criteria. If both bits are set then the packet is forwarded if the receive port is in the Forwarding/Blocking/Learning state. If both bits are not set then the packet is forwarded if the receive port is in the Forwarding state.

### Secure (SECURE)

This bit indicates that a packet with a matching source address should be dropped if the received port number is not equal to the table entry PORT\_NUMBER.

0h = Received port number is a don't care.

1h = Drop the packet if the received port is not the secure port for the source address and do not update the address (block must be zero)

### Touch Indicator (TOUCH)

Only valid when AGEABLE is a 1h.

0h = Ageable unicast address has not been touched

1h = Ageable unicast address that has been touched

### Ageable (AGEABLE)

This bit indicates that the address is ageable.

0h = Unicast address that is not ageable

1h = Unicast address that is ageable

#### Table Entry Type (ENTRY\_TYPE)

Address entry. Unicast or multicast determined by address bit 40.

3h: VLAN address entry. Unicast or multicast determined by address bit 40.

#### VLAN ID (VLAN\_ID)

The unique identifier for VLAN identification. This is the 12-bit VLAN ID.

#### Packet Address (UNICAST\_ADDRESS)

This is the 48-bit packet MAC address. All 48-bits are used in the lookup.

#### 12.2.2.4.6.1.9.7 VLAN/ Multicast Address Table Entry (Bit 40 == 1)

**Table 12-211. Multicast Address Table Entry Bit Values**

74:66	65	64	63:62	61:60	59:48	47:0
PORT_MASK	SUPER	IGNMBITS	FWDSTLVL	ENTRY_TYPE (3h)	VLAN_ID	UNICAST_ADD RESS

#### Port Mask(1:0) (PORT\_MASK)

This 2-bit field is the port bit mask that is returned with a found multicast destination address. There may be multiple bits set indicating that the multicast packet may be forwarded to multiple ports (but not the receiving port).

#### Supervisory Packet (SUPER)

When set, this field indicates that the packet with a matching multicast destination address is a supervisory packet.

0h = Non-supervisory packet

1h = Supervisory packet

#### Ignore Multicast Bits (IGNMBITS)

Indication that the Multicast Address has ignored bits.

#### Forward State Level (FWDSTLVL)

Indicates the port state(s) required for the received port on a destination address lookup in order for the multicast packet to be forwarded to the transmit port(s).

A transmit port must be in the Forwarding state in order to forward the packet. If the transmit port\_mask has multiple set bits then each forward decision is independent of the other transmit port(s) forward decision.

0h = Forwarding

1h = Blocking/Forwarding/Learning

2h = Forwarding/Learning

3h = Forwarding

The forward state test returns a true value if both the RX and TX ports are in the required state.

#### Table Entry Type (ENTRY\_TYPE)

Address entry type. Unicast or multicast determined by address bit 40.

3h = VLAN address entry. Unicast or multicast determined by address bit 40.

## VLAN ID (VLAN\_ID)

The VLAN ID associated with the Address entry.

## Packet Address (MULTICAST\_ADDRESS)

This is the 48-bit packet MAC address. For an OUI address, only the upper 24-bits of the address are used in the source or destination address lookup. Otherwise, all 48-bits are used in the lookup.

### 12.2.2.4.6.1.9.8 Inner VLAN Table Entry

**Table 12-212. Inner VLAN Table Entry**

74:66	65	64:62	61:60	59:48	47	46:45	44:36
NO_LEARN_MASK	VLAN_FORCE_INGRESS_CHECK	RESERVED	ENTRY_TYPE (1h)	VLAN_ID	NOFRAG	RESERVED	REG_MCAST_FLOOD_INDEX
35:33	32:24	23	22:21	20:12	11:9	8:0	
RESERVED	FORCE_UNTAGGED_EGRESS	LMTNXTHDR	RESERVED	UREGMSK	RESERVED	VLAN_MEMBER_LIST	

## No Learn Mask (NO\_LEARN\_MASK)

When a bit is set in this mask, a packet with an unknown source address received on the associated port will not be learned (i.e. When a VLAN packet is received and the source address is not in the table, the source address will not be added to the table).

## VLAN Force Ingress Check (VLAN\_FORCE\_INGRESS\_CHECK)

If the receive port is not a member of this VLAN then the packet is dropped. This is similar to the `ly_REG_Py_VID_INGRESS_CHECK` bit in the `CPSW_ly_ALE_PORTCTL0_y` registers except this check is for this VLAN only (not all VLANs).

## Table Entry Type (ENTRY\_TYPE)

2h: VLAN entry

## VLAN ID (VLAN\_ID)

The unique identifier for VLAN identification. This is the 12-bit VLAN ID.

## (NOFRAG)

VLAN No IPv4 Fragmented frames Control - Causes IPv4 fragmented IP frames to be dropped.

## Registered Multicast Flood Index (REG\_MCAST\_FLOOD\_INDEX)

Index into `CPSW_ALE_MSK_MUX0` to `CPSW_lx_ALE_MSK_MUXx` register array that is used to create the registered multicast flood mask.

## Force Untagged Packet Egress (FORCE\_UNTAGGED\_EGRESS)

This field causes the packet VLAN tag to be removed on egress for the specified port(s) (except on port 0).

## VLAN Limit Next Header Control (LMTNXTHDR)

This bit causes frames to be dropped if the Protocol/Nxt Header does not match the `CPSW_ALE_NXT_HDR` register values.

## VLAN Unregister Multicast Mask (UREGMSK)

This field indicates which port(s) are the unregistered multicast flood mask.

## VLAN Member List (VLAN\_MEMBER\_LIST)

This field indicates which port(s) are members of the associated VLAN. One bit per port.



### 12.2.2.4.6.1.9.9 Outer VLAN Table Entry

**Table 12-213. Outer VLAN Table Entry**

74:66	65	64:62	61:60	59:48	47	46:45	44:36
NO_LEARN_MASK	VLAN_FORCE_INGRESS_CHECK	RESERVED	ENTRY_TYPE (2h)	VLAN_ID	NOFRAG	RESERVED	REG_MCAST_FLOOD_INDEX
35:33	32:24	23	22:21	20:12	11:9	8:0	
RESERVED	FORCE_UNTAGGED_EGRESS	LMTNXTHDR	RESERVED	UREGMSK	RESERVED	VLAN_MEMBER_LIST	

#### No Learn Mask (NO\_LEARN\_MASK)

When a bit is set in this mask, a packet with an unknown source address received on the associated port will not be learned (i.e. When a VLAN packet is received and the source address is not in the table, the source address will not be added to the table).

#### VLAN Force Ingress Check (VLAN\_FORCE\_INGRESS\_CHECK)

If the receive port is not a member of this VLAN then the packet is dropped. This is similar to the `ly_REG_Py_VID_INGRESS_CHECK` bit in the `CPSW_ly_ALE_PORTCTL0_y` registers except this check is for this VLAN only (not all VLANs).

#### Table Entry Type (ENTRY\_TYPE)

2h: VLAN entry

#### VLAN ID (VLAN\_ID)

The unique identifier for VLAN identification. This is the 12-bit VLAN ID.

#### (NOFRAG)

VLAN No IPv4 Fragmented frames Control - Causes IPv4 fragmented IP frames to be dropped.

#### Registered Multicast Flood Index (REG\_MCAST\_FLOOD\_INDEX)

Index into `CPSW_ALE_MSK_MUX0` to `CPSW_lx_ALE_MSK_MUXx` register array that is used to create the registered multicast flood mask.

#### Force Untagged Packet Egress (FORCE\_UNTAGGED\_EGRESS)

This field causes the packet VLAN tag to be removed on egress for the specified port(s) (except on port 0).

#### VLAN Limit Next Header Control (LMTNXTHDR)

This bit causes frames to be dropped if the Protocol/Nxt Header does not match the `CPSW_ALE_NXT_HDR` register values.

#### VLAN Unregister Multicast Mask (UREGMSK)

This field indicates which port(s) are the unregistered multicast flood mask.

#### VLAN Member List (VLAN\_MEMBER\_LIST)

This field indicates which port(s) are members of the associated VLAN. One bit per port.

### 12.2.2.4.6.1.9.10 EtherType Table Entry

**Table 12-214. EtherType Table Entry**

74:65	64:62	61:60	59:16	15:0
RESERVED	RESERVED	ENTRY_TYPE (1h)	RESERVED	ETHERTYPE

#### Table Entry Type (ENTRY\_TYPE)

2h: VLAN entry

### Ether Type (ETHERTYPE)

16-bits Ether Type field.

#### 12.2.2.4.6.1.9.11 IPv4 Table Entry

**Table 12-215. IPv4 Table Entry**

74:70	69:65	64:62	61:60	59:32	31:0
RESERVED	IGNMBITS	RESERVED	ENTRY_TYPE (2h)	RESERVED	IPv4ADR

### Ignore Multicast Bits (IGNMBITS)

Indication that the Multicast Address has ignored bits.

### Table Entry Type (ENTRY\_TYPE)

2h: VLAN entry

### IPv4 Address (IPv4ADR)

32-bit IPv4 Address. Any ignored bits must be zero value in the table entry.

#### 12.2.2.4.6.1.9.12 IPv6 Table Entry High

**Table 12-216. IPv6 Table Entry High**

74:71	70:64	63	62	61:60	59:0
RESERVED	IGNMBITS	RESERVED	(1h)	ENTRY_TYPE (2h)	IPv6ADR[127:68]

### Ignore Multicast Bits (IGNMBITS)

Indication that the Multicast Address has ignored bits.

### Table Entry Type (ENTRY\_TYPE)

2h: VLAN entry

### IPv6 Address - upper 64 bits (IPv6ADR[127:68])

This address is split into three fields in the IPv6 High and IPv6 Low table entry. Any ignored bits must be zero in the table entry(s).

#### 12.2.2.4.6.1.9.13 IPv6 Table Entry Low

**Table 12-217. IPv6 Table Entry Low**

74:71	70:63	62	61:60	59:0
RESERVED	IPv6ADR[67:60]	RESERVED	ENTRY_TYPE (2h)	IPv6ADR[59:0]

### Table Entry Type (ENTRY\_TYPE)

2h: VLAN entry

### IPv6 Address - upper 8 bits (IPv6ADR[67:60])

### IPv6 Address - upper 60 bits (IPv6ADR[59:0])

This address is split into three fields in the IPv6 High and IPv6 Low table entry. Any ignored bits must be zero in the table entry(s).

Note: IPv6 table address entries operate differently than all other table entry types. IPv6 table entries have a high entry concatenated with a low entry. The high entry must have an entry\_pointer value of the low entry\_pointer plus 0x40. Bit six of the entry\_pointer must be set for the high entry and must be zero for the low entry.

#### 12.2.2.4.6.1.10 Multicast Address

Multicast addresses are addresses with bit 40 set. Only destination addresses can be Multicast addresses. The group bit (bit 40) of the source address is reserved in the IEEE standard.

A multicast address of all ones is the broadcast address which can be added to the lookup table if forwarding of broadcast packets need be modified.

##### 12.2.2.4.6.1.10.1 Multicast Ranges

Added IgnMbits to indicate at least one bit of the multicast address is ignored. Up to 10 bits of the multicast address can be ignored to provide the ability to create multiple multicast address ranges.

```
if ((IgnMbits)&(MultiCastAddress[0] == 0x000)) { MultiCastAddress[0] is ignored in compare}
if ((IgnMbits)&(MultiCastAddress[1:0] == 0x001)) { MultiCastAddress[1:0] is ignored in compare}
if ((IgnMbits)&(MultiCastAddress[2:0] == 0x003)) { MultiCastAddress[2:0] is ignored in compare}
if ((IgnMbits)&(MultiCastAddress[3:0] == 0x007)) { MultiCastAddress[3:0] is ignored in compare}
if ((IgnMbits)&(MultiCastAddress[4:0] == 0x00F)) { MultiCastAddress[4:0] is ignored in compare}
if ((IgnMbits)&(MultiCastAddress[5:0] == 0x01F)) { MultiCastAddress[5:0] is ignored in compare}
if ((IgnMbits)&(MultiCastAddress[6:0] == 0x03F)) { MultiCastAddress[6:0] is ignored in compare}
if ((IgnMbits)&(MultiCastAddress[7:0] == 0x07F)) { MultiCastAddress[7:0] is ignored in compare}
if ((IgnMbits)&(MultiCastAddress[8:0] == 0x0FF)) { MultiCastAddress[8:0] is ignored in compare}
if ((IgnMbits)&(MultiCastAddress[9:0] == 0x1FF)) { MultiCastAddress[9:0] is ignored in compare}
```

Below is 'C' code to modify ALE MultiCastAddress and IgnMbits when iNumOfBitsToIgnore is greater than zero. Where fGenMask(iOffset,iBitsToMask) creates a Mask for the value provided. For example fGenMask(0,5) will return 0x1F.

```
if(iNumOfBitsToIgnore)
{
    int iIgnClrMsk,iIgnSetMsk;
    iIgnClrMsk=fGenMask(0, iNumOfBitsToIgnore);
    iIgnSetMsk=fGenMask(0, iNumOfBitsToIgnore -1);
    MultiCastAddress &= ~iIgnClrMsk;
    MultiCastAddress |= iIgnSetMsk;
    IgnMbits = 1;
}
else
{
    IgnMbits = 0;
}
```

Multicast Addresses or Ranges can overlap, in the event of an overlap; the higher ALE index will be used.

##### 12.2.2.4.6.1.11 Supervisory Packets

Multicast supervisory packets are designated by the SUPER bit in the table entry. Unicast supervisory packets are indicated when BLOCK and SECURE are both set. Supervisory packets are not dropped due to rate limiting, OUI, or VLAN processing. The purpose of supervisory packets is to allow packets that would be otherwise

blocked to be forwarded for special purposes. For example the BPDU packets that sent during link start up must also be seen on ports in a blocking state. This allows the bridge software to remove loops in the fabric.

#### **12.2.2.4.6.1.12 Aging and Auto Aging**

The ALE supports software control or automatic aging of agable addresses.

Any time an agable address is seen as a source address entering from a port the source address entry will be marked as touched.

If the aging timer expires or the software sets the [29] AGE\_OUT\_NOW bit in the CPSW\_ALE\_CONTROL, the aging process will be started.

The aging process will read each ALE entry and for all entries that are an address with or without VLAN that is also agable, the touch check process will be done.

The touch check process will test the TOUCH bit and if clear, the entry will be marked as free, else the TOUCH bit will be cleared.

What this means is that if the aging interval was programmed as one second, any unused entry could stay in the ALE table for 1.000001 to 1.999999 seconds

#### **12.2.2.4.6.1.13 ALE Policing and Classification**

The ALE has a number of configurable classifier engines (policers) that can be used for classification. Classification is a subset of the policing function and uses a policer without the color marking or rate limiting functions. A policer is a hardware engine that is used for policing. The POLCNTDIV8 field in the CPSW\_ALE\_STATUS register indicates the number of policers available to be used for classification. Each policer can be enabled to match on one or more of any of the below packet fields for classification. All but Port and Priority are index references to the ALE table entries.

- Port Number
- Priority extracted from VLAN, mapped from DSCP if enabled, or Default Port Priority
- Organization Network Unique identifier - ONU
- Destination Address - DA
- Source Address - SA
- Outer VLANID -S-VLANID
- Inner VLANID -C-VLANID
- Ether Type
- IP Source Address - IPSA with full CIDR masking for IPv4 and IPv6
- IP Destination Address - IPSA with full CIDR masking for IPv4 and IPv6
- Support Host Thread/Flow ID mapping based on any packet classification above

##### **12.2.2.4.6.1.13.1 ALE Policing**

The policing function on each policer engine is implemented as dual-counter three-color marking engine as described in the IETF RFC2698. The first counter is the Committed Information Rate (CIR) counter and the second counter is the Peak Information Rate (PIR) counter. The policing function can use either or both counters. Based on the counter values the packet color is determined. The color is used to determine whether the packet is dropped or forwarded. The ALE has a local feature that can drop packets regardless of queue state.

The policing rates are determined by the below equations:

CIR policing rate in Mbit/s = ((ALE frequency in Mhz) \* CPSW\_ALE\_POLICECFG7[31-0] CIR\_IDLE\_INC\_VAL) / 32768

PIR policing rate in Mbit/s = ((ALE frequency in Mhz) \* CPSW\_ALE\_POLICECFG6[31-0] PIR\_IDLE\_INC\_VAL) / 32768

Each policer has 10 different match operations (see [Section 12.2.2.4.6.1.13](#)). Since multiple policing entries can be hit on a single packet this provides the ability to create precise traffic stream control.

Packets are colored at ALE lookup time. Packets can be colored RED, YELLOW, or GREEN. If multiple policers are configured for a packet stream then the packet color is merged from all matching (hit) policers. If any policer is RED then the packet is marked RED. Else if any policer is YELLOW then the packet is marked YELLOW. Otherwise the packet is marked GREEN.

The Policing engine supports several modes such that packets that don't hit a policing/classifier match can be treated as RED, YELLOW, GREEN or policer 0 color. Using policer 0 allows for a system to regulate unregulated traffic.

#### **12.2.2.4.6.1.13.2 Classifier to Host Thread Mapping**

The ALE module allows Host Thread mapping based on any packet classification. That is the ALE can generate a thread ID used by the host based on ALE classifier matches.

When enabled the highest classifier match can map to a particular thread ID value.

The ALE also supports an optional default Thread ID value in the event that no classifier match.

Each Thread ID including the default thread ID has an enable such that if no matches occur with the enable bit set are seen the default is used, if the default is not enabled, the switch will use the {port,priority} value instead. If multiple classifier matches occur, the highest matching entry with a thread enable bit set will be used.

Three registers are used for ALE classification thread mapping configuration (CPSW\_ALE\_THREADMAPDEF, CPSW\_ALE\_THREADMAPCTL and CPSW\_ALE\_THREADMAPVAL). The three thread mapping registers are used independently and are separate from the other ALE policing registers. The CPSW\_ALE\_THREADMAPCTL register allows the CPSW\_ALE\_THREADMAPVAL register contents to be written to the selected classifier. There is a CPSW\_ALE\_THREADMAPDEF register that is used for all classifiers. The thread mapping registers can be written or changed at any time but any packets that are already processed will not have their thread altered.

#### **12.2.2.4.6.1.13.3 ALE Classification**

When the policers are configured as classifiers, the color marking and policing functions of the policing/classifier engines are not used. One or multiple classifiers can be configured to match on a single packet. For example, a classifier can be enabled to match on priority while another classifier could match IP address.

##### **12.2.2.4.6.1.13.3.1 Classifier to CPPI Transmit Flow ID Mapping**

The ALE can generate a 6-bit transmit CPPI Flow ID based on classifier matches that can be used instead of the switch default transmit Flow ID mapping. The switch default flow ID is the remapped received packet priority (0 to 7). Thread and flow ID are used interchangeably for this since there is a single hardware thread (TXST\_THREAD\_MREADY) but there are 6-bits of FLOW\_ID in the transmit CPPI INFO word 0. When enabled, the highest classifier match can map to a particular 6-bit flow ID value that is associated with the classifier. The ALE also supports an optional ALE default thread/flow ID value in the event that no classifiers match. Each thread/flow ID, including the ALE default thread/flow ID, has an enable such that the ALE default thread/Flow ID is used if enabled and if no matches occur (instead of the remapped received packet priority). If the ALE default is not enabled and no matches occur then the switch default value will be used. If multiple classifier matches occur, the highest match with a thread enable bit set will be used. The resultant flow ID has the CPSW\_P0\_FLOW\_ID\_OFFSET\_REG register value added to it to determine the actual value in the INFO 0 Flow ID field.

Three registers are used for ALE classification thread/flow ID mapping configuration (CPSW\_ALE\_THREADMAPDEF, CPSW\_ALE\_THREADMAPCTL and CPSW\_ALE\_THREADMAPVAL). The three thread mapping registers are used independently and are separate from the other ALE policing registers. The CPSW\_ALE\_THREADMAPCTL register allows the CPSW\_ALE\_THREADMAPVAL register contents to be written to the selected classifier. There is a single CPSW\_ALE\_THREADMAPDEF that is used for all classifiers. The thread mapping registers can be written or changed at any time but any packets that are already processed will not have their thread altered.

#### 12.2.2.4.6.1.14 Mirroring

The ALE supports three mirroring modes: destination port, source port and or table entry.

**Destination port mirroring** allows packets from any ingress port or trunk which ends up switching to a particular egress destination port or trunk to be mirrored to yet another egress destination port or trunk. For example any traffic from any port that is switched to port 'A' can be also mirrored to port 'B'. (MIRROR\_DP=A, MIRROR\_DEN=1h, MIRROR\_TOP=B in the CPSW\_ALE\_CONTROL register).

**Source port mirroring** allows packets received on any enabled ingress source port or trunk to be switched to the mirror egress port as well as the actual egress destination ports. For example traffic received on ingress port 'A' can be switched to egress port 'B' as well as the intended egress destination port. (ly\_REG\_Py\_MIRROR\_SP=1h in the CPSW\_ly\_ALE\_PORTCTL0\_y register, MIRROR\_SEN=1h, MIRROR\_TOP=B in the CPSW\_ALE\_CONTROL register).

**Table entry mirroring** allows for any MAC Address, MAC Address with VLAN, ONU Address or VLAN entry that matches on ingress to be switched to the egress destination as well as the actual egress destination. For example all traffic for VLAN ID of 35 can be mirrored to port 'B'. That is any traffic switched on VLAN ID of 35 will be mirrored. ({VLAN ID of 35 in ALE Table entry index=C}, MIRROR\_MIDX=C, MIRROR\_MEN=1h, MIRROR\_TOP=B)

In the event that mirrored packets are mirrored to or from a port that is also the mirror port the packet will not be duplicated or marked as a mirror packet since the packet has already been on the port as ingress or egress. The packet sent to the mirror port may have modified VLAN info based on the port and VLAN lookup table entries. The mirror port need not be a member of the VLAN ID it is mirroring, the ALE will forward traffic to the mirror port after ingress and egress filters are applied.

The switch may decide to drop any mirror traffic based on switch buffer thresholds as to prevent required traffic from becoming congested.

Port mirroring is controlled by register fields in CPSW\_ALE\_CONTROL, CPSW\_ALE\_CTRL2 and the port control registers.

- MIRROR\_DP - The destination port that will have its traffic mirrored (CPSW\_ALE\_CONTROL register).
- MIRROR\_TOP - The port to which mirrored traffic is sent (CPSW\_ALE\_CONTROL register).
- MIRROR\_MEN - The enable for mirroring traffic that matches a supported lookup table entry (CPSW\_ALE\_CONTROL register).
- MIRROR\_DEN - The Enable for destination port mirroring (CPSW\_ALE\_CONTROL register).
- MIRROR\_SEN - The Enable for source port mirroring (CPSW\_ALE\_CONTROL register).
- MIRROR\_MIDX - The index of a lookup table entry that will be mirrored CPSW\_ALE\_CTRL2 register).
- ly\_REG\_Py\_MIRROR\_SP - The enable for the Source port to be mirrored. Although multiple source ports can be mirrored concurrently, a mirror traffic bandwidth issue may occur on the mirror egress port (CPSW\_ly\_ALE\_PORTCTL0\_y register).

#### 12.2.2.4.6.1.15 Trunking

The ALE supports port trunking of any port in any of four trunk groups. That is, four trunk groups can be supported with up to eight ports in each trunk group. There are no port adjacency rules for trunk groups. When ports are a member of a trunk group, addresses added and used in the lookup table will refer to the trunk group rather than port as indicated in the lookup table entries. If ports are removed from a trunk group, the ALE will redistribute the traffic based on the crc polynomial of enabled fields and the remaining ports within the trunk group. A trunk group may contain only one port. Packet priority, DA, SA, C-VLAN ID, IPv4SA, IPv4DA, IPv6SA, and/or IPv6DA can be used in the hash to generate destination port within the trunk group. If all hash enables are disabled, the packet can be directed to a particular port within the trunk group which allows for testing paths etc. A host directed frame is directed to the directed port regardless of trunk group settings.

Trunking is controlled through fields in the CPSW\_ALE\_CTRL2 register and in each ALE CPSW\_ly\_ALE\_PORTCTL0\_y register:

- TRK\_EN\_DST - Enable destination address hashing for trunk port calculation.



- TRK\_EN\_SRC - Enable source address hashing for trunk port calculation.
- TRK\_EN\_PRI - Enable priority hashing for trunk port calculation.
- TRK\_EN\_IVLAN - Enable inner C-VLAN ID hashing for trunk port calculation.
- TRK\_EN\_SIP - Enable source IP address hashing for trunk port calculation.
- TRK\_EN\_DIP - Enable destination IP address hashing for trunk port calculation.
- TRK\_BASE - Hashing formula starting value and test port offset.
- ly\_REG\_Py\_TRUNKEN - Enable this port as a trunk group
- ly\_REG\_Py\_TRUNKNUM - Trunk group number defines this port as a member of a particular trunk group.

#### 12.2.2.4.6.1.16 DSCP

The ALE can map DSCP field to priority prior to classification matching. When enabled the DSCP is mapped via 64 priority entries such that any DSCP value can be mapped to any of the eight priorities. When a packet is received without a VLAN priority this remapped priority can be used instead of the default Port VLAN priority field. See CPSW\_P0\_RX\_DSCP\_MAP\_REG\_y and CPSW\_PN\_RX\_DSCP\_MAP\_REG\_k\_y registers in the Register Manual section for DSCP mapping.

#### 12.2.2.4.6.1.17 Packet Forwarding Processes

There are four processes that an incoming received packet may go through to determine packet forwarding. The processes are *Ingress Filtering*, *VLAN\_Aware Lookup*, and *Egress*.

Packet processing begins in the Ingress Filtering process. Each port has an associated packet forwarding state that can be one of four values (Disabled, Blocked, Learning, or Forwarding). The default state for all ports is Disabled. The host sets the packet forwarding state for each port.

In the packet ingress process (receive packet process), there is a forward state test for unicast destination addresses and a forward state test for multicast addresses. The multicast forward state test indicates the port states required for the receiving port in order for the multicast packet to be forwarded to the transmit port(s). A transmit port must be in the Forwarding state for the packet to be forwarded for transmission. The MCAST\_FWD\_STATE indicates the required port state for the receiving port as indicated in the preceding table. The unicast forward state test indicates the port state required for the receiving port in order to forward the unicast packet. The transmit port must be in the Forwarding state in order to forward the packet. The BLOCK and SECURE bits determine the unicast forward state test criteria. If both bits are set then the packet is forwarded if the receive port is in the Forwarding/Blocking/Learning state. If both bits are not set then the packet is forwarded if the receive port is in the Forwarding state. The transmit port must be in the Forwarding state regardless. The forward state test used in the ingress process is determined by the destination address packet type (multicast/unicast).

In general, packets received with errors are dropped by the address lookup engine without learning, updating, or touching the address. The error condition and the abort are indicated by the Ethernet port to the ALE. Packets with errors may be passed to the host (not aborted) by a Ethernet port port, if the port has the RX\_CMF\_EN, RX\_CEF\_EN, or RX\_CSF\_EN bit(s) set in the CPSW\_PN\_MAC\_CONTROL\_REG\_k register. Error packets that are passed to the host by the Ethernet port are considered to be bypass packets by the ALE and are sent only to the host. Error packets do not learn, update, or touch addresses regardless of whether they are aborted or sent to the host. Packets with long or short errors received by the host are dropped. Packets with errors received by the host are forwarded as normal.

The following control bits are in the CPSW\_PN\_MAC\_CONTROL\_REG\_k register:

- [22] RX\_CEF\_EN - enables frames that are fragments, long, jabber, CRC, code, and alignment errors to be forwarded
- [23] RX\_CSF\_EN - enables short frames to be forwarded
- [24] RX\_CMF\_EN - enables MAC control frames to be forwarded.

#### 12.2.2.4.6.1.17.1 Ingress Filtering Process

##### Condition and action

<p>If ((ALE BYPASS) and (host port is not the receive port)) then use host portmask and go to Egress process</p>
<p>if (directed packet) then use directed port number and go to Egress process</p>
<p>If (Rx ly_REG_Py_PORTSTATE is Disabled) then discard the packet</p>
<p>if ((ALE BYPASS or error packet) and (host port is not the receive port)) then use host portmask and go to Egress process</p>
<p>if (((BLOCK) and (unicast source address found)) or ((BLOCK) and (unicast destination address found))) then discard the packet</p>
<p>if ((ENABLE_RATE_LIMIT) and (rate limit exceeded) and (not BCAST_MCAST_CTL)) then if (((Multicast/Broadcast destination address found) and (not SUPER)) or (Multicast/Broadcast destination address not found)) then discard the packet</p>
<p>if ((not forward state test valid) and (destination address found)) then discard the packet to any port not meeting the requirements</p> <ul style="list-style-type: none"> <li>Unicast destination addresses use the unicast forward state test and multicast destination addresses use the multicast forward state test.</li> </ul>
<p>if ((destination address not found) and ((not transmit port forwarding) or (not receive port forwarding))) then discard the packet to any ports not meeting the above requirements</p>
<p>if (source address found) and (secure) and (not block) and (receive port number != port_number)) then discard the packet</p>
<p>if ((not super) and (drop_untagged) and ((non-tagged packet) or ((priority tagged) and not(en_vid0_mode))) then discard the packet</p>
<p>If (VLAN_Unaware)</p> <p>CPSW_ALE_UVLAN_UNTAG = "000000"</p> <p>CPSW_ALE_UVLAN_URCAST = "111111"</p> <p>CPSW_ALE_UVLAN_URCAST = "111111"</p> <p>UVLAN_MEMBER_LIST = "111111"</p> <p>else if (VLAN not found)</p> <p>CPSW_ALE_UVLAN_UNTAG = CPSW_ALE_UVLAN_UNTAG</p> <p>CPSW_ALE_UVLAN_RMCAST = CPSW_ALE_UVLAN_RMCAST</p> <p>CPSW_ALE_UVLAN_RMCAST = CPSW_ALE_UVLAN_RMCAST</p> <p>CPSW_ALE_UVLAN_MEMBER = CPSW_ALE_UVLAN_MEMBER</p> <p>else</p> <p>CPSW_ALE_UVLAN_UNTAG = found CPSW_ALE_UVLAN_UNTAG</p> <p>CPSW_ALE_UVLAN_URCAST = found CPSW_ALE_UVLAN_URCAST</p> <p>CPSW_ALE_UVLAN_RMCAST = found CPSW_ALE_UVLAN_RMCAST</p> <p>UVLAN_MEMBER_LIST = found UVLAN_MEMBER_LIST</p>
<p>if ((not SUPER) and (ly_REG_Py_VID_INGRESS_CHECK) and (Rx port is not VLAN member)) then discard the packet</p>
<p>if ((ENABLE_AUTH_MODE) and (source address not found) and not(destination address found and (SUPER))) then discard the packet</p>



if (destination address equals source address) then discard the packet
if (VLAN_AWARE) goto VLAN_Aware_Lookup process else goto VLAN_Unaware_Lookup process

#### 12.2.2.4.6.1.17.2 VLAN\_Aware Lookup Process

Condition and action
if ((unicast packet) and (destination address found with or without VLAN) and (not SUPER)) then portmask is the logical "AND" of the PORT_NUMBER and UVLAN_MEMBER_LIST less the host port and goto Egress process
if ((unicast packet) and (destination address found with or without VLAN) and (not SUPER)) then portmask is the logical "AND" of the PORT_NUMBER and the UVLAN_MEMBER_LIST and goto Egress process
if ((unicast packet) and (destination address found with or without VLAN) and (SUPER)) then portmask is the PORT_NUMBER and goto Egress process
if (Unicast packet) # destination address not found then portmask is VLAN member LIST less host port and goto Egress process
if ((Multicast packet) and (destination address found with or without VLAN) and (not SUPER)) then portmask is the logical "AND" of CPSW_ALE_UVLAN_URCAST and found destination address/VLAN portmask (PORT_MASK) and UVLAN_MEMBER_LIST and goto Egress process
if ((Multicast packet) and (destination address found with or without VLAN) and (SUPER)) then portmask is the PORT_MASK and goto Egress process
if (Multicast packet) # destination address not found then portmask is the logical "AND" of CPSW_ALE_UVLAN_URCAST and UVLAN_MEMBER_LIST then goto Egress process
if (Broadcast packet) then use found UVLAN_MEMBER_LIST and goto Egress process

#### Note

The UVLAN\_MEMBER\_LIST, UVLAN\_UNREG\_MCAST\_FLOOD\_MASK, UVLAN\_REG\_MCAST\_FLOOD\_MASK and UVLAN\_FORCE\_UNTAGGED\_EGRESS are set in the [Section 12.2.2.4.6.1.17.1 Ingress Filtering Process](#), based on VLAN\_Unaware, Unknown\_VLAN rules and VLAN table entries.

#### 12.2.2.4.6.1.17.3 Egress Process

Condition and action
Clear Rx port from portmask (don't send packet to Rx port).
Clear disabled ports from portmask.
if ((ENABLE_OUI_DENY) and (OUI source address not found) and (not ALE BYPASS) and (not error packet) and ((not mcast destination address) and (SUPER))) then Clear host port from portmask

if ((not ENABLE_OUI_DENY) and (OUI source address found) and (not ALE BYPASS) and (not error packet) and not ((mcast destination address) and (SUPER)))
then Clear host port from portmask
if ((ENABLE_RATE_LIMIT) and (BCAST_MCAST_CTL))
then if (not SUPER) and (rate limit exceeded on any tx port)
then clear rate limited tx port from portmask
If address not found then SUPER cannot be set.
If portmask is zero then discard packet
Send packet to portmask ports.

#### 12.2.2.4.6.1.17.4 Learning/Updating/Touching Processes

The learning, updating, and touching processes are applied to each receive packet that is not aborted. The processes are concurrent with the packet forwarding process. In addition to the following, a packet must be received without error in order to learn/update/touch an address.

##### 12.2.2.4.6.1.17.4.1 Learning Process

The learning process is applied to each receive packet that is not aborted. The learning process is a concurrent process with the packet forwarding process.

Condition and action
If (directed) then do not learn, update, or set touched else continue
If (not (Learning or Forwarding) or (ENABLE_AUTH_MODE) or (packet error) or (ly_REG_Py_NO_LEARN)) then do not learn address
if ((Non-tagged packet) and (ly_REG_Py_DROP_UN_TAGGED)) then do not learn address
if ((VLAN_AWARE) and (VLAN not found) and (unknown UVLAN_MEMBER_LIST = "000")) then do not learn address
if ((ly_REG_Py_VID_INGRESS_CHECK) and (Rx port is not VLAN member) and (VLAN found)) then do not learn address
if ((source address found) and (receive port_number != PORT_NUMBER) and (SECURE or BLOCK)) then do not update address else continue
if ((source address found) and (receive port number != PORT_NUMBER)) then update address else continue
if ((source address not found) and (VLAN_AWARE) and not (LEARN_NO_VLANID)) then learn address with VLAN
if ((source address not found) and ((not VLAN_AWARE) or (VLAN_AWARE and LEARN_NO_VLANID))) then learn address without VLAN

##### 12.2.2.4.6.1.17.4.2 Updating Process

Condition and action
if (dlr_unicast) then do not update address
If (not(Learning or Forwarding) or (ENABLE_AUTH_MODE) or (packet error) or (ly_REG_Py_NO_SA_UPDATE)) then do not update address
if ((Non-tagged packet) and (ly_REG_Py_DROP_UN_TAGGED)) then do not update address

if ((VLAN_AWARE) and (VLAN not found) and (unknown UVLAN_MEMBER_LIST = "000")) then do not update address
if ((ly_REG_Py_VID_INGRESS_CHECK) and (Rx port is not VLAN member) and (VLAN found)) then do not update address
if ((source address found) and (receive port number != PORT_NUMBER) and (SECURE or BLOCK)) then do not update address
if ((source address found) and (receive port number != PORT_NUMBER)) then update address

#### 12.2.2.4.6.1.17.4.3 Touching Process

if ((source address found) and (ageable) and (not touched)) then set touched
---

#### 12.2.2.4.6.1.18 VLAN Aware Mode

The CPSW is in VLAN aware mode when the VLAN\_AWARE bit is set in the CPSW\_CONTROL\_REG register.

In VLAN aware mode, transmitted packet data is changed depending on the packet type (PKT\_TYPE), packet priority (PKT\_PRI), and VLAN information.

The VLAN\_LTYPE\_SEL value is selected by the S\_CN\_SWITCH bit in the CPSW\_CONTROL\_REG register and is either the VLAN\_LTYPE\_INNER (8100h default) or VLAN\_LTYPE\_OUTER (88A8h default) value.

#### 12.2.2.4.6.1.19 VLAN Unaware Mode

An egress port is operating in the VLAN unaware mode when the VLAN\_AWARE bit in the CPSW\_CONTROL\_REG register is cleared to 0h. In VLAN unaware mode, transmit (egress) packets are not modified on egress.

#### 12.2.2.4.6.2 Packet Priority Handling

Packets are received on 9 ports, eight Ethernet ports and one CPPI host port. Received packets have a received packet priority (0 to 7, with 7 being the highest priority).

The received packet priority is determined as follows:

1. If the first packet LTYPE = VLAN\_LTYPE\_SEL then the received packet priority is the packet priority (VLAN tagged and priority tagged packets).
2. Else if the first packet LTYPE = 0x0800 and byte 14 (following the LTYPE) is equal to 0x4X, and DSCP\_IPV4\_EN is set in CPSW\_P0\_CONTROL\_REG or CPSW\_PN\_CONTROL\_REG\_k, then the received packet priority is the 6-bit TOS field in byte 15 (upper 6 bits) mapped through the port's DSCP priority mapping registers (IPv4 packet).
3. Else if the first packet LTYPE = 0x86DD and the most significant nibble of byte 14 (following the LTYPE) is equal to 0x6, and DSCP\_IPV6\_EN is set in CPSW\_P0\_CONTROL\_REG or CPSW\_PN\_CONTROL\_REG\_k, then the received packet priority is the 6-bit priority (in the 6-bits following the upper nibble 0x6) mapped through the port's DSCP priority mapping registers (IPv6 packet).
4. Else the received packet priority is the source (ingress) port priority.

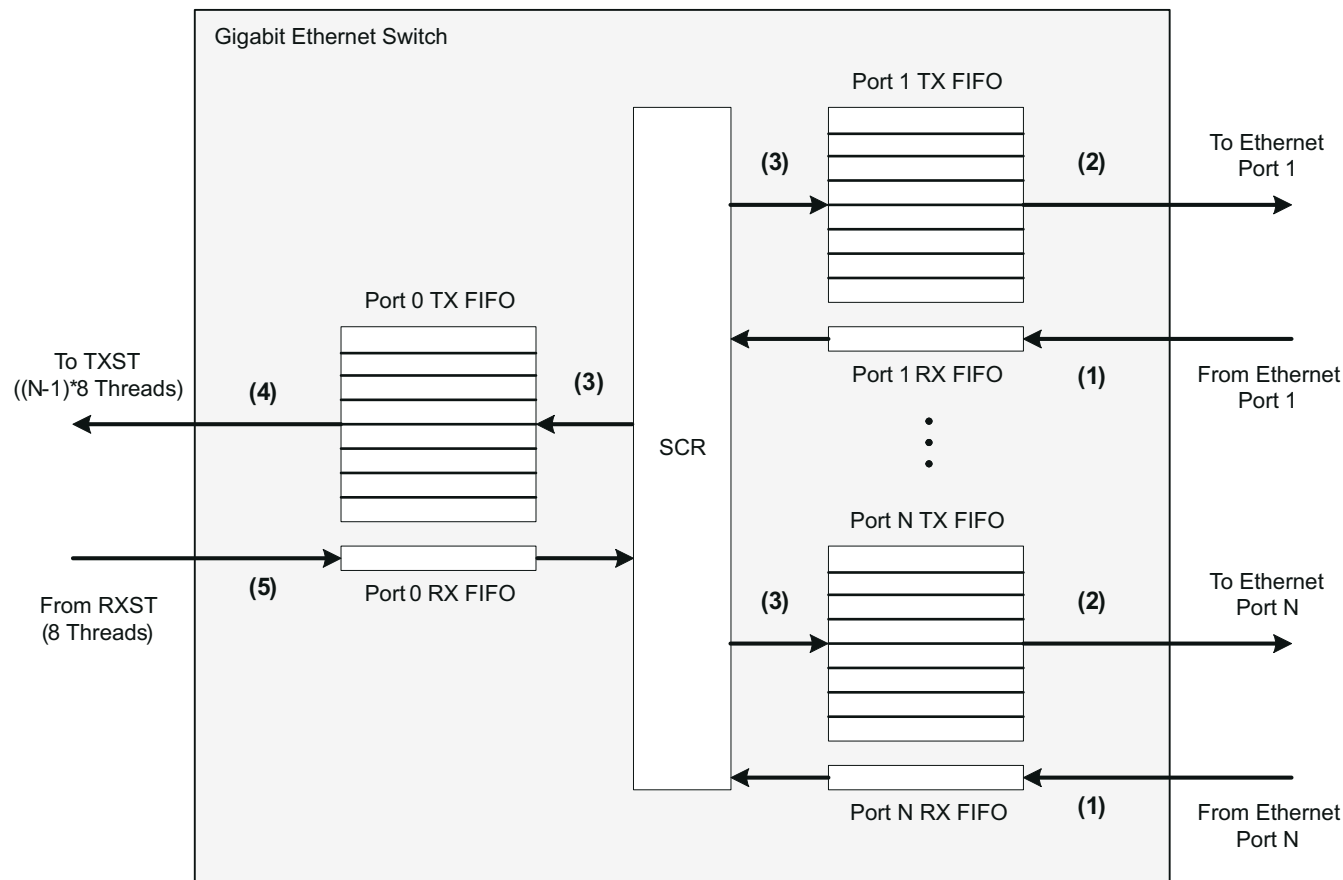
The received packet priority is mapped through the receive ports associated packet-priority-to-header-packet-priority-mapping register (CPSW\_PN\_RX\_PRI\_MAP\_REG\_k) to obtain the header packet priority. The header packet priority is the hardware switch priority. The header packet priority is also used as the actual transmit packet priority if the VLAN information is to be sent on egress.

The header packet priority is mapped at each destination FIFO through the CPSW\_PN\_RX\_PRI\_MAP\_REG\_k register (header priority to switch priority mapping register) to obtain the hardware switch priority (hardware queue 0 through 7).

### 12.2.2.4.6.2.1 Priority Mapping and Transmit VLAN Priority

There are three priorities that are used inside the Gigabit Ethernet Switch: **packet priority**, **header packet priority**, and **switch priority**. The **header packet priority** is used as the outgoing VLAN priority if the packet is egressing from the switch with a VLAN tag. The **switch priority** determines which of the 8 FIFO priority queues the packet uses during egress.

Figure 12-166 below, as well as the corresponding explanation that follows, explains each of the priorities, how they are determined, and how they are used. A number in parentheses in the figure indicates a process (Ethernet port ingress, host port egress, etc.). Each bullet in the text following the diagram explains one of the 5 processes pointed out in the figure.



**Figure 12-166. Gigabit Ethernet Switch Priority Mapping and Transmit VLAN Processing**

From [Figure 12-166](#) above:

- (1) is the ingress process that occurs at the external Ethernet ports
  - The incoming packet is assigned a **packet priority** based on either its VLAN priority, IPv4 or IPv6 DSCP value, or the ingress port's priority. This **packet priority** is then mapped to a **header packet priority** using the CPSW\_PN\_RX\_PRI\_MAP\_REG\_k register where N is the port where the packet entered the switch. This process is explained in further detail in [Section 12.2.2.4.6.2](#).
- (2) is the egress process that occurs at the external Ethernet ports
  - If the switch is in VLAN Aware mode then the VLAN header may be added, replaced, or removed during the egress process. If the VLAN header is to be added or replaced, the VLAN priority will come from the **header packet priority** that was determined in process (1) or (5). Transmit VLAN processing is the same for both the host port and the external Ethernet ports and is described in [Section 12.2.2.4.6.4.1](#).
- (3) is the process by which it is decided which priority TX queue to place the packet on in the Port N TX FIFO during egress
  - Each Port's TX FIFO has 8 queues that each correspond to a priority that is used when determining which packet will egress from the switch next at that port. The **header packet priority** (Ethernet port ingress, process (1)) or the receive packet thread (host port 0 ingress, process (5)) gets mapped through the CPSW\_PN\_RX\_PRI\_MAP\_REG\_k register (where N is the egress port number) to determine the **switch priority** of the packet. The **switch priority** determines which TX FIFO queue to place the packet in. The FIFO architecture is described in [Section 12.2.2.4.6.10.11](#). The header packet priority to switch priority mapping is discussed in [Section 12.2.2.4.6.2](#).
- (4) is the egress process that occurs at Host Port 0 toward the transmit streaming interface (TXST)
  - The TXST has 8 egress threads for each external Ethernet port (32 threads in a 5-port switch and 64 threads in a 9-port switch). The TX thread that is selected for a packet is determined by the Ethernet port of ingress and the **switch priority** of the packet that was determined in process (3).
  - If the switch is in VLAN Aware mode then the VLAN header may be added, replaced, or removed during the egress process. If the VLAN header is to be added or replaced, the VLAN priority will come from the **header packet priority** that was determined in process (1). Transmit VLAN processing is the same for both the host port and the external Ethernet ports and is described in [Section 12.2.2.4.6.4.1](#).
- (5) is the ingress process that occurs at Host Port 0
  - The incoming packet is assigned a **packet priority** based on either its VLAN priority, IPv4 or IPv6 DSCP value, or the host port's priority. This packet priority is then mapped to a **header packet priority** using the CPSW\_PN\_RX\_PRI\_MAP\_REG\_k register.
  - Host port 0 also has a received packet thread. The receive packet thread is based on either the packet's VLAN priority, IPv4 or IPv6 DSCP value, or the streaming interface (RXST) thread that the packet entered host port 0 on.

### 12.2.2.4.6.3 CPPI Port Ingress

Packets received on the CPPI host port have a received packet priority (0 to 7 with 7 being the highest priority).

The received packet priority is determined as follows:

1. If the first packet LTYPE = VLAN\_LTYPE\_SEL then the received packet priority is the packet priority (VLAN tagged and priority tagged packets).
2. Else if the first packet LTYPE = 0x0800 and byte 14 (following the LTYPE) is equal to 0x4X, and DSCP\_IPV4\_EN is set in CPSW\_P0\_CONTROL\_REG or CPSW\_PN\_CONTROL\_REG\_k register, then the received packet priority is the 6-bit TOS field in byte 15 (upper 6 bits) mapped through the port's DSCP priority mapping registers (IPv4 packet).
3. Else if the first packet LTYPE = 0x86DD and the most significant nibble of byte 14 (following the LTYPE) is equal to 0x6, and DSCP\_IPV6\_EN is set in CPSW\_P0\_CONTROL\_REG or CPSW\_PN\_CONTROL\_REG\_k register, then the received packet priority is the 6-bit priority (in the 6-bits following the upper nibble 0x6) mapped through the port's DSCP priority mapping registers (IPv6 packet).
4. Else the received packet priority is the source (ingress) port priority

The CPPI Port (port 0) also has a received packet thread. The received packet thread is determined exactly as the received packet priority except for untagged packets. For untagged packets, the received packet thread is the packet streaming interface thread that the packet was received on (instead of the port VLAN priority or DSCP priority). The received packet thread is the hardware switch priority. The received packet thread only determines which hardware switch priority the packet should be sent to, the egress VLAN rules are identical to packets that were received on Ethernet ports (as determined by the header packet priority).

For CPPI ingress packets, the destination port hardware switch priority is the below selected value remapped through CPSW\_PN\_RX\_PRI\_MAP\_REG\_k:

1. If the ingress packet is priority tagged or vlan tagged:
  - If RX\_REMAP\_VLAN in CPSW\_P0\_CONTROL\_REG register is clear then the destination hardware switch priority is the CPPI receive thread number.
  - If RX\_REMAP\_VLAN in CPSW\_P0\_CONTROL\_REG register is set then the destination hardware switch priority is the packet priority value. Port transmit remapping (CPSW\_PN\_RX\_PRI\_MAP\_REG\_k should remain the default value) is not compatible with this bit being set, but remapping can be configured on port 0 receive.
2. Else if the ingress packet has the first packet LTYPE = 0x0800 and byte 14 (following the LTYPE) is equal to 0x4X, and DSCP\_IPV4\_EN is set in CPSW\_P0\_CONTROL\_REG register:
  - If RX\_REMAP\_DSCP\_V4 bit in CPSW\_P0\_CONTROL\_REG register is clear then the destination hardware switch priority is the CPPI receive thread number.
  - If RX\_REMAP\_DSCP\_V4 bit in CPSW\_P0\_CONTROL\_REG register is set then the destination hardware switch priority is the 6-bit TOS field in byte 15 (upper 6-bits) mapped through the port's DSCP priority mapping registers (IPv4 packet). Port 1 transmit remapping (CPSW\_PN\_RX\_PRI\_MAP\_REG\_k should remain the default value) is not compatible with this bit being set, but remapping can be configured on port 0 receive.
3. Else if the ingress packet has the first packet LTYPE = 0x86DD and the most significant nibble of byte 14 (following the LTYPE) is equal to 0x6, and DSCP\_IPV6\_EN is set in CPSW\_P0\_CONTROL\_REG register:
  - If RX\_REMAP\_DSCP\_V6 bit in CPSW\_P0\_CONTROL\_REG register is clear then the destination hardware switch priority is the CPPI receive thread number.
  - If RX\_REMAP\_DSCP\_V6 bit in CPSW\_P0\_CONTROL\_REG register is set then the destination hardware switch priority is the 6-bit priority (in the 6-bits following the upper nibble 0x6) mapped through the port's DSCP priority mapping registers (IPv6 packet). Port 1 transmit remapping (CPSW\_PN\_RX\_PRI\_MAP\_REG\_k should remain the default value) is not compatible with this bit being set, but remapping can be configured on port 0 receive.
4. Else the ingress packet is non-tagged and the destination hardware switch priority is the CPPI receive thread number.

#### 12.2.2.4.6.4 Packet CRC Handling

The P0\_TX\_CRC\_REMOVE bit in the CPSW\_CONTROL\_REG register determines if host port egress packets have CRC included or not. If P0\_TX\_CRC\_REMOVE is set to 1h then all packets that are transmitted from port 0 do not contain CRC. If P0\_TX\_CRC\_REMOVE bit is cleared to 0h then all packets that are transmitted from port 0 contain CRC. The CRC type, if present, is determined by the CRC\_TYPE bit in the CPSW\_PN\_MAC\_CONTROL\_REG\_k register. If the CRC\_TYPE bit is cleared to 0h then the CRC present in each packet after host port egress is Ethernet CRC. If the CRC\_TYPE bit is set to 1h then the CRC present in each packet after host port egress is Castagnoli CRC.

#### Note

The CRC type present in the packet after host port egress is determined solely by the CRC\_TYPE bit in the CPSW\_PN\_MAC\_CONTROL\_REG\_k register regardless of the CRC type present in the packet during Ethernet port ingress.



#### 12.2.2.4.6.4.1 Transmit VLAN Processing

Transmit packets are NOT modified during switch egress when the VLAN\_AWARE bit in the CPSW\_CONTROL\_REG register is cleared to 0h. This means that the switch is not in VLAN-aware mode.

The next three sections cover transmit processing when the switch is in VLAN-aware mode for different packet types. The Gigabit Ethernet switch is in VLAN-aware mode when the VLAN\_AWARE bit is set in the CPSW\_CONTROL\_REG register. While in VLAN-aware mode, VLAN is added, removed, or replaced according to the type of packet as well as the CPSW\_ALE\_UVLAN\_UNTAG[1-0] UVLAN\_FORCE\_UNTAGGED\_EGRESS bit in the packet header as explained below.

##### 12.2.2.4.6.4.1.1 Untagged Packets (No VLAN or Priority Tag Header)

Untagged packets are all packets that are not a VLAN packet or a priority tagged packet. According to the CPSW\_ALE\_UVLAN\_UNTAG[1-0] UVLAN\_FORCE\_UNTAGGED\_EGRESS bit in the packet header the packet may exit the switch with a VLAN tag inserted or the packet may leave the switch unchanged. The two cases are discussed below.

- Insert VLAN Case:

Untagged input packets have the header packet VLAN inserted when the CPSW\_ALE\_UVLAN\_UNTAG[1-0] UVLAN\_FORCE\_UNTAGGED\_EGRESS bit in the transmit packet header is de-asserted. For untagged packets, the VLAN EtherType = 0x8100 is inserted after the source address followed by the two byte header packet VLAN. The header packet VLAN is composed of the header packet priority along with the PORT\_CFI and PORT\_VID values from the CPSW\_PN\_PORT\_VLAN\_REG\_k register (where N is the port that the untagged packet entered the switch) through. The packet length/type field is output four bytes later than it is input and is not removed or replaced. If the CRC is present in the packet data (PASS\_CRC is asserted), then the packet CRC is replaced with a hardware generated CRC.

- No Change Case:

Untagged input packets are output unchanged when the CPSW\_ALE\_UVLAN\_UNTAG[1-0] UVLAN\_FORCE\_UNTAGGED\_EGRESS transmit packet header bit is asserted.

##### 12.2.2.4.6.4.1.2 Priority Tagged Packets (VLAN VID == 0 && EN\_VID0\_MODE == 0h)

Priority tagged packets are packets that contain a VLAN header with VID = 0. According to the CPSW\_ALE\_UVLAN\_UNTAG[1-0] UVLAN\_FORCE\_UNTAGGED\_EGRESS bit in the packet header, priority tagged packets may exit the switch with their VLAN ID and priority replaced or they may have their priority tag completely removed. The two cases are discussed below.

#### Note

In order for a priority tagged packet to fall into this category the ENABLE\_VID0\_MODE bit in the CPSW\_ALE\_CONTROL register must also be set to 0h. If the ENABLE\_VID0\_MODE bit in the CPSW\_ALE\_CONTROL register is set to 1h, then packets with a VLAN VID of 0 will fall into the VLAN Tagged Packets category in [Section 12.2.2.4.6.4.1.3](#) below.

- Replace Priority and VLAN ID Case:

Priority tagged input packets have the packet VLAN ID and the packet priority replaced with the header packet VLAN ID and the header packet priority when the transmit packet header CPSW\_FORCE\_UNTAGGED\_EGRESS\_REG[1-0] MASK bit is de-asserted. The header packet VLAN ID comes from the PORT\_VID bits in the CPSW\_PN\_PORT\_VLAN\_REG\_k register (where N is the port where the packet entered the switch). The header packet priority is based on the packet priority to header packet priority mapping in the CPSW\_PN\_RX\_PRI\_MAP\_REG\_k register (where N is the port where the packet entered the switch).

- Remove VLAN Header Case:

Priority tagged input packets have the 4-byte packet VLAN information removed when the transmit packet header CPSW\_ALE\_UVLAN\_UNTAG[1-0] UVLAN\_FORCE\_UNTAGGED\_EGRESS bit is asserted. The 0x8100 EtherType is removed as is the two byte packet VLAN. Input 64-67 byte priority tagged packets go out with the VLAN removed and padded to 64-bytes if the PASS\_CRC input bit is asserted. The input CRC bytes are used as the pad data. Input 64-byte priority-tagged packets use all four input CRC bytes as pad, input 65-byte priority-tagged packets use three of the input CRC bytes as pad, and so on. No pad is performed if the PASS\_CRC input bit is not asserted - input 64-67 byte (on the wire) priority-tagged packets go out as 60-63 byte packets. The output CRC is replaced with a generated CRC when the VLAN is removed.

#### **12.2.2.4.6.4.1.3 VLAN Tagged Packets (VLAN VID != 0 || (EN\_VID0\_MODE == 1h && VLAN VID == 0))**

VLAN tagged packets are packets that contain a VLAN header specifying the VLAN the packet belongs to (VID), the packet priority (PRI), and the drop eligibility indicator (CFI). According to the CPSW\_ALE\_UVLAN\_UNTAG[1-0] UVLAN\_FORCE\_UNTAGGED\_EGRESS bit in the packet header, VLAN tagged packets may exit the switch with their VLAN priority replaced or they may have their VLAN header completely removed. The two cases are discussed below.

- Replace Priority Case:

VLAN tagged input packets are output with the packet priority replaced with the header packet priority when the transmit packet header CPSW\_ALE\_UVLAN\_UNTAG[1-0] UVLAN\_FORCE\_UNTAGGED\_EGRESS bit is deasserted. The header packet priority is based on the packet priority to header packet priority mapping in the CPSW\_PN\_RX\_PRI\_MAP\_REG\_k register (where N is the port where the packet entered the switch). If the CRC is present in the packet data (PASS\_CRC is asserted), then the packet CRC is replaced with a generated CRC.

- Remove VLAN Header Case:

VLAN tagged input packets have the 4-byte packet VLAN information removed when the transmit packet header CPSW\_ALE\_UVLAN\_UNTAG[1-0] UVLAN\_FORCE\_UNTAGGED\_EGRESS bit is asserted. The 0x8100 EtherType is removed as is the two byte packet VLAN. Input 64-67 byte priority tagged packets go out with the VLAN removed and padded to 64-bytes if the PASS\_CRC input bit is asserted. The input CRC bytes are used as the pad data. Input 64-byte priority tagged packets use all four input CRC bytes as pad, input 65-byte priority tagged packets use three of the input CRC bytes as pad, and so on. No pad is performed if the PASS\_CRC input bit is not asserted - input 64-67 byte (on the wire) priority tagged packets go out as 60-63 byte packets. The output CRC is replaced with a generated CRC when the VLAN is removed.

#### **12.2.2.4.6.4.2 Ethernet Port Ingress Packet CRC**

All Ethernet ports check the ingress packet CRC in all modes/speeds. The receive port can check either Ethernet CRC or Castagnoli CRC as determined by the CRC\_TYPE bit in the CPSW\_PN\_MAC\_CONTROL\_REG\_k register.

#### **12.2.2.4.6.4.3 Ethernet Port Egress Packet CRC**

Ethernet ports transmit each egress packet with the CRC selected by the CRC\_TYPE bit in the CPSW\_PN\_MAC\_CONTROL\_REG\_k register, regardless of the type of CRC that the packet had on ingress to the switch. At the egress port after passing through the switch, the packet CRC is checked for correctness and if the CRC is correct then the packet is output with the generated selected output CRC. If the packet CRC is incorrect, due either to a bit flip in a memory or an error CRC passed in on host ingress, then the generated egress CRC type is used with at least a single byte of the CRC inverted to indicate the error. If the packet length including CRC is divisible by 4 then all 4 CRC bytes will be inverted on error. If there are three bytes remainder after dividing the packet length by 4 then three bytes will be inverted (and so on down to one byte remainder).

#### **12.2.2.4.6.4.4 CPPI Port Ingress Packet CRC**

CPPI port ingress packets can be passed in with or without a CRC. The host port is Ethernet CRC only. CPPI ingress packets are not checked for CRC correctness on CPPI ingress, however they are checked for



correctness on Ethernet egress and are output with a CRC error if they came in with a CRC error. If a CPPI ingress packet does not have the INFO0 PASSED\_CRC bit set then a CRC will be generated for the packet on CPPI ingress. If the PASSED\_CRC bit is set then the packet is received and forwarded unchanged. The CRC type is input in the INFO0 word CRC\_TYPE bit and must be Ethernet CRC.

#### 12.2.2.4.6.4.5 CPPI Port Egress Packet CRC

The P0\_TX\_CRC\_REMOVE bit in the CPSW\_CONTROL\_REG register determines if CPPI egress packet have an Ethernet CRC included or not.

#### 12.2.2.4.6.5 FIFO Memory Control

Each of the two CPSW\_9G ports has an identical associated FIFO. Each FIFO contains a single logical receive queue and eight logical transmit queues (priority 0 through 7 with 7 the highest priority). Each FIFO memory contains 20,480 bytes (20k) total organized as 2560 by 64-bit words contained in a single memory instance. The FIFO memory is used for the associated port transmit and receive queues. The TX\_MAX\_BLKs field in the FIFOs associated CPSW\_PN\_MAX\_BLKs\_REG register determines the maximum number of 1k FIFO memory blocks to be allocated to the eight logical transmit queues (transmit total). The RX\_MAX\_BLKs field in the FIFO's associated CPSW\_PN\_MAX\_BLKs\_REG register determines the maximum number of 1k memory blocks to be allocated to the logical receive queue. The TX\_MAX\_BLKs value plus the RX\_MAX\_BLKs value must sum to 20 (the total number of blocks in the FIFO). If the sum were less than 20, then some memory blocks would be unused. The default is 17 (decimal) transmit blocks and three receive blocks. The FIFOs follow the naming convention of the Ethernet ports. Host Port is Port0 and External Ports is Port1.

Each transmit FIFO contains a configurable number of blocks (20 max) that are either 1k or 4k byte blocks that can be allocated to any priority.

#### 12.2.2.4.6.6 FIFO Transmit Queue Control

There are eight transmit queues in each transmit FIFO. Software has some flexibility in determining how packets are loaded into the queues and on how packet priorities are selected for transmission (how packets are removed and transmitted from queues). All ports on the switch have identical FIFO's. For the purposes of the below the transmit FIFO is switch egress even though the port 0 transmit FIFO is connected to the CPPI receive (also switch egress). The CPPI nomenclature is reversed from the Ethernet port nomenclature due to legacy reasons.

##### 12.2.2.4.6.6.1 CPPI Port Receive Rate Limiting

Port 0 receive operations can be configured to rate limit the host ingress data for each receive thread (priority). CPPI Receive has 8 threads for QOS. There is a committed information rate (CPSW\_P0\_PRI\_CIR\_REG\_y, where y = 0 to 7) and an excess information rate for each thread (CPSW\_P0\_PRI\_EIR\_REG\_y, where y = 0 to 7). Rate limiting is enabled for a thread when the committed information rate for the thread is non-zero. The excess information rate for a priority is enabled when the excess information rate for the priority is non-zero. The committed information rate must be non-zero if the excess information rate is configured to be non-zero. That is, there must be a configured non-zero committed information rate for there to be a configured non-zero excess information rate. Bulk traffic on other non-rate limited priorities does not impact the committed information traffic on a priority. However, bulk traffic on other non-rate limited threads does impact the excess information rates. No bulk thread will be enabled to send unless there are CPSW\_PN\_PRI\_CTL\_REG\_k[15-12] TX\_HOST\_BLKs\_REM number of unused blocks remaining in each of the Ethernet port transmit FIFOs. The "blocks remaining check" ensures that bulk traffic from the host will not block rate-limited traffic from the host. Rate limited channels must be the highest priority channels. For example, if two rate limited channels are required then threads 7 and 6 should be configured for committed information (and excess information if desired). When any channels are configured to be rate-limited, the priority type must be fixed for receive. Round-robin priority type is not allowed when rate-limiting is configured for any thread. The configured transfer rate includes the inter-packet gap (12 bytes) and the preamble (8 bytes). The rate in Mbits/second that each priority is configured to receive is controlled by the below equation. If the configured excess information rate is zero, then only the committed information rate is transferred:

Priority Transfer rate [Mbit/s] = (((Frequency in MHZ) \* CPSW\_P0\_PRI\_CIR\_REG\_y) / 32768) + (((Frequency in MHZ) \* CPSW\_P0\_PRI\_EIR\_REG\_y) / 32768))

Where the *frequency* is the CPPI\_ICLK frequency (in MHz) and priority 0 to 7.

For example, 10Mbps on priority 7 would give the below:

10Mbps =  $\sim ((350 * 936) / 32768)$ , at 350Mhz and CPSW\_P0\_PRI\_CIR\_REG\_y[27-0] PRI\_CIR value = 936 (no excess information rate)

#### 12.2.2.4.6.2 Ethernet Port Transmit Rate Limiting

Ethernet port transmit operations can be configured to rate limit egress data for each egress priority. There is a committed information rate (CPSW\_P0\_PRI\_CIR\_REG\_y, where y = 0 to 7) and an excess information rate for each priority (CPSW\_P0\_PRI\_EIR\_REG\_y, where y = 0 to 7). Rate limiting is enabled for a priority when the committed information rate for the priority is non-zero. The excess information rate for a priority is enabled when the excess information rate for the priority is non-zero. The committed information rate must be non-zero if the excess information rate is configured to be non-zero. That is, there must be a configured non-zero committed information rate for there to be a configured non-zero excess information rate. Bulk traffic on other non-rate limited priorities does not impact the committed information traffic on a priority. However, bulk traffic on other non-rate limited threads does impact the excess information rates. Rate limited channels must be the highest priority channels. For example, if two rate limited channels are required then priorities 7 and 6 should be configured for committed information (and excess information if desired). The configured transfer rate includes the inter-packet gap (12 bytes) and the preamble (8 bytes). The rate in Mbits/second that each priority is configured to send is controlled by the below equation. If the excess information rate is disabled then the committed information rate only is transferred:

Priority Transfer rate [Mbit/s] =  $(((((\text{Frequency in MHz}) * \text{CPSW\_P0\_PRI\_CIR\_REG\_y}) / 32768) + (((\text{Frequency in MHz}) * \text{CPSW\_P0\_PRI\_EIR\_REG\_y}) / 32768)))$

Where the *frequency* is the CPPI\_ICLK frequency (in MHz) and priority 0 to 7.

#### 12.2.2.4.6.7 Interspersed Express Traffic (IET – P802.3br/D2.0)

When IET is enabled through CPSW\_CONTROL\_REG[17] IET\_ENABLE = 1h and configured, IET allows preemptable traffic on selected transmit priorities to be preempted by express traffic on selected express priorities. Received traffic is separated onto express and preempt receive queues. When IET is disabled (IET\_ENABLE = 0h), all Ethernet traffic is express traffic and switch operation is as if IET does not exist. Traffic is only intended to be moved to the preempt queue after preemption is verified and enabled.

##### 12.2.2.4.6.7.1 IET Configuration

- Using the preempt receive queue requires more blocks to be allocated to the ports receive FIFO. Write a value of decimal 7 to the CPSW\_PN\_MAX\_BLKs\_REG[7-0] RX\_MAX\_BLKs bit field, and a value of decimal 13 to the CPSW\_PN\_MAX\_BLKs\_REG[15-8] TX\_MAX\_BLKs register for every port to be enabled for IET.
- Write the [23-0]MAC\_VERIFY\_CNT bit field in the Ethernet port CPSW\_PN\_IET\_VERIFY\_REG\_k register to set the verify/response timeout count. The default is 10ms for Gigabit mode. For other time values or link speeds the verify count should be updated. If CPSW\_PN\_IET\_CONTROL\_REG\_k[2] MAC\_DISABLEVERIFY bit is to be set (this is forced mode) then this step is unneeded.
- The receive FIFO block allocation is insufficient if CPSW\_STAT\_RX\_BOTTOM\_OF\_FIFO\_DROP\_k[31-0] COUNT is nonzero, indicating that receive packets are being dropped due to FIFO block allocation.
- Write the CPSW\_PN\_IET\_CONTROL\_REG\_k register in the Ethernet port as below:
  - Set the CPSW\_PN\_CONTROL\_REG\_k[16] IET\_PORT\_EN bit. The port will not actually be enabled until bit IET\_ENABLE is set in the CPSW\_CONTROL\_REG.
  - The CPSW\_PN\_IET\_CONTROL\_REG\_k[0] MAC\_PENABLE bit can be set as desired. No effect will occur until IET\_ENABLE is set. This bit enables preemptable packets to be preempted by express traffic but does not preclude packets from being sent to the preempt queue.
  - If verify/response is desired then CPSW\_PN\_IET\_CONTROL\_REG\_k[3] MAC\_LINKFAIL should be cleared by software to enable verify and response packets. Otherwise, MAC\_DISABLEVERIFY bit should be set for forced mode. Verification and response will occur immediately after clearing this bit.
  - Configure the remaining CPSW\_PN\_IET\_CONTROL\_REG\_k register bits as desired.

5. Set the IET\_ENABLE bit in the CPSW\_CONTROL\_REG register to enable IET operations.
6. After preemption has been verified, the CPSW\_PN\_IET\_CONTROL\_REG\_k[23-16] MAC\_PREMPT field is written to configure the FIFO priorities to be sent to the preempt queue (the other priorities with cleared bits go to the express queue). The hardware switch for each queue from express to preempt happens only when there are no packets queued on the priority.

#### **12.2.2.4.6.8 Enhanced Scheduled Traffic (EST – P802.1Qbv/D2.2)**

##### **12.2.2.4.6.8.1 Enhanced Scheduled Traffic Overview**

- When enabled and configured, EST allows express queue traffic to be scheduled (placed) on the wire at specific repeatable time intervals.
- EST operates on a repeating time interval generated by the CPTS EST function generator. For example, a 125us repeating time interval can be configured.
- Each Ethernet port has 128 EST fetch commands maximum in the global EST fetch RAM.
- Each 22-bit fetch command consists of a 14-bit fetch count (14 MSB's) and an 8-bit priority fetch allow (8 LSB's) that will be applied for the fetch count time in wireside clocks.
- The configured port fetch commands are executed in sequence, beginning at port address zero each time through the time interval beginning at cycle start.
- EST allows non-scheduled express and preempt queue traffic to be cleared from the wire to ensure that the scheduled traffic is transmitted at the proper time (zero allow).
- EST can be used with or without preemption. The CPSW\_PN\_IET\_CONTROL\_REG\_k[23-16] MAC\_PREMPT value determines whether the priority is enabled on the express or preempt queue. Whether a priority is on the express or preempt queue only effects the wire clear time from an EST operation perspective.
- Software should not move priorities to the preempt queue unless preemption is configured, enabled, and verified allowing preemption to occur.
- Express packet time stamp events can be enabled to assist software in configuring and timing EST operations.

##### **12.2.2.4.6.8.2 Enhanced Scheduled Traffic Fetch RAM**

- The EST fetch RAM is read/writable in the CPSW configuration address space.
- Each Ethernet transmit port has 128 locations in the global EST fetch RAM.
  - Ethernet port 1 has EST fetch RAM addresses 0x080-0x0FF, and so on.
- **One buffer operation:** When CPSW\_PN\_EST\_CONTROL\_REG\_k[0] EST\_ONEBUF is set to 1h, the 128 port locations operate as one buffer. The EST\_BUFFACT bit in CPSW\_PN\_FIFO\_STATUS\_REG\_k register is the upper address bit of the port's fetch RAM address indicating whether operation is currently in the upper or lower 64 locations of the port's fetch RAM.
- **Two buffer operation:** When CPSW\_PN\_EST\_CONTROL\_REG\_k[0] EST\_ONEBUF is cleared there are two 64-location buffers with CPSW\_PN\_EST\_CONTROL\_REG\_k[1] EST\_BUFSEL selecting the buffer to be used. When the buffer is switched by changing the CPSW\_PN\_EST\_CONTROL\_REG\_k[1] EST\_BUFSEL value, the actual switch occurs on cycle start. The actual buffer being used is indicated by the EST\_BUFFACT bit in CPSW\_PN\_FIFO\_STATUS\_REG\_k. Software should avoid writing the switched out buffer fetch RAM locations until it detects that the actual switch has occurred.
- The first address location in the port's fetch RAM space (location zero) is read at the beginning of each EST time interval (cycle start). Addresses are then read in ascending order for the duration of the interval. The port address zero is then read again at the beginning of the next cycle repeating the time interval packet operations.

##### **12.2.2.4.6.8.3 Enhanced Scheduled Traffic Time Interval**

- Each Ethernet port has an Enhanced Scheduled Traffic Function (ESTF) generator in the CPTS submodule.
- The EST function generator generates the EST time interval as a configured number of CPTS reference clocks (CPTS\_RFT\_CLK).
- The EST function generator rising edge is the cycle start time and the cycle repeats (cycle start occurs) after every time interval.

- The first fetch allowed value is at the port base address zero in the EST fetch RAM and is actually applied 16 wireside clocks after cycle start. The 16 clock cycle delay allows the first fetch value time to be fetched from the EST fetch RAM (prefetch time at cycle start).
- Each successive fetch allow is applied for the associated fetch count thereafter. The minimum non-zero fetch count is 16. The minimum value of 16 guarantees that the next fetch value has time to be fetched before the current fetch count is over. There are 64 maximum fetch values when CPSW\_PN\_EST\_CONTROL\_REG\_k[0] EST\_ONEBUF = 0h, and 128 maximum fetch values when CPSW\_PN\_EST\_CONTROL\_REG\_k[0] EST\_ONEBUF = 1h.
- The next cycle start then causes the fetch to once again start at the port address zero.

#### 12.2.2.4.6.8.4 Enhanced Scheduled Traffic Fetch Values

- The 22-bit fetch value is made up of the 14-bit fetch count and the 8-bit fetch allow.
- The fetch time indicates the number of wireside clocks that the fetch allow will be active.
- The fetch count is in Ethernet wireside clocks which is bytes in Gigabit mode (CPSW\_PN\_MAC\_CONTROL\_REG\_k[7] GIG = 1h) and nibbles in 10/100Mbps mode.
- When a fetch allow bit is set, the corresponding priority is enabled to begin packet transmission on an allowed priority subject to rate limiting. The actual packet transmission on the wire may carry over into the next fetch count and is the reason for the wire clear time in the fetch zero allow.
- When a fetch allow bit is cleared, the corresponding priority is not enabled to transmit for the fetch count time.
- A non-zero fetch allow value with a non-zero fetch count causes the fetch allow value to be applied for the fetch count number of wireside clocks.
- A zero fetch count causes the associated fetch allow to be held for the duration of the cycle (until the next cycle start).
- A zero fetch allow with a non-zero fetch count is intended to clear the wire for a scheduled (timed) express packet in the next fetch. A zero fetch allow indicates that no packet can be started for transmission for the associated fetch count. The associated fetch count must be sufficient to guarantee that the wire is cleared given that a packet on an allowed priority in the previous fetch could have been started on the previous clock and that there is hardware latency in the clear time. The timed packet should be sent on a priority that is enabled in the next fetch but disabled in the current zero allow fetch. The fetch allow previous to a zero allow should have only preempt priorities enabled or only express priorities enabled but not both.
- The number of clocks required to clear the wire varies depending Ethernet wire speed and on whether express or preempt priorities were allowed in the previous fetch command.

#### 12.2.2.4.6.8.5 Enhanced Scheduled Traffic Packet Fill

Packet fill can be enabled (CPSW\_PN\_EST\_CONTROL\_REG\_k[8] EST\_FILL\_EN = 1h) to occur in the fetch count time associated with a fetched zero allow. The intention with fill is that a smaller packet on a non-timed priority might be able to be inserted on the wire during the wire clear time which would increase wire utilization. Fill is intended to be used with either only express priorities in the previous non-zero fetched allow, or only preempt priorities in the previous non-zero fetched allow but not both. Fill must be configured to ensure that any fill packet does not conflict with the timed express packet allowed in the next fetch.

- **Express fill:**
  1. CPSW\_PN\_EST\_CONTROL\_REG\_k[8] EST\_FILL\_EN is set and
  2. A fetch contains a non-zero fetch count with a zero fetch allow, and
  3. The previous allow contained only express priorities.
  4. The fetch count loaded should be at least TBD clocks and the CPSW\_PN\_EST\_CONTROL\_REG\_k[25-16] EST\_FILL\_MARGIN should be at least TBD clocks in Gigabit mode to ensure that the wire is cleared for the timed express packet allowed in the next fetch.
  5. The fetch count loaded should be at least TBD clocks and the CPSW\_PN\_EST\_CONTROL\_REG\_k[25-16] EST\_FILL\_MARGIN should be at least TBD clocks in 10/100Mbps mode to ensure that the wire is cleared for the timed express packet allowed in the next fetch.
- **Preempt fill:**

1. CPSW\_PN\_EST\_CONTROL\_REG\_k[8] EST\_FILL\_EN is set and
  2. A fetch contains a non-zero fetch count with a zero fetch allow, and
  3. The previous allow contained only preempt priorities
  4. The fetch count loaded should be at least TBD clocks in Gigabit mode to ensure that the wire is cleared for the timed express packet allowed in the next fetch.
  5. The fetch count loaded should be at least TBD clocks in 10/100Mbps mode to ensure that the wire is cleared for the timed express packet allowed in the next fetch.
- **Express non-fill:**
    1. CPSW\_PN\_EST\_CONTROL\_REG\_k[8] EST\_FILL\_EN is clear and
    2. The previous allow contained only express priorities.
    3. The fetch count loaded should be at least TBD clocks in Gigabit mode to ensure that the wire is cleared for the timed express packet allowed in the next fetch.
    4. The fetch count loaded should be at least TBD clocks in 10/100Mbps mode to ensure that the wire is cleared for the timed express packet allowed in the next fetch.
  - **Preempt non-fill:**
    1. CPSW\_PN\_EST\_CONTROL\_REG\_k[8] EST\_FILL\_EN is clear and
    2. The previous allow contained only preempt priorities.
    3. The fetch count loaded should be at least TBD clocks in Gigabit mode to ensure that the wire is cleared for the timed express packet allowed in the next fetch.
    4. The fetch count loaded should be at least TBD clocks in 10/100Mbps mode to ensure that the wire is cleared for the timed express packet allowed in the next fetch.

#### 12.2.2.4.6.8.6 Enhanced Scheduled Traffic Time Stamp

The EST can be configured to generate CPTS timestamp events for selected express traffic. The EST timestamp events use the CPTS host event type (CPSW\_CPTS\_EVENT\_1\_REG[23-20] EVENT\_TYPE = 7 decimal). The EST timestamps will not override host sent timestamps for packets that were sent from the host with an enabled host timestamp.

- EST Events (host events) contain the below information:
  - Time Stamp of the selected express packet.
  - The event CPSW\_CPTS\_EVENT\_1\_REG[28-24] PORT\_NUMBER indicates the transmit port number.
  - The event CPSW\_CPTS\_EVENT\_1\_REG[23-20] EVENT\_TYPE is decimal 7 (host event).
  - The event CPSW\_CPTS\_EVENT\_1\_REG[23-20] MESSAGE\_TYPE indicates the packet transmit hardware switch priority.
  - The event CPSW\_CPTS\_EVENT\_1\_REG[15-0] SEQUENCE\_ID upper nibble indicates the packet receive port number.
  - The event CPSW\_CPTS\_EVENT\_1\_REG[15-0] SEQUENCE\_ID lower byte indicates the sequence number of the express packet in numerical order. The first event is event one, the second is event two and so on. The sequence ID rolls over to zero after 0xFF (8-bits).
  - The event domain is the value from the CPSW\_EST\_TS\_DOMAIN\_REG[7-0] EST\_TS\_DOMAIN register.
- When CPSW\_PN\_EST\_CONTROL\_REG\_k[2] EST\_TS\_EN is set, timestamp events will be generated on selected express traffic.
- When CPSW\_PN\_EST\_CONTROL\_REG\_k[3] EST\_TS\_FIRST is also set, events will be generated only on the first express packet in each time interval. If CPSW\_PN\_EST\_CONTROL\_REG\_k[4] EST\_TS\_ONEPRI is also set then the event will only be on the first CPSW\_PN\_EST\_CONTROL\_REG\_k[7-5] EST\_TS\_PRI express packet in the time interval. If CPSW\_PN\_EST\_CONTROL\_REG\_k[4] EST\_TS\_ONEPRI is clear then the event will be generated on the first express packet in the time interval on any priority.
- When CPSW\_PN\_EST\_CONTROL\_REG\_k[3] EST\_TS\_FIRST is clear, events will be generated on every express packet. If CPSW\_PN\_EST\_CONTROL\_REG\_k[4] EST\_TS\_ONEPRI is set then the event will be generated on every CPSW\_PN\_EST\_CONTROL\_REG\_k[7-5] EST\_TS\_PRI express packet. If CPSW\_PN\_EST\_CONTROL\_REG\_k[4] EST\_TS\_ONEPRI is clear then event will be generated on every express packet on any priority.



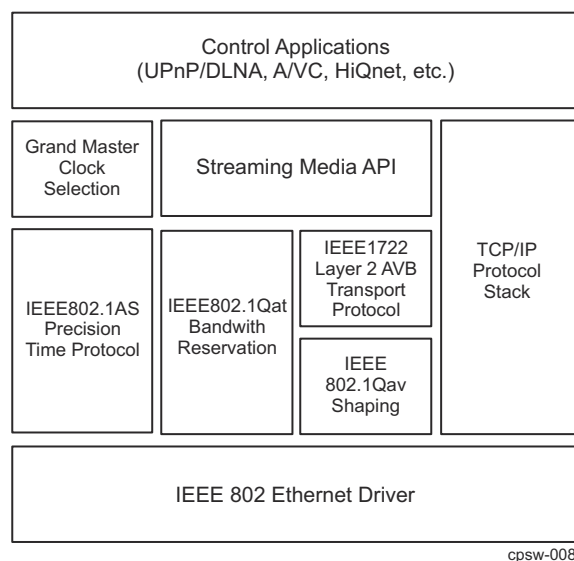
### 12.2.2.4.6.9 Audio Video Bridging

Audio Video Bridging is an ongoing project of IEEE 802.1 concerned with enabling low-latency streaming of time-sensitive audiovisual data over networks. Devices are designated as talkers (transmitters), bridges, or listeners (receivers). It is suggested that the maximum latency could be 2 ms over 7 hops for Class A devices and 20 ms over 7 hops for Class B devices. A hop is essentially a single local area network stage in the journey of a packet. Every time a bridge is encountered between one network section and another a hop is involved. One of the performance goals is that AVB streams will not use more than 75 percent of a link's bandwidth, leaving the remaining capacity for non-AVB streams.

The goal of developing AVB is simply--extend Ethernet's data-networking capabilities to the realm of reliable real-time audio/video networking.

An "Audio Video Bridging" network is one that implements a set of protocols being developed by the IEEE 802.1 Audio/Video Bridging Task Group. There are four primary differences between the proposed Audio Video Bridging architecture and existing 802 architectures (from now on the term "AVB" will be used instead of "Audio Video Bridging"):

1. Precise synchronization - IEEE 802.1AS: "*Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks*. "a.k.a Precision Time Protocol (PTP).
2. Traffic shaping for media streams - IEEE 802.1Qav: "*Virtual Bridged Local Area Networks: Forwarding and Queuing for Time-Sensitive Streams*."
3. Admission controls - IEEE 802.1Qat: "*Virtual Bridged Local Area Networks - Amendment 9: Stream Reservation Protocol (SRP)*."
4. Identification of non-participating devices - IEEE 802.1BA: "Audio/Video Bridging (AVB) Systems"



**Figure 12-167. The Network Static with AVB**

The following sections describe the media transport protocols that work within the AVB framework.

#### 12.2.2.4.6.9.1 IEEE 802.1AS: *Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks (Precision Time Protocol (PTP))*

The protocol defined by 802.1AS automatically selects a device to be the master clock, and then distributes this clock throughout the bridged LAN / IP subnet to all other network devices using link-specific transmit/receive time-stamping. However, we only use a two-step solution only on transmit. That is, we do not modify a packet with the timestamp on the way out. The timestamp packet is sent out and then a separate message with the timestamp is sent by the host afterward. Receive can be one or two step.

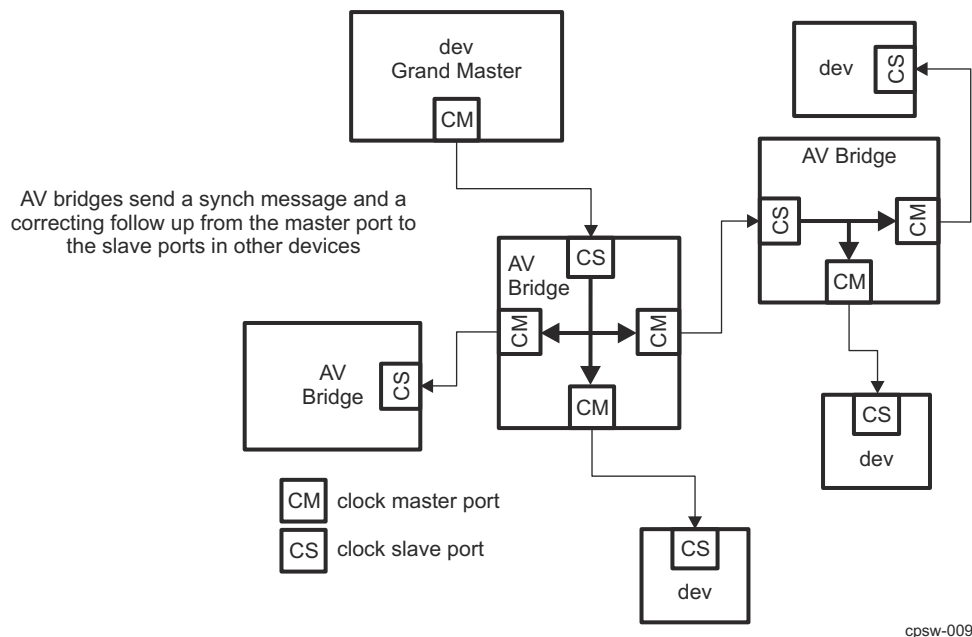
### Note

The 802.1AS-distributed clock is not used as a media clock. Rather, the shared 802.1AS clock reference is used to regenerate the media clock at the listener/renderer. Such a reference removes the need to force the latency of the network to be constant, or compute long running averages in order to estimate the actual media rate of the transmitter in the presence of substantial network jitter. IEEE 802.1AS is based on the ratified IEEE 1588 standard.

Based on IEEE 1588:2002, PTP devices exchange standard Ethernet messages that synchronize network nodes to a common time reference by defining clock master selection and negotiation algorithms, link delay measurement and compensation, and clock rate matching and adjustment mechanisms.

Designed as a simplified profile of IEEE 1588, a primary difference between 1588 and IEEE 802.1AS is that PTP is a layer 2-in other words, a non-IP routable protocol. Like IEEE 1588, PTP defines an automatic method for negotiating the network clock master, the Best Master Clock Algorithm (BMCA). PTP nodes can be assigned one of eight priority levels, presumably based on clock quality. BMCA defines the underlying negotiation and signaling mechanism whose purpose is to identify the AVB LAN Grandmaster. Once a Grandmaster has been selected, synchronization automatically begins.

At the core of 802.1AS synchronization is time-stamping. In short, during PTP message ingress/egress from the 802.1AS-capable MAC, the PTP Ether type triggers the sampling of the value of a local real-time counter (RTC). Slave nodes compare the value of their RTC against the PTP Grandmaster and, by use of link delay measurement and compensation techniques, match their RTC value to the time of the AVB LAN PTP domain. After network time throughout the AVB LAN has converged, periodic SYNC and FOLLOW\_UP messages provide the information that enables the PTP rate matching adjustment algorithms. The result is all PTP nodes are then synchronized to the same "Wall Clock" time. PTP assures 1- $\mu$ s accuracy over seven network hops.



**Figure 12-168. AVB Network & PTP Clock Entities**

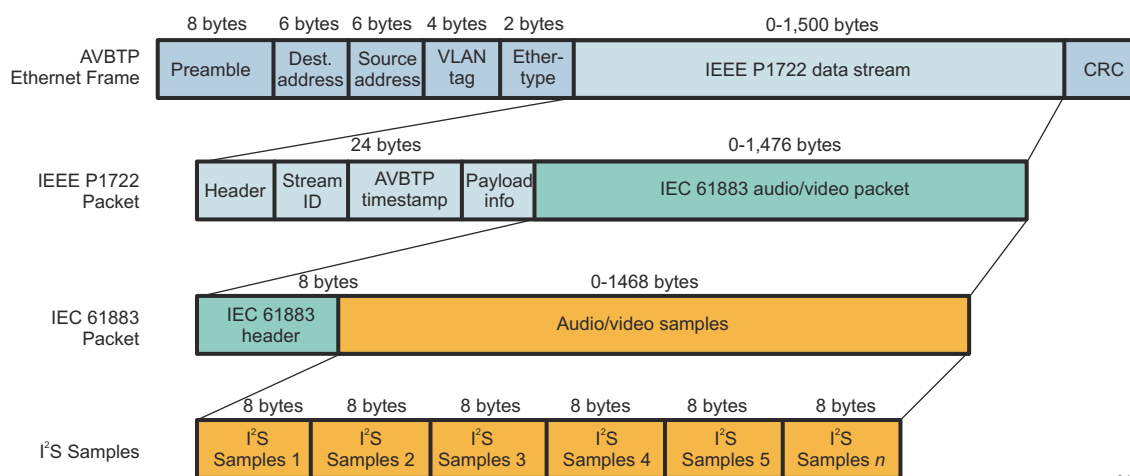
The media transport protocols that work within the AVB framework are:

#### 12.2.2.4.6.9.1.1 IEEE 1722: "Layer 2 Transport Protocol for Time-Sensitive Streams"

AVBTP or 1722 sits above the IEEE 802.1 AVB plumbing and below the application layer. It acts as the conduit between an Ethernet MAC and a streaming application. AVBTP abstracts the underlying network transmission channel to enable the virtual connection of distributed audio and video CODECs over reliable Ethernet networks.

A complete AVBTP Ethernet packet is shown in [Figure 12-169](#) and illustrates how IEC 61883-6 AM824 uncompressed audio samples are encapsulated in an Ethernet frame.

### IEEE 1722 Packet Construction



cpsw-010

**Figure 12-169. IEEE 1722 Packets**

1722 or AVBTP Presentation Time and Synchronization:

Synchronization in an AVB network starts with the Precision Time Protocol but ends with synchronized media clocks. PTP is responsible for synchronizing all nodes in an AVB network to identical wall clock time; not for synchronizing media clocks. In other words, PTP does not actually transport synchronized media clocks but instead provides a low-level building block crucial for managing a distributed media synchronization system.

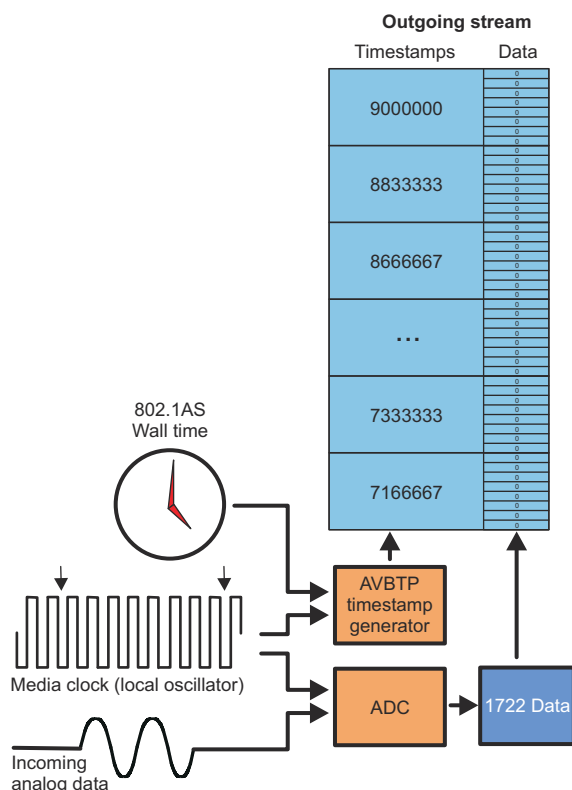
A crucial benefit of this approach is coexistence of multiple, independent media clock domains on an AVB network. Unrelated audio and video streams can simultaneously exist in the same LAN.

#### 12.2.2.4.6.9.1.1.1 Cross-timestamping and Presentation Timestamps

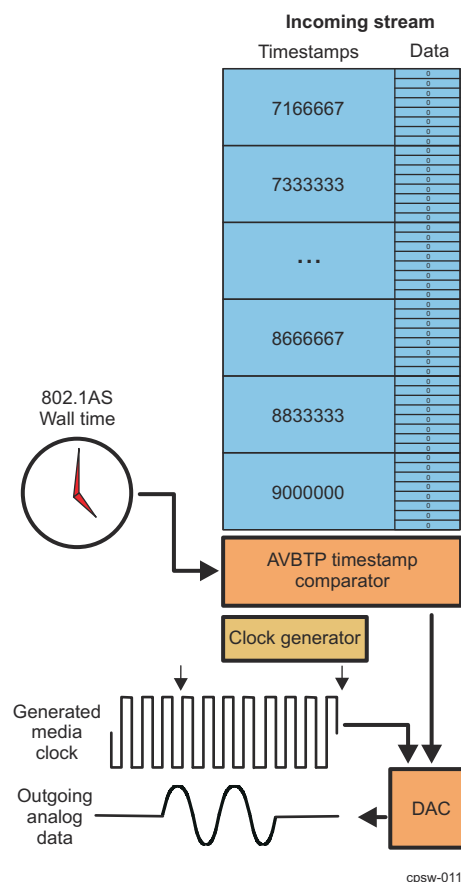
AVBTP assumes that AVB node media clocks are clocked by free-running oscillators. It is also assumed that the node's internal concept of wall clock time has been synchronized to the PTP Grandmaster. AVBTP media clock sources embed "AVBTP Presentation Timestamps" in AVBTP streaming packets. [Figure 12-170](#) illustrates the relationship between PTP network time and AVBTP Presentation Timestamps.



### Presentation timestamps



### Cross-time stamped clock recovery



**Figure 12-170. Cross Time Stamping and Presentation Timestamps**

#### 12.2.2.4.6.9.1.2 IEEE 1733: Extends RTCP for RTP Streaming over AVB-supported Networks

This standard specifies the protocol, data encapsulations, connection management and presentation time procedures used to ensure interoperability between audio and video based end stations that use standard networking services provided by all IEEE 802 networks meeting QoS requirements for time-sensitive applications by leveraging the Real-time Transport Protocol (RTP) family of protocols and IEEE 802.1 Audio/Video Bridging (AVB) protocols.

#### 12.2.2.4.6.9.2 IEEE 802.1Qav: "Virtual Bridged Local Area Networks: Forwarding and Queuing for Time-Sensitive Streams"

This standard allows bridges to provide guarantees for time-sensitive (that is, bounded latency and delivery variation), loss-sensitive real-time audio video (AV) data transmission (AV traffic). It specifies per priority ingress metering, priority regeneration, and timing-aware queue draining algorithms. This standard uses the timing derived from IEEE 802.1AS. Virtual Local Area Network (VLAN) tag encoded priority values are allocated, in aggregate, to segregate frames among controlled and non-controlled queues, allowing simultaneous support of both AV traffic and other bridged traffic over and between wired and wireless Local Area Networks (LANs).

Such a guarantee in bandwidth is provided by two functional entities:

- A registration protocol, which registers the service and its maximum network utilization with a device or switch (IEEE 802.1Qat: "Virtual Bridged Local Area Networks - Amendment 9: Stream Reservation Protocol (SRP)")
- A hardware bandwidth management service.
  - Receive policing
  - Transmit rate control.

## End Station Behavior

In order for an end station to successfully participate in the transmission and reception of time-sensitive streams, it is necessary for their behavior to be compatible with the operation of the forwarding and queuing mechanisms employed in bridges.

The requirements for end stations that participate as "talkers" i.e., sources of time-sensitive streams are different from the requirements that apply to "listeners", the destination station(s) for the streams.

## Talker Behavior

In order for Talker-originated data streams to make use of the credit-based shaper behavior in Bridges, it is a requirement for a Talker to use the priorities that the Bridges in the network recognize as being associated with SR classes exclusively for transmitting stream data.

It is also necessary for the Talker and the Bridges in the path to the Listener(s), to have a common view of the bandwidth required in order to transmit the Talker's streams, and for that bandwidth to be reserved along the path to the Listener(s). This latter requirement can be met by means of stream reservation mechanisms, such as defined in SRP, or by other management means.

End stations that are Talkers shall exhibit transmission behavior for frames that are part of "time-sensitive streams" that is consistent with the operation of the credit-based shaper algorithm, both in terms of the way they transmit frames that are part of an individual data stream, and in terms of the way they transmit stream data frames from a Port.

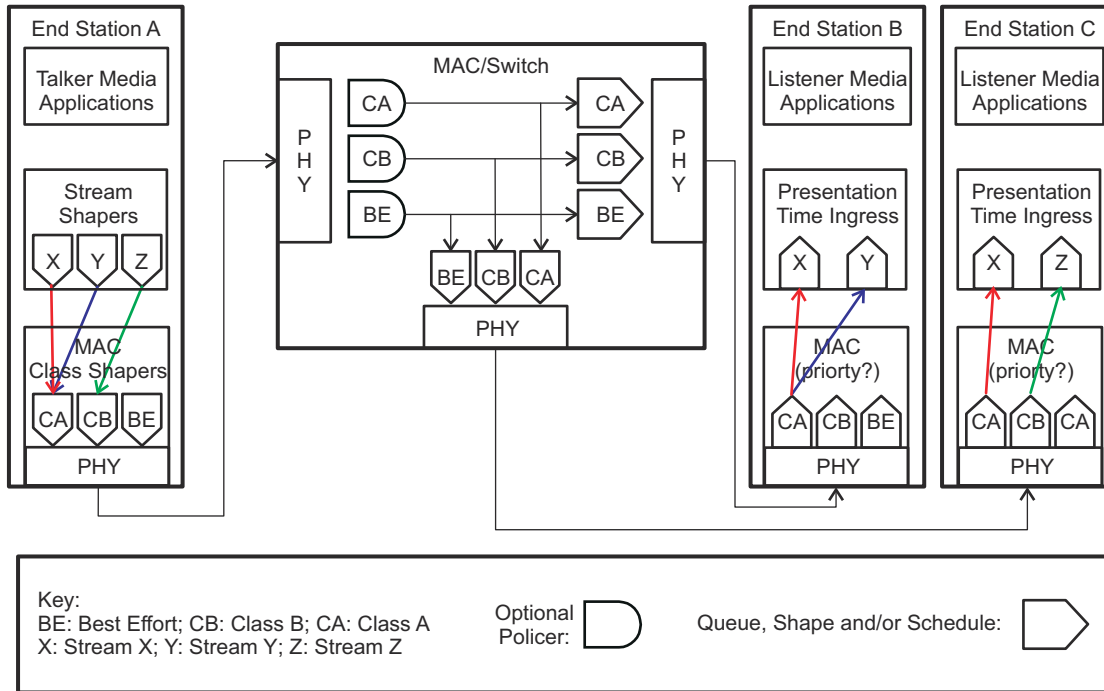
In effect, the queuing model for a Talker Port (and a Listener port), and for given priorities, can be considered to look like [Figure 12-171](#).

## Listener Behavior

The primary requirement for a listener station is that it is capable of buffering the amount of data that could be transmitted for a stream during a time period equivalent to the accumulated maximum jitter that could be experienced by stream data frames in transmission between Talker and Listener.

From the point of view of the specification of the forwarding and queuing requirements for time-sensitive streams, it is assumed that the listener will assess the buffering required for a stream as part of the stream bandwidth reservation mechanisms employed by the implementation.

The credit-based shaper's operation details are beyond the scope of this document.



cpsw-012

**Figure 12-171. AV Stream Queuing/Policing**

#### 12.2.2.4.6.9.2.1 Configuring the Device for 802.1Qav Operation

There is no dedicated register-set to be configured for the time-sensitive stream handling. The list of functional features of CPSW\_9G that will have to be configured are:

- DESCRIPTORS and CHANNEL CONFIGURATIONS:
  - CPPI TX and RX descriptors
  - VLAN and Priority tags

**Table 12-218. Example of TX Configuration**

TX DMA CHANNEL	Packet Priority	Switch Queue Priority
7	7	3
6	5	2
5	3	1
4	1	0

**Table 12-219. Example of RX Configuration**

RX DMA CHANNEL	Packet Priority	Switch Queue Priority
0	7	0
0	5	0
0	3	0
0	1	0

- ALE Configuration:
  - ALE in VLAN-ware mode, Non-ALE in bypass mode.

#### 12.2.2.4.6.10 Ethernet MAC Sliver

The Ethernet port peripheral is compliant to the IEEE Std 802.3 Specification. Half-duplex mode is supported in 10/100 Mbps mode, but not in 1000 Mbps (gigabit) mode.

## Features:

- Synchronous 10/100/1000 Mbit operation
- RMII/RGMII Interface
- Hardware Error handling including CRC
- Full-Duplex Gigabit operation (half-duplex gigabit is not supported)
- EtherStats and 802.3Stats RMON statistics gathering support for external statistics collection module
- Transmit CRC generation selectable on a per channel basis
- Emulation Support
- VLAN Aware Mode Support
- Hardware flow control
- Programmable Inter Packet Gap (IPG).

**12.2.2.4.6.10.1 CRC Insertion**

The MAC generates and appends a 32-bit Ethernet CRC onto the transmitted data, if the transmit packet header PASS\_CRC bit is 0h. For the Ethernet port generated CRC case, a CRC at the end of the input packet data is not allowed.

If the header word PASS\_CRC bit is set, then the last four bytes of the TX data are transmitted as the frame CRC. The four CRC data bytes should be the last four bytes of the frame and should be included in the packet byte count value. The MAC performs no error checking on the outgoing CRC when the PASS\_CRC bit is set.

**12.2.2.4.6.10.2 MTXER**

The MTXER signal is only used for EEE. If an underflow condition occurs on a transmitted frame, the frame CRC will be inverted to indicate the error to the network. Underflow is a hardware error.

**12.2.2.4.6.10.3 Adaptive Performance Optimization (APO)**

The Ethernet MAC port incorporates Adaptive Performance Optimization (APO) logic that may be enabled by setting the TX\_PACE bit in the CPSW\_PN\_MAC\_CONTROL\_REG\_k register. Transmission pacing to enhance performance is enabled when set. Adaptive performance pacing introduces delays into the normal transmission of frames, delaying transmission attempts between stations, reducing the probability of collisions occurring during heavy traffic (as indicated by frame deferrals and collisions) thereby increasing the chance of successful transmission.

When a frame is deferred, suffers a single collision, multiple collisions or excessive collisions, the pacing counter is loaded with an initial value of 31. When a frame is transmitted successfully (without experiencing a deferral, single collision, multiple collision or excessive collision) the pacing counter is decremented by one, down to zero.

With pacing enabled, a new frame is permitted to immediately (after one IPG) attempt transmission only if the pacing counter is zero. If the pacing counter is non zero, the frame is delayed by the pacing delay, a delay of approximately four inter-packet gap delays. APO only affects the IPG preceding the first attempt at transmitting a frame. It does not affect the back-off algorithm for re-transmitted frames.

**12.2.2.4.6.10.4 Inter-Packet-Gap Enforcement**

The measurement reference for the IPG of 96-bit times is changed depending on frame traffic conditions. If a frame is successfully transmitted without collision, and MCRC is de-asserted within approximately 48-bit times of MTXEN being de-asserted, then 96-bit times is measured from MTXEN. If the frame suffered a collision, or if MCRC is not de-asserted until more than approximately 48-bit times after MTXEN is de-asserted, then 96-bit times (approximately, but not less) is measured from MCRC.

The Ethernet port transmit inter-packet gap (IPG) may be shortened by eight bit times when short gap is enabled and triggered. Setting the [10] TX\_SHORT\_GAP\_ENABLE bit in the CPSW\_PN\_MAC\_CONTROL\_REG\_k register enables the gap to be shortened when triggered. The condition is triggered when the ports associated transmit packet FIFO has a user defined number of FIFO blocks used. The associated transmit FIFO blocks used value determines if the gap is shortened, and so on. The CPSW\_GAP\_THRESH\_REG register value

determines the short gap threshold. If the FIFO blocks used is greater than or equal to the GAP\_THRESH value then short gap is triggered.

#### **12.2.2.4.6.10.5 Back Off**

The Gigabit Ethernet Mac Sliver implements the 802.3 binary exponential back-off algorithm.

#### **12.2.2.4.6.10.6 Programmable Transmit Inter-Packet Gap**

The transmit inter-packet gap (IPG) is programmable through the CPSW\_PN\_MAC\_CONTROL\_REG\_k register. The default value is decimal 12. The transmit IPG may be increased to the maximum value of 1FFh. Increasing the IPG is not compatible with transmit pacing. The short gap feature will override the increased gap value, so the short gap feature may not be compatible with an increased IPG.

#### **12.2.2.4.6.10.7 Speed, Duplex and Pause Frame Support Negotiation**

The Ethernet port can operate in half duplex or full duplex in 10/100 Mbit modes, and can operate in full duplex only in 1000 Mbit mode. Pause frame support is included in 10/100/1000 Mbit modes as configured by the host.

#### **12.2.2.4.6.10.8 RMII Interface**

The CPRMII peripheral is compliant to the RMII specification document.

##### **12.2.2.4.6.10.8.1 Features**

- Source Synchronous 10/100 Mbit operation
- Full and Half Duplex support

##### **12.2.2.4.6.10.8.2 RMII Receive (RX)**

The CPRMII receive (RX) interface converts the input data from the external RMII PHY (or switch) into the required MII (CPGMAC) signals. The carrier sense and collision signals are determined from the RMII input data stream and transmit inputs as defined in the RMII specification.

An asserted RMII\_RXER on any di-bit in the received packet will cause an MII\_RXER assertion to the CPGMAC during the packet. In 10Mbps mode, the error is not required to be duplicated on 10 successive clocks. Any di-bit which has an asserted RMII\_RXER during any of the 10 replications of the data will cause the error to be propagated.

Any received packet that ends with an improper nibble boundary aligned RMII\_CRS\_DV toggle will issue an MII\_RXER during the packet to the CPGMAC. Also, a change in speed or duplex mode during packet operations will cause packet corruption.

The CPRMII can accept receive packets with shortened preambles, but 0x55 followed by a 0x5D is the shortest preamble that will be recognized (1 preamble byte with the start of frame byte). At least one byte of preamble with the start of frame indicator is required to begin a packet. An asserted RMII\_CRS\_DV without at least a single correct preamble byte followed by the start of frame indicator will be ignored.

##### **12.2.2.4.6.10.8.3 RMII Transmit (TX)**

The CPRMII transmit (TX) interface converts the CPGMAC MII input data into the RMII transmit format. The data is then output to the external RMII PHY.

The CPGMAC does not source the transmit error (MII\_TXERR) signal. Any transmit frame from the CPGMAC with an error (underrun) will be indicated as an error by an error CRC. Transmit error is assumed to be de-asserted at all times and is not an input into the CPRMII module. Zeroes are output on RMII\_TXD[1:0] for each clock that RMII\_TXEN is de-asserted.

#### **12.2.2.4.6.10.9 RGMII Interface**

The CPRGMII peripheral is compliant to the RGMII specification document.

##### **12.2.2.4.6.10.9.1 Features**

- Supports 1000/100/10 Mbps speed
- Full and Half Duplex support (CPGMAC supports only Full duplex in Gigabit mode).

- MII mode support
- Energy Efficient Ethernet Support

#### 12.2.2.4.6.10.9.2 RGMII Receive (RX)

The CPRGMII receive (RX) interface converts the source synchronous DDR input data from the external RGMII PHY into the required G/MII (CPGMAC) signals.

#### 12.2.2.4.6.10.9.3 In-Band Mode of Operation

The CPRGMII is operating in the in-band mode of operation when the RGMII\_RX\_INBAND input is asserted. RGMII\_RX\_INPUT is asserted by configuring the CTL\_EN bit to 1h of the CPSW\_PN\_MAC\_CONTROL\_REG\_k register. The link status, duplexity, and speed are determined from the RGMII input data stream RXD[3:0] when RX\_CTL is deasserted, as defined in the RGMII specification. The PHY might need to be configured beforehand to output in-band data. The in-band data is indicated as shown in [Table 12-220](#).

**Table 12-220. In-Band Data**

RXD3	RXD[2:1]		RXD0
Duplex status:	Link Speed:	RXC_CLK Speed:	Link Status:
0h: half-duplex	0h: 10-Mbps mode	2.5 MHz	0h: Link is down
1h: full-duplex	1h: 100-Mbps mode	25 MHz	1h: Link is up
	2h: 1000-Mbps mode	125 MHz	
	3h: Reserved	Reserved	

#### 12.2.2.4.6.10.9.4 Forced Mode of Operation

The CPRGMII is operating in the forced mode of operation when the RGMII\_RX\_INBAND input is deasserted by setting to 0h bit CTL\_EN of the CPSW\_PN\_MAC\_CONTROL\_REG\_k register. In the forced mode of operation, the in-band data is ignored if present. The link status is forced high, and the duplexity and speed are determined from the CPSW\_PN\_MAC\_CONTROL\_REG\_k[0] FULLDUPLEX and CPSW\_PN\_MAC\_CONTROL\_REG\_k[7] GIG bits. If bit [7] GIG = 1h, then CPRGMII is operating in Gigabit mode. If bit [7] GIG is cleared (0h), then CPRGMII is operating in 100 Mbps mode.

#### 12.2.2.4.6.10.9.5 RGMII Transmit (TX)

The CPRGMII transmit (TX) interface converts the CPGMAC G/MII input data into the DDR RGMII format. The DDR data is then output to the external PHY.

The CPGMAC does not source the transmit error (TXERR) signal. Any transmit frame from the CPGMAC with an error (underrun) will be indicated as an error by an error CRC. Transmit error is assumed to be deasserted at all times and is not an input into the CPRGMII module.

In 10/100 Mbps mode, the TXD[7:0] data bus uses only the lower nibble. The CPRGMII will output the lower nibble twice in 10/100 Mbps mode to avoid unnecessary signal switching.

Packets will be precluded from transmission through the CPRGMII module for 4096 transmit clocks after the rising edge of RGMII\_LINK. Packet transmission will begin on the first TX\_CTL rising edge after the 4096 transmit clock count has expired.

#### 12.2.2.4.6.10.10 Frame Classification

Received frames are proper (good) frames if they are between 64 and CPSW\_P0\_RX\_MAXLEN\_REG[13:0] RX\_MAXLEN in length (inclusive) and contain no errors (code/align/CRC).

Received frames are long frames if their frame count exceeds the value in the CPSW\_P0\_RX\_MAXLEN\_REG/ CPSW\_PN\_RX\_MAXLEN\_REG\_k register. The register reset (default) value is 1518 (decimal). Long received frames are either oversized or jabber frames. Long frames with no errors are oversized frames. Long frames with CRC, code, or alignment errors are jabber frames.

Received frames are short frames if their frame count is less than 64 bytes. Short frames that contain no errors are undersized frames. Short frames with CRC, code, or alignment errors are fragment frames. If RX\_CSF\_EN

bit in CPSW\_PN\_MAC\_CONTROL\_REG\_k is set to 1h, undersized frames from 33 to 63 bytes will be forwarded only to the host on a best effort basis (meaning that the ALE may or may not be able to keep up with the packet rate and the short packet may be dropped due to bandwidth limitations). If RX\_CSF\_EN and RX\_CEF\_EN in CPSW\_PN\_MAC\_CONTROL\_REG\_k are set, fragment frames from 33 to 63 bytes will also be forwarded only to the host on a best effort basis. Ethernet port received frames shorter than 33 bytes are dropped in all cases.

A received long packet will always contain RX\_MAXLEN number of bytes transferred to memory (if CPSW\_PN\_MAC\_CONTROL\_REG\_k[22]RX\_CEF\_EN = 1h). An example with RX\_MAXLEN = 1518 is:

- If the frame length is 1518, then the packet is not a long packet and there will be 1518 bytes transferred to memory.
- If the frame length is 1519, there will be 1518 bytes transferred to memory. The last three bytes will be the first three CRC bytes.
- If the frame length is 1520, there will be 1518 bytes transferred to memory. The last two bytes will be the first two CRC bytes.
- If the frame length is 1521, there will be 1518 bytes transferred to memory. The last byte will be the first CRC byte.

If the frame length is 1522, there will be 1518 bytes transferred to memory. The last byte will be the last data byte.

#### 12.2.2.4.6.10.11 Receive FIFO Architecture

This section describes the architecture of the Ethernet port's receive FIFOs. Internal to the Gigabit Ethernet switch, all Ethernet ports have an identical associated packet FIFO. Each transmit packet FIFO contains eight logical transmit queues (priority 0 through 7 with 7 the highest priority). Each transmit FIFO memory contains 81,920 bytes total organized as 2560 by 256-bit words. Each FIFO also contains a single memory for the receive queue. Each receive FIFO memory contains a total of 32768 bytes total organized as 1024 by 256-bit words.

#### 12.2.2.4.6.11 Embedded Memories

**Table 12-221. Embedded Memories**

Memory Type Description	Number of Instances	
Single-port 3072-word × 64 RAM	1	(Combined FIFO RAM)
Single-port 128-word × 28-bit RAM	1	1 (EST)



#### 12.2.2.4.6.12 Memory Error Detection and Correction

The CPSW\_9G error detection and correction logic uses the ECC Aggregator Module.

The ECC CPSW\_ECC\_VECTOR register is used to select which ECC RAM's status and control registers are currently being read or written as shown in [Table 12-222](#). The CPSW FIFO RAMs implement ECC only on packet headers. The packet data is protected by Ethernet CRC. The ALE and EST RAMs have complete ECC as normal.

**Table 12-222. ECC RAM to CPSW RAM Mapping**

ECC RAM Number	CPSW RAM
0	ALE RAM
1	Port 0 FIFO RAM

##### 12.2.2.4.6.12.1 Packet Header ECC

Only packet headers bits are protected by ECC in the FIFO RAMs. The ECC\_ERR\_CTRL1[31-0] ECC\_ROW bit is used to activate a row address where force single-bit error or force double-bit error needs to be applied. ECC\_ERR\_CTRL2 [15-0] ECC\_BIT1 is implemented to determine which bit of the header is flipped for an SEC error when the ECC\_CRC\_MODE bit is cleared in the CPSW\_CONTROL\_REG register. The ECC status registers return the RAM row address where the single or double-bit error has occurred (ECC\_ERR\_STAT2[31-0] ECC\_ROW) along with the bit position in the RAM data that is in error (ECC\_ERR\_STAT1[31-16] ECC\_BIT1 value). Forcing double-bit errors in testing can cause indeterminate operation if multiple used packet header bits are flipped given that only single-bit errors are fixed by the ECC logic. Header bits 207 down to 200 are not currently used in the CPSW and may be used to test double bit errors without the possibility of requiring a reset for the switch to recover from the double bit error. No header bits are flipped when ECC\_CRC\_MODE is set to 1h. Either the RX\_ECC\_ERR\_EN (enable receive ECC error operations) or the TX\_ECC\_ERR\_EN (enable transmit ECC error operations) bits must be set in the CPSW\_P0\_CONTROL\_REG register to test ECC header errors.

The header ECC code is stored in bits 255 down to 208. If any bit is flipped in the ECC code, the flipped bit will be corrected, but the index of the flipped bit will be reported as bit zero. This implies that when the aggregator reports that there is a SEC on bit 0, it can mean two things: either SEC on data bit 0 or SEC somewhere inside the ECC code. Any packet header with ECC error issues a pulse interrupt (ECC\_PULSE\_INTR) as does an ALE RAM ECC error.

##### 12.2.2.4.6.12.2 Packet Protect CRC

Each packet received without error is passed through the CPSW\_9G memories with a generated Ethernet protect CRC. The protect CRC is checked on egress for correctness and removed. If the CRC is correct (no RAM bit errors), then the packet is output with the selected port CRC type. If a protect CRC error is detected on host egress then the TXST\_DROP signal will be asserted so that the packet is dropped to the host. If a protect CRC error is detected on Ethernet egress then the egress CRC will be generated on the packet and at least one byte of the CRC will be inverted on output. CRC memory protect errors do not assert the ECC\_PULSE\_INTR signal. CRC memory protect errors are counted in the associated port statistics registers and issue an interrupt on STAT\_PEND\_INTR if any CRC memory protect error occurs (and the statistics for that port are enabled). When the ECC\_CRC\_MODE bit in the CPSW\_CONTROL\_REG register is set, the ECC\_ERR\_CTRL2 [15-0] ECC\_BIT1 bit field will flip the associated column bit in any FIFO memory read operation, inducing a CRC protect error when the protect CRC is checked. No header bits are flipped when ECC\_CRC\_MODE is set. Either the RX\_ECC\_ERR\_EN or the TX\_ECC\_ERR\_EN bits must be set in the CPSW\_P0\_CONTROL\_REG register to test packet CRC errors.

##### 12.2.2.4.6.12.3 Aggregator RAM Control

The ECC logic for each FIFO RAM (receive and transmit) is divided into eight separate ECC encoders/decoders that encode/decode 26-bits of data each. Each of the 8 encoders (0 to 7) generates 6-bits of ECC code (48 code bits total), and each of the eight decoders (0 to 7) checks 6-bits of ECC code across the 26-bits of data (208 data bits total). The 48-bits of ECC code are passed through the RAM in the upper 48 unused bits in the header word.



The header data bits and ECC code bits are shown in [Table 12-223](#). The [15-0] ECC\_BIT1 value returned on error is a 16-bit value that is the concatenation of 5 bits of zero, 3 bits of the encoder/decoder number (0 to 7), 3 bits of zero, and 5 bits of index into the indicated 26-bit encoder/decoder.

For example, an ECC\_BIT1 value of 0x0308 is bit 8 of encoder/decoder 3, which is header bit 86 (that is,  $(26 \times 3) + 8$ ).

**Table 12-223. ECC Submodule Header Data Bit to Encoder/Decoder Mapping**

Header Data Bits	Encoder/Decoder
25:0	Encoder/Decoder 0 Data
51:26	Encoder/Decoder 1 Data
77:52	Encoder/Decoder 2 Data
103:52	Encoder/Decoder 3 Data
129:104	Encoder/Decoder 4 Data
155:130	Encoder/Decoder 5 Data
181:156	Encoder/Decoder 6 Data
207:182	Encoder/Decoder 7 Data
213:208	Encoder/Decoder 0 ECC
219:214	Encoder/Decoder 1 ECC
225:220	Encoder/Decoder 2 ECC
231:226	Encoder/Decoder 3 ECC
237:232	Encoder/Decoder 4 ECC
243:238	Encoder/Decoder 5 ECC
249:244	Encoder/Decoder 6 ECC
255:250	Encoder/Decoder 7 ECC

#### 12.2.2.4.6.13 Ethernet Port Flow Control

The Ethernet port have flow control available for transmit and receive. Transmit flow control stops the Ethernet port from transmitting packets to the wire (switch egress) in response to a received pause frame. Transmit flow control does not depend on FIFO usage.

The Ethernet port have flow control available for receive operations (packet ingress). Ethernet port receive flow control is initiated when enabled and triggered. Packets received on an Ethernet port can be sent to the CPPI port. The destination port can trigger the receive Ethernet port flow control. An Ethernet destination port triggers another Ethernet receive flow control when the destination port is full.

When a packet is received on an Ethernet port interface with enabled flow control the below occurs:

- The packet will be sent to all ports that currently have room to take the entire packet.
- The packet will be retried until successful to all ports that indicate they don't have room for the packet.

The flow control trigger to the Ethernet port will be asserted until the packet has been sent, and there is room in the logical receive FIFO for packet runout from another flow control trigger (RX\_BLK\_CNT = 0h). Ethernet port receive flow control is disabled by default on reset. Ethernet port receive flow control requires that the RX\_FLOW\_EN bit in CPSW\_PN\_MAC\_CONTROL\_REG\_k be set to 1h. When receive flow control is enabled on a port, the port's associated FIFO block allocation must be adjusted. The port RX allocation must increase from the default three blocks to accommodate the flow control runout. A corresponding decrease in the TX block allocation is required. If a sending port ignores a pause frame then packets may overrun on receive (and be dropped) but will not be dropped on transmit.

### 12.2.2.4.6.13.1 Ethernet Receive Flow Control

For every Ethernet port to be configured for full-duplex receive flow control, write a value of decimal 7 to the CPSW\_PN\_MAX\_BLKs\_REG[7-0] RX\_MAX\_BLKs bit field, and a value of decimal 13 to the CPSW\_PN\_MAX\_BLKs\_REG[15-8] TX\_MAX\_BLKs register. This re-allocation allows for flow control runout on the receive FIFO at the expense of FIFO memory on the Ethernet transmit side. 10/100Mbps half-duplex collision based receive flow control does not need this re-allocation. Receive flow control is enabled by the RX\_FLOW\_EN bit in the CPSW\_PN\_MAC\_CONTROL\_REG\_k register.

#### 12.2.2.4.6.13.1.1 Collision Based Receive Buffer Flow Control

Collision-based receive buffer flow control provides a means of preventing frame reception when the port is operating in half-duplex mode (FULLDUPLEX is cleared in CPSW\_PN\_MAC\_CONTROL\_REG\_k). When receive flow control is enabled and triggered, the port will generate collisions for received frames. The jam sequence transmitted will be the twelve byte sequence C3.C3.C3.C3.C3.C3.C3.C3.C3.C3.C3.C3 (hex). The jam sequence will begin no later than approximately as the source address starts to be received. Note that these forced collisions will not be limited to a maximum of 16 consecutive collisions, and are independent of the normal back-off algorithm. Receive flow control does not depend on the value of the incoming frame destination address. A collision will be generated for any incoming packet, regardless of the destination address.

#### 12.2.2.4.6.13.1.2 IEEE 802.3X Based Receive Flow Control

IEEE 802.3x based receive flow control provides a means of preventing frame reception when the port is operating in full-duplex mode (FULLDUPLEX bit is set in the CPSW\_PN\_MAC\_CONTROL\_REG\_k register). When receive flow control is enabled and triggered, the port will transmit a pause frame to request that the sending station stop transmitting for the period indicated within the transmitted pause frame.

The Ethernet port will transmit a pause frame to the reserved multicast address at the first available opportunity (immediately if currently idle, or following the completion of the frame currently being transmitted). The pause frame will contain the maximum possible value for the pause time (FFFFh). The MAC will count the receive pause frame time (decrements FF00h down to 0) and retransmit an outgoing pause frame if the count reaches zero. When the flow control request is removed, the MAC will transmit a pause frame with a zero pause time to cancel the pause request.

Note that transmitted pause frames are only a request to the other end station to stop transmitting. Frames that are received during the pause interval will be received normally (provided the RX FIFO is not full at which time the receive FIFO will overrun and CPSW\_STAT\_RX\_BOTTOM\_OF\_FIFO\_DROP\_k[31-0] COUNT value will increment).

Pause frames will be transmitted if enabled and triggered regardless of whether or not the port is observing the pause time period from an incoming pause frame.

The Ethernet port will transmit pause frames as described below:

- The 48-bit reserved multicast destination address 01.80.C2.00.00.01.
- The 48-bit source address - from SL\_SA[47-0] input.
- The 16-bit length/type field containing the value 88.08
- The 16-bit pause opcode equal to 00.01
- The 16-bit pause time value FF.FF. A pause-quantum is 512 bit-times. Pause frames sent to cancel a pause request will have a pause time value of 00.00.
- Zero padding to 64-byte data length (The Ethernet port will transmit only 64 byte pause frames).
- The 32-bit frame-check sequence (CRC word).

All quantities above are hexadecimal and are transmitted most-significant byte first. The least-significant bit is transferred first in each byte.

If CPSW\_PN\_MAC\_CONTROL\_REG\_k[3] RX\_FLOW\_EN is cleared to 0h while the pause time is nonzero, then the pause time will be cleared to 0h and a 0 count pause frame will be sent.

#### 12.2.2.4.6.13.2 Qbb (10/100/1G/10G) Receive Priority Based Flow Control (PFC)

IEEE 802.1Qbb based receive flow control provides a means of preventing frame reception when the port is operating in full-duplex mode (FULLDUPLEX bit must be set in the CPSW\_PN\_MAC\_CONTROL\_REG\_k register). When receive PFC (Priority based Flow Control) flow control is enabled and triggered for a priority, the port will transmit a PFC pause frame to request that the sending station stop transmitting on that priority (and perhaps others) for the period indicated within the transmitted pause frame (FFFFh pause time). When the triggering condition is removed, or when PFC flow control is disabled, the port will transmit a pause frame to cancel the pause request (00.00 pause time). Priority based pause frames can have one to eight priorities enabled as determined by the priority enable vector in the pause frame. Pause frames can give some priorities pause on and others pause off in the same frame. An enabled priority will have either a pause on value (FFFFh) or a pause off value (00.00). Priorities with a zero enable bit in the pause frame priority enable vector are unchanged by the pause frame on the sending station side. The [23-16] RX\_FLOW\_PRI value in the CPSW\_PN\_PRI\_CTL\_REG\_k register indicates which port receive priorities are enabled for PFC. The priority enable vector in the pause frame only refers to the priorities that have a valid pause on or pause off sent in the sent pause frame.

The Ethernet port will transmit a pause frame to the reserved multicast address at the first available opportunity (immediately if currently idle, or following the completion of the frame currently being transmitted). The pause frame will contain the maximum possible value for the pause time (FFFFh) on each priority to be paused along with a set priority enable vector bit. The MAC will count the receive pause frame time (decrements FF00h down to 0) and retransmit an outgoing pause frame if the count reaches zero. When the flow control trigger is removed for a specific priority, the MAC will transmit a pause frame with a set priority enable vector bit and a zero pause time on that priority to cancel the pause request.

Note that transmitted pause frames are only a request to the other end station to stop transmitting. Frames that are received during the pause interval will be received normally (provided the Rx FIFO is not full at which time the receive FIFO will overrun and CPSW\_STAT\_RX\_TOP\_OF\_FIFO\_DROP\_k[31-0] COUNT value will increment).

Pause frames will be transmitted if enabled and triggered regardless of whether or not the port is observing the pause time period from an incoming pause frame on the priority to be paused or not.

The Ethernet port will transmit pause frames as described below:

- The 48-bit reserved multicast destination address 01.80.C2.00.00.01.
- The 48-bit source address - from SL\_SA[47-0] input.
- The 16-bit length/type field containing the value 88.08
- The 16-bit pause opcode equal to 01.01
- The 16-bit priority enable vector. The lower 8-bits are used for priorities 7 down to 0.
- A 16-bit pause time value for each priority. The pause on value is FFFFh and the pause off value is 00.00. A pause-quantum is 512 bit-times.
- Zero padding to 64-byte data length (The Ethernet port will transmit only 64 byte pause frames).
- The 32-bit frame-check sequence (CRC word).

All quantities above are hexadecimal and are transmitted most-significant byte first. The least-significant bit is transferred first in each byte.

If RX\_FLOW\_EN bit in the CPSW\_PN\_MAC\_CONTROL\_REG\_k is cleared to 0h while the pause time is nonzero, then the pause time will be cleared to zero and a zero count pause frame will be sent. For any priority that has flow control enabled then associated priority in the CPSW\_PN\_TX\_BLKs\_PRI\_REG\_k register should be written with zero.

#### 12.2.2.4.6.13.3 Ethernet Transmit Flow Control

Incoming pause frames are acted upon, when enabled, to prevent the Ethernet port from transmitting any further frames. Incoming pause frames are only acted upon when the [0] FULLDUPLEX and [4] TX\_FLOW\_EN bits in the CPSW\_PN\_MAC\_CONTROL\_REG\_k register are set. Pause frames are not acted upon in half-duplex mode. Pause frame action will be taken if enabled, but normally the frame will be filtered and not transferred

to memory. MAC control frames will be transferred to memory if the [24] RX\_CMF\_EN (RX Copy MAC Control Frames Enable) bit in the CPSW\_PN\_MAC\_CONTROL\_REG\_k register is set. The [4] TX\_FLOW\_EN and [0] FULLDUPLEX bits effect whether or not MAC control frames are acted upon, but they have no effect upon whether or not MAC control frames are transferred to memory or filtered.

Pause frames are a subset of MAC Control Frames with an opcode field = 0001h. Incoming pause frames will only be acted upon by the port if:

- [4] TX\_FLOW\_EN is set in CPSW\_PN\_MAC\_CONTROL\_REG\_k register, and
- the RX maximum frame length is 64 bytes inclusive (CPSW\_PN\_RX\_MAXLEN\_REG\_k[13-0] RX\_MAXLEN), and
- the frame contains no CRC error or align/code errors.

The pause time value from valid frames will be extracted from the two bytes following the opcode. The pause time will be loaded into the port's transmit pause timer and the transmit pause time period will begin.

If a valid pause frame is received during the transmit pause time period of a previous transmit pause frame then:

- if the destination address is not equal to the reserved multicast address or any enabled or disabled unicast address, then the transmit pause timer will immediately expire, or
- if the new pause time value is zero then the transmit pause timer will immediately expire, else the port transmit pause timer will immediately be set to the new pause frame pause time value. (Any remaining pause time from the previous pause frame will be discarded).

If [4] TX\_FLOW\_EN in CPSW\_PN\_MAC\_CONTROL\_REG\_k register is cleared, then the pause-timer will immediately expire.

The port will not start the transmission of a new data frame any sooner than 512-bit times after a pause frame with a non-zero pause time has finished being received (MRXDV going inactive). No transmission will begin until the pause timer has expired (the port may transmit pause frames in order to initiate outgoing flow control). Any frame already in transmission when a pause frame is received will be completed and unaffected.

Incoming pause frames consist of the below:

- A 48-bit destination address equal to:
  - The reserved multicast destination address 01.80.C2.00.00.01, or the Ethernet port SL\_SA [47:0] input.
- The 48-bit source address of the transmitting device.
- The 16-bit length/type field containing the value 88.08
- The 16-bit pause opcode equal to 00.01
- The 16-bit CPSW\_PN\_MAC\_TX\_PAUSETIMER\_REG\_k[15-0] TX\_PAUSETIMER. A pause-quantum is 512 bit-times.
- Padding to 64-byte data length.
- The 32-bit frame-check sequence (CRC word).

All quantities above are hexadecimal and are transmitted most-significant byte first. The least-significant bit is transferred first in each byte.

The padding is required to make up the frame to a minimum of 64 Bytes. The standard allows pause frames longer than 64 Bytes to be discarded or interpreted as valid pause frames. The Ethernet port will recognize any pause frame between 64 Bytes and CPSW\_PN\_RX\_MAXLEN\_REG\_k[13-0] RX\_MAXLEN bytes in length.

#### 12.2.2.4.6.14 PFC Trigger Rules

There are five rules that can each set and clear a PFC (Priority based Flow Control) trigger condition for each enabled priority. The five rules are the destination based rule, the sum of outflows rule, the sum of blocks per port rule, the sum of blocks total rule, and the top of receive FIFO rule. Ethernet ports issue transmit PFC pause frames when any priority is triggered for any rule. Port 0 issues flow control on the associated receive thread when any priority is triggered for any rule. There are no pause frames sent on port 0 transmit (no port 0 outflow). Receive priority remapping is taken into account for all rules on all ports. Transmit priority remapping is taken into account for the sum of outflows rule, the sum of blocks per port rule, and the destination based rule.

Transmit priority remapping is not taken into account for the sum of blocks total rule or the top of receive FIFO rule. The intent of PFC is to shutdown lower priorities as traffic congestion increases so that the higher priorities can continue to operate. Systems with only 1518 byte packets may only need to configure the destination based rule.

#### 12.2.2.4.6.14.1 Destination Based Rule

The destination based rule is the primary rule for PFC and should be configured when PFC is enabled. The rule is evaluated for PFC trigger when the destination ports are determined on address lookup at packet reception (bottom of receive FIFO). The conditions of all packet destination port transmit FIFO's at packet reception determine whether or not one or more receive port priorities will be triggered. Each packet destination port is evaluated for each priority and the results are combined to produce the priority flow control trigger(s) on the receive port. A priority trigger is cleared at any time the clear condition is satisfied for that priority on all destination port FIFO's (all transmit FIFO's that set the trigger). The intent of the rule is to progressively trigger receive PFC with decreasing priority as the FIFO allocated blocks (CPSW\_PN\_BLK\_CNT\_REG\_k[12-8] TX\_BLK\_CNT) increases, and have the effect of priority outflow increase with decreasing priority. The add amount for outflow is zero if the outflow is not set and CPSW\_PN\_TX\_D\_OFLOW\_ADDVAL\_L\_REG\_k/ CPSW\_PN\_TX\_D\_OFLOW\_ADDVAL\_H\_REG\_k if the outflow is set for a particular priority.

A PFC priority is triggered by conditions in any destination transmit FIFO by the below equation at packet reception:

$$\text{priX\_trig\_set} = ((\text{addvalX}[4:0] \& \{5\{\text{tx\_d\_trig\_setX}\}\}) + \text{pn\_tx\_blk\_cnt}[4:0]) \geq \text{tx\_d\_thresh\_setX}[4:0]$$

A PFC priority is cleared by conditions in all destination transmit FIFO's by the below equation at any time after the set has occurred:

$$\text{priX\_trig\_clr} = ((\text{addvalX}[4:0] \& \{5\{\text{tx\_d\_trig\_setX}\}\}) + \text{pn\_tx\_blk\_cnt}[4:0]) \leq \text{tx\_d\_thresh\_clrX}[4:0]$$

Where,

- **X** is the priority
- **addvalX** is from CPSW\_PN\_TX\_D\_OFLOW\_ADDVAL\_L\_REG\_k and CPSW\_PN\_TX\_D\_OFLOW\_ADDVAL\_H\_REG\_k registers and has a maximum value of 11 decimal
- **tx\_d\_trig\_setX** is set when transmit flow control (outflow) is triggered on the destination TX FIFO priority (by receiving a pause frame with the associated priority paused).
- **pn\_tx\_blk\_cnt** is the total number of blocks allocated on all priorities in the destination transmit FIFO (0 to 20).
- **tx\_d\_thresh\_setX** is from CPSW\_PN\_TX\_D\_THRESH\_SET\_L\_REG\_k and CPSW\_PN\_TX\_D\_THRESH\_SET\_H\_REG\_k registers
- **tx\_d\_thresh\_clrX** is from CPSW\_PN\_TX\_D\_THRESH\_CLR\_L\_REG\_k and CPSW\_PN\_TX\_D\_THRESH\_CLR\_H\_REG\_k registers
- **tx\_d\_thresh\_setX** must be greater than **tx\_d\_thresh\_clrX**

#### 12.2.2.4.6.14.2 Sum of Outflows Rule

The Sum of Outflows rule is a global control that sets and clears PFC triggers on a receive port due to conditions in all other Ethernet port transmit FIFOs regardless of receive traffic. An outflow is set for a transmit FIFO priority when a PFC pause frame is received on the port with a non-zero pause quanta for that priority indicating that the port should not transmit on that priority. An outflow is clear for a priority when the transmit pause condition is removed either due to receiving a pause frame with a zero pause quanta or by the pause timer expiring for the priority. Port 0 does not have transmit flow control. The sum of outflows rule is intended to be used in some systems to pass along outflow on lower priorities with larger packet sizes. Triggering PFC on priorities due to this rule also prevents packet transmission from Ethernet ports to the host for the duration of the pause time on the triggered priorities. When configured to trigger due to this rule, pause frames will be sent on Ethernet ports regardless of receive traffic.

A PFC priority is triggered by conditions in all other Ethernet transmit FIFOs by the below equation at any time:

$$\text{priX\_trig\_set} = \text{sum of all other Ethernet port priX\_outflows} \geq \text{tx\_g\_oflow\_thresh\_setX}[4:0]$$

A PFC priority is cleared by conditions in all other Ethernet transmit FIFO's by the below equation at any time:



**priX\_trig\_clr = sum of all other Ethernet port priX\_outflows <= tx\_g\_oflow\_thresh\_clrX[4:0]**

Where,

- **X** is the priority
- **tx\_g\_oflow\_thresh\_setX** is from CPSW\_TX\_G\_OFLOW\_THRESH\_SET\_REG
- **tx\_g\_oflow\_thresh\_clrX** is from CPSW\_TX\_G\_OFLOW\_THRESH\_CLR\_REG
- **tx\_g\_oflow\_thresh\_setX** must be greater than **tx\_g\_oflow\_thresh\_clrX**.

#### 12.2.2.4.6.14.3 Sum of Blocks Per Port Rule

The Sum of Blocks Per Port rule is an aggressive global control that sets and clears PFC triggers on receive ports due to conditions in any other port transmit FIFO regardless of receive traffic. When any transmit FIFO has a trigger condition set, all other receive ports will trigger PFC for that priority. This rule is intended to be configured in systems with high traffic rate jumbo packets on lower priorities.

A PFC priority is triggered by conditions in any other transmit FIFO by the below equation at any time:

**priX\_trig\_set when the sum of all Tx FIFO blocks allocated on the port >= thresh\_setX[4:0]**

A PFC priority is cleared when conditions in all other Ethernet transmit FIFO's satisfy the below equation at any time:

**priX\_trig\_clr when the sum of all tx FIFO blocks allocated on the port <= thresh\_clrX[4:0]**

Where,

- **X** is the priority
- **thresh\_setX** is from CPSW\_PN\_TX\_G\_BUF\_THRESH\_SET\_L\_REG\_k/  
CPSW\_PN\_TX\_G\_BUF\_THRESH\_SET\_H\_REG\_k (or CPPI port0  
CPSW\_P0\_TX\_G\_BUF\_THRESH\_SET\_L\_REG/CPSW\_P0\_TX\_G\_BUF\_THRESH\_SET\_H\_REG)
- **thresh\_clrX** is from CPSW\_PN\_TX\_G\_BUF\_THRESH\_CLR\_L\_REG\_k/  
CPSW\_PN\_TX\_G\_BUF\_THRESH\_CLR\_H\_REG\_k (or CPPI port0  
CPSW\_P0\_TX\_G\_BUF\_THRESH\_CLR\_L\_REG/CPSW\_P0\_TX\_G\_BUF\_THRESH\_CLR\_H\_REG)
- **thresh\_setX** must be greater than **thresh\_clrX** for each priority.

#### 12.2.2.4.6.14.4 Sum of Blocks Total Rule

The Sum of Blocks Total rule is an extremely aggressive global control that sets and clears PFC triggers on receive ports due to conditions in all transmit FIFOs regardless of receive traffic. When any priority has a trigger condition set, all receive ports will trigger PFC for that priority. The intent of this rule is to shutdown lower priorities with jumbo packets as the total switch congestion increases.

A PFC priority is triggered by conditions in all transmit FIFO by the below equation at any time:

**priX\_trig\_set = sum of all tx FIFO blocks allocated in all ports >= thresh\_setX[4:0]**

A PFC priority is cleared when conditions in all transmit FIFO's satisfy the below equation at any time:

**priX\_trig\_clr = sum of all tx FIFO blocks allocated in all ports <= thresh\_clrX[4:0]**

Where,

- **X** is the priority
- **thresh\_setX** is from CPSW\_TX\_G\_BUF\_THRESH\_SET\_L\_REG/  
CPSW\_TX\_G\_BUF\_THRESH\_SET\_H\_REG
- **thresh\_clrX** is from CPSW\_TX\_G\_BUF\_THRESH\_CLR\_L\_REG/  
CPSW\_TX\_G\_BUF\_THRESH\_CLR\_H\_REG
- **thresh\_setX** must be greater than **thresh\_clrX** for each priority.

#### 12.2.2.4.6.14.5 Top of Receive FIFO Rule

The Top of Receive FIFO Rule sets PFC triggers on the associated receive port for all priorities. This rule requires no configuration and is enabled when PFC flow control is enabled. The Top of Receive FIFO rule triggers when a received packet cannot be transferred to an intended destination port due to insufficient space in that destination port transmit FIFO. The receive FIFO exists to accept flow control runout (especially for 802.3 flow control). The trigger condition for this rule is cleared when the receive FIFO is emptied after runout.

Preferable PFC configurations preclude this rule from triggering. If a lower priority packet triggers this rule then higher priority packets could be in the runout after the lower priority packet(s), which would increase the switch latency of the higher priority packets. This rule ensures that there is no packet loss on any priority in any configuration. However, PFC configuration should ensure that this rule is triggered only by high priority packets or preferably not at all. The [27] TORF bit in the CPSW\_PN\_MAC\_STATUS\_REG\_k register indicates that this rule has triggered. The CPSW\_PN\_MAC\_STATUS\_REG\_k[26-24] TORF\_PRI field indicates the lowest priority that has triggered this rule since the last clearing of the status register fields. More aggressive shutdown of lower priorities in the other rules is preferable to having this rule trigger.

#### 12.2.2.4.6.15 Energy Efficient Ethernet Support (802.3az)

Energy Efficient Ethernet (EEE) allows the LPSC to turn off the module clock during inactive periods as determined by network and host traffic. The module can then be awakened by host queued transmit packet(s) or by a port's external Ethernet PHY. The module EEE clock stop interface is used by the external controller to control module EEE operations. EEE operations are configured as shown below:

1. The 12-bit EEE clock pre-scale value is written to the CPSW\_EEE\_PRESCALE\_REG register. The pre-scaler is used to clock all EEE-related counters
2. The port Idle to LPI count values (CPSW\_PN\_IDLE2LPI\_REG\_k[23-0] COUNT) are written with the desired values
3. The port LPI to Wake count values (CPSW\_PN\_LPI2WAKE\_REG\_k[23-0] COUNT) are written with the desired values
4. The [0] EEE\_EN bit is set in the switch CPSW\_SS\_CONTROL\_REG register

EEE operation can begin after configuration. The host allows (through LPSC) the CPSW to enter a low power state by asserting the EEE\_CLKSTOP\_REQ signal. There are no requirements on host queues or traffic in order for the host to assert or de-assert EEE\_CLKSTOP\_REQ to the CPSW.

Each Ethernet port has a transmit and a receive LPI (low power indicate) state. The PHY indicates LPI by asserting MRXER with a MRXD[7:0] value of 0x01 while MRXDV is deasserted (inter-packet gap). The Ethernet transmit port indicates LPI after the CPSW\_PN\_IDLE2LPI\_REG\_k value has been counted (the transmit port has gone idle for the configured amount of time). If another packet is received for transmit during the count then the count is restarted. When the transmit port has been idle for the Idle to LPI time, the transmit port enters the LPI state and indicates LPI to the associated PHY. The LPI is indicated to the external PHY by an asserted MTXER with a MTXD[7:0] while MTXEN is deasserted (inter-packet gap). The CPPI (port 0) LPI state includes transmit and receive. The CPPI LPI state is entered when the CPPI transmit and receive have both been idle for the Idle to LPI time (CPSW\_P0\_IDLE2LPI\_REG). The Idle to LPI time value for all ports must be large relative to the switch latency to ensure that the count is not able to complete between successive packets.

#### Note

External PHY signaling has the following conditions:

- RGMII is a DDR interface. TXEN and TXER are the sampled values of TX\_CTL at the rising and the falling TXC edges, respectively. RXDV and RXER are the sampled values of RX\_CTL at the rising and the falling RXC clock edges, respectively
- In RMII mode, EEE is not supported.

When all transmit and receive ports are in the LPI state (CPSW LPI state), the EEE\_CLKSTOP\_ACK signal is asserted, and the LPSC is allowed to stop the module clock. When EEE\_CLKSTOP\_ACK is asserted, the clock may be turned on and off as desired by the host. The host is allowed to restart the clock, perform slave read/write operations to the CPSW memory address space, and then turn off the clock again while EEE\_CLKSTOP\_ACK is asserted.

The software can remove and disable from re-entering the CPSW LPI state by restarting the module clock and then de-asserting EEE\_CLKSTOP\_REQ. There must be at least one rising edge of the clock before EEE\_CLKSTOP\_REQ is de-asserted. The module EEE\_CLKSTOP\_ACK output signal will be deasserted on the clock after the de-assertion of EEE\_CLKSTOP\_REQ. The host may queue CPPI receive packets at any time

without regard to the CPSW module LPI state. The Host must deassert `EEE_CLKSTOP_REQ` on wakeup for a minimum of two clock periods. If `EEE_CLKSTOP_REQ` is deasserted for less than 5 clock periods for a wakeup event from the host to a particular Ethernet port (or visa versa), then the wakeup event will not cause the other Ethernet port port to awaken.

The external Ethernet PHY's can also wakeup the LPSC by removing the Ethernet receive LPI indication. If the CPSW module is in Idle state with `EEE_CLKSTOP_ACK` asserted and the receive LPI indication is removed, the `EEE_CLKSTOP_WAKEUP` signal will be asynchronously asserted. On wakeup, the LPSC restarts the clock and de-assert the `EEE_CLKSTOP_REQ` signal. The `EEE_CLKSTOP_WAKEUP` signal will be synchronously deasserted with `EEE_CLKSTOP_ACK`. Upon the deassertion of `EEE_CLKSTOP_REQ`, the Ethernet ports will count the `CPSW_PN_LPI2WAKE_REG_k` time for each port at which time the port is available for transmit.

#### 12.2.2.4.6.16 Ethernet Switch Latency

When CPSW is configured as a store and forward switch, the switch latency is defined as the amount of time between the end of packet reception of the received packet to the start of the output packet transmit.

The store and forward latency is shown in [Table 12-224](#):

**Table 12-224. Switch Latency**

Mode	Latency
Gig (1000)	880 ns
100	1.3 $\mu$ s
10	6.5 $\mu$ s

#### 12.2.2.4.6.17 MAC Emulation Control

The emulation control input (EMUSUSP) and submodule emulation control registers allow CPSW operation to be completely or partially suspended. Each Ethernet port has associated emulation control registers (`CPSW_EM_CONTROL_REG` and `CPSW_PN_MAC_EMCONTROL_REG_k`). The submodule emulation control registers must be accessed to facilitate CPSW emulation control. The CPSW module enters the emulation suspend state if all three submodules are configured for emulation suspend and the emulation suspend input is asserted. A partial emulation suspend state is entered if one or two submodules is configured for emulation suspend and the emulation suspend input is asserted. Emulation suspend occurs at packet boundaries. The emulation control feature is implemented for compatibility with other peripherals.

#### Ethernet port Emulation Control

The emulation control input (TBEMUSUP) and register bits (SOFT and FREE bits in the `CPSW_PN_MAC_EMCONTROL_REG_k` register) allow Ethernet port operation to be suspended. When the emulation suspend state is entered, the Ethernet port will stop processing receive and transmit frames at the next frame boundary. Any frame currently in reception or transmission will be completed normally without suspension. For receive, frames that are detected by the Ethernet port after the suspend state is entered are ignored.

[Table 12-225](#) shows the operations of the emulation control input and register bits.

**Table 12-225. Emulation Control Input**

EMUSUSP	SOFT	FREE	Description
0	X	X	Normal Operation
1	0	0	Normal Operation
1	1	0	Emulation Suspend
1	X	1	Normal Operation

#### 12.2.2.4.6.18 MAC Command IDLE

The `CMD_IDLE` bit in the `CPSW_PN_MAC_CONTROL_REG_k` register allows MAC operation to be suspended. When the idle state is commanded, the MAC will stop processing receive and transmit frames



at the next frame boundary. Any frame currently in reception or transmission will be completed normally without suspension. For transmission, any complete or partial frame in the TX cell FIFO will be transmitted. For receive, frames that are detected by the MAC after the suspend state is entered are ignored. No statistics will be kept for ignored frames. Commanded idle is similar in operation to emulation control and clock stop.

#### 12.2.2.4.6.19 CPSW Network Statistics

The CPSW has a set of statistics that record events associated with frame traffic on selected switch ports. The statistics values are cleared to zero 38 clocks after the rising edge of CPSW0\_RST. When one or more port enable (Pn\_STAT\_EN) bits in the CPSW\_STAT\_PORT\_EN\_REG register are set, all statistics registers are write to decrement. The value written will be subtracted from the register value with the result being stored in the register. If a value greater than the statistics value is written, then zero will be written to the register (writing 0xFFFF FFFF clears a statistics location). When all port enable bits are cleared to zero, all statistics registers are read/write (normal write direct, so writing 0x0000 0000 clears a statistics location). All write accesses must be 32-bit accesses.

The statistics interrupt (STAT\_PEND0) will be issued if enabled when any statistics value is greater than or equal to 0x8000 0000. The statistics interrupt is removed by writing to decrement any statistics value greater than 0x8000 0000. The statistics are mapped into internal memory space and are 32-bits wide. All statistics rollover from 0xFFFF FFFF to 0x0000 0000.

[Table 12-226](#) and [Table 12-227](#) summarize network statistics.

#### 12.2.2.4.6.19.1 Rx-only Statistics Descriptions

##### 12.2.2.4.6.19.1.1 Good Rx Frames (Offset = 3A000h)

#### All ports

The total number of good frames received on the port. A good frame is defined to be:

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Had a length of 64 to RX\_MAXLEN bytes inclusive
- Had no CRC error, alignment error or code error.

See the [Section 12.2.2.4.6.19.1.6, Rx Align/Code Errors](#) and [Section 12.2.2.4.6.19.1.5, Rx CRC errors](#) statistic descriptions for definitions of alignment, code and CRC errors.

Overruns have no effect upon this statistic.

##### 12.2.2.4.6.19.1.2 Broadcast Rx Frames (Offset = 3A004h)

#### All ports

The total number of good broadcast frames received on the port. A good broadcast frame is defined to be:

- Any data or MAC control frame which was destined for address FF.FF.FF.FF.FF.FF
- Had a length of 64 to RX\_MAXLEN bytes inclusive
- Had no CRC error, alignment error or code error.

See the [Section 12.2.2.4.6.19.1.6, Rx Align/Code Errors](#) and [Section 12.2.2.4.6.19.1.5, Rx CRC errors](#) statistic descriptions for definitions of alignment, code and CRC errors.

Overruns have no effect upon this statistic.

##### 12.2.2.4.6.19.1.3 Multicast Rx Frames (Offset = 3A008h)

#### All ports

The total number of good multicast frames received on the port. A good multicast frame is defined to be:

- Any data or MAC control frame which was destined for any multicast address other than FF.FF.FF.FF.FF.FF
- Had a length of 64 to RX\_MAXLEN bytes inclusive
- Had no CRC error, alignment error or code error

See the [Section 12.2.2.4.6.19.1.6, Rx Align/Code Errors](#) and [Section 12.2.2.4.6.19.1.5, Rx CRC errors](#) statistic descriptions for definitions of alignment, code and CRC errors.

Overruns have no effect upon this statistic.

#### **12.2.2.4.6.19.1.4 Pause Rx Frames (Offset = 3A00Ch)**

##### **Ethernet port N (where N = 1 to 8)**

The total number of IEEE 802.3X pause frames received by the port (whether acted upon or not). Such a frame:

- Contained any unicast, broadcast, or multicast address
- Contained the length/type field value 88.08 (hex) and the opcode 0x0001
- Was of length 64 to RX\_MAXLEN bytes inclusive
- Had no CRC error, alignment error or code error
- Pause-frames had been enabled on the port (TX\_FLOW\_EN = 1h).

The port could have been in either half or full-duplex mode.

See the [Section 12.2.2.4.6.19.1.6, Rx Align/Code Errors](#) and [Section 12.2.2.4.6.19.1.5, Rx CRC errors](#) statistic descriptions for definitions of alignment, code and CRC errors.

Overruns have no effect upon this statistic.

#### **12.2.2.4.6.19.1.5 Rx CRC Errors (Offset = 3A010h)**

##### **All ports**

The total number of frames received on the port that experienced a CRC error. Such a frame:

- Was any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Was of length 64 to RX\_MAXLEN bytes inclusive
- Had no code/align error
- Had a CRC error

Overruns have no effect upon this statistic.

A CRC error is defined to be:

- A frame containing an even number of nibbles, and
- Failing the Frame Check Sequence test.

#### **12.2.2.4.6.19.1.6 Rx Align/Code Errors (Offset = 3A014h)**

##### **Ethernet port N (where N = 1 to 8)**

The total number of frames received on the port that experienced an alignment error or code error. Such a frame:

- Was any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Was of length 64 to RX\_MAXLEN bytes inclusive
- Had either an alignment error, or a code error.

Overruns have no effect upon this statistic.

An alignment error is defined to be:

- A frame containing an odd number of nibbles
- Failing the Frame Check Sequence test if the final nibble is ignored

A code error is defined to be a frame which has been discarded because the port's MRXER pin driven with a one for at least one bit-time's duration at any point during the frame's reception.

### Note

RFC 1757 etherStatsCRCAAlignErrors Ref. 1.5 can be calculated by summing Rx Align/Code Errors and Rx CRC errors.

#### 12.2.2.4.6.19.1.7 Oversize Rx Frames (Offset = 3A018h)

##### All ports

The total number of oversized frames received on the port. An oversized frame is defined to be:

- Was any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Was greater than RX\_MAXLEN in bytes
- Had no CRC error, alignment error, or code error.

See the [Section 12.2.2.4.6.19.1.6, Rx Align/Code Errors](#) and [Section 12.2.2.4.6.19.1.5, Rx CRC errors](#) statistic descriptions for definitions of alignment, code and CRC errors.

Overruns have no effect upon this statistic.

#### 12.2.2.4.6.19.1.8 Rx Jabbers (Offset = 3A01Ch)

##### All ports

The total number of jabber frames received on the port. A jabber frame:

- Was any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Was greater than RX\_MAXLEN in bytes
- Had no CRC error, alignment error or code error

See the [Section 12.2.2.4.6.19.1.6, Rx Align/Code Errors](#) and [Section 12.2.2.4.6.19.1.5, Rx CRC errors](#) statistic descriptions for definitions of alignment, code and CRC errors.

Overruns have no effect upon this statistic.

#### 12.2.2.4.6.19.1.9 Undersize (Short) Rx Frames (Offset = 3A020h)

##### All ports

The total number of undersized frames received on the port. An undersized frame is defined to be:

- Was any data frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Was less than 64 bytes
- Had no CRC error, alignment error, or code error

See the [Section 12.2.2.4.6.19.1.6, Rx Align/Code Errors](#) and [Section 12.2.2.4.6.19.1.5, Rx CRC errors](#) statistic descriptions for definitions of alignment, code and CRC errors.

Overruns have no effect upon this statistic.

#### 12.2.2.4.6.19.1.10 Rx Fragments (Offset = 3A024h)

##### Ethernet port N (where N = 1 to 8)

The total number of frame fragments received on the port. A frame fragment is defined to be:

- Any data frame (address matching does not matter)
- Less than 64 bytes long
- Having a CRC error, an alignment error, or a code error
- Not the result of a collision caused by half-duplex, collision-based flow control

See the [Section 12.2.2.4.6.19.1.6, Rx Align/Code Errors](#) and [Section 12.2.2.4.6.19.1.5, Rx CRC errors](#) statistic descriptions for definitions of alignment, code and CRC errors.

Overruns have no effect upon this statistic.

#### **12.2.2.4.6.19.1.11 RX IPG Error**

The total number of 10G frames received on a port that had a correct preamble but did not have at least five bytes of IDLE preceding the frame. This does not indicate if the frame with the IPG error was kept or ignored.

#### **12.2.2.4.6.19.1.12 ALE Drop (Offset = 3A028h)**

##### **All ports**

The total number of frames received on a port such that the destination address was not equal to the source address and the packet was not destined to the port it was received on, but the frame was not forwarded to any port (the PORT\_MASK was zero).

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Any length (including less than 64 bytes and greater than RX\_MAXLEN bytes)
- had no CRC error, alignment error, or code error
- the destination address was not equal to the source address
- the packet was not destined for the port it was receive on
- had a zero PORT\_MASK

#### **12.2.2.4.6.19.1.13 ALE Overrun Drop (Offset = 3A02Ch)**

##### **All ports**

The total number of frames received on a port that were dropped (zero PORT\_MASK) due to exceeding the maximum ALE lookup rate (Port 0 should not have ALE Overrun Drops because the ingress rate is controlled to prevent it). This statistic should be zero and when non-zero indicates a system clock issue or indicates that short packets were sent with RX\_CSF\_EN at a rate that exceeded the maximum lookup rate.

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Any length (including less than 64 bytes and greater than RX\_MAXLEN bytes)
- the maximum ALE lookup rate was exceeded so the lookup was aborted and the packet was dropped.

#### **12.2.2.4.6.19.1.14 Rx Octets (Offset = 3A030h)**

##### **All ports**

The total number of bytes in all good frames received on the port. A good frame is defined to be:

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Of length 64 to RX\_MAXLEN bytes inclusive
- Had no CRC error, alignment error or code error

See the [Section 12.2.2.4.6.19.1.6, Rx Align/Code Errors](#) and [Section 12.2.2.4.6.19.1.5, Rx CRC errors](#) statistic descriptions for definitions of alignment, code and CRC errors.

Overruns have no effect upon this statistic.

#### **12.2.2.4.6.19.1.15 Rx Bottom of FIFO Drop (Offset = 3A084h)**

##### **Ethernet port N (where N = 1 to 8)**

The total number of frames received on a port that overran the port's receive FIFO and were dropped (bottom of receive FIFO). Port 0 (CPPI receive port) should not drop packets on receive because port 0 receive flow control should be enabled. The Ethernet ports will only drop packets in the receive FIFO when receive flow control is

enabled and the sending port ignores sent pause frame and then overruns the receive FIFO. An overrun frame is defined to be:

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Any length (including less than 64 bytes and greater than RX\_MAXLEN bytes)
- Was dropped on port 0 due to a lack of memory space in the receive FIFO.

---

#### Note

This statistic should be zero if proper flow control is being followed.

---

### Host port 0

This statistic also counts frames dropped on port 0 that were 17 to 63 bytes (only for port 0). For Ethernet ports, the drop count for frames shorter than 33 bytes is included in the undersized or fragment count. Port 0 simply gives an indication that a packet was dropped. No other statistics are counted for frames shorter than 33 bytes.

#### 12.2.2.4.6.19.1.16 Portmask Drop (Offset = 3A088h)

##### All ports

The total number of frames received on a port that were dropped by the ALE (the ALE did not forward the packet to any port). Port mask drop frame is defined to be:

- Any data or MAC control frame
- Any length greater than 32 bytes
- Was dropped by the ALE due to PORT\_MASK=0 (was not sent to any destination port)
- The frame could have been dropped due to error or other counted reason, so it could be counted elsewhere also.

---

#### Note

This statistic does not count in the overall total as it includes every packet received greater than 32 bytes that had a zero PORT\_MASK.

---

#### 12.2.2.4.6.19.1.17 Rx Top of FIFO Drop (Offset = 3A08Ch)

##### All ports

The total number of frames received on a port that had a start-of-frame (SOF) overrun on any destination port egress (when attempting to load the packet from the top of the ingress port receive FIFO into any other port's transmit FIFO). If a multicast/broadcast packet is dropped by multiple destination ports then this statistic will increment by the number of ports that dropped the packet. Rx Top Of FIFO Drop is defined to be:

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Any length (including less than 64 bytes and greater than RX\_MAXLEN bytes)
- had no CRC error, alignment error or code error
- had a SOF of frame overrun on another port egress.

#### 12.2.2.4.6.19.1.18 ALE Rate Limit Drop (Offset = 3A090h)

##### All ports

The total number of frames received on a port that were dropped (zero PORT\_MASK) due to receive rate limiting on this port or due to transmit rate limiting on any destination port (not sent to all expected destination ports if transmit rate limiting).

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Any length (including less than 64 bytes and greater than RX\_MAXLEN bytes)

- had no CRC error, alignment error, or code error
- the receive rate was exceeded and the packet was dropped, or the transmit rate was exceeded to any destination port and the packet was dropped to one or more expected destination ports (indicates that the destinations were reduced due to rate limiting).

#### **12.2.2.4.6.19.1.19 ALE VLAN Ingress Check Drop (Offset = 3A094h)**

##### **All ports**

The total number of frames received on a port that were dropped (zero PORT\_MASK) due to VLAN ingress check failure.

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Any length (including less than 64 bytes and greater than RX\_MAXLEN bytes)
- had no CRC error, alignment error, or code error
- the VLAN ID ingress check failed (the receive port was not in the group)
- The address lookup did not return a match with the SUPER bit set.

#### **12.2.2.4.6.19.1.19.1 ALE DA=SA Drop (Offset = 3A098h)**

##### **All ports**

The total number of frames received on a port that were dropped (zero PORT\_MASK) due to destination address equal to source address.

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Any length (including less than 64 bytes and greater than RX\_MAXLEN bytes)
- had no CRC error, alignment error, or code error
- the destination address was equal to the source address
- the source address was not an entry in the table.

#### **12.2.2.4.6.19.1.19.2 Block Address Drop (Offset = 3A09Ch)**

The total number of frames received on a port that were dropped (zero PORT\_MASK) due to the destination or source address being blocked.

- was any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode, and
- Any length (including less than 64 bytes and greater than RX\_MAXLEN bytes)
- had no CRC error, alignment error, or code error, and
- the source or destination address matched a table entry with the block bit set.

#### **12.2.2.4.6.19.1.19.3 ALE Secure Drop (Offset = 3A0A0h)**

The total number of frames received on a port that were dropped (zero port\_mask) due to a secure violation (the source address is owned by a different receive port).

- was any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode, and
- Any length (including less than 64 bytes and greater than RX\_MAXLEN bytes)
- had no CRC error, alignment error, or code error, and
- the source address is an entry in the table with the SECURE bit set and a port number for a different receive port.

#### **12.2.2.4.6.19.1.19.4 ALE Authentication Drop (Offset = 3A0A4h)**

The total number of frames received on a port that were dropped (zero port\_mask) due to authentication failure.

- was any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode, and
- Any length (including less than 64 bytes and greater than RX\_MAXLEN bytes)
- had no CRC error, alignment error, or code error, and

- CPSW\_ALE\_CONTROL[1] ENABLE\_AUTH\_MODE is set to 1h, and
- the source address is not equal to the destination address, and
- the source address is not a table entry, and
- the destination address is not a table entry with the SUPER bit set.

#### **12.2.2.4.6.19.1.19.5 ALE Unknown Unicast (Offset = 3A0A8h)**

##### **All ports**

The total number of frames received on a port that had a unicast destination address with an unknown source address.

- was any data frame with a unicast destination address
- the source address was not a table entry
- was of length 64 to RX\_MAXLEN bytes inclusive
- had no CRC error, alignment error, or code error

Note: The ALE Unknown Unicast Bytecount statistic is the number of bytes contained in the ALE Unknown Unicast frames.

#### **12.2.2.4.6.19.1.19.6 ALE Unknown Unicast Bytecount (Offset = 3A0ACh)**

The total number of bytes received on a port that had a unicast destination address with an unknown source address.

#### **12.2.2.4.6.19.1.19.7 ALE Unknown Multicast (Offset = 3A0B0h)**

The total number of frames received on a port that had a multicast destination address with an unknown source address. The frame is defined to be:

- was any data frame with a unicast destination address
- the source address was not a table entry, and
- was of length 64 to RX\_MAXLEN bytes inclusive
- had no CRC error, alignment error or code error

Note: The ALE Unknown Multicast Bytecount statistic is the number of bytes contained in the ALE Unknown Multicast frames.

#### **12.2.2.4.6.19.1.19.8 ALE Unknown Multicast Bytecount (Offset = 3A0B4h)**

The total number of bytes received on a port that had a multicast destination address with an unknown source address.

#### **12.2.2.4.6.19.1.19.9 ALE Unknown Broadcast (Offset = 3A0B8h)**

The total number of frames received on a port that had a broadcast destination address with an unknown source address. The frame is defined to be:

- was any data frame with a unicast destination address
- the source address was not a table entry, and
- was of length 64 to RX\_MAXLEN bytes inclusive
- had no CRC error, alignment error or code error

Note: The ALE Unknown Broadcast Bytecount statistic is the number of bytes contained in the ALE Unknown Broadcast frames.

#### **12.2.2.4.6.19.1.19.10 ALE Unknown Broadcast Bytecount (Offset = 3A0BCh)**

The total number of bytes received on a port that had a broadcast destination address with an unknown source address.

#### **12.2.2.4.6.19.1.19.11 ALE Policier/Classifier Match (Offset = 3A0C0h)**

##### **All ports**

The total number of frames received on a port that matched a policier. The frame is defined to be:



- was any data frame
- matched a condition on a policer,
- was of length 64 to RX\_MAXLEN bytes inclusive
- had no CRC error, alignment error, or code error

#### **12.2.2.4.6.19.2 ALE Policer Match Red (Offset = 3A0C4h)**

The total number of frames received on a port that had matched a policer and the condition was red. The frame is defined to be:

- was any data frame
- matched a condition on a policer,
- was of length 64 to RX\_MAXLEN bytes inclusive
- had no CRC error, alignment error, or code error

#### **12.2.2.4.6.19.3 ALE Policer Match Yellow (Offset = 3A0C8h)**

The total number of frames received on a port that had matched a policer and the condition was red. The frame is defined to be:

- was any data frame
- matched a condition on a policer,
- was of length 64 to RX\_MAXLEN bytes inclusive
- had no CRC error, alignment error, or code error

#### **12.2.2.4.6.19.4 IET Receive Assembly Error (Offset = 3A140h)**

The total number of preemptable received frames with IET assembly errors.

- any frame received
- was any size, and
- was a non-initial fragment that mismatched the frame count or fragment count (went to the assembly error state in the IET receive state machine)

#### **12.2.2.4.6.19.5 IET Receive Assembly OK (Offset = 3A144h)**

The total number of correctly received and re-assembled preemptable frames.

- any preemptable frame received
- was any size, and
- was correctly received and re-assembled without error

#### **12.2.2.4.6.19.6 IET Receive SMD Error (Offset = 3A148h)**

The total number of received frames rejected due to an unknown SMD value or received frames rejected with an SMD-C when no frame is in progress.

- any frame received
- was any size, and
- was rejected because of an unknown SMD value or SMD-C with no frame in progress.

Note: If IET\_ENABLE is not set, this statistic counts any received frame with any non express SMD.

#### **12.2.2.4.6.19.7 IET Receive Merge Fragment Count (Offset = 3A14Ch)**

The total number of received non-initial fragments that did not have an assembly error. The IET stat CPSW\_STAT\_RXFRAGMENTS\_k is derived by adding the Receive Assembly Error count to this value.

- any frame received
- was any size, and
- was a non-initial fragment that did not contain an assembly error

#### **12.2.2.4.6.19.8 Tx-only Statistics Descriptions**

The maximum and minimum transmit frame size is software controllable.



#### 12.2.2.4.6.19.8.1 Good Tx Frames (Offset = 3A034h)

##### All ports

The total number of good frames received on the port. A good frame is defined to be:

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Any length
- Had no late or excessive collisions, no carrier loss and no underrun

#### 12.2.2.4.6.19.8.2 Broadcast Tx Frames (Offset = 3A038h)

##### All ports

The total number of good broadcast frames transmitted on the port. A good broadcast frame is defined to be:

- Any data or MAC control frame which was destined for only address FF.FF.FF.FF.FF.FF
- Any length
- Had no late or excessive collisions, no carrier loss and no underrun

#### 12.2.2.4.6.19.8.3 Multicast Tx Frames (Offset = 3A03Ch)

##### All ports

The total number of good multicast frames transmitted on the port. A good multicast frame is defined to be:

- Any data or MAC control frame which was destined for any multicast address other than FF.FF.FF.FF.FF.FF
- Any length
- Had no late or excessive collisions, no carrier loss and no underrun

#### 12.2.2.4.6.19.8.4 Pause Tx Frames (Offset = 3A040h)

##### Ethernet port N (where N = 1 to 8)

This statistic indicates the number of IEEE 802.3X pause frames transmitted by the port.

Pause frames cannot contain a CRC error because they are created in the transmitting MAC, so these error conditions have no effect upon the statistic. Pause frames sent by software will not be included in this count.

Since pause frames are only transmitted in full duplex, carrier loss and collisions have no effect upon this statistic.

Transmitted pause frames are always 64-byte multicast frames so will appear in the *Tx Multicast Frames* and *64octet Frames* statistics.

#### 12.2.2.4.6.19.8.5 Deferred Tx Frames (Offset = 3A044h)

##### Ethernet port N (where N = 1 to 8)

The total number of frames transmitted on the port that first experienced deferment. Such a frame:

- Was any data or MAC control frame destined for any unicast, broadcast or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced no collisions before being successfully transmitted
- Found the medium busy when transmission was first attempted, so had to wait.

CRC errors have no effect upon this statistic.

#### 12.2.2.4.6.19.8.6 Collisions (Offset = 3A048h)

##### Ethernet port N (where N = 1 to 8)

This statistic records the total number of times that the port experienced a collision. Collisions occur under two circumstances.

1. When a transmit data or MAC control frame:

- Was destined for any unicast, broadcast or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced a collision. A jam sequence is sent for every non-late collision, so this statistic will increment on each occasion if a frame experiences multiple collisions (and increments on late collisions)

CRC errors have no effect upon this statistic.

2. When the port is in half-duplex mode, flow control is active, and a frame reception begins.

#### **12.2.2.4.6.19.8.7 Single Collision Tx Frames (Offset = 3A04Ch)**

##### **Ethernet port N (where N = 1 to 8)**

The total number of frames transmitted on the port that experienced exactly one collision. Such a frame:

- Was any data or MAC control frame destined for any unicast, broadcast or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced one collision before successful transmission. The collision was not late.

CRC errors have no effect upon this statistic.

#### **12.2.2.4.6.19.8.8 Multiple Collision Tx Frames (Offset = 3A050h)**

##### **Ethernet port N (where N = 1 to 8)**

The total number of frames transmitted on the port that experienced multiple collisions. Such a frame:

- Was any data or MAC control frame destined for any unicast, broadcast or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced 2 to 15 collisions before being successfully transmitted. None of the collisions were late.

CRC errors have no effect upon this statistic.

#### **12.2.2.4.6.19.8.9 Excessive Collisions (Offset = 3A054h)**

##### **Ethernet port N (where N = 1 to 8)**

The total number of frames for which transmission was abandoned due to excessive collisions. Such a frame:

- Was any data or MAC control frame destined for any unicast, broadcast or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced 16 collisions before abandoning all attempts at transmitting the frame. None of the collisions were late.

CRC errors have no effect upon this statistic.

#### **12.2.2.4.6.19.8.10 Late Collisions (Offset = 3A058h)**

##### **Ethernet port N (where N = 1 to 8)**

The total number of frames on the port for which transmission was abandoned because they experienced a late collision. Such a frame:

- Was any data or MAC control frame destined for any unicast, broadcast or multicast address
- Was any size
- Experienced a collision later than 512 bit-times into the transmission. There may have been up to 15 previous (non-late) collisions which had previously required the transmission to be re-attempted. The *Late Collisions* statistic dominates over the single-, multiple-, and excessive- collision statistics. If a late collision occurs, the frame will not be counted in any of these other three statistics.

CRC errors have no effect upon this statistic.

#### **12.2.2.4.6.19.8.11 Carrier Sense Errors (Offset = 3A060h)**

##### **Ethernet port N (where N = 1 to 8)**

The total number of frames on the port that experienced carrier loss. Such a frame:

- Was any data or MAC control frame destined for any unicast, broadcast or multicast address
- Was any size
- The carrier sense condition was lost or never asserted when transmitting the frame (the frame is not retransmitted). This is a transmit only statistic. Carrier Sense is a don't care for received frames. Transmit frames with carrier sense errors are sent until completion and are not aborted.

CRC errors have no effect upon this statistic.

#### **12.2.2.4.6.19.8.12 Tx Octets (Offset = 3A064h)**

##### **All ports**

The total number of bytes in all good frames transmitted on the port. A good frame is defined to be:

- Any data or MAC control frame which was destined for any unicast, broadcast or multicast address
- Was any size
- Had no late or excessive collisions, no carrier loss and no underrun.

#### **12.2.2.4.6.19.8.13 Transmit Priority 0-7 (Offset = 3A180h to 3A1A8h)**

The total number of frames transmitted on the port from transmit FIFO priority 0-7. Collision retries do not affect this statistic. Pause frames do not affect this statistic.

- Any frame transmitted from priority 0-7, and
- Was less than or equal to CPSW\_TX\_PRI0\_MAXLEN\_REG to CPSW\_TX\_PRI7\_MAXLEN\_REG
- Collision retries are not counted in this statistic.
- Pause frames are not counted in this statistic.
- Carrier sense errors do not affect this statistic.

Note: The Transmit Priority 0-7 Bytecount statistic is the number of bytes contained in the frames of the Transmit Priority 0-7 statistic.

#### **12.2.2.4.6.19.8.14 Transmit Priority 0-7 Drop (Offset = 3A1C0h to 3A1E8h)**

The total number of transmit frames on the port that overran the transmit FIFO priority 0-7 and were dropped. This count includes frames dropped due to CPSW\_TX\_PRI0\_MAXLEN\_REG to CPSW\_TX\_PRI7\_MAXLEN\_REG.

- Any frame destined to be transmitted from priority 0-7, and
- Was any size, and
- Was dropped due to priority 0-7 FIFO overrun (Start of packet overrun).
- Was dropped due to frame size larger than CPSW\_TX\_PRI0\_MAXLEN\_REG to CPSW\_TX\_PRI7\_MAXLEN\_REG.

Note: The Transmit Priority 0-7 Drop Bytecount statistic is the number of bytes contained in the frames of the Transmit Priority 0-7 Drop statistic.

#### **12.2.2.4.6.19.8.15 Tx Memory Protect Errors (Offset = 3A17Ch)**

##### **All ports**

The total number of transmit frames on the port that had a memory protect CRC error on egress:

- Any frame destined to be transmitted,
- Was any size
- Had a memory protect CRC error on egress.

### Note

1. Frames to the host with memory protect errors are dropped via TXST\_DROP. Ethernet frames will have at least one byte of the generated port type CRC inverted on egress.
2. This statistic is 8-bits wide only and will not rollover but will limit at 0xFF.
3. A non-zero value in this statistic will issue a STAT\_PEND0 interrupt for the associated port.

#### 12.2.2.4.6.19.8.16 IET Transmit Merge Fragment Count (Offset = 3A14Ch)

The total number of non-initial preemptable transmit fragments on preemptable transmit.

- Any frame destined to be transmitted on the preemptable port, and
- Was any size, and
- was a non-initial fragment.

#### 12.2.2.4.6.19.8.17 IET Transmit Merge Hold Count (Offset = 3A150h)

The total number of preemptable frames that were preempted and reassembled by the assertion of MAC\_HOLD in the CPSW\_PN\_IET\_CONTROL\_REG\_k register or were preempted by Enhanced Scheduled Traffic (EST). The IET statistic can be derived and maintained by software.

- Any frame destined to be transmitted on the preemptable port, and
- was any size, and
- was preempted by the assertion of MAC\_HOLD.
- was preempted by Enhanced Scheduled Traffic (EST).

#### 12.2.2.4.6.19.9 Rx- and Tx (Shared) Statistics Descriptions

**Table 12-226. Rx Statistics Summary**

Rx Statistic	Frame/ Oct	Rx/ Tx	Frame Type					Frame Size (bytes)								Event				
			MAC control		Data <sup>(5)</sup>			<64	64	65-127	128-255	256-511	512-1023	1024- rx_max len	>rx_max len	Flow Coll. (8)	CRC Error	Align/ Code	Overrun	Add. Disc.
			Pause frame	Non-pause <sup>(4)</sup>	Multicast	Broadcast	Unicast													
Good Rx Frames	F	Rx	(y  <sup>(1)</sup>	y	y	y	y	n	(y	y	y	y	y	y)	n	.(2)	n	n	-	n
Broadcast Rx Frames	F	Rx	(%  (6)	%	n	y)	n	n	(y	y	y	y	y	y)	n	-	n	n	-	n
Multicast Rx Frames	F	Rx	(%	%	y)	n	n	n	(y	y	y	y	y	y)	n	-	n	n	-	n
Pause Rx Frames	F	Rx	y	n	n	n	n	n	(y	y	y	y	y	y)	n	-	n	n	-	-
Rx CRC Errors	F	Rx	(y	y	y	y	y)	n	(y	y	y	y	y	y)	n	-	y	n	-	n
Rx Align/Code Errors	F	Rx	(y	y	y	y	y)	n	(y	y	y	y	y	y)	n	-	-	y	-	n
Oversized Rx Frames	F	Rx	(y	y	y	y	y)	n	n	n	n	n	n	n	y	-	n	n	-	n
Rx Jabbers	F	Rx	(y	y	y	y	y)	n	n	n	n	n	n	n	y	-	(y	y)	-	n
Undersized Rx Frames	F	Rx	n	n	(y	y	y)	y	n	n	n	n	n	n	n	-	n	n	-	n
Rx Fragments	F	Rx	n	n	(y	y	y)	y <sup>(7)</sup>	n	n	n	n	n	n	n	-	(y	y)	-	-
Rx Overruns <sup>(9)</sup>	F	Rx	(y	y	y	y	y)	(y	y	y	y	y	y	y)	y)	-	-	-	y	n
64octet Frames	F	Rx+ Tx <sup>(3)</sup>	(y	y	y	y	y)	n	y	n	n	n	n	n	n	-	-	-	-	n

**Table 12-226. Rx Statistics Summary (continued)**

Rx Statistic	Frame/ Oct	Rx/ Rx+ Tx	Frame Type					Frame Size (bytes)										Event				
			MAC control		Data <sup>(5)</sup>			<64	64	65-127	128-255	256-511	512-1023	1024-rx_max len	>rx_max len	Flow Coll. <sup>(8)</sup>	CRC Error	Alignment/ Code	Overrun	Add. Disc.		
			Pause frame	Non-pause <sup>(4)</sup>	Multicast	Broadcast	Unicast															
65-127octet Frames	F	Rx+Tx	(y	y	y	y	y)	n	n	y	n	n	n	n	n	-	-	-	-	n		
128-255octet Frames	F	Rx+Tx	(y	y	y	y	y)	n	n	n	y	n	n	n	n	-	-	-	-	n		
256-511octet Frames	F	Rx+Tx	(y	y	y	y	y)	n	n	n	n	y	n	n	n	-	-	-	-	n		
512-1023octet Frames	F	Rx+Tx	(y	y	y	y	y)	n	n	n	n	n	y	n	n	-	-	-	-	n		
1024-UPoctet Frames	F	Rx+Tx	(y	y	y	y	y)	n	n	n	n	n	n	y	n	-	-	-	-	n		
Rx Octets	O	Rx	(y	y	y	y	y)	n	(y	y	y	y	y	y)	n	-	n	n	-	n		
Net Octets	O	Rx+Tx	(y	y	y	y	y)	(y	y	y	y	y	y	y	y)	y)	-	-	-	-		

- (1) "AND" is assumed horizontally across the table between all conditions which form the statistic (marked y or n) except where (y|y), meaning "OR" is indicated. Parentheses are significant.
- (2) "-" indicates conditions which are ignored in the formations of the statistic.
- (3) Statistics marked "Rx+Tx" are formed by summing the Rx and Tx statistics, each of which is formed independently.
- (4) The non-pause column refers to all MAC control frames (for example, frames with length/type=88.08) with opcodes other than 0x0001. The pauseframe column refers to MAC frames with the opcode=0x0001.
- (5) The multicast, broadcast and unicast columns in the table refer to non-MAC Control/non-pause frames (i.e. data frames).
- (6) "%" If either a MAC control frame or pause frame has a multicast or broadcast destination address then the appropriate statistics will be updated.
- (7) "y^" Frame fragments are not counted if less than 8 bytes.
- (8) Flow coll. are half-duplex collisions forced by the MAC to achieve flow-control. A collision will be forced during the first 8 bytes so should not show in frame fragments. Some of the '-'s in this column might in reality be 'n's.
- (9) The rx\_overruns stat is for RX\_MOF\_OVERRUNS and RX\_SOF\_OVERRUNS added together.

**Table 12-227. Tx Statistics Summary**

Tx Statistic <sup>(9)</sup>	Fra me/ Oct	Tx/ Rx +Tx	Frame Type					Frame Size (bytes)							Event											
			MAC control (4)		Data			64	65- 127	128 -25 5	256 -51 1	512 -10 23	102 4-1 535	>15 35	CR C Err or	Collision Type					No Car rier	Qu eue d	Def err ed	Un der run		
			Pau se- MA C	An y- CP U	Mul ti cas t	Bro ad cas t	Uni cas t									Flo w <sup>(8)</sup>	1	2-1 5	16	Lat e						
Good Tx Frames	F	Tx	(y  (1)	y	y	y	y)	(y	y	y	y	y	y	y)	-(2)	-	-	-	n	n	n	-	-	n		
Broadcast Tx Frames	F	Tx	n	(%  (5)	n	y)	n	(y	y	y	y	y	y	y)	-	-	-	-	n	n	n	-	-	n		
Multicast Tx Frames	F	Tx	(y	%	y)	n	n	(y	y	y	y	y	y	y)	-	-	-	-	n	n	n	-	-	n		
Pause Tx Frames	F	Tx	y	n	n	n	n	y	n	n	n	n	n	n	-	-	-	-	-	-	-	-	-	-		
Collisions	F	Tx	n	(y	y	y	y)	(y	y	y	y	y	y	y)	-	(+ (6)	+	+	+	+	n	-	-	-		
Single Collision Tx Frames	F	Tx	n	(y	y	y	y)	(y	y	y	y	y	y	y)	-	-	y	n	n	n	n	-	-	-		

**Table 12-227. Tx Statistics Summary (continued)**

Tx Statistic <sup>(9)</sup>	Fra me/ Oct	Tx/ Rx +Tx	Frame Type					Frame Size (bytes)								Event										
			MAC control (4)		Data			64	65- 127	128 -25 5	256 -51 1	512 -10 23	102 4-1 535	>15 35	CR C Err or	Collision Type					No Car rier	Qu eue d	Def err ed	Un der run		
			Pau se- MA C	An y- CP U	Mul ti cas t	Bro ad cas t	Uni cas t									Flo w <sup>(8)</sup>	1	2-1 5	16	Lat e						
Multiple Collision Tx Frames	F	Tx	n	(y	y	y	y)	(y	y	y	y	y	y)	-	-	n	y	n	n	n	-	-	-			
Excessive Collisions	F	Tx	n	(y	y	y	y)	(y	y	y	y	y	y)	-	-	n	n	y	n	n	-	-	-			
Late Collisions	F	Tx	n	(y	y	y	y)	n	(y	y	y	y	y)	-	-	-	-	-	y	-	-	-	-			
Deferred Tx Frames	F	Tx	n	(y	y	y	y)	(y	y	y	y	y	y)	-	-	n	n	n	n	n	-	y	n			
Carrier Sense Errors	F	Tx	(y	y	y	y	y)	(y	y	y	y	y	y)	-	-	-	-	-	-	y	-	-	-			
64octet Frames	F	Rx+ Tx <sup>(3)</sup>	(y	y	y	y	y)	y	n	n	n	n	n	-	-	-	-	n	n	n	-	-	-			
65-127octet Frames	F	Rx+ Tx	(y	y	y	y	y)	n	y	n	n	n	n	-	-	-	-	n	n	n	-	-	-			
128-255octe t Frames	F	Rx+ Tx	(y	y	y	y	y)	n	n	y	n	n	n	-	-	-	-	n	n	n	-	-	-			
256-511octe t Frames	F	Rx+ Tx	(y	y	y	y	y)	n	n	n	y	n	n	-	-	-	-	n	n	n	-	-	-			
512-1023oct et Frames	F	Rx+ Tx	(y	y	y	y	y)	n	n	n	n	y	n	-	-	-	-	n	n	n	-	-	-			
1024- UPoctet Frames	F	Rx+ Tx	(y	y	y	y	y)	n	n	n	n	n	y	-	-	-	-	n	n	n	-	-	-			
Tx Octets	O	Tx	(y	y	y	y	y)	(y	y	y	y	y	y)	-	-	-	-	n	n	n	-	-	n			
Net Octets	O	Rx+ Tx	(y	y	y	y	y)	(y	y	y	y	y	y)	-	-	<sup>(7)</sup> \$	\$	\$	\$	\$	-	-	-			

- (1) "AND" is assumed horizontally across the table between all conditions which form the statistic (marked y or n) except where (y|y), meaning "OR" is indicated. Parentheses are significant.
- (2) "-" indicates conditions which are ignored in the formations of the statistic.
- (3) Statistics marked "Rx+Tx" are formed by summing the Rx and Tx statistics, each of which is formed independently.
- (4) Pause (MAC) frames are issued in the MAC as perfect (no CRC error) 64 byte frames in full duplex only, so they cannot collide.
- (5) "%" If a CPU sourced MAC control frame has a multicast or broadcast destination address then the appropriate statistics will be updated.
- (6) "+" indicates collisions which are "summed" (i.e. every collision is counted in the Collisions statistic). Jam sequences used for halfduplex flow control are also counted.
- (7) "\$" Every byte written on the wire during each retry attempt is also counted in addition to frames which experience no collisions or carrier loss.
- (8) The flow collision type is for half-duplex collisions forced by the MAC to achieve flow control. Some of the '-'s in this column might in reality be 'n's. To prevent double-counting, Net Octets are unaffected by the jam sequence – the 'received' bytes, however, are counted. (See Rx Statistics Summary.)
- (9) When the transmit Tx FIFO is drained due to the MAC being disabled or link being lost, then the frames being purged will not appear in the Tx statistics.

#### 12.2.2.4.6.19.9.1 Rx + Tx 64 Octet Frames (Offset = 3A068h)

#### All ports

The total number of 64-byte frames received and transmitted on the port. Such a frame is defined to be:

- Any data or MAC control frame which was destined for any unicast, broadcast or multicast address
- Did not experience late collisions, excessive collisions, or carrier sense error
- Was exactly 64 bytes long. (If the frame was being transmitted and experienced carrier loss that resulted in a frame of this size being transmitted, then the frame will be recorded in this statistic).

CRC errors, code/align errors and overruns do not affect the recording of frames in this statistic.

#### **12.2.2.4.6.19.9.2 Rx + Tx 65–127 Octet Frames (Offset = 3A06Ch)**

##### **All ports**

The total number of frames of size 65 to 127 bytes received and transmitted on the port. Such a frame is defined to be:

- Any data or MAC control frame which was destined for any unicast, broadcast or multicast address
- Did not experience late collisions, excessive collisions, or carrier sense error
- Was 65 to 127 bytes long

CRC errors, code/align errors, underruns and overruns do not affect the recording of frames in this statistic.

#### **12.2.2.4.6.19.9.3 Rx + Tx 128–255 Octet Frames (Offset = 3A070h)**

##### **All ports**

The total number of frames of size 128 to 255 bytes received and transmitted on the port. Such a frame is defined to be:

- Any data or MAC control frame which was destined for any unicast, broadcast or multicast address
- Did not experience late collisions, excessive collisions, or carrier sense error
- Was 128 to 255 bytes long

CRC errors, code/align errors, underruns and overruns do not affect the recording of frames in this statistic.

#### **12.2.2.4.6.19.9.4 Rx + Tx 256–511 Octet Frames (Offset = 3A074h)**

##### **All ports**

The total number of frames of size 256 to 511 bytes received and transmitted on the port. Such a frame is defined to be:

- Any data or MAC control frame which was destined for any unicast, broadcast or multicast address
- Did not experience late collisions, excessive collisions, or carrier sense error
- Was 256 to 511 bytes long

CRC errors, code/align errors, underruns and overruns do not affect the recording of frames in this statistic.

#### **12.2.2.4.6.19.9.5 Rx + Tx 512–1023 Octet Frames (Offset = 3A078h)**

##### **All ports**

The total number of frames of size 512 to 1023 bytes received and transmitted on the port. Such a frame is defined to be:

- Any data or MAC control frame which was destined for any unicast, broadcast or multicast address
- Did not experience late collisions, excessive collisions, or carrier sense error
- Was 512 to 1023 bytes long

CRC errors, code/align errors, underruns and overruns do not affect the recording of frames in this statistic.

#### **12.2.2.4.6.19.9.6 Rx + Tx 1024\_Up Octet Frames (Offset = 3A07Ch)**

##### **All ports**

The total number of frames of size 1024 to RX\_MAXLEN bytes for receive or 1024 up for transmit on the port. Such a frame is defined to be:

- Any data or MAC control frame which was destined for any unicast, broadcast or multicast address
- Did not experience late collisions, excessive collisions, or carrier sense error
- Was 1024 to RX\_MAXLEN bytes long on receive, or any size on transmit

CRC errors, code/align errors, underruns and overruns do not affect the recording of frames in this statistic.

#### 12.2.2.4.6.19.9.7 Net Octets (Offset = 3A080h)

#### All ports

The total number of bytes of frame data received and transmitted on the port. Each frame counted:

- was any data or MAC control frame destined for any unicast, broadcast or multicast address (address match does not matter)
- Any length (including less than 64 bytes and greater than RX\_MAXLEN bytes)

Also counted in this statistic is:

- Every byte transmitted before a carrier-loss was experienced
- Every byte transmitted before each collision was experienced, (that is, multiple retries are counted each time)
- Every byte received if the port is in half-duplex mode until a jam sequence was transmitted to initiate flow control. (The jam sequence was not counted to prevent double-counting)

Error conditions such as alignment errors, CRC errors, code errors, overruns and underruns do not affect the recording of bytes by this statistic.

The objective of this statistic is to give a reasonable indication of Ethernet utilization.

#### 12.2.2.4.6.19.10

**Table 12-228. Rx Statistics Summary**

Rx Statistic	Fra me/ Oct	Rx/ Rx+ Tx	Frame Type					Frame Size (bytes)								Event				
			MAC control		Data <sup>(5)</sup>			<64	64	65-127	128-255	256-511	512-1023	1024-rx_maxlen	>rx_maxlen	Flow Coll. <sup>(8)</sup>	CRC Error	Align/Code	Overrun	Add. Disc.
			Pause frame	Non-pause <sup>(4)</sup>	Multicast	Broadcast	Unicast													
Good Rx Frames	F	Rx	(y  <sup>(1)</sup>	y	y	y	y	n	(y	y	y	y	y	y)	n	.(2)	n	n	-	n
Broadcast Rx Frames	F	Rx	(%  <sup>(6)</sup>	%	n	y)	n	n	(y	y	y	y	y	y)	n	-	n	n	-	n
Multicast Rx Frames	F	Rx	(%	%	y)	n	n	n	(y	y	y	y	y	y)	n	-	n	n	-	n
Pause Rx Frames	F	Rx	y	n	n	n	n	n	(y	y	y	y	y	y)	n	-	n	n	-	-
Rx CRC Errors	F	Rx	(y	y	y	y	y)	n	(y	y	y	y	y	y)	n	-	y	n	-	n
Rx Align/Code Errors	F	Rx	(y	y	y	y	y)	n	(y	y	y	y	y	y)	n	-	-	y	-	n
Oversized Rx Frames	F	Rx	(y	y	y	y	y)	n	n	n	n	n	n	n	y	-	n	n	-	n
Rx Jabbers	F	Rx	(y	y	y	y	y)	n	n	n	n	n	n	n	y	-	(y	y)	-	n
Undersized Rx Frames	F	Rx	n	n	(y	y	y)	y	n	n	n	n	n	n	n	-	n	n	-	n
Rx Fragments	F	Rx	n	n	(y	y	y)	y <sup>(7)</sup>	n	n	n	n	n	n	n	-	(y	y)	-	-
Rx Overruns <sup>(9)</sup>	F	Rx	(y	y	y	y	y)	(y	y	y	y	y	y	y)	y)	-	-	-	y	n
64octet Frames	F	Rx+Tx <sup>(3)</sup>	(y	y	y	y	y)	n	y	n	n	n	n	n	n	-	-	-	-	n



**Table 12-228. Rx Statistics Summary (continued)**

Rx Statistic	Frame/ Oct	Rx/ Rx+ Tx	Frame Type					Frame Size (bytes)										Event				
			MAC control		Data <sup>(5)</sup>			<64	64	65-127	128-255	256-511	512-1023	1024-rx_max len	>rx_max len	Flow Coll. <sup>(8)</sup>	CRC Error	Alignment/ Code	Overrun	Add. Disc.		
			Pause frame	Non-pause <sup>(4)</sup>	Multicast	Broadcast	Unicast															
65-127octet Frames	F	Rx+Tx	(y	y	y	y	y)	n	n	y	n	n	n	n	n	-	-	-	-	n		
128-255octet Frames	F	Rx+Tx	(y	y	y	y	y)	n	n	n	y	n	n	n	n	-	-	-	-	n		
256-511octet Frames	F	Rx+Tx	(y	y	y	y	y)	n	n	n	n	y	n	n	n	-	-	-	-	n		
512-1023octet Frames	F	Rx+Tx	(y	y	y	y	y)	n	n	n	n	n	y	n	n	-	-	-	-	n		
1024-UPoctet Frames	F	Rx+Tx	(y	y	y	y	y)	n	n	n	n	n	n	y	n	-	-	-	-	n		
Rx Octets	O	Rx	(y	y	y	y	y)	n	(y	y	y	y	y	y)	n	-	n	n	-	n		
Net Octets	O	Rx+Tx	(y	y	y	y	y)	(y	y	y	y	y	y	y	y)	y)	-	-	-	-		

- (1) "AND" is assumed horizontally across the table between all conditions which form the statistic (marked y or n) except where (y|y), meaning "OR" is indicated. Parentheses are significant.
- (2) "-" indicates conditions which are ignored in the formations of the statistic.
- (3) Statistics marked "Rx+Tx" are formed by summing the Rx and Tx statistics, each of which is formed independently.
- (4) The non-pause column refers to all MAC control frames (for example, frames with length/type=88.08) with opcodes other than 0x0001. The pauseframe column refers to MAC frames with the opcode=0x0001.
- (5) The multicast, broadcast and unicast columns in the table refer to non-MAC Control/non-pause frames (i.e. data frames).
- (6) "%" If either a MAC control frame or pause frame has a multicast or broadcast destination address then the appropriate statistics will be updated.
- (7) "y^" Frame fragments are not counted if less than 8 bytes.
- (8) Flow coll. are half-duplex collisions forced by the MAC to achieve flow-control. A collision will be forced during the first 8 bytes so should not show in frame fragments. Some of the '-'s in this column might in reality be 'n's.
- (9) The rx\_overruns stat is for RX\_MOF\_OVERRUNS and RX\_SOF\_OVERRUNS added together.

**Table 12-229. Tx Statistics Summary**

Tx Statistic <sup>(9)</sup>	Fra me/ Oct	Tx/ Rx +Tx	Frame Type					Frame Size (bytes)							Event											
			MAC control (4)		Data			64	65- 127	128 -25 5	256 -51 1	512 -10 23	102 4-1 535	>15 35	CR C Err or	Collision Type					No Car rier	Qu eue d	Def err ed	Un der run		
			Pau se- MA C	An y- CP U	Mul ti cas t	Bro ad cas t	Uni cas t									Flo w <sup>(8)</sup>	1	2-1 5	16	Lat e						
Good Tx Frames	F	Tx	(y  (1)	y	y	y	y)	(y	y	y	y	y	y	y)	-(2)	-	-	-	n	n	n	-	-	n		
Broadcast Tx Frames	F	Tx	n	(%  (5)	n	y)	n	(y	y	y	y	y	y	y)	-	-	-	-	n	n	n	-	-	n		
Multicast Tx Frames	F	Tx	(y	%	y)	n	n	(y	y	y	y	y	y	y)	-	-	-	-	n	n	n	-	-	n		
Pause Tx Frames	F	Tx	y	n	n	n	n	y	n	n	n	n	n	n	-	-	-	-	-	-	-	-	-	-		
Collisions	F	Tx	n	(y	y	y	y)	(y	y	y	y	y	y	y)	-	(+ (6)	+	+	+	+	n	-	-	-		
Single Collision Tx Frames	F	Tx	n	(y	y	y	y)	(y	y	y	y	y	y	y)	-	-	y	n	n	n	n	-	-	-		

**Table 12-229. Tx Statistics Summary (continued)**

Tx Statistic <sup>(9)</sup>	Fra me/ Oct	Tx/ Rx +Tx	Frame Type					Frame Size (bytes)										Event									
			MAC control (4)		Data			64	65- 127	128- 255	256- 511	512- 1023	1024- 1535	>1535	CR C Error	Collision Type					No Carrier	Queue d	Def erred	Under run			
			Pause- MAC	Any- CPU	Multi- cast	Broadcast	Unicast									Flow <sup>(8)</sup>	1	2-15	16	Late							
Multiple Collision Tx Frames	F	Tx	n	(y	y	y	y)	(y	y	y	y	y	y	y)	-	-	n	y	n	n	n	-	-	-			
Excessive Collisions	F	Tx	n	(y	y	y	y)	(y	y	y	y	y	y	y)	-	-	n	n	y	n	n	-	-	-			
Late Collisions	F	Tx	n	(y	y	y	y)	n	(y	y	y	y	y	y)	-	-	-	-	-	y	-	-	-	-			
Deferred Tx Frames	F	Tx	n	(y	y	y	y)	(y	y	y	y	y	y	y)	-	-	n	n	n	n	n	-	y	n			
Carrier Sense Errors	F	Tx	(y	y	y	y)	(y	y	y	y	y	y	y	y)	-	-	-	-	-	-	y	-	-	-			
64octet Frames	F	Rx+ Tx <sup>(3)</sup>	(y	y	y	y)	y	n	n	n	n	n	n	n	-	-	-	-	n	n	n	-	-	-			
65-127octet Frames	F	Rx+ Tx	(y	y	y	y)	n	y	n	n	n	n	n	n	-	-	-	-	n	n	n	-	-	-			
128-255octet Frames	F	Rx+ Tx	(y	y	y	y)	n	n	y	n	n	n	n	n	-	-	-	-	n	n	n	-	-	-			
256-511octet Frames	F	Rx+ Tx	(y	y	y	y)	n	n	n	y	n	n	n	n	-	-	-	-	n	n	n	-	-	-			
512-1023octet Frames	F	Rx+ Tx	(y	y	y	y)	n	n	n	n	y	n	n	n	-	-	-	-	n	n	n	-	-	-			
1024-UPoctet Frames	F	Rx+ Tx	(y	y	y	y)	n	n	n	n	n	y	y	y	-	-	-	-	n	n	n	-	-	-			
Tx Octets	O	Tx	(y	y	y	y)	(y	y	y	y	y	y	y	y)	-	-	-	-	n	n	n	-	-	n			
Net Octets	O	Rx+ Tx	(y	y	y	y)	(y	y	y	y	y	y	y	y)	-	-	\$ <sup>(7)</sup>	\$	\$	\$	\$	-	-	-			

- (1) "AND" is assumed horizontally across the table between all conditions which form the statistic (marked y or n) except where (y|y), meaning "OR" is indicated. Parentheses are significant.
- (2) "-" indicates conditions which are ignored in the formations of the statistic.
- (3) Statistics marked "Rx+Tx" are formed by summing the Rx and Tx statistics, each of which is formed independently.
- (4) Pause (MAC) frames are issued in the MAC as perfect (no CRC error) 64 byte frames in full duplex only, so they cannot collide.
- (5) "%" If a CPU sourced MAC control frame has a multicast or broadcast destination address then the appropriate statistics will be updated.
- (6) "+" indicates collisions which are "summed" (i.e. every collision is counted in the Collisions statistic). Jam sequences used for halfduplex flow control are also counted.
- (7) "\$" Every byte written on the wire during each retry attempt is also counted in addition to frames which experience no collisions or carrier loss.
- (8) The flow collision type is for half-duplex collisions forced by the MAC to achieve flow control. Some of the '-'s in this column might in reality be 'n's. To prevent double-counting, Net Octets are unaffected by the jam sequence – the 'received' bytes, however, are counted. (See [Table 12-228](#).)
- (9) When the transmit Tx FIFO is drained due to the MAC being disabled or link being lost, then the frames being purged will not appear in the Tx statistics.

#### 12.2.2.4.7 Common Platform Time Sync (CPTS)

The Common Platform Time Sync (CPTS) module is used to facilitate host control of time sync operations. It enables compliance with the IEEE 1588-2008 standard for a precision clock synchronization protocol.

Main features of CPTS module are:

- Supports the selection of multiple external clock sources
- Software control of time sync events via interrupt or polling
- Supports up to 4 hardware timestamp push inputs
- Supports timestamp counter compare output (CPTS\_COMP)
- Supports timestamp counter bit output (CPTS\_SYNC)
- Supports a configurable number of timestamp Generator bit outputs (CPTS\_GENFn).
- Supports Ethernet Enhanced Scheduled Traffic Operations (CPTS\_ESTFn).
- 32-bit and 64-bit timestamp modes with PPM and nudge adjustment.

##### 12.2.2.4.7.1 CPSW0 CPTS Integration

This section describes CPTS module integration in the CPSW0 module, including information about clocks, resets, and hardware requests.

Figure 12-172 shows CPTS integration in the device CPSW0 module.

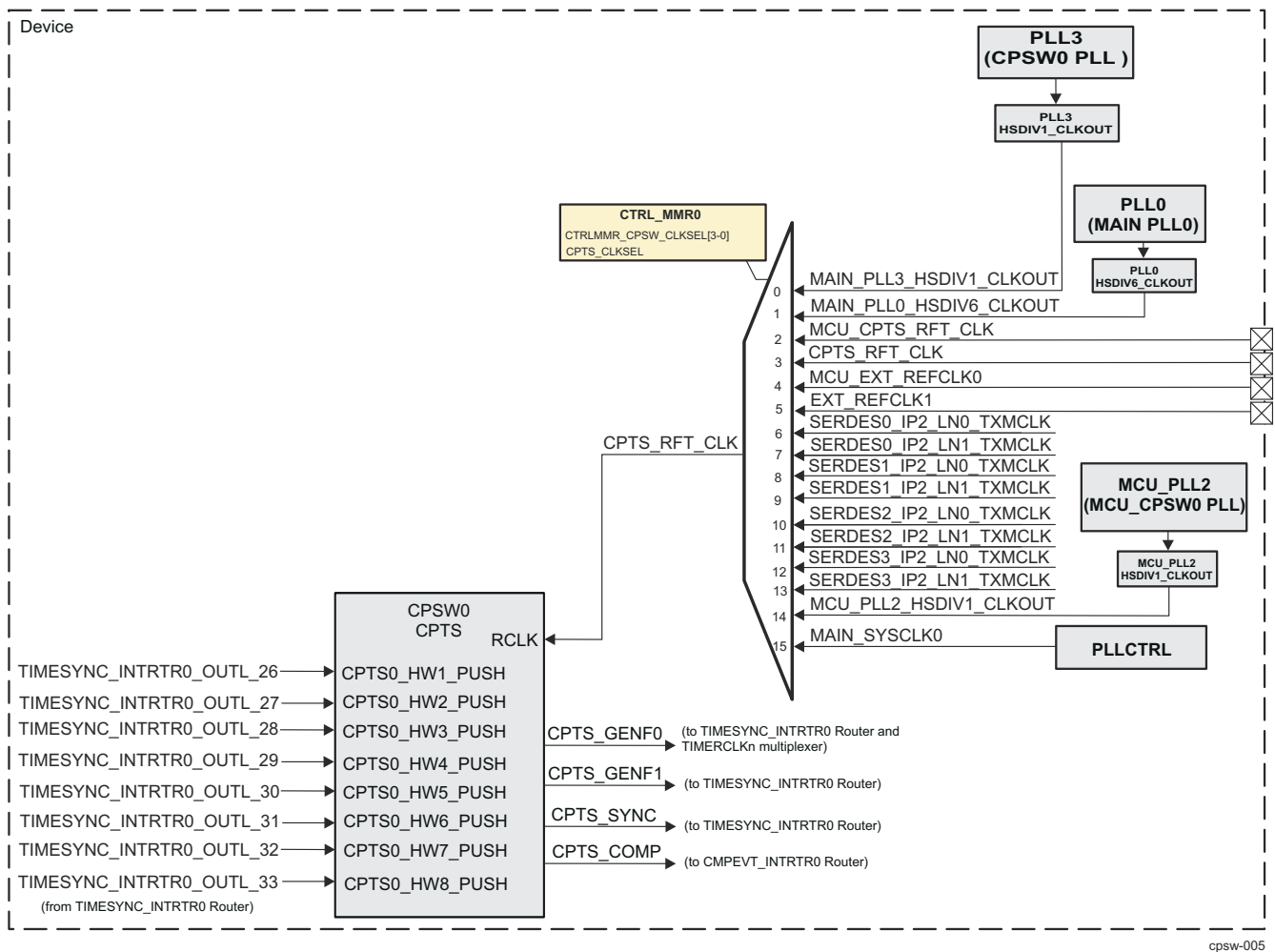


Figure 12-172. CPSW0 CPTS Integration

CPTS IEEE 1588 clock (RCLK) is selected through the CTRLMMR\_MCU\_ENET\_CLKSEL register.

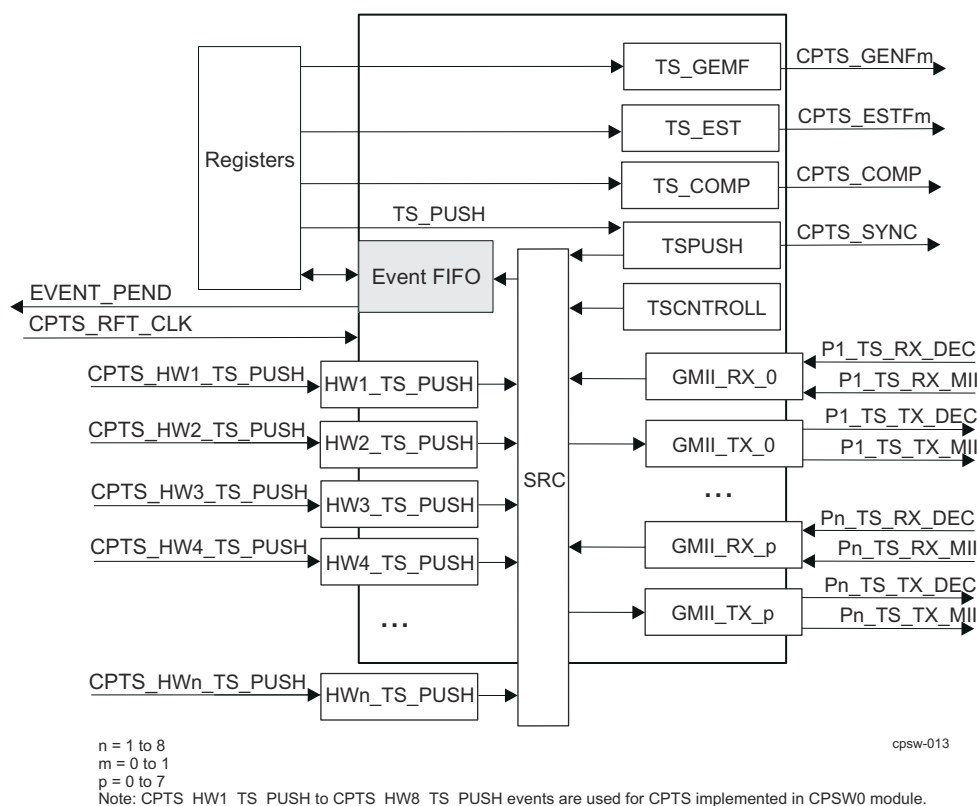
### Note

For more information about CPTS clocks and resets, see *MCU\_CPSW0 Clocks and Resets* in *CPSW0 Integration*.

#### 12.2.2.4.7.2 CPTS Architecture

Figure 12-173 shows the architecture of the CPTS module inside the CPSW Ethernet Subsystem. Time stamp values for every packet transmitted or received on either port of the CPSW are recorded. At the same time, each packet is decoded to determine if it is a valid time sync event. If so, an event is loaded into the Event FIFO for processing containing the recorded time stamp value when the packet was transmitted or received.

In addition, both hardware (HWN\_TS\_PUSH) and software (TS\_PUSH) can be used to read the current time stamp value through the Event FIFO. The reference clock used for the time stamp (CPTS\_RFT\_CLK) can be derived from several sources.



**Figure 12-173. CPTS Block Diagram**

### Note

See [Section 12.2.2.4.7.1](#), *CPSW0 CPTS Integration* for CPTS integration in the device CPSW0 module.

#### 12.2.2.4.7.3 CPTS Initialization

The CPTS module should be configured as follows:

1. Reset the CPTS module.
2. Write the CPTS\_CLKSEL value in the CTRLMMR\_CPSW\_CLKSEL register with the desired reference clock selection. This value is allowed to be written only when the CPTS\_EN bit in the CPSW\_CPTS\_CONTROL\_REG register is cleared to zero.
3. Set the CPTS\_EN bit in the CPSW\_CPTS\_CONTROL\_REG register.

4. If using interrupts and not polling, enable the interrupt by setting the TS\_PEND\_EN bit in the CPSW\_CPTS\_INT\_ENABLE\_REG register.

#### 12.2.2.4.7.4 32-bit Time Stamp Value

The time stamp value is a 32-bit value that increments on each CPTS\_RFT\_CLK rising edge when CPTS\_EN bit is set to 1h. When CPTS\_EN bit is cleared to 0h, the time stamp value is reset to 0h.

If more than 32-bits of time stamp are required by the application, the host software must maintain the necessary number of upper bits. The upper time stamp value should be incremented by the host when the rollover event is detected.

For test purposes, the time stamp can be written via the time stamp load function (CPSW\_CPTS\_TS\_LOAD\_VAL\_REG / CPSW\_CPTS\_TS\_LOAD\_HIGH\_VAL\_REG and CPSW\_CPTS\_TS\_LOAD\_EN\_REG registers).

#### 12.2.2.4.7.5 64-bit Time Stamp Value

The time stamp value is a 64-bit value that increments on each CPTS\_RFT\_CLK rising edge when CPTS\_EN bit is set to 1h. When CPTS\_EN bit is cleared to 0h, the time stamp value is reset to 0h.

64-bit mode is selected when CPSW\_CPTS\_CONTROL\_REG[5] MODE bit set to 1h.

For test purposes, the time stamp value can be written via the time stamp load function (CPSW\_CPTS\_TS\_LOAD\_EN\_REG, CPSW\_CPTS\_TS\_LOAD\_VAL\_REG, and CPSW\_CPTS\_TS\_LOAD\_HIGH\_VAL\_REG registers). The CPSW\_CPTS\_TS\_ADD\_VAL\_REG feature is included to allow 1ns timestamp operations with an CPTS\_RFT\_CLK rate less than 1Ghz. [Table 12-230](#) shows the CPTS\_RFT\_CLK and CPSW\_CPTS\_TS\_ADD\_VAL\_REG values for 1ns operations.

**Table 12-230. ADD\_VAL feature**

CPTS_RFT_CLK (MHz)	CPSW_CPTS_TS_ADD_VAL_REG[2-0] ADD_VAL
1000 MHz	0
500 MHz	1
333.33 MHz	2
250 MHz	3
200 MHz	4
166.66 MHz	5
142.85714 MHz	6
125 MHz	7

#### 12.2.2.4.7.6 64-Bit Timestamp Nudge

The 64-bit TIME\_STAMP value can be adjusted by writing the CPSW\_CPTS\_TS\_NUDGE\_VAL\_REG[7-0] TS\_NUDGE\_VAL bit field value which is a two's complement value. A value of FFh will subtract 1 clock cycle from the next incremented 64-bit time stamp value (CPSW\_CPTS\_EVENT\_0\_REG[31-0] TIME\_STAMP and CPSW\_CPTS\_EVENT\_3\_REG[31-0] TIME\_STAMP value). A nudge value of 1h will add 1 clock cycle to the next incremented TIME\_STAMP[63-0] value. For example, if the current TIME\_STAMP value is F06h, and CPSW\_CPTS\_TS\_ADD\_VAL\_REG[2-0] ADD\_VAL = 3h, the next incremented timestamp value would be F0Ah without a nudge and F0Ah +/- [7-0] TS\_NUDGE\_VAL with a nudge. The [7-0] TS\_NUDGE\_VAL value is cleared to zero when the nudge has occurred.

#### 12.2.2.4.7.7 64-bit Timestamp PPM

The 64-bit TIME\_STAMP can be adjusted by parts per million or by parts per hour. Writing a non-zero value to the CPSW\_CPTS\_TS\_PPM\_LOW\_VAL\_REG[31-0] TS\_PPM\_LOW\_VAL (Time stamp PPM Low value) and CPSW\_CPTS\_TS\_PPM\_HIGH\_VAL\_REG[9-0] TS\_PPM\_HIGH\_VAL (Time stamp PPM High value) enables PPM operations. The adjustment is up or down depending on the [7] TS\_PPM\_DIR bit in the CPSW\_CPTS\_CONTROL\_REG register. The TIME\_STAMP value is increased by the PPM value when [7]

TS\_PPM\_DIR bit is cleared. The TIME\_STAMP value is decreased by the PPM value when [7] TS\_PPM\_DIR bit is set.

#### Parts Per Million example:

To adjust for 100 parts per million the configured value for TS\_PPM[41-0] (through CPSW\_CPTS\_TS\_PPM\_LOW\_VAL\_REG[31-0] TS\_PPM\_LOW\_VAL and CPSW\_CPTS\_TS\_PPM\_HIGH\_VAL\_REG[9-0] TS\_PPM\_HIGH\_VAL) is:  
 $1,000,000/100 = 10,000(\text{decimal})$

#### Parts Per Hour example:

To adjust for 1 part per hour at 1 GHz CPTS\_RFT\_CLK the configured value for TS\_PPM[41-0] (through CPSW\_CPTS\_TS\_PPM\_LOW\_VAL\_REG[31-0] TS\_PPM\_LOW\_VAL and CPSW\_CPTS\_TS\_PPM\_HIGH\_VAL\_REG[9-0] TS\_PPM\_HIGH\_VAL) is:  
 $(1,000,000,000\text{Hz}/1\text{pph}) * (3600 \text{ seconds/hour}) = 34630\text{B8A000} (\text{hex})$

#### 12.2.2.4.7.8 Event FIFO

All time sync events are pushed onto the Event FIFO. There are 10 locations in the event FIFO with no overrun indication supported. Software must service the event FIFO in a timely manner to prevent FIFO overrun.

#### 12.2.2.4.7.9 Timestamp Compare Output

CPTS features one Time Stamp Compare (CPTS\_COMP) output. The CPTS\_COMP function is a software oriented feature that is intended to be replaced going forward by the hardware oriented GENF function. CPTS\_COMP is not compatible with timestamp PPM or a non-zero CPSW\_CPTS\_TS\_ADD\_VAL\_REG[2-0] ADD\_VAL value.

##### 12.2.2.4.7.9.1 Non-Toggle Mode: 32-bit

The CPTS\_COMP output is asserted for CPSW\_CPTS\_TS\_COMP\_LEN\_REG[31-0] TS\_COMP\_LENGTH periods when the CPSW\_CPTS\_EVENT\_0\_REG[31-0] TIME\_STAMP value (lower 32-bits) compares with the CPSW\_CPTS\_TS\_COMP\_VAL\_REG[31-0] TS\_COMP\_VAL and the length value is non-zero. The CPTS\_COMP rising edge occurs three CPTS\_RFT\_CLK clock periods after the values compare. A timestamp compare event is pushed into the event FIFO when CPTS\_COMP is asserted. The polarity of the CPTS\_COMP output is determined by the CPSW\_CPTS\_CONTROL\_REG[2] TS\_COMP\_POLARITY bit. The output is asserted low when the polarity bit is 0h.

##### 12.2.2.4.7.9.2 Non-Toggle Mode: 64-bit

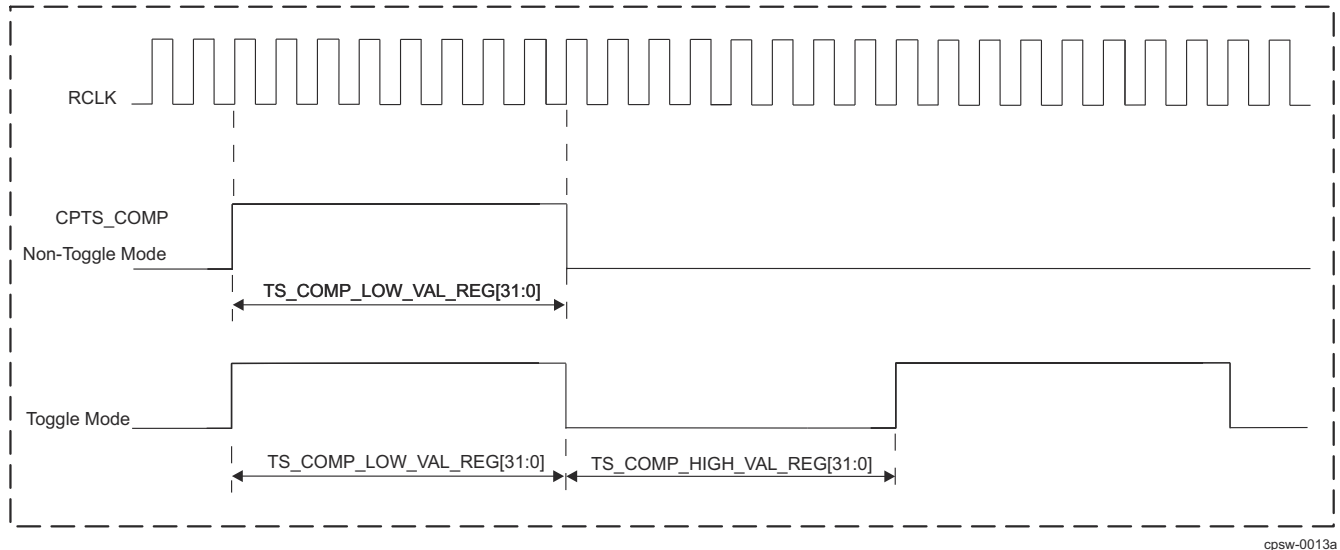
64-bit mode operation is identical to 32-bit mode except that all 64-bits of the TIME\_STAMP are used (CPSW\_CPTS\_EVENT\_0\_REG and CPSW\_CPTS\_EVENT\_3\_REG). In 32-bit mode only the lower 32-bits (CPSW\_CPTS\_EVENT\_0\_REG) are used.

##### 12.2.2.4.7.9.3 Toggle Mode: 32-bit

The CPTS\_COMP output is asserted (CPSW\_CPTS\_TS\_COMP\_LEN\_REG[31-0] TS\_COMP\_LENGTH) for CPTS\_RFT\_CLK clock periods when the TIME\_STAMP[31:0] value compares with the CPSW\_CPTS\_TS\_COMP\_VAL\_REG and the length value is non-zero. The CPTS\_COMP toggles thereafter on CPSW\_CPTS\_TS\_COMP\_VAL\_REG[31-0] TS\_COMP\_LENGTH for CPTS\_RFT\_CLK periods. The length high or low can be adjusted by writing the CPSW\_CPTS\_TS\_COMP\_NUDGE\_REG[7-0] NUDGE bit field value which is a two's complement value. A value of FFh will subtract one CPTS\_RFT\_CLK period from the CPSW\_CPTS\_TS\_COMP\_VAL\_REG[31-0] TS\_COMP\_LENGTH value. A value of 0x01h will add one CPTS\_RFT\_CLK period to the CPSW\_CPTS\_TS\_COMP\_LEN\_REG[31-0] TS\_COMP\_LENGTH value. Only a single high or low time is adjusted (nudged) and the CPSW\_CPTS\_TS\_COMP\_NUDGE\_REG[7-0] NUDGE value is cleared to zero when the nudge has occurred. The CPTS\_COMP output is asserted low when the CPSW\_CPTS\_CONTROL\_REG[2] TS\_COMP\_POLARITY bit is 0h. No compare events and no CPTS\_EVNT interrupts are generated in toggle mode. The CPSW\_CPTS\_CONTROL\_REG[6] TS\_COMP\_TOG bit must be set for toggle mode (value 1h). Note this bit must be set before writing a non-zero value to CPSW\_CPTS\_TS\_COMP\_VAL\_REG register.

#### 12.2.2.4.7.9.4 Toggle Mode: 64-bit

64-bit mode operation is identical to 32-bit mode except that all 64-bits of the **TIME\_STAMP** are used (**CPSW\_CPTS\_EVENT\_0\_REG** and **CPSW\_CPTS\_EVENT\_3\_REG**). In 32-bit mode only the lower 32-bits (**CPSW\_CPTS\_EVENT\_0\_REG**) are used.



**Figure 12-174. CPTS\_COMP Output in Toggle and Non-Toggle Mode**

#### 12.2.2.4.7.10 Timestamp Sync Output

The **CPTS\_SYNC** output is a selected bit of the [31:0]**TIME\_STAMP** counter value. One of bits 17-31 can be selected in **CPSW\_CPTS\_CONTROL\_REG[31-28] TS\_SYNC\_SEL**. The **CPTS\_SYNC** output is disabled when **CPSW\_CPTS\_CONTROL\_REG[31-28] TS\_SYNC\_SEL** is zero.

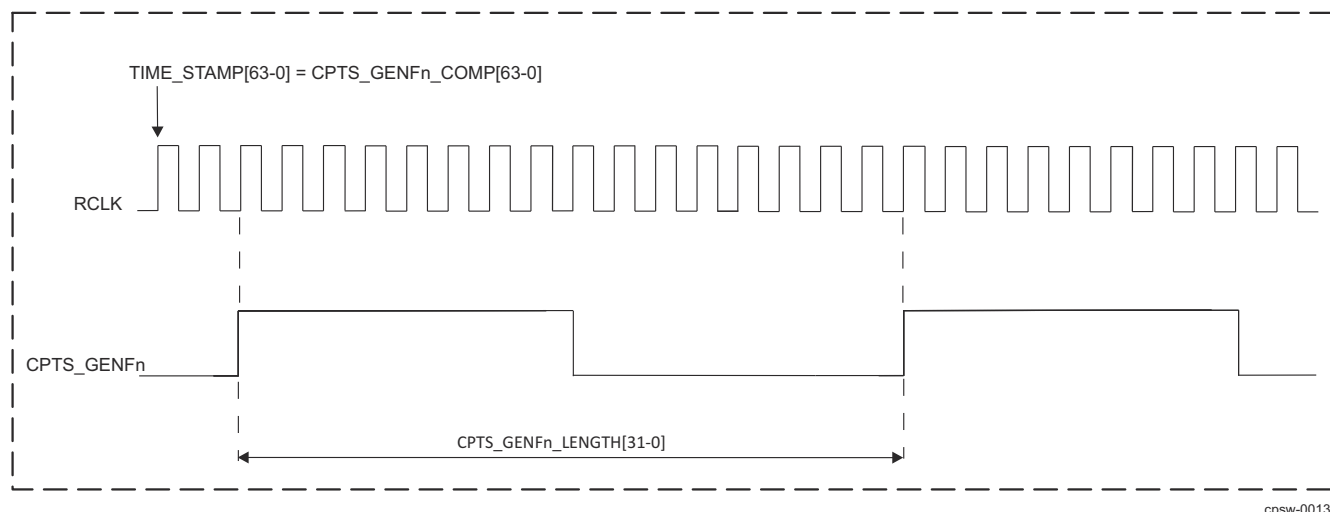
If the selected counter bit is 1 at the time when **TS\_SYNC\_SEL** value is written then a rising edge will not occur on the **CPTS\_SYNC** output. A rising edge will occur on the **CPTS\_SYNC** output upon the next transition to 1 of the selected counter bit. The **TS\_SYNC\_SEL** value must be written to zero before changing to a different non-zero value. No events are generated due to the **CPTS\_SYNC** operation. The **CPTS\_SYNC** output is two **CPTS\_RFT\_CLK** periods after the actual count value.

#### 12.2.2.4.7.11 Timestamp GENFn Output

The **CPTS\_GENFn** outputs have a programmable cycle (frequency) with a PPM feature and software nudge feature. The **CPTS\_GENFn** output cycle is **CPSW\_GENF0\_LENGTH\_REG/ CPSW\_GENF1\_LENGTH\_REG[31:0] LENGTH CPTS\_RFT\_CLK** periods (which is different than **CPTS\_COMP** operation). [Figure 12-175](#) represents the **CPTS\_GENFn** output signal.

The **CPTS\_GENFn** output cycle is **CPSW\_GENF0\_LENGTH\_REG/ CPSW\_GENF1\_LENGTH\_REG[31:0] CPTS\_RFT\_CLK** periods beginning when the 64-bit **TIME\_STAMP** value compares with the 64-bit **GENFn\_COMP** value (**CPSW\_GENF0\_COMP\_LOW\_REG** and **CPSW\_GENF0\_COMP\_HIGH\_REG** registers) and the length value is non-zero. The **CPTS\_GENFn** output cycle repeats thereafter every **CPSW\_GENF0\_LENGTH\_REG/ CPSW\_GENF1\_LENGTH\_REG[31:0] CPTS\_RFT\_CLK** periods. The upper 32-bit word should be written first for 64-bit values. The length should be zero while the comparison value and other configuration parameters are being configured. The length should be written non-zero to enable operations last. The first cycle after comparison is active high when the **CPSW\_CPTS\_CONTROL\_REG[2] TS\_COMP\_POLARITY** bit is low. No compare events and no **CPTS\_EVNT** interrupts are generated.





**Figure 12-175. CPTS\_GENFn Output Signal Diagram**

#### 12.2.2.4.7.11.1 GENFn Nudge

The cycle length can be adjusted by writing the CPSW\_CPTS\_TS\_COMP\_NUDGE\_REG[7-0] NUDGE register value which is a two's complement value. A value of FFh will subtract 1 CPTS\_RFT\_CLK from the CPSW\_GENF0\_LENGTH\_REG/ CPSW\_GENF1\_LENGTH\_REG[31-0] value. A value of 1h will add 1 CPTS\_RFT\_CLK to the CPSW\_GENF0\_LENGTH\_REG/ CPSW\_GENF1\_LENGTH\_REG[31-0] value. The CPSW\_CPTS\_TS\_COMP\_NUDGE\_REG[7-0] NUDGE value is cleared to zero when the nudge has occurred.

#### 12.2.2.4.7.11.2 GENFn PPM

The CPTS\_GENFn output cycle can be adjusted by parts per million or by parts per hour. Writing a non-zero value to CPSW\_GENF0\_PPM\_LOW\_REG/ CPSW\_GENF0\_PPM\_HIGH\_REG enables PPM operations. The PPM counter continually loads and decrements to zero and then loads again. A single CPTS\_RFT\_CLK adjustment is made when the PPM counter decrements to zero. The adjustment is up or down depending on the CPSW\_GENF0\_CONTROL\_REG/ CPSW\_GENF1\_CONTROL\_REG[0] PPM\_DIR bit. When PPM\_DIR bit is set a single CPTS\_RFT\_CLK time is subtracted from the generate function counter which has the effect of increasing the generate function frequency by the PPM amount. When PPM\_DIR bit is cleared a single CPTS\_RFT\_CLK time is added to the generate function counter which has the effect of decreasing the generate function frequency by the PPM amount.

#### Parts Per Million example:

To adjust for 100 parts per million the configured value for GENF\_PPM[41-0] (through CPSW\_GENF0\_PPM\_LOW\_REG and CPSW\_GENF0\_PPM\_HIGH\_REG) is:  
 $1,000,000/100 = 10,000(\text{decimal})$

#### Parts Per Hour example:

To adjust for 1 part per hour at 1 GHz CPTS\_RFT\_CLK the configured value for GENF\_PPM[41-0] (through CPSW\_GENF0\_PPM\_LOW\_REG and CPSW\_GENF0\_PPM\_HIGH\_REG) is:  
 $(1,000,000,000\text{Hz}/1\text{pph}) * (3600 \text{ seconds/hour}) = 34630\text{B8A000} (\text{hex})$

#### 12.2.2.4.7.12 Timestamp ESTFn

Each Ethernet port has a dedicated ESTFn generator which operates identically to the GENFn function.

#### 12.2.2.4.7.13 Time Sync Events

Time Sync events are 96-bit values that are pushed onto the event FIFO and read by software in 32-bit reads. Four 32-bit registers, CPSW\_CPTS\_EVENT\_0\_REG through CPSW\_CPTS\_EVENT\_3\_REG hold the data of a time sync event. There are eight types of sync events:

- Time Stamp Push Event



- Time Stamp Counter Rollover Event (32-bit mode only)
- Time Stamp Counter Half-rollover Event (32-bit mode only)
- Hardware Time Stamp Push Event
- Ethernet Receive Event
- Ethernet Transmit Event
- Time Stamp Compare Event
- Host Transmit Event

#### **12.2.2.4.7.13.1 Time Stamp Push Event**

Software can obtain the current time stamp value (at the time of the write) by initiating a time stamp push event. The push event is initiated by setting the [0]TS\_PUSH bit of the CPSW\_CPTS\_TS\_PUSH\_REG register. The time stamp value is returned in the event, along with a time stamp push event code. The upper 32-bits (CPSW\_CPTS\_EVENT\_3\_REG register) of the timestamp are zero in 32-bit mode.

#### **12.2.2.4.7.13.2 Time Stamp Counter Rollover Event (32-bit mode only)**

The CPTS module contains a 32-bit time stamp value (CPSW\_CPTS\_EVENT\_0\_REG). The counter upper bits are maintained by host software. The rollover event indicates to software that the time stamp counter has rolled over from 0xFFFF FFFF to 0x0000 0000 and the software-maintained upper count value should be incremented. This event occurs only in 32-bit mode.

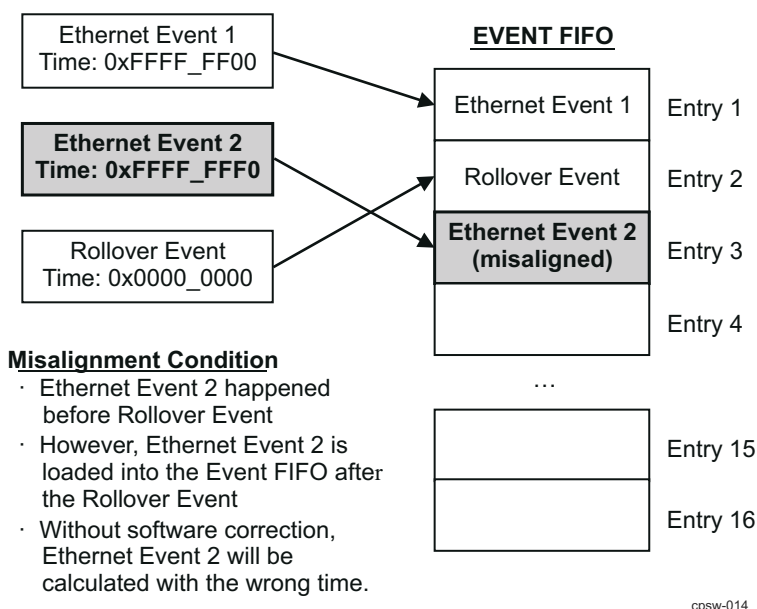
#### **12.2.2.4.7.13.3 Time Stamp Counter Half-rollover Event (32-bit mode only)**

The CPTS includes a time stamp counter half-rollover event. The half-rollover event indicates to software that the time stamp value (CPSW\_CPTS\_EVENT\_0\_REG[31:0] TIME\_STAMP) has incremented from 0x7FFF FFFF to 0x8000 0000. The half-rollover event is included to enable software to correct a misaligned event condition. This event occurs only in 32-bit mode.

The half-rollover event is included to enable software to determine the correct time for each event that contains a valid time stamp value, such as an Ethernet event. If an Ethernet event occurs around a counter rollover (full rollover), the rollover event could possibly be loaded into the event FIFO before the Ethernet event, even though the Ethernet event time was actually taken before the rollover. [Figure 12-176](#) shows a misalignment condition. This misaligned event condition arises because an Ethernet event time stamp occurs at the beginning of a packet and time passes before the packet is determined to be a valid synchronization packet. The misaligned event condition occurs if the rollover occurs in the middle, after the packet time stamp has been taken, but before the packet has been determined to be a valid time sync packet.

Host software must detect and correct for misaligned event conditions. For every event time stamp after a rollover and before a half-rollover, software must examine the time stamp most significant bit. If bit 31 of the time stamp value is low (0x0000 0000 through 0x7FFF FFFF), then the event time stamp was taken after the rollover and no correction is required. If the value is high (0x8000 0000 through 0xFFFF FFFF), the time stamp value was taken before the rollover and a misalignment is detected. The misaligned case indicates to software that it must subtract one from the upper count value stored in software to calculate the correct time for the misaligned event. The misaligned event occurs only on the rollover boundary and not on the half-rollover boundary. Software only needs to check for misalignment from a rollover event to a half-rollover event.

When a rollover occurs, software increments the software time stamp upper value. The misaligned case indicates to software that the misaligned event time stamp has a valid upper value that is pre-increment, so one must be subtracted from the upper value to allow software to calculate the correct time for the misaligned event.



**Figure 12-176. Event FIFO Misalignment Condition**

#### 12.2.2.4.7.13.4 Hardware Time Stamp Push Event

There are four hardware time stamp inputs ( CPTS\_HW[1:4]\_TS\_PUSH events) that can cause hardware time stamp push events to be loaded into the Event FIFO. Each time stamp input is mapped in the device as shown in [Figure 12-172](#). The event is loaded into the event FIFO on the rising edge of the timer, and the PORT\_NUMBER field in the CPSW\_CPTS\_EVENT\_1\_REG register indicates the hardware push input that caused the event (encoded).

The hardware time stamp inputs are asynchronous and are low frequency signals. The CPTS logic synchronizes and performs a rising edge detect on the incoming asynchronous input.

Each hardware time stamp input must be asserted for at least 10 periods of the selected CPTS\_RFT\_CLK clock. Each input can be enabled or disabled by setting the respective bits in the CPSW\_CPTS\_CONTROL\_REG register.

Hardware time stamps are intended to be an extremely low frequency signals, such that the event FIFO does not overrun. Software must keep up with the event FIFO and ensure that there is no overrun, or events will be lost.

#### 12.2.2.4.7.13.5 Ethernet Port Events

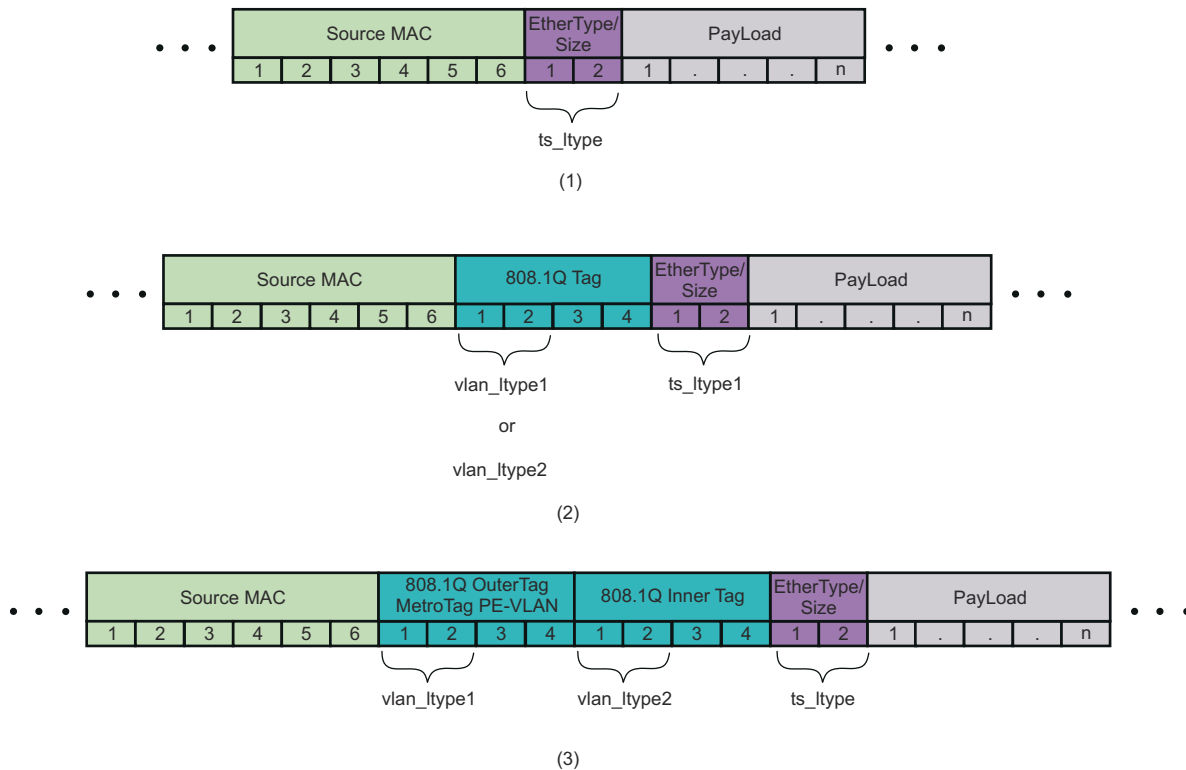
Packets transmitted or received on each Ethernet port can generate Ethernet Transmit Events or Ethernet Receive Events, respectively. The CPTS hardware will decode each packet to determine if it is a valid CPTS time sync event.

According to the IEEE 802.3 Ethernet standard, each Ethernet frame contains a 2-octet EtherType field to indicate which protocol is encapsulated in the PayLoad field, as shown in [Figure 12-177](#). For standard time sync packets, this will contain the EtherType for the Precision Time Protocol (IEEE 1588), which is defined as 0x88F7. The CPTS hardware will compare this field to the TS\_LTYPE1 field in the CPSW\_PN\_TS\_SEQ\_LTYPE\_REG\_k or the TS\_LTYPE2 field in CPSW\_PN\_TS\_CTL\_LTYPE2\_REG\_k register (depending on which enable bit was set) , which should also be programmed to 88F7h.

When a virtual LAN is used, an additional 4-octet 802.1Q tag is inserted in the Ethernet frame before the EtherType field, as shown in [Figure 12-177](#). To indicate to the CPTS hardware that a virtual LAN is in use, the TS\_TX\_VLAN\_LTYPE1\_EN (or TS\_TX\_VLAN\_LTYPE2\_EN) enable bit must be set in the

CPSW\_PN\_TS\_CTL\_REG\_k register. The EtherType for the 802.1Q tag is defined as 0x8100, and the CPTS hardware will compare this value to the TS\_VLAN\_LTYPE1 (or TS\_VLAN\_LTYPE2 depending on which enable bit was set) field in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG\_k register, which should also be programmed to 0x8100.

When two stacked VLANs are used, two additional 4-octet 801.Q tags are inserted in the Ethernet frame before the EtherType field, as shown in Figure 12-177. In this case, both TS\_VLAN\_LTYPE1 and TS\_VLAN\_LTYPE2 must be enabled. The outer tag must match the value of the TS\_VLAN\_LTYPE1 field, and the inner tag must match the value of the TS\_VLAN\_LTYPE2 field.



cpsw-015

**Figure 12-177. Partial Ethernet-II Frames Showing Register Mapping of EtherTypes for a Simple Frame (1), a Single 1Q Tag Added (2), and Two 1Q Tags Added (3)**

#### 12.2.2.4.7.13.5.1 Ethernet Port Receive Event

This section describes Ethernet port receive events. Ethernet port generates time synchronization events for valid received time sync packets. For every packet received on the Ethernet port, a timestamp will be captured by the receive module inside the CPTS for the corresponding port. The time stamp will be captured by the receive module regardless of whether or not the packet is a time synchronization packet to make sure that the time stamp is captured as soon as possible. The packet is sampled on both the rising and falling edges of the CPTS\_RFT\_CLK, and the time stamp will be captured once the start of frame delimiter for the receive packet is detected.

After the time stamp has been captured, the receive interface will begin parsing the packet to determine if it is a valid Ethernet time synchronization packet. The CPSW decoder determines if the packet is a valid Ethernet receive time synchronization event. The receive interface for the port will use the following criteria to determine if the packet is a valid Annex D, Annex E, or Annex F time synchronization Ethernet receive event:

#### Annex D (IPv4)

1. Receive annex D time sync is enabled (TS\_RX\_ANNEX\_D\_EN is set in the CPSW\_PN\_TS\_CTL\_REG\_k register).

2. One of the sequences below is true.
  - a. The first packet LTYPE matches 0x0800
  - b. The first packet LTYPE matches TS\_VLAN\_LTYPE1 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG\_k register and TS\_RX\_VLAN\_LTYPE1\_EN is set in the CPSW\_PN\_TS\_CTL\_REG\_k register and the second packet LTYPE matches 0x0800
  - c. The first packet LTYPE matches TS\_VLAN\_LTYPE2 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG\_k register and TS\_RX\_VLAN\_LTYPE2\_EN is set in the CPSW\_PN\_TS\_CTL\_REG\_k register and the second packet LTYPE matches 0x0800
  - d. The first packet LTYPE matches TS\_VLAN\_LTYPE1 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG\_k register and TS\_RX\_VLAN\_LTYPE1\_EN is set in the CPSW\_PN\_TS\_CTL\_REG\_k register and the second packet LTYPE matches TS\_VLAN\_LTYPE2 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG\_k register and TS\_RX\_VLAN\_LTYPE2\_EN is set in the CPSW\_PN\_TS\_CTL\_REG\_k register and the third packet LTYPE matches 0x0800
3. Byte 14 (the byte after the LTYPE) contains 0x45 (IPv4).

#### Note

The byte numbering assumes that there are no VLANs. The byte number is intended to show the relative order of the bytes.

4. Byte 20 contains 0bXXX00000 (5 lower bits zero) and Byte 21 contains 0x00 (fragment offset zero)
5. Byte 22 contains 0x01 (HOP Limit = 1) if the TS\_TTL\_NONZERO bit in the switch CPSW\_PN\_TS\_CTL\_LTYPE2\_REG\_k register is cleared to 0h, or byte 22 contains any value if CPSW\_PN\_TS\_CTL\_LTYPE2\_REG\_k is set to 1h. Byte 22 is the TTL/HOP field.
6. Byte 23 contains 0x11 (Next Header UDP Fixed).
7. The TS\_UNI\_EN bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG\_k register is cleared to 0h and Bytes 30 through 33 contain:
  - a. Decimal 224.0.1.129 and the TS\_129 bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG\_k register is set, or
  - b. Decimal 224.0.1.130 and the TS\_130 bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG\_k register is set, or
  - c. Decimal 224.0.1.131 and the TS\_131 bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG\_k register is set, or
  - d. Decimal 224.0.1.132 and the TS\_132 bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG\_k register is set, or
  - e. Decimal 224.0.0.107 and the TS\_107 bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG\_k register is set

-OR-

The TS\_UNI\_EN bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG\_k register is set and Bytes 30 through 33 contain any values.

8. Bytes 36 and 37 contain:
  - a. Decimal 0x01 and 0x3F respectively and the TS\_319 bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG\_k register is set -OR-
  - b. Decimal 0x01 and 0x40 respectively and the TS\_320 bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG\_k register is set.
9. The PTP message begins in byte 42.
10. The packet message type is enabled in the TS\_MSG\_TYPE\_EN field in the CPSW\_PN\_TS\_CTL\_REG\_k register.
11. The packet was received without error (not long/short/mac\_ctl/CRC/code/align).

#### Annex E (IPv6)

1. Receive annex E time sync is enabled (TS\_RX\_ANNEX\_E\_EN bit is set in the switch CPSW\_PN\_TS\_CTL\_REG\_k register).
2. One of the sequences below is true.
  - a. The first packet LTYPE matches 0x86dd.
  - b. The first packet LTYPE matches TS\_VLAN\_LTYPE1 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG\_k register and TS\_RX\_VLAN\_LTYPE1\_EN is set in the CPSW\_PN\_TS\_CTL\_REG\_k register and the second packet LTYPE matches 0x86dd

- c. The first packet LTYPE matches TS\_VLAN\_LTYPE2 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG\_k register and TS\_RX\_VLAN\_LTYPE2\_EN is set in the CPSW\_PN\_TS\_CTL\_REG\_k register and the second packet LTYPE matches 0x86dd
- d. The first packet LTYPE matches TS\_VLAN\_LTYPE1 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG\_k register and TS\_RX\_VLAN\_LTYPE1\_EN is set in the CPSW\_PN\_TS\_CTL\_REG\_k register and the second packet LTYPE matches TS\_VLAN\_LTYPE2 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG\_k register and TS\_RX\_VLAN\_LTYPE2\_EN is set in the CPSW\_PN\_TS\_CTL\_REG\_k register and the third packet LTYPE matches 0x86dd
3. Byte 14 (the byte after the LTYPE) contains 0x6X (IPv6).
4. Byte 20 contains 0x11 (UDP Fixed Next Header).
5. Byte 21 contains 0x01 (Hop Limit = 1) if the TS\_TTL\_NONZERO bit in the switch CPSW\_PN\_TS\_CTL\_LTYPE2\_REG\_k register is cleared to 0h, or byte 21 contains any value if TS\_TTL\_NONZERO is set to 1h. Byte 21 is the TTL/HOP field.
6. The TS\_UNI\_EN bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG\_k register is cleared to 0 and Bytes 38 through 53 contain:
  - a. FF0M:0:0:0:0:0:0:0:0181 and the TS\_129 bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG\_k register is set, or
  - b. FF0M:0:0:0:0:0:0:0:0182 and the TS\_130 bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG\_k register is set, or
  - c. FF0M:0:0:0:0:0:0:0:0183 and the TS\_131 bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG\_k register is set, or
  - d. FF0M:0:0:0:0:0:0:0:0184 and the TS\_132 bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG\_k register is set, or
  - e. FF0M:0:0:0:0:0:0:0:006B and the TS\_107 bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG\_k register is set

#### Note

All values above are 16-bit hex numbers where M is enabled in the TS\_MCAST\_TYPE\_EN field in the CPSW\_PN\_TS\_CTL\_REG\_k register.

-OR-

The TS\_UNI\_EN bit in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG\_k register is set to 1h and Bytes 38 through 53 contain any value.

7. Bytes 56 and 57 contain (UDP Header in bytes 54 through 61):
  - a. Decimal 0x01 and 0x3F respectively and the TS\_319 bit in the CPSW\_PN\_TS\_CTL2\_REG\_k register is set, or
  - b. Decimal 0x01 and 0x40 respectively and the TS\_320 bit in the CPSW\_PN\_TS\_CTL2\_REG\_k register is set.
8. The PTP message begins in byte 62.
9. The packet message type is enabled in the MSG\_TYPE\_EN field in the CPSW\_PN\_TS\_CTL2\_REG\_k register.
10. The packet was received without error (not long/short/mac\_ctl/CRC/code/align).

#### Annex F (IEEE 802.3)

1. Receive Annex F time sync is enabled (TS\_RX\_ANNEX\_F\_EN is set in the switch CPSW\_PN\_TS\_CTL\_REG\_k register).
2. One of the sequences below is true:
  - a. The first packet LTYPE matches TS\_LTYPE1 in the CPSW\_PN\_TS\_SEQ\_LTYPE\_REG\_k register. LTYPE 1 should be used when only one time sync LTYPE is to be enabled.
  - b. The first packet LTYPE matches TS\_LTYPE2 in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG\_k register and LTYPE2\_EN is set in the CPSW\_PN\_TS\_CTL\_REG\_k register.
  - c. The first packet LTYPE matches TS\_VLAN\_LTYPE1 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG\_k register and TS\_RX\_VLAN\_LTYPE1\_EN is set in the CPSW\_PN\_TS\_CTL\_REG\_k register and the second packet LTYPE matches TS\_LTYPE1 in the CPSW\_PN\_TS\_SEQ\_LTYPE\_REG\_k register

- d. The first packet LTYPE matches TS\_VLAN\_LTYPE1 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG\_k register and TS\_RX\_VLAN\_LTYPE1\_EN is set in the CPSW\_PN\_TS\_CTL\_REG\_k register and the second packet LTYPE matches TS\_LTYPE2 in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG\_k register and TS\_LTYPE2\_EN is set in the CPSW\_PN\_TS\_CTL\_REG\_k register.
- e. The first packet LTYPE matches TS\_VLAN\_LTYPE2 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG\_k register and TS\_RX\_VLAN\_LTYPE2\_EN is set in the CPSW\_PN\_TS\_CTL\_REG\_k register and the second packet LTYPE matches TS\_LTYPE1 in the CPSW\_PN\_TS\_SEQ\_LTYPE\_REG\_k register.
- f. The first packet LTYPE matches TS\_VLAN\_LTYPE2 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG\_k register and TS\_RX\_VLAN\_LTYPE2\_EN is set in the CPSW\_PN\_TS\_CTL\_REG\_k register and the second packet LTYPE matches TS\_LTYPE2 in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG\_k register and TS\_LTYPE2\_EN is set in the CPSW\_PN\_TS\_CTL\_REG\_k register.
- g. The first packet LTYPE matches TS\_VLAN\_LTYPE1 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG\_k register and TS\_RX\_VLAN\_LTYPE1\_EN is set in the CPSW\_PN\_TS\_CTL\_REG\_k register and the second packet LTYPE matches TS\_VLAN\_LTYPE2 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG\_k register and TS\_RX\_VLAN\_LTYPE2\_EN is set in the CPSW\_PN\_TS\_CTL\_REG\_k register and the third packet LTYPE matches TS\_LTYPE1 in the CPSW\_PN\_TS\_SEQ\_LTYPE\_REG\_k register.
- h. The first packet LTYPE matches TS\_VLAN\_LTYPE1 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG\_k register and TS\_RX\_VLAN\_LTYPE1\_EN is set in the CPSW\_PN\_TS\_CTL\_REG\_k register and the second packet LTYPE matches TS\_VLAN\_LTYPE2 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG\_k register and TS\_RX\_VLAN\_LTYPE2\_EN is set in the CPSW\_PN\_TS\_CTL\_REG\_k register and the third packet LTYPE matches TS\_LTYPE2 in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG\_k register and TS\_LTYPE2\_EN is set in the CPSW\_PN\_TS\_CTL\_REG\_k register
3. The PTP message begins in the byte after the LTYPE.
4. The packet message type is enabled in the TS\_MSG\_TYPE\_EN field in the CPSW\_PN\_TS\_CTL\_REG\_k register.
5. The packet was received without error (not long/short/mac\_ctl/CRC/code/align).

If all of the criteria described above are met for either Annex D, Annex E, or Annex F, and the packet is determined to be a valid time synchronization packet, then the RX interface will push an Ethernet receive event into the event FIFO.

#### 12.2.2.4.7.13.5.2 Ethernet Port Transmit Event

This section describes Ethernet port transmit events. For every packet transmitted on the Ethernet ports, the port transmit interface will begin parsing the packet to determine if it is a valid Ethernet time synchronization packet. The CPTS transmit interface for the port will use to the following criteria to determine if the packet is a valid time synchronization Ethernet transmit event. The CPSW decoder determines if the packet is a valid ethernet receive time synchronization event. To be a valid Ethernet transmit time synchronization event, the conditions listed below must be true for either Annex D, Annex E, or Annex F:

#### Annex D (IPv4)

1. Transmit time sync is enabled (TS\_TX\_ANNEX\_D\_EN is set in the CPSW\_PN\_TS\_CTL\_REG\_k register).
2. One of the sequences below is true.
  - a. The first packet LTYPE matches 0x0800
  - b. The first packet LTYPE matches TS\_VLAN\_LTYPE1 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG\_k register and TS\_TX\_VLAN\_LTYPE1\_EN is set in the CPSW\_PN\_TS\_CTL\_REG\_k register and the second packet LTYPE matches 0x0800
  - c. The first packet LTYPE matches TS\_VLAN\_LTYPE2 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG\_k register and TS\_TX\_VLAN\_LTYPE2\_EN is set in the CPSW\_PN\_TS\_CTL\_REG\_k register and the second packet LTYPE matches 0x0800
  - d. The first packet LTYPE matches TS\_VLAN\_LTYPE1 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG\_k register and TS\_TX\_VLAN\_LTYPE1\_EN is set in the CPSW\_PN\_TS\_CTL\_REG\_k register and the second packet LTYPE matches TS\_VLAN\_LTYPE2 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG\_k register and TS\_TX\_VLAN\_LTYPE2\_EN is set in the CPSW\_PN\_TS\_CTL\_REG\_k register and the third packet LTYPE matches 0x0800
3. Byte 14 (the byte after the LTYPE) contains 0x45 (IPv4).









second packet LTYPE matches TS\_VLAN\_LTYPE2 in the CPSW\_PN\_TS\_VLAN\_LTYPE\_REG\_k register and TS\_TX\_VLAN\_LTYPE2\_EN is set in the CPSW\_PN\_TS\_CTL\_REG\_k register and the third packet LTYPE matches TS\_LTYPE2 in the CPSW\_PN\_TS\_CTL\_LTYPE2\_REG\_k register and TS\_LTYPE2\_EN is set in the CPSW\_PN\_TS\_CTL\_REG\_k register

3. The packet message type is enabled in the TS\_MSG\_TYPE\_EN field in the CPSW\_PN\_TS\_CTL\_REG\_k register.
4. The packet was sent by host port 0.

If all of the criteria described above are met, and the packet is determined to be a valid time synchronization packet, then the time stamp for the transmit event will not be generated until the start of frame delimiter of the packet is actually transmitted. The start of frame delimiter will be sampled on every rising and falling edge of the CPTS\_RFT\_CLK. Once the packet is transmitted, then the TX interface will push an Ethernet transmit event into the event FIFO.

**Table 12-231. Values of Message Type Field**

Message Type	Value (hex)
Sync	0
Delay_Req	1
Pdelay_Req	2
Pdelay_Resp	3
Reserved	4:7
Follow_Up	8
Delay_Resp	9
Pdelay_Resp_Follow_Up	A
Announce	B
Signaling	C
Management	D
Reserved	E:F

Once a transmitted or received packet is determined to be a valid time sync packet, the Ethernet Transmit Event or Ethernet Receive Event is loaded onto the Event FIFO.

The CPSW\_CPTS\_EVENT\_1\_REG register contains the Message Type and Sequence ID values from the original time sync packet. The CPSW\_CPTS\_EVENT\_0\_REG (and CPSW\_CPTS\_EVENT\_3\_REG) register contains the time stamp value when the packet arrived at the corresponding port.

#### 12.2.2.4.7.13.5.3

**Table 12-232. Values of Message Type Field**

Message Type	Value (hex)
Sync	0
Delay_Req	1
Pdelay_Req	2
Pdelay_Resp	3
Reserved	4:7
Follow_Up	8
Delay_Resp	9
Pdelay_Resp_Follow_Up	A
Announce	B
Signaling	C
Management	D
Reserved	E:F

Once a transmitted or received packet is determined to be a valid time sync packet, the Ethernet Transmit Event or Ethernet Receive Event is loaded onto the Event FIFO.

The CPSW\_CPTS\_EVENT\_1\_REG register contains the Message Type and Sequence ID values from the original time sync packet. The CPSW\_CPTS\_EVENT\_0\_REG (and CPSW\_CPTS\_EVENT\_3\_REG) register contains the time stamp value when the packet arrived at the corresponding port.

#### 12.2.2.4.7.14 Timestamp Compare Event

##### Note

Timestamp compare events are generated for non-toggle mode only.

The CPTS can generate an event for a time stamp comparison in 32-bit or 64-bit mode.

##### 12.2.2.4.7.14.1 32-Bit Mode

The CPTS\_COMP output is also asserted when the event is generated. The event is generated when the 32-bit time stamp value (CPSW\_CPTS\_EVENT\_0\_REG) compares with the CPSW\_CPTS\_TS\_COMP\_VAL\_REG register and the CPSW\_CPTS\_TS\_COMP\_LEN\_REG value is non-zero. The CPSW\_CPTS\_TS\_COMP\_LEN\_REG value should be written by software after the CPSW\_CPTS\_TS\_COMP\_VAL\_REG register is written and should be zero when the comparison value is written.

##### 12.2.2.4.7.14.2 64-Bit Mode

The CPTS\_COMP output is also asserted when the event is generated. The event is generated when the 64-bit time stamp value (CPSW\_CPTS\_EVENT\_0\_REG and CPSW\_CPTS\_EVENT\_3\_REG) compares with the CPSW\_CPTS\_TS\_COMP\_VAL\_REG and CPSW\_CPTS\_TS\_COMP\_HIGH\_VAL\_REG registers and the CPSW\_CPTS\_TS\_COMP\_LEN\_REG value is non-zero. The CPSW\_CPTS\_TS\_COMP\_LEN\_REG value should be written by software after the CPSW\_CPTS\_TS\_COMP\_VAL\_REG register is written and should be zero when the comparison value is written.

##### 12.2.2.4.7.15 Host Transmit Event

The host can send a packet to be transmitted on an Ethernet port that will generate a time synchronization event. The host sets the TSTAMP\_EN bit and sends the DOMAIN, MESSAGE\_TYPE, and SEQUENCE\_ID in the additional control information that resides in the protocol specific section of the descriptor that is transmitted to the CPSW\_9G. An event is then generated and placed on the event FIFO once the packet is transmitted. Host events allow the user to timestamp exactly when a software generated packet exits the device.

##### 12.2.2.4.7.16 CPTS Interrupt Handling

When an event is push onto the Event FIFO, an interrupt can be generated to indicate to software that a time sync event occurred. The following steps should be taken to process time sync events using interrupts:

1. Enable the TS\_PEND interrupt by setting the TS\_PEND\_EN bit of the CPSW\_CPTS\_INT\_ENABLE\_REG register.
2. Upon interrupt, read the CPSW\_CPTS\_EVENT\_0\_REG through CPSW\_CPTS\_EVENT\_3\_REG registers values.
3. Set the CPSW\_CPTS\_EVENT\_POP\_REG[0] EVENT\_POP bit to 1h to pop the previously read value off of the event FIFO.
4. Process the interrupt as required by the application software.

Software has the option of processing more than a single event from the event FIFO in the interrupt service routine in the following way:

1. Enable the TS\_PEND interrupt by setting the TS\_PEND\_EN bit of the CPSW\_CPTS\_INT\_ENABLE\_REG
2. Upon interrupt, enter the CPTS service routine.
3. Read the CPSW\_CPTS\_EVENT\_0\_REG through CPSW\_CPTS\_EVENT\_3\_REG registers values.

4. Set the CPSW\_CPTS\_EVENT\_POP\_REG[0] EVENT\_POP bit to 1h to pop the previously read value off of the event FIFO.
5. Wait for an amount of time greater than four CPTS\_RFT\_CLK periods plus four CPPI\_ICLK periods.
6. Read the TS\_PEND\_RAW bit in the CPSW\_CPTS\_INTSTAT\_RAW\_REG register to determine if another valid event is in the event FIFO. If bit TS\_PEND\_RAW is asserted, go to step 3. If bit TS\_PEND\_RAW is not asserted proceed with step 7.
7. Process the interrupt(s) as required by the application software.

Software also has the option of disabling the interrupt and polling the TS\_PEND\_RAW bit of the CPSW\_CPTS\_INTSTAT\_RAW\_REG register to determine if a valid event is on the event FIFO.

#### 12.2.2.4.8 CPPI Streaming Packet Interface

The receive streaming interface on port 0 of the CPSW is responsible for receiving packet for Ethernet egress data from the packet streaming switch in the NAVSS. The CPPI receive port is equivalent to an Ethernet port with the difference being that the data is provided to the CPSW in the 128-bit streaming interface data format instead RGMII data format.

In addition to the packet data, the receive streaming interface also can provide additional control information that resides in the information words of the descriptor that was transmitted to the CPSW.

The tables below show the information that may be passed along with which descriptor information word to put it in.

##### 12.2.2.4.8.1 Port 0 CPPI Transmit Packet Streaming Interface (CPSW\_9G Egress)

The CPSW2G has a single transmit packet streaming interface. All Ethernet packet data destined for the host (Port 0) is transferred on the transmit packet streaming interface. The transmit packet streaming interface is equivalent to an Ethernet MAC output with the difference being that 128-bit streaming interface data is output. Egress packet data is packed on the 128-bit data bus with all words having 16-bytes except possibly the last packet data word which is the word previous to the EOP word. INFO Word 0–3 is transferred on SOP. The EOP word contains the packet status 0–3 (data type 24) which includes the timestamp and checksum data. Packets are not dropped on the transmit streaming interface due to pushback, but packets may be dropped in the associated priority FIFO.

INFO Word 0–3 and Status Data Word 0–3 (on EOP) are the only non-payload data word types that are transferred. Long packets are truncated at the CPSW\_PN\_RX\_MAXLEN\_REG\_k[13-0] RX\_MAXLEN byte value of the ingress port (only the CPSW\_PN\_RX\_MAXLEN\_REG\_k number of bytes are kept if long packets are transferred due to CPSW\_PN\_MAC\_CONTROL\_REG\_k register copy error frames set - RX\_CEF\_EN). MAC control frames are only transferred if the receiving Ethernet port has the CPSW\_PN\_MAC\_CONTROL\_REG\_k[24] RX\_CMF\_EN bit set.

The error encoding on the TXST\_PKT\_ERR[3:0] output is shown in [Table 12-233](#):

**Table 12-233. Error Encoding on the TXST\_PKT\_ERR[3:0] Output**

TXST_PKT_ERR[3:0]	Description
0000	No Error
0001	CRC Error
0010	Code or alignment error
0011	Short (no code/align/crc error)
0100	FragCRC (short with crc error)
0101	Frag Code/Align (short with code/align error)
0110	Long
0111	Jabber CRC (long with crc error)
1000	Jabber Code/Align (long with code/align error)
1001	Mac ontrl packet (CPSW_PN_MAC_CONTROL_REG[24] RX_CMF_EN set on ingress Ethernet port)
1010	Mac control CRC

**Table 12-233. Error Encoding on the TXST\_PKT\_ERR[3:0] Output (continued)**

TXST_PKT_ERR[3:0]	Description
1011	Mac control Code/Align
1100	Mac control short/frag (short MAC control frame with CRC/Code/Align)
1101	Mac control long/jabber (long MAC control frame with CRC/Code/Align)
1110	Reserved
1111	Reserved

The CPPI Port 0 transmit packet streaming interface has a single output thread. If a packet transmission has begun then the entire packet will be transmitted before the next packet is sent (packet data from multiple packets are not interleaved on the transmit streaming interface). The default egress flow is the port number minus 1, concatenated with the 3-bit Port 0 transmit FIFO hardware switch priority. For example, a packet that was received on port 4 with a Port 0 transmit FIFO hardware switch priority of 5 would be sent on flow 29 (decimal) or flow 0b0011101 (binary). Priority remapping on ingress and Port 0 transmit egress effects the output flow.

INFO Words () are a contiguous block of four 32-bit data words aligned on a 32-bit word boundary.

**Figure 12-178. TX INFO Word 0 Format**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PKT_TYPE					RESERVED			PASS_CRC	CRC_T YPE	RESERVED					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								FLOW_ID							

Bit	Field	Description
31-27	PKT_TYPE	Always set to 0b00111. Host PD (Packet Descriptor) Word 2. Packet Type: bits[31-27].
26-24	RESERVED	Reserved.
23	PASS_CRC	This bit is cleared to zero (no CRC passed) when the P0_TX_CRC_REMOVE bit in the CPSW_CONTROL_REG register is set (and the egress packet has no errors). When the remove bit is cleared to zero then this bit is cleared and no CRC is passed with the output packet. The packet length includes the CRC if it is present.
22	CRC_TYPE	The packet CRC type. The type of CRC passed is determined by CRC_TYPE field in the CPSW_PN_MAC_CONTROL_REG_k register (not by the type of CRC the packet had on Ethernet port ingress). Host PD Word 1. Protocol Specific Flags: bits[27-24]. 0h: Ethernet CRC 1h: Castagnoli CRC
21-8	RESERVED	Reserved.
7-0	FLOW_ID	This is the packet output transmit streaming interface flow. The default flow ID can be overridden by ALE classification (Thread mapping). The switch default flow is the 3-bit "From Port" value concatenated with the 3-bit "Switch Priority" {From_Port[2:0], Switch_Priority[2:0]} as shown below: Host PD (Packet Descriptor) Word 1. Flow ID: bits[13-0]. 0h: The packet was received on Ethernet port 1 1h: The packet was received on Ethernet port 2 Switch Priority – The actual hardware switch priority that the packet was stored in on the CPPI transmit FIFO.

**Figure 12-179. TX INFO Word 1 Format**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

**Figure 12-179. TX INFO Word 1 Format (continued)**

0x4 (fixed_ps_size)												0			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		PKT_LENGTH													
Bit	Field		Description												
31-20	FIXED_PS_SIZE		Fixed packet size: 0x4												
19-14	RESERVED		Reserved.												
13-0 (Host PD (Packet Descriptor) Word 1. Packet Length: bits[ 21-0])	PKT_LENGTH		Specifies the number of bytes in the entire packet. Offset bytes are not included. Valid only on SOP. The packet length must be greater than zero. The packet data will be truncated to the packet length if the packet length is shorter than the sum of the packet buffer descriptor buffer lengths. A host error occurs if the packet length is greater than the sum of the packet buffer descriptor buffer lengths.												

**Figure 12-180. TX INFO Word 2 Format**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFFFF															

**Figure 12-181. TX INFO Word 3 Format**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								SRC_ID							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
Bit	Field		Description												
31-24	RESERVED		Reserved.												
23-16	SRC_ID		The packet SRC_ID value comes from the PORT1 field in the CPSW_P0_SRC_ID_A_REG register. (src_tag) PD (Packet Descriptor) Word 3. Source Tag Low bits[23-16] if RFLOW[a]_RFC.rx_src_tag_lo_sel = 0x4 or (src_tag) PD (Packet Descriptor) Word 3. Source Tag High bits[31-24] if RFLOW[a]_RFC.rx_src_tag_hi_sel = 0x4												
15-0	RESERVED		Reserved.												

### Note

TX Status Data Word [0..3] are mapped to Host Packet Descriptor Protocol Specific Words if RFLOW[a]\_RFA.rx\_psinfo\_present = 1 and RFLOW[a]\_RFA.rx\_ps\_location = 0

**Figure 12-182. TX Status Data Word 0 Format**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMESTAMP[31:0]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMESTAMP[31:0]															

Bit	Field	Description
31-0	TIMESTAMP[31:0]	Contains the lower 32-bits of the time stamp value.

**Figure 12-183. TX Status Data Word 1 Format**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMESTAMP[63:32]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMESTAMP[63:32]															

Bit	Field	Description
31-0	TIMESTAMP[63:32]	Contains the upper 32-bits of the time stamp value.

**Figure 12-184. TX Status Data Word 2 Format**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED											IPV4_VALID	IPV6_VALID	TCP_UDP_N	FRAGMENT	CHECKSUM_ERROR
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHECKSUM_ADD															

Bit	Field	Description
31-21	RESERVED	Reserved.
20	IPV4_VALID	An IPV4 TCP or UDP Packet was detected.
19	IPV6_VALID	An IPV6 TCP or UDP Packet was detected.
18	TCP_UDP_N	Valid only when either the IPV4_VALID or IPV6_VALID bits are set. 0h: Indicates UDP packet was detected. 1h: Indicates TCP packet was detected.
17	FRAGMENT	Indicates that an IP fragment was detected. Valid only when either the IPV4_VALID or IPV6_VALID bits are set.
16	CHECKSUM_ERROR	Valid only when either the IPV4_VALID or IPV6_VALID bits are set.
15-0	CHECKSUM_ADD	This is the value that was summed during the checksum computation. This value is FFFFh for IPV4/6 UDP/TCP packets with no checksum error.

**Figure 12-185. TX Status Data Word 3 Format**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															

Bit	Field	Description
31-0	RESERVED	Reserved.

#### 12.2.2.4.8.2 CPPI Receive Packet Streaming Interface (CPSW Ingress)

Info Word 0/1/2/3 (INFO1 and INFO3 are ignored) are transferred on SOP. If a timestamp word (Extended Packet Info Word 0/1/2/3) is to be transferred it must be after SOP and before any packet data is transferred. Any following non-data type words will be dropped. The EOP word must be payload data.

Input receive packets cannot be aborted by the host. The INFO Word bit descriptions and Extended Packet INFO Word bit descriptions are shown below. The PASS\_CRC bit indicates that the CRC is passed with the packet data. Packets that have a passed CRC that is an error CRC will be output on the Ethernet port with at least one CRC byte inverted to indicate the error if P0\_RX\_PASS\_CRC\_ERR bit is set, otherwise they are dropped. The packet is a directed packet when any of the TO\_PORT bits are nonzero. A packet may be directed only to a single port. The packet will be sent to the port number indicated. For directed packets the lookup process is skipped to determine the destination. However, in vlan aware mode (when VLAN\_AWARE bit in the CPSW\_CONTROL\_REG register is set to 1h) the lookup is performed to determine untagged egress. Packets longer than the value in CPSW\_P0\_RX\_MAXLEN\_REG[13:0] RX\_MAXLEN bit field are dropped. Packets shorter than 60-Bytes are padded to 64-Bytes (after adding pad and CRC) if P0\_RX\_PAD bit in the CPSW\_CONTROL\_REG register is set and if PASS\_CRC is clear, otherwise they are dropped. This means that packets shorter than 64-Bytes are dropped if the PASS\_CRC info bit is set regardless of P0\_RX\_PAD bit (packets are padded only if they are short and do not have CRC).

A RX INFO word () is a contiguous block of four 32-bit data words aligned on a 32-bit word boundary.

### Note

RX Control Data Words [0..2] are mapped to Host Packet Descriptor Protocol Specific Words if (TCHAN[a]\_TCFG.tx\_filt\_pswords = 0) and (Host PD Word 1.Protocol Specific Region Location.bit[28] = 0h) and (Host PD Word 1.Protocol Specific Valid Word Count.bits[22-27] = 4h)

**Figure 12-186. RX INFO Word 0 Format**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED								PASS_CRC	CRC_T YPE	RESERVED					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															

Bit	Field	Description
31-24	RESERVED	Reserved.
23	PASS_CRC	The PASS_CRC bit indicates that the CRC is passed with the packet data. 0h: CRC is not passed with packet (CRC_TYPE is don't care) 1h: CRC of type CRC_TYPE is passed with the packet.
22 (Host PD (Packet Descriptor) Word 1. Protocol Specific Flags: bits[27-24])	CRC_TYPE	CRC Type 0h: Ethernet CRC 1h: Castagnoli CRC
21-0	RESERVED	Reserved.

**Figure 12-187. RX INFO Word 2 Format**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED											TO_PORT				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															

Bit	Field	Description
31-21	RESERVED	Reserved.

Bit	Field	Description
20-16 (Host PD (Packet Descriptor) Word 3. Dest Tag Low bits[8-0])	TO_PORT	Port number to send the directed packet to. This field is set by the host. This field is valid on SOP. Directed packets go to the directed port, but an ALE lookup is performed to determine untagged egress in VLAN_AWARE mode. 0h: Not directed 1h: Send the packet to port 1.
15-0	RESERVED	Reserved.

**Figure 12-188. RX Control Data Word 1 Format**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMES TAMP_ EN	RESERVED				DOMAIN								MSG_TYPE		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEQUENCE_ID															

Bit	Field	Description
31	TIMESTAMP_EN	When set, this bit indicates that the packet will generate a timesync event on Ethernet egress (if the CPTS is configured properly) with the associated DOMAIN, MSG_TYPE, and SEQUENCE_ID.
30-28	RESERVED	Reserved.
27-20	DOMAIN	Timesync domain.
19-16	MSG_TYPE	Timesync message type.
15-0	SEQUENCE_ID	Timesync sequence ID.

**Figure 12-189. RX Control Data Word 2 Format**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CHECKSUM_RESULT								CHECKSUM_START_BYTE							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHEC KSUM _INV	RESE RVED	CHECKSUM_BYTECOUNT													

Bit	Field	Description
31-24	CHECKSUM_RESULT	This is the packet byte number where the checksum result will be placed in the egress packet. The first packet byte which is the first byte of the destination address is Byte 1 (not byte zero).
23-16	CHECKSUM_START_BYTE	This is the packet byte number to start the checksum calculation on. The first packet byte is Byte 1.
15	CHECKSUM_INV	When set, a zero checksum value will be inverted and sent as FFFFh.
14	RESERVED	Reserved.
13-0	CHECKSUM_BYTECOUNT	This is the number of bytes to calculate the checksum on. The outgoing Ethernet packet will have a checksum inserted when this value is non-zero.

Other INFO words are not taken into account.



### 12.2.2.4.8.3 CPPI Checksum Offload

The CPPI host port can be enabled to perform checksum offload on host port packet ingress and egress. UDP (User Datagram Protocol) and TCP (Transmission Control Protocol) over IPV4 and IPV6 are supported. For the purposes of checksum description, the first packet byte (the first byte of the destination address) is byte 1 (not byte 0). That is, a 64 byte packet goes from byte 1 to byte 64. For all packet types, the S\_CN\_SWITCH bit in the CPSW\_CONTROL\_REG register must be set for the Outer VLAN L type to be supported.

#### 12.2.2.4.8.3.1 CPPI Transmit Checksum Offload

IPV4 and IPV6 UDP and TCP packets that are received on any Ethernet port and destined for port 0 egress are checked for correct checksum as described below. The byte counts below are shown for packets with no VLAN's. The byte counts vary with one or two packet VLAN's. Packets received on an Ethernet port with errors are not checked for a correct checksum if they are passed to the host.

##### 12.2.2.4.8.3.1.1 IPV4 UDP

- Byte 15 Upper Nibble = 4 for IPV4
- Byte 15 Lower Nibble = IHL - Nibble with number of 32-bit words in IPV4 header (5 to 15 supported).
- Bytes 20-21 = fragment[15-0] – Bit 13 is the MF bit and bits [12-0] are the Fragment offset. A packet is a fragment if the MF bit is set or if the fragment offset is non-zero. The first packet fragment has MF=1 with a zero offset. Middle fragments have MF=1 with a nonzero offset. The last packet fragment has MF=0 with a nonzero offset. Non-fragmented packets have MF=0 and a zero offset. A count is output for packet fragments but no errors are reported. First fragments have the UDP header included in the count. Middle and last fragments have only data included in the count (there is no UDP header).
- Byte 24 = 0x11 for UDP protocol.
- Received packet UDP checksum of zero means that there is no IPV4 checksum sent with the packet so no error will be issued.
- Received packet UDP checksum of 0xFFFF means that the checksum was calculated to be 0xFFFF or 0x0000 but was sent in the transmitted packet as 0xFFFF by the sending originating entity.

##### 12.2.2.4.8.3.1.2 IPV4 TCP

- Byte 15 Upper Nibble = 4 for IPV4
- Byte 15 Lower Nibble = IHL - Nibble with number of 32-bit words in IPV4 header (5 to 15 supported).
- Bytes 20-21 = fragment[15-0] – Bit 13 is the MF bit and bits [12-0] are the Fragment offset. A packet is a fragment if the MF bit is set or if the fragment offset is non-zero. The first packet fragment has MF=1 with a zero offset. Middle fragments have MF=1 with a nonzero offset. The last packet fragment has MF=0 with a nonzero offset. Non-fragmented packets have MF=0 and a zero offset. A count is output for packet fragments but no errors are reported. First fragments have the UDP header included in the count. Middle and last fragments have only data included in the count (there is no TCP header).
- Byte 24 = 0x06 for TCP protocol.

##### 12.2.2.4.8.3.1.3 IPV6 UDP

- Byte 15 upper nibble = 6 for IPV6.
- Byte 21 = 0x11 for UDP protocol as next header.
- Fragment extension headers are supported. First fragments have a fragment extension header (byte 21 = 0x2C) followed by a UDP header (byte 55 = 0x11). Middle and last fragments have a fragment extension header followed by data only (no UDP header). The first packet fragment has MF=1 with a zero offset. Middle fragments have MF=1 with a nonzero offset. The last packet fragment has MF=0 with a nonzero offset. Non-fragmented packets do not have a fragment extension header. A count is output for packet fragments but no errors are reported.
- Received packet UDP checksum of zero means that there is no IPV6 checksum sent with the packet so no error will be issued.
- Received packet UDP checksum of 0xFFFF means that the checksum was calculated to be 0xFFFF or 0x0000 but was sent in the transmitted packet as 0xFFFF by the sending originating entity.

**12.2.2.4.8.3.1.4 IPV6 TCP**

- Byte 15 upper nibble = 6 for IPV6.
- Byte 21 = 0x06 for TCP protocol as next header.
- Fragment extension headers are supported. First fragments have a fragment extension header (byte 21 = 0x2C) followed by a UDP header (byte 55 = 0x06). Middle and last fragments have a fragment extension header followed by data only (no TCP header). The first packet fragment has MF=1 with a zero offset. Middle fragments have MF=1 with a nonzero offset. The last packet fragment has MF=0 with a nonzero offset. Non-fragmented packets do not have a fragment extension header. A count is output for packet fragments but no errors are reported.

**12.2.2.4.8.4 CPPI Receive Checksum Offload**

Packets sent from host port 0 (switch ingress) to any Ethernet port can have a checksum calculated and inserted into the Ethernet egress packet. The RX\_CHECKSUM\_EN bit in the CPSW\_P0\_CONTROL\_REG register must be set for receive checksum operation to be enabled. When bit RX\_CHECKSUM\_EN is enabled, Control Data Word 2 input on CPPI receive PSI interface determines how the checksum is calculated. The CHECKSUM\_RESULT field in Control Data Word 2 determines where the checksum is inserted. The checksum result location is adjusted by the egress port if a VLAN is to be inserted or removed on Ethernet port egress.

**12.2.2.4.8.5 Egress Packet Operations**

Each CPSW egress port (Ethernet and Host) is capable of performing egress packet processing operations (CPSW\_ALE\_EGRESSOP). IntraVLAN processing either adds, removes, or replaces VLAN information or does nothing. InterVLAN routing allows hardware routing between a limited number of VLANs - thereby allowing high-bandwidth or other routing operations to be offloaded from software to the CPSW (hardware). IntraVLAN processing and InterVLAN routing operations are mutually exclusive. In addition, the packet source and destination addresses can be swapped on egress to facilitate OAM or generic testing operations.

#### 12.2.2.4.9 MII Management Interface (MDIO)

The MII Management interface module implements the 802.3 serial management interface to interrogate and control external Ethernet PHY using a two-wire bus.

##### 12.2.2.4.9.1 MDIO Frame Formats

Table 12-234 shows the address, Table 12-235 shows the read format and Table 12-236 shows the write format of the supported Clause 45 MII Management interface frames. Post-increment accesses are not supported.

**Table 12-234. MDIO Clause 45 Address Frame Format**

Pre-amble	Start Delimiter	Operation Code	PHY Address	MMD Number	Turnaround	Data
FFFF FFFFh	00	00	AAAAA	RRRRR	10	AAAA.AAAA.AAAA.AAAA

**Table 12-235. MDIO Clause 45 Read Frame Format**

Pre-amble	Start Delimiter	Operation Code	PHY Address	MMD Number	Turnaround	Data
FFFF FFFFh	00	11	AAAAA	RRRRR	Z0	DDDD.DDDD.DDDD.DDDD

**Table 12-236. MDIO Clause 45 Write Frame Format**

Pre-amble	Start Delimiter	Operation Code	PHY Address	MMD Number	Turnaround	Data
FFFF FFFFh	00	01	AAAAA	RRRRR	10	DDDD.DDDD.DDDD.DDDD

The default or idle state of the two wire serial interface is a logic one. All tri-state drivers should be disabled and the PHY's pull-up resistor will pull the MDIO line to a logic 1. Prior to initiating any other transaction, the station management entity shall send a preamble sequence of 32 contiguous logic 1 bits on the MDIO line with 32 corresponding cycles on MDCLK to provide the PHY with a pattern that it can use to establish synchronization. A PHY shall observe a sequence of 32 contiguous logic one bits on MDIO with 32 corresponding MDCLK cycles before it responds to any other transaction. The MDIO CPSW\_MDIO\_USER\_ADDR0\_REG register must be written before a read or write operation is performed to set the address used in the operation. Each read or write operation has a preceeding address frame.

#### Preamble

The start of a frame is indicated by a preamble, which consists of a sequence of 32 contiguous bits all of which are a 1. This sequence provides the PHY a pattern to use to establish synchronization. The preamble is required in clause 45 operation.

#### Start Delimiter

The preamble is followed by the start delimiter which is indicated by a 00 pattern.

#### Operation Code

The operation code for an address transaction is 00. The operation code for a read is 11, while the operation code for a write is a 01.

#### PHY Address

The PHY address is 5 bits allowing 32 unique values. The first bit transmitted is the MSB of the PHY address.

#### MMD Number

The MMD number is the 5 bits allowing 32 unique values. The first bit transmitted is the MSB.

#### Turnaround

An idle bit time during which no device actively drives the MDIO signal shall be inserted between the register address field and the data field of a read frame in order to avoid contention. During a read frame, the PHY shall

drive a zero bit onto MDIO for the first bit time following the idle bit and preceding the Data field. During a write frame, this field shall consist of a one bit followed by a zero bit.

### Address

The address field is 16 bits on address operations. The first bit transmitted is the MSB of the address word. Each read/write operation initiated has an automatic address operation initiated first that uses the MDIO CPSW\_MDIO\_USER\_ADDR0\_REG/ CPSW\_MDIO\_USER\_ADDR1\_REG register values as the 16-bit address.

### Data

The Data field is 16 bits on read and write operations. The first bit transmitted and received is the MSB of the data word.

#### 12.2.2.4.9.2 MDIO Functional Description

The MII Management I/F will remain idle until enabled by setting the ENABLE bit in the CPSW\_MDIO\_CONTROL\_REG register. The MII Management I/F will then continuously poll the link status from within the Generic Status Register of all possible 32 PHY addresses in turn recording the results in the MDIO CPSW\_MDIO\_LINK\_REG register. Individual PHY's can be enabled or disabled for polling the associated bit in the CPSW\_MDIO\_POLL\_EN\_REG register. The CPSW\_MDIO\_LINK\_REG and CPSW\_MDIO\_ALIVE\_REG register bit values are updated on the poll of each PHY. The LINKSEL bit in the CPSW\_MDIO\_USER\_PHY\_SEL\_REG\_k register determines the status input that is used. A change in the link status of the two PHYs being monitored will set the appropriate bit in the MDIO CPSW\_MDIO\_LINK\_INT\_RAW\_REG register and the MDIO CPSW\_MDIO\_LINK\_INT\_MASKED\_REG register, if enabled by the LINKINT\_ENABLE bit in the CPSW\_MDIO\_USER\_PHY\_SEL\_REG\_k register.

The MDIO CPSW\_MDIO\_ALIVE\_REG register is updated by the MII Management I/F module if the PHY acknowledged the read of the generic status register. In addition, any PHY register read transactions initiated by the host also cause the MDIO CPSW\_MDIO\_ALIVE\_REG register to be updated.

At any time, the host can define a transaction for the MII Management interface module to undertake using the DATA, PHYADR, REGADR, and WRITE fields in a CPSW\_MDIO\_USER\_ACCESS\_REG\_k register. When the host sets the GO bit in this register, the MII Management interface module will begin the transaction without any further intervention from the host. Upon completion, the MII Management interface will clear the GO bit and set the USERINTRAW field in the CPSW\_MDIO\_USER\_INT\_RAW\_REG register corresponding to the CPSW\_MDIO\_USER\_ACCESS\_REG\_k register being used. The corresponding bit in the CPSW\_MDIO\_USER\_INT\_MASKED\_REG register may also be set depending on the mask setting in the MDIO CPSW\_MDIO\_USER\_INT\_MASK\_SET\_REG and CPSW\_MDIO\_USER\_INT\_MASK\_CLEAR\_REG registers. A round-robin arbitration scheme is used to schedule transactions that may be queued by the host in different CPSW\_MDIO\_USER\_ACCESS\_REG\_k registers. The host should check the status of the GO bit in the MDIO CPSW\_MDIO\_USER\_ACCESS\_REG\_k register before initiating a new transaction to ensure that the previous transaction has completed. The host can use the ACK bit in the MDIO CPSW\_MDIO\_USER\_ACCESS\_REG\_k register to determine the status of a read transaction.

It is necessary for software to use the MII Management interface module to setup the auto-negotiation parameters of each PHY attached to a MAC port, retrieve the negotiation results, and setup the CPSW\_PN\_MAC\_CONTROL\_REG\_k register in the corresponding MAC.

#### 12.2.2.5 CPSW0 Programming Guide

##### 12.2.2.5.1 Initialization and Configuration of CPSW Subsystem

To configure the CPSW Ethernet Subsystem for operation, the host must perform the following:

1. Select the Interface (RMII, RGMII, SGMII, QSGMII or QSGMII\_SUB ) Mode.  
See CTRL\_MMR0\_ENET1\_CTRL[2-0] PORT\_MODE\_SEL to CTRL\_MMR0\_ENET8\_CTRL[2-0] PORT\_MODE\_SEL registers.
2. Configure pads (pin muxing), as per the interface selected. Refer to *Pad Configuration Registers* and the device-specific Datasheet.
3. Enable the CPSW Ethernet Subsystem clocks. See *CPSW Integration*

4. Ensure that at least 2000 CPPI\_ICLK periods are run after reset is de-asserted.
5. Configure the CPSW\_CONTROL\_REG register
6. Configure the Ethernet Port Source Address registers (CPSW\_PN\_SA\_L\_REG\_k and CPSW\_PN\_SA\_H\_REG\_k)
7. Configure the CPSW statistic port enable register CPSW\_STAT\_PORT\_EN\_REG
8. Configure the ALE ([Section 12.2.2.4.6.1, Address Lookup Engine](#))
9. Configure the MDIO ([Section 12.2.2.5.3.1, Initializing the MDIO Module](#))
10. Configure Ethernet port, as per the desired mode of operations

#### **12.2.2.5.2 Ethernet MAC Reset or XGMII/GMII Mode Change Configuration**

To reset the Ethernet port, the host must perform the following:

1. Set CMD\_IDLE bit to 1h in the Ethernet port control registers: CPSW\_PN\_MAC\_CONTROL\_REG\_k.
2. Wait for IDLE bit to be set to 1h, which is indicated in the Ethernet port status registers: CPSW\_PN\_MAC\_STATUS\_REG\_k.
3. Set SOFT\_RESET bit to 1h in the Ethernet port software reset registers: CPSW\_PN\_MAC\_SOFT\_RESET\_REG\_k.
4. Wait for SOFT\_RESET bit in the CPSW\_PN\_MAC\_SOFT\_RESET\_REG\_k registers to be cleared to confirm reset completion.
5. Configure the Ethernet ports.

#### **12.2.2.5.3 MDIO Software Interface**

##### **12.2.2.5.3.1 Initializing the MDIO Module**

The following steps are performed by the application software or device driver to initialize the MDIO device:

1. Configure the PREAMBLE and CLKDIV bits in the MDIO Control register (CPSW\_MDIO\_CONTROL\_REG).
2. Enable the MDIO module by setting the ENABLE bit in CPSW\_MDIO\_CONTROL\_REG.
3. The MDIO PHY alive status register (MDIO CPSW\_MDIO\_ALIVE\_REG) can be read in polling fashion until a PHY connected to the system responded, and the MDIO PHY link status register (MDIO CPSW\_MDIO\_LINK\_REG) can determine whether this PHY already has a link.
4. Set the appropriate PHY addresses in the MDIO user PHY select register (CPSW\_MDIO\_USER\_PHY\_SEL\_REG\_k, where k = 0 or 1), and set the LINKINT\_ENABLE bit to enable a link change event interrupt if desirable.
5. Set the appropriate LINKSEL bit in the CPSW\_MDIO\_USER\_PHY\_SEL\_REG\_k register (where k = 0 or 1).
6. Set the appropriate USERINTMASKSET bit field in the CPSW\_MDIO\_USER\_INT\_MASK\_SET\_REG register.
7. If an interrupt on general MDIO register access is desired, set the corresponding bit in the MDIO user command complete interrupt mask set register (MDIO CPSW\_MDIO\_USER\_INT\_MASK\_SET\_REG) to use the MDIO user access register (MDIO CPSW\_MDIO\_USER\_ACCESS\_REG\_k, where k = 0 or 1).

##### **12.2.2.5.3.2 Writing Data To a PHY Register**

The MDIO module includes a user access register (MDIO CPSW\_MDIO\_USER\_ACCESS\_REG\_k, where k = 0 or 1) to directly access a specified PHY device. To write a PHY register, perform the following:

1. Check to ensure that the GO bit in the MDIO user access register (MDIO CPSW\_MDIO\_USER\_ACCESS\_REG\_k) is cleared.
2. Write to the GO, WRITE, REGADR, PHYADR, and DATA bits in MDIO CPSW\_MDIO\_USER\_ACCESS\_REG\_k corresponding to the PHY and PHY register SW wants to write.
3. The write operation to the PHY is scheduled and completed by the MDIO module. Completion of the write operation can be determined by polling the GO bit in MDIO CPSW\_MDIO\_USER\_ACCESS\_REG\_k for a 0.
4. Completion of the operation sets the corresponding USERINTRAW bit (0 or 1) in the MDIO user command complete interrupt register (CPSW\_MDIO\_USER\_INT\_RAW\_REG) corresponding to MDIO CPSW\_MDIO\_USER\_ACCESS\_REG\_k used. If interrupts have been enabled on this bit using the MDIO user command complete interrupt mask set register (CPSW\_MDIO\_USER\_INT\_MASK\_SET\_REG), then the bit is also set in the MDIO user command complete interrupt register (CPSW\_MDIO\_USER\_INT\_MASKED\_REG) and an interrupt is triggered on the host processor.

### 12.2.2.5.3.3 Reading Data From a PHY Register

The MDIO module includes a user access register (MDIO CPSW\_MDIO\_USER\_ACCESS\_REG\_k, where k = 0 or 1) to directly access a specified PHY device. To read a PHY register, perform the following:

1. Check to ensure that the GO bit in the MDIO user access register (CPSW\_MDIO\_USER\_ACCESS\_REG\_k, where k = 0 or 1) is cleared.
2. Write to the GO, REGADR, and PHYADR bits in the CPSW\_MDIO\_USER\_ACCESS\_REG\_k register corresponding to the PHY and PHY register SW wants to read.
3. The read data value is available in the DATA bit field in MDIO CPSW\_MDIO\_USER\_ACCESS\_REG\_k register after the module completes the read operation on the serial bus. Completion of the read operation can be determined by polling the GO and ACK bits in CPSW\_MDIO\_USER\_ACCESS\_REG\_k register. After the GO bit has cleared, the ACK bit is set on a successful read.
4. Completion of the operation sets the corresponding USERINTRAW bit (0 or 1) in the MDIO user command complete interrupt register (CPSW\_MDIO\_USER\_INT\_RAW\_REG) corresponding to MDIO CPSW\_MDIO\_USER\_ACCESS\_REG\_k used. If interrupts have been enabled on this bit using the MDIO user command complete interrupt mask set register (CPSW\_MDIO\_USER\_INT\_MASK\_SET\_REG), then the bit is also set in the MDIO user command complete interrupt register (CPSW\_MDIO\_USER\_INT\_MASKED\_REG) and an interrupt is triggered on the host processor.



### 12.2.3 Peripheral Component Interconnect Express (PCIe) Subsystem

This chapter describes the features and functions of the device Peripheral Component Interconnect Express (PCIe) subsystem.

#### 12.2.3.1 PCIe Subsystem Overview

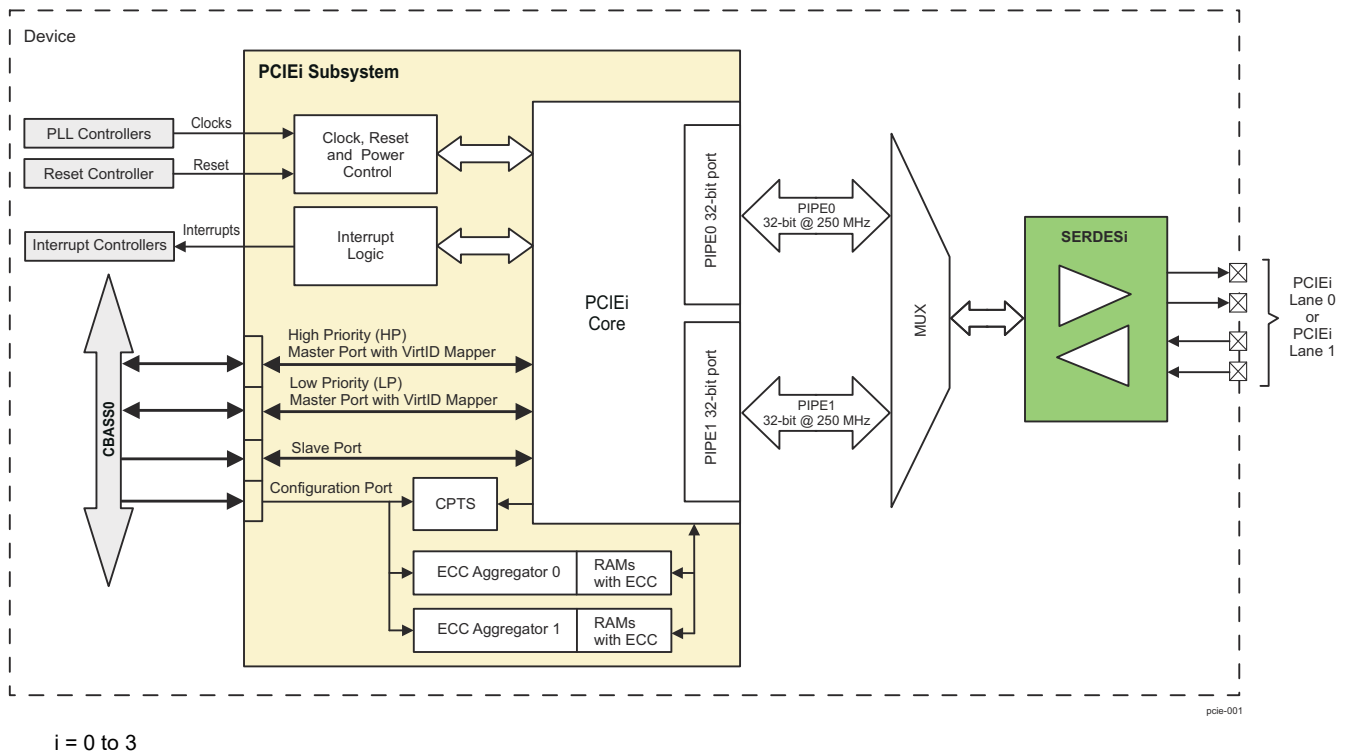
The Peripheral Component Interconnect Express (PCIe) subsystem is built around a multi-lane dual-mode PCIe controller that provides low pin-count, high reliability, and high-speed data transfers at rates of up to 8.0 Gbps per lane for serial links on backplanes and printed wiring boards.

The device includes four instantiations of PCIe subsystem named PCIE0, PCIE1, PCIE2 and PCIE3. [Table 12-237](#) shows the PCIe subsystem allocation within device domains:

**Table 12-237. PCIe Subsystem Allocation within Device Domains**

Module Instance	Domain		
	WKUP	MCU	MAIN
PCIE0	-	-	✓
PCIE1	-	-	✓
PCIE2	-	-	✓
PCIE3	-	-	✓

[Figure 12-190](#) provides PCIe subsystems overview.



**Figure 12-190. PCIe Subsystems Overview**

#### 12.2.3.1.1 PCIe Subsystem Features

Each PCIe subsystem supports the following main features:

- Compliance to PCIe® Base Specification, Revision 4.0 (Version 0.7)
- One or two-lane configuration with up to 8.0 Gbps/lane (Gen3). Can be used as 2-lane controller, configurable in 1x1 or 1x2 mode.

- Gen3 (8 Gbps 128/130-bit encoding), Gen2 (5 Gbps 8/10-bit encoding), and Gen1 (2.5 Gbps 8/10-bit encoding) with auto-negotiation
- Dual mode: Root Port (RP) or End Point (EP) operation modes, selectable via bootstrap pins
- Dynamic PIPE width change when switching between Gen1/2/3 modes
- Constant 32-bit PIPE width for Gen1/2/3 modes
- Maximum payload size of 256 bytes
- Maximum remote read request size of 4K bytes
- Address Translation Services (ATS)
- Single-root I/O Virtualization (SR-IOV) with Physical Functions (PF) and Virtual Functions (VF) in End Point mode
  - Six Physical Functions (PF)
  - Sixteen Virtual Functions (4 VF for each of PF0, PF1, PF2, and PF3; 0 VF for PF4 and PF5)
- Four virtual channels (VC)
- PCI Power Management states are:
  - L1 Active State Power Management
  - L1 Power Management substates support
  - D1 Device Power Management state
- Maximum number of non-posted outstanding transactions: 32
- Resizable BAR capability
- Legacy, MSI and MSI-X Interrupt Support
- 32 outbound address translation regions
- Precision time measurement (PTM)

#### **12.2.3.1.2 PCIe Subsystem Not Supported Features**

The PCIe subsystems do not support the following features:

- Gen4 (16GT/s) operation
- 2-lane controller, configurable in 2x1 mode
- PCIe beacon for in-band wake
- Vendor Messaging
- I/O access in inbound direction in RP or EP mode
- Addressing modes other than incremental for burst transactions. As a result, the PCIe addresses cannot be in cacheable memory space.
- L2 power state
- Hot-plug
- Separate Reference Clock with Independent Spread (SRIS)

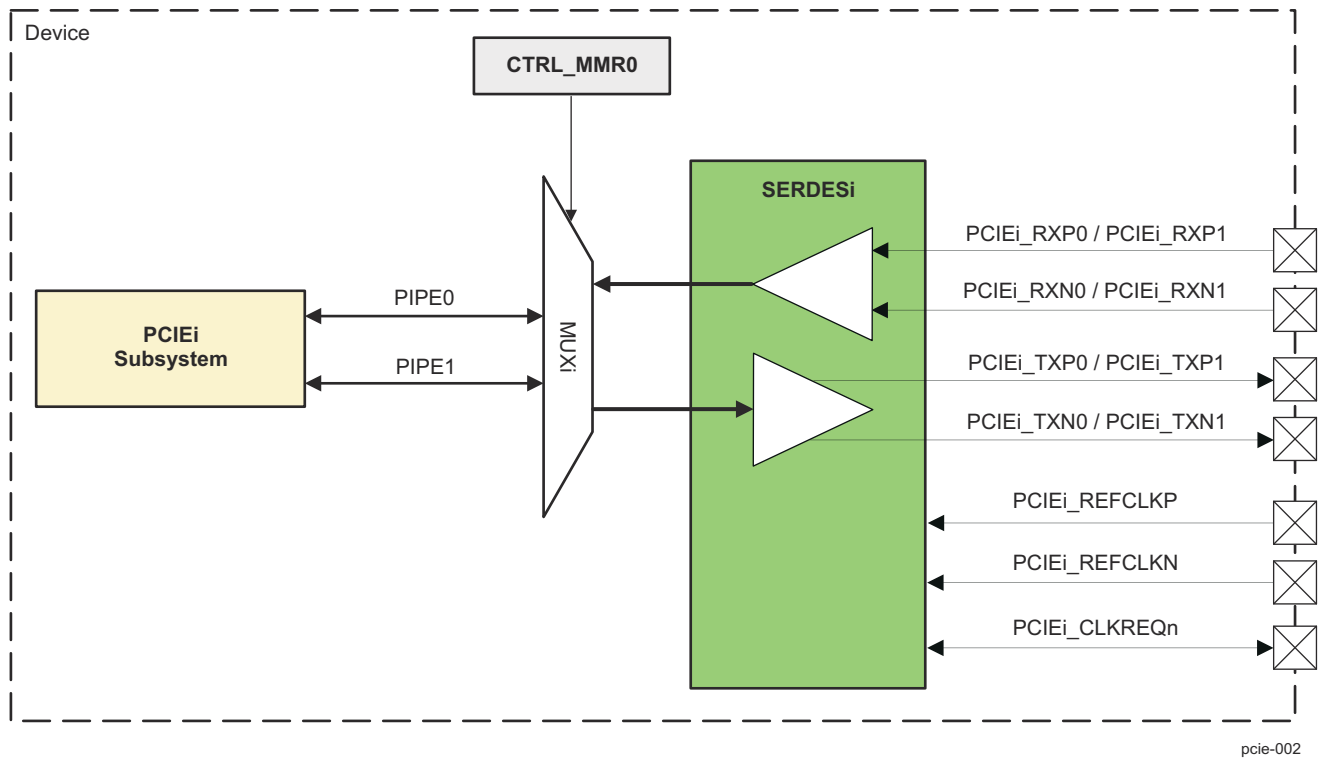


### 12.2.3.2 PCIe Subsystem Environment

This section describes the PCIe subsystem application fields from an environment point of view (external connections).

#### CAUTION

The PCIe subsystems do not have any direct external interface pins. PCIe data transactions are implemented through the corresponding SERDES interface pins.



**Figure 12-191. PCIe Subsystem Enviroment**

[Table 12-238](#) describes the SERDES signal names at device level related to PCIe subsystems, and specifies their functions. For more information on the SERDES operation and interface signals, refer to *Serializer/Deserializer (SerDes)*.

**Table 12-238. PCIe Subsystem I/O Signals**

Device Level Signal	I/O <sup>(1)</sup>	Description
<b>PCIE0 Subsystem</b>		
PCIE0_RXN0	I	PCle Lane 0 Receive Differential Data (-)
PCIE0_RXP0	I	PCle Lane 0 Receive Differential Data (+)
PCIE0_TXN0	O	PCle Lane 0 Transmit Differential Data (-)
PCIE0_TXP0	O	PCle Lane 0 Transmit Differential Data (+)
PCIE0_RXN1	I	PCle Lane 1 Receive Differential Data (-)
PCIE0_RXP1	I	PCle Lane 1 Receive Differential Data (+)
PCIE0_TXN1	O	PCle Lane 1 Transmit Differential Data (-)
PCIE0_TXP1	O	PCle Lane 1 Transmit Differential Data (+)
PCIE0_REFCLKN	I	PCle differential reference clock (-)
PCIE0_REFCLKP	I	PCle differential reference clock (+)
PCIE0_CLKREQn	I/O	PCle active-low clock request
<b>PCIE1 Subsystem</b>		
PCIE1_RXN0	I	PCle Lane 0 Receive Differential Data (-)
PCIE1_RXP0	I	PCle Lane 0 Receive Differential Data (+)
PCIE1_TXN0	O	PCle Lane 0 Transmit Differential Data (-)
PCIE1_TXP0	O	PCle Lane 0 Transmit Differential Data (+)
PCIE1_RXN1	I	PCle Lane 1 Receive Differential Data (-)
PCIE1_RXP1	I	PCle Lane 1 Receive Differential Data (+)
PCIE1_TXN1	O	PCle Lane 1 Transmit Differential Data (-)
PCIE1_TXP1	O	PCle Lane 1 Transmit Differential Data (+)
PCIE1_REFCLKN	I	PCle differential reference clock (-)
PCIE1_REFCLKP	I	PCle differential reference clock (+)
PCIE1_CLKREQn	I/O	PCle active-low clock request
<b>PCIE2 Subsystem</b>		
PCIE2_RXN0	I	PCle Lane 0 Receive Differential Data (-)
PCIE2_RXP0	I	PCle Lane 0 Receive Differential Data (+)
PCIE2_TXN0	O	PCle Lane 0 Transmit Differential Data (-)
PCIE2_TXP0	O	PCle Lane 0 Transmit Differential Data (+)
PCIE2_RXN1	I	PCle Lane 1 Receive Differential Data (-)
PCIE2_RXP1	I	PCle Lane 1 Receive Differential Data (+)
PCIE2_TXN1	O	PCle Lane 1 Transmit Differential Data (-)
PCIE2_TXP1	O	PCle Lane 1 Transmit Differential Data (+)
PCIE2_REFCLKN	I	PCle differential reference clock (-)
PCIE2_REFCLKP	I	PCle differential reference clock (+)
PCIE2_CLKREQn	I/O	PCle active-low clock request
<b>PCIE3 Subsystem</b>		
PCIE3_RXN0	I	PCle Lane 0 Receive Differential Data (-)
PCIE3_RXP0	I	PCle Lane 0 Receive Differential Data (+)
PCIE3_TXN0	O	PCle Lane 0 Transmit Differential Data (-)
PCIE3_TXP0	O	PCle Lane 0 Transmit Differential Data (+)
PCIE3_RXN1	I	PCle Lane 1 Receive Differential Data (-)
PCIE3_RXP1	I	PCle Lane 1 Receive Differential Data (+)
PCIE3_TXN1	O	PCle Lane 1 Transmit Differential Data (-)
PCIE3_TXP1	O	PCle Lane 1 Transmit Differential Data (+)

**Table 12-238. PCIe Subsystem I/O Signals (continued)**

Device Level Signal	I/O <sup>(1)</sup>	Description
PCIE3_REFCLKN	I	PCIe differential reference clock (-)
PCIE3_REFCLKP	I	PCIe differential reference clock (+)
PCIE3_CLKREQn	I/O	PCIe active-low clock request

(1) I = Input; O = Output.

#### Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

### 12.2.3.3 PCIe Subsystem Integration

This section describes PCIe subsystems integration in the device MAIN domain, including information about clocks, resets, and hardware requests.

There are four PCIe subsystems integrated in the device MAIN domain - PCIE0, PCIE1, PCIE2 and PCIE3. Figure 12-192 shows the integration of PCIE0 and PCIE1.

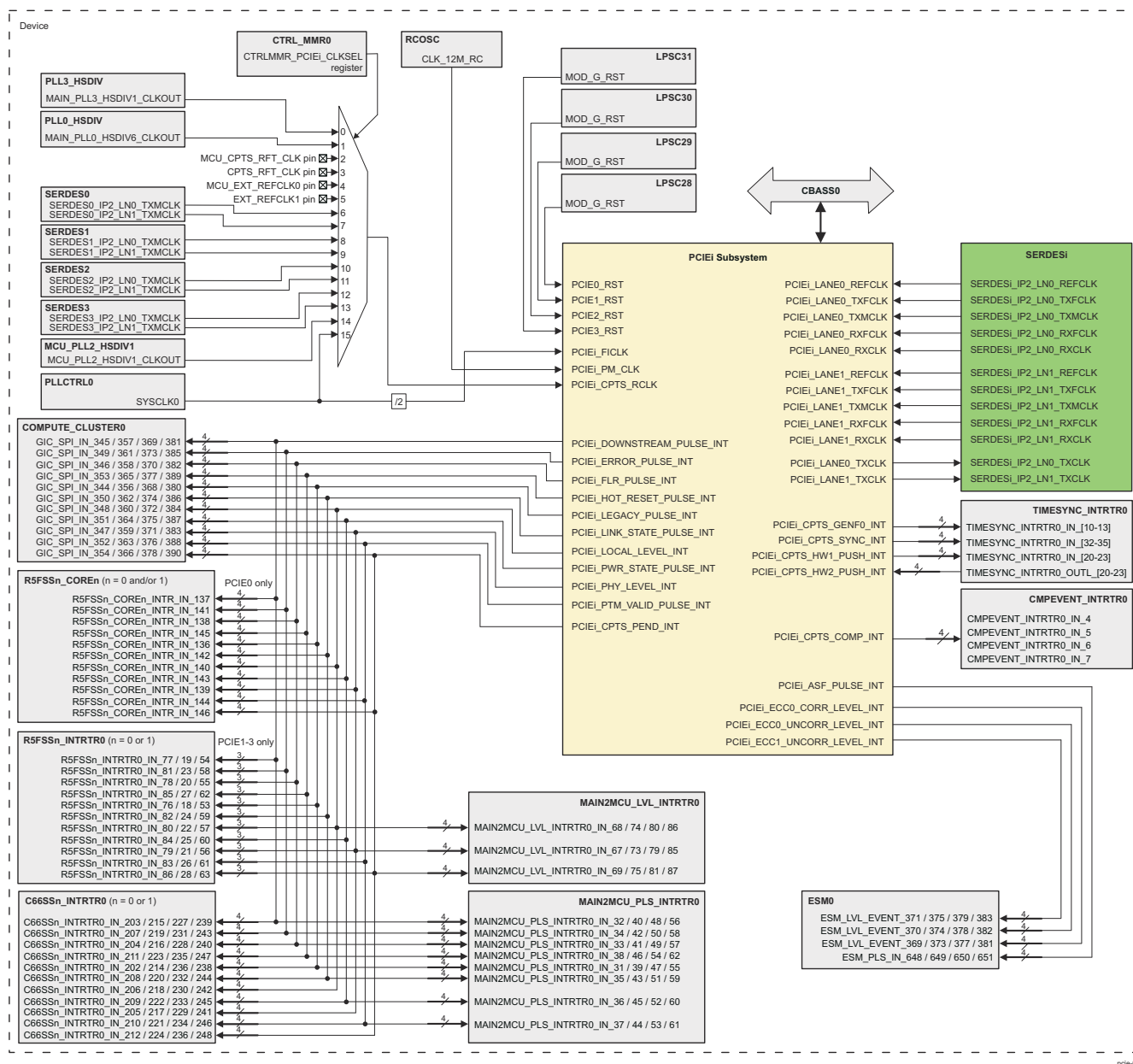


Figure 12-192. PCIe Subsystem Integration

Table 12-239 through Table 12-241 summarize the integration of PCIE0, PCIE1, PCIE2, and PCIE3 in device MAIN domain.

**Table 12-239. PCIe Subsystem Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
PCIE0	PSC0	PD0	LPSC28	CBASS0
PCIE1	PSC0	PD0	LPSC29	CBASS0
PCIE2	PSC0	PD0	LPSC30	CBASS0
PCIE3	PSC0	PD0	LPSC31	CBASS0

**Table 12-240. PCIe Subsystem Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
PCIE0	PCIE0_FICLK	SYSCLK0 / 2	PLLCTRL0	PCIE0 functional and interface clock
	PCIE0_PM_CLK	CLK_12M_RC	RCOSC	PCIE0 clock from RC oscillator
	PCIE0_CPTS_RCLK	MAIN_PLL3_HSDIV1_CLKOUT	PLL3_HSDIV	PCIE0 CPTS reference clock (RCLK). The CPTS RCLK clock frequency should be greater than or equal to the PCIE FICLK clock frequency. Otherwise, the software will have to add some wait cycles before a correct event is generated by the CPTS module.
		MAIN_PLL0_HSDIV6_CLKOUT	PLL0_HSDIV	
		MCU_CPTS_RFT_CLK	I/O pin	
		CPTS_RFT_CLK	I/O pin	
		MCU_EXT_REFCLK0	I/O pin	The selection of the source signal (see <a href="#">Figure 12-192, PCIe Subsystem Integration</a> ) can be done via the CTRLMMR_PCIE0_CLKSEL[3-0] CPTS_CLKSEL register field in the device Control Module.
		EXT_REFCLK1	I/O pin	
		SERDES0_IP2_LN0_TXMCLK	SERDES0	
		SERDES0_IP2_LN1_TXMCLK	SERDES0	
		SERDES1_IP2_LN0_TXMCLK	SERDES1	PCIE0 PIPE lane 0 reference clock from SERDES0
		SERDES1_IP2_LN1_TXMCLK	SERDES1	
		SERDES2_IP2_LN0_TXMCLK	SERDES2	
		SERDES2_IP2_LN1_TXMCLK	SERDES2	
		SERDES3_IP2_LN0_TXMCLK	SERDES3	PCIE0 PIPE lane 0 TX full rate clock from SERDES0
		SERDES3_IP2_LN1_TXMCLK	SERDES3	
		MCU_PLL2_HSDIV1_CLKOUT	MCU_PLL2_HSDIV1	
		SYSCLK0	PLLCTRL0	
	PCIE0_LANE0_REFCLK	SERDES0_IP2_LN0_REFCLK	SERDES0	PCIE0 PIPE lane 0 reference clock from SERDES0
	PCIE0_LANE0_TXFCLK	SERDES0_IP2_LN0_TXFCLK		PCIE0 PIPE lane 0 TX full rate clock from SERDES0
	PCIE0_LANE0_TXMCLK	SERDES0_IP2_LN0_TXMCLK		PCIE0 core clock from SERDES0 via lane 0
	PCIE0_LANE0_RXFCLK	SERDES0_IP2_LN0_RXFCLK		PCIE0 PIPE lane 0 RX full rate clock from SERDES0
	PCIE0_LANE0_RXCLK	SERDES0_IP2_LN0_RXCLK		PCIE0 PIPE lane 0 RX clock from SERDES0
	PCIE0_LANE1_REFCLK	SERDES0_IP2_LN1_REFCLK	SERDES0	PCIE0 PIPE lane 1 reference clock from SERDES0
	PCIE0_LANE1_TXFCLK	SERDES0_IP2_LN1_TXFCLK		PCIE0 PIPE lane 1 TX full rate clock from SERDES0
	PCIE0_LANE1_TXMCLK	SERDES0_IP2_LN1_TXMCLK		PCIE0 core clock from SERDES0 via lane 1
	PCIE0_LANE1_RXFCLK	SERDES0_IP2_LN1_RXFCLK		PCIE0 PIPE lane 1 RX full rate clock from SERDES0
	PCIE0_LANE1_RXCLK	SERDES0_IP2_LN1_RXCLK		PCIE0 PIPE lane 1 RX clock from SERDES0

**Table 12-240. PCIe Subsystem Clocks and Resets (continued)**

SERDES0	SERDES0_IP2_LN0_TX CLK	PCIE0_LANE0_TXCLK	PCIE0	PCIE0 PIPE lane 0 transmit clock to SERDES0
	SERDES0_IP2_LN1_TX CLK	PCIE0_LANE1_TXCLK		PCIE0 PIPE lane 1 transmit clock to SERDES0
PCIE1	PCIE1_FICLK	SYSCLK0 / 2	PLLCTRL0	PCIE1 functional and interface clock
	PCIE1_PM_CLK	CLK_12M_RC	RCOSC	PCIE1 clock from RC oscillator
	PCIE1_CPTS_RCLK	MAIN_PLL3_HSDIV1_CLKOUT	PLL3_HSDIV	PCIE1 CPTS reference clock (RCLK).
		MAIN_PLL0_HSDIV6_CLKOUT	PLL0_HSDIV	The CPTS RCLK clock frequency should be greater than or equal to the PCIE FICLK clock frequency. Otherwise, the software will have to add some wait cycles before a correct event is generated by the CPTS module.
		MCU_CPTS_RFT_CLK	I/O pin	The selection of the source signal (see <a href="#">Figure 12-192, PCIe Subsystem Integration</a> ) can be done via the
		CPTS_RFT_CLK	I/O pin	CTRLMMR_PCIE1_CLKSEL[3-0]
		MCU_EXT_REFCLK0	I/O pin	CPTS_CLKSEL register field in the device Control Module.
		EXT_REFCLK1	I/O pin	
		SERDES0_IP2_LN0_TXMCLK	SERDES0	
		SERDES0_IP2_LN1_TXMCLK	SERDES0	
		SERDES1_IP2_LN0_TXMCLK	SERDES1	
		SERDES1_IP2_LN1_TXMCLK	SERDES1	
		SERDES2_IP2_LN0_TXMCLK	SERDES2	
		SERDES2_IP2_LN1_TXMCLK	SERDES2	
		SERDES3_IP2_LN0_TXMCLK	SERDES3	
		SERDES3_IP2_LN1_TXMCLK	SERDES3	
		MCU_PLL2_HSDIV1_CLKOUT	MCU_PLL2_HSDIV1	
		MAIN_SYSCLK0	PLLCTRL0	
	PCIE1_LANE0_REFCLK	SERDES1_IP2_LN0_REFCLK	SERDES1	PCIE1 PIPE lane 0 reference clock from SERDES1
	PCIE1_LANE0_TXFCLK	SERDES1_IP2_LN0_TXFCLK		PCIE1 PIPE lane 0 TX full rate clock from SERDES1
	PCIE1_LANE0_TXMCLK	SERDES1_IP2_LN0_TXMCLK		PCIE1 core clock from SERDES1 via lane 0
	PCIE1_LANE0_RXFCLK	SERDES1_IP2_LN0_RXFCLK		PCIE1 PIPE lane 0 RX full rate clock from SERDES1
	PCIE1_LANE0_RXCLK	SERDES1_IP2_LN0_RXCLK		PCIE1 PIPE lane 0 RX clock from SERDES1
	PCIE1_LANE1_REFCLK	SERDES1_IP2_LN1_REFCLK	SERDES1	PCIE1 PIPE lane 1 reference clock from SERDES1
	PCIE1_LANE1_TXFCLK	SERDES1_IP2_LN1_TXFCLK		PCIE1 PIPE lane 1 TX full rate clock from SERDES1
	PCIE1_LANE1_TXMCLK	SERDES1_IP2_LN1_TXMCLK		PCIE1 core clock from SERDES1 via lane 1
	PCIE1_LANE1_RXFCLK	SERDES1_IP2_LN1_RXFCLK		PCIE1 PIPE lane 1 RX full rate clock from SERDES1
	PCIE1_LANE1_RXCLK	SERDES1_IP2_LN1_RXCLK		PCIE1 PIPE lane 1 RX clock from SERDES1
SERDES1	SERDES1_IP2_LN0_TX CLK	PCIE1_LANE0_TXCLK	PCIE1	PCIE1 PIPE lane 0 transmit clock to SERDES1
	SERDES1_IP2_LN1_TX CLK	PCIE1_LANE1_TXCLK		PCIE1 PIPE lane 1 transmit clock to SERDES1
PCIE2	PCIE2_FICLK	SYSCLK0 / 2	PLLCTRL0	PCIE2 functional and interface clock
	PCIE2_PM_CLK	CLK_12M_RC	RCOSC	PCIE2 clock from RC oscillator

**Table 12-240. PCIe Subsystem Clocks and Resets (continued)**

PCIE2_CPTS_RCLK	MAIN_PLL3_HSDIV1_CLKOUT	PLL3_HSDIV	PCIE2 CPTS reference clock (RCLK). The CPTS RCLK clock frequency should be greater than or equal to the PCIE FICLK clock frequency. Otherwise, the software will have to add some wait cycles before a correct event is generated by the CPTS module. The selection of the source signal (see <a href="#">Figure 12-192, PCIe Subsystem Integration</a> ) can be done via the CTRLMMR_PCIE2_CLKSEL[3-0] CPTS_CLKSEL register field in the device Control Module.	
	MAIN_PLL0_HSDIV6_CLKOUT	PLL0_HSDIV		
	MCU_CPTS_RFT_CLK	I/O pin		
	CPTS_RFT_CLK	I/O pin		
	MCU_EXT_REFCLK0	I/O pin		
	EXT_REFCLK1	I/O pin		
	SERDES0_IP2_LN0_TXMCLK	SERDES0		
	SERDES0_IP2_LN1_TXMCLK	SERDES0		
	SERDES1_IP2_LN0_TXMCLK	SERDES1		
	SERDES1_IP2_LN1_TXMCLK	SERDES1		
	SERDES2_IP2_LN0_TXMCLK	SERDES2		
	SERDES2_IP2_LN1_TXMCLK	SERDES2		
	SERDES3_IP2_LN0_TXMCLK	SERDES3		
	SERDES3_IP2_LN1_TXMCLK	SERDES3		
	MCU_PLL2_HSDIV1_CLKOUT	MCU_PLL2_HSDIV1		
MAIN_SYSCLK0	PLLCTRL0			
PCIE2_LANE0_REFCLK	SERDES2_IP2_LN0_REFCLK	SERDES2	PCIE2 PIPE lane 0 reference clock from SERDES2	
PCIE2_LANE0_TXFCLK	SERDES2_IP2_LN0_TXFCLK		PCIE2 PIPE lane 0 TX full rate clock from SERDES2	
PCIE2_LANE0_TXMCLK	SERDES2_IP2_LN0_TXMCLK		PCIE2 core clock from SERDES2 via lane 0	
PCIE2_LANE0_RXFCLK	SERDES2_IP2_LN0_RXFCLK		PCIE2 PIPE lane 0 RX full rate clock from SERDES2	
PCIE2_LANE0_RXCLK	SERDES2_IP2_LN0_RXCLK		PCIE2 PIPE lane 0 RX clock from SERDES2	
PCIE2_LANE1_REFCLK	SERDES2_IP2_LN1_REFCLK	SERDES2	PCIE2 PIPE lane 1 reference clock from SERDES2	
PCIE2_LANE1_TXFCLK	SERDES2_IP2_LN1_TXFCLK		PCIE2 PIPE lane 1 TX full rate clock from SERDES2	
PCIE2_LANE1_TXMCLK	SERDES2_IP2_LN1_TXMCLK		PCIE2 core clock from SERDES2 via lane 1	
PCIE2_LANE1_RXFCLK	SERDES2_IP2_LN1_RXFCLK		PCIE2 PIPE lane 1 RX full rate clock from SERDES2	
PCIE2_LANE1_RXCLK	SERDES2_IP2_LN1_RXCLK		PCIE2 PIPE lane 1 RX clock from SERDES2	
SERDES2	SERDES2_IP2_LN0_TX CLK	PCIE2_LANE0_TXCLK	PCIE2	PCIE2 PIPE lane 0 transmit clock to SERDES2
	SERDES2_IP2_LN1_TX CLK	PCIE2_LANE1_TXCLK		PCIE2 PIPE lane 1 transmit clock to SERDES2
PCIE3	PCIE3_FICLK	SYSCLK0 / 2	PLLCTRL0	PCIE3 functional and interface clock
	PCIE3_PM_CLK	CLK_12M_RC	RCOSC	PCIE3 clock from RC oscillator

**Table 12-240. PCIe Subsystem Clocks and Resets (continued)**

PCIE3_CPTS_RCLK	MAIN_PLL3_HSDIV1_CLKOUT	PLL3_HSDIV	PCIE3 CPTS reference clock (RCLK). The CPTS RCLK clock frequency should be greater than or equal to the PCIE FICLK clock frequency. Otherwise, the software will have to add some wait cycles before a correct event is generated by the CPTS module. The selection of the source signal (see <a href="#">Figure 12-192, PCIe Subsystem Integration</a> ) can be done via the CTRLMMR_PCIE3_CLKSEL[3-0] CPTS_CLKSEL register field in the device Control Module.	
	MAIN_PLL0_HSDIV6_CLKOUT	PLL0_HSDIV		
	MCU_CPTS_RFT_CLK	I/O pin		
	CPTS_RFT_CLK	I/O pin		
	MCU_EXT_REFCLK0	I/O pin		
	EXT_REFCLK1	I/O pin		
	SERDES0_IP2_LN0_TXMCLK	SERDES0		
	SERDES0_IP2_LN1_TXMCLK	SERDES0		
	SERDES1_IP2_LN0_TXMCLK	SERDES1		
	SERDES1_IP2_LN1_TXMCLK	SERDES1		
	SERDES2_IP2_LN0_TXMCLK	SERDES2		
	SERDES2_IP2_LN1_TXMCLK	SERDES2		
	SERDES3_IP2_LN0_TXMCLK	SERDES3		
	SERDES3_IP2_LN1_TXMCLK	SERDES3		
	MCU_PLL2_HSDIV1_CLKOUT	MCU_PLL2_HSDIV1		
MAIN_SYSCLK0	PLLCTRL0			
PCIE3_LANE0_REFCLK	SERDES3_IP2_LN0_REFCLK	SERDES3	PCIE3 PIPE lane 0 reference clock from SERDES3	
PCIE3_LANE0_TXFCLK	SERDES3_IP2_LN0_TXFCLK		PCIE3 PIPE lane 0 TX full rate clock from SERDES3	
PCIE3_LANE0_TXMCLK	SERDES3_IP2_LN0_TXMCLK		PCIE3 core clock from SERDES3 via lane 0	
PCIE3_LANE0_RXFCLK	SERDES3_IP2_LN0_RXFCLK		PCIE3 PIPE lane 0 RX full rate clock from SERDES3	
PCIE3_LANE0_RXCLK	SERDES3_IP2_LN0_RXCLK		PCIE3 PIPE lane 0 RX clock from SERDES3	
PCIE3_LANE1_REFCLK	SERDES3_IP2_LN1_REFCLK	SERDES3	PCIE3 PIPE lane 1 reference clock from SERDES3	
PCIE3_LANE1_TXFCLK	SERDES3_IP2_LN1_TXFCLK		PCIE3 PIPE lane 1 TX full rate clock from SERDES3	
PCIE3_LANE1_TXMCLK	SERDES3_IP2_LN1_TXMCLK		PCIE3 core clock from SERDES3 via lane 1	
PCIE3_LANE1_RXFCLK	SERDES3_IP2_LN1_RXFCLK		PCIE3 PIPE lane 1 RX full rate clock from SERDES3	
PCIE3_LANE1_RXCLK	SERDES3_IP2_LN1_RXCLK		PCIE3 PIPE lane 1 RX clock from SERDES3	
SERDES3	SERDES3_IP2_LN0_TXCLK	PCIE3_LANE0_TXCLK	PCIE3	PCIE3 PIPE lane 0 transmit clock to SERDES3
	SERDES3_IP2_LN1_TXCLK	PCIE3_LANE1_TXCLK		PCIE3 PIPE lane 1 transmit clock to SERDES3
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
PCIE0	PCIE0_RST	MOD_G_RST	LPSC28	PCIE0 reset
PCIE1	PCIE1_RST	MOD_G_RST	LPSC29	PCIE1 reset
PCIE2	PCIE2_RST	MOD_G_RST	LPSC30	PCIE2 reset
PCIE3	PCIE3_RST	MOD_G_RST	LPSC31	PCIE3 reset

**Table 12-241. PCIe Subsystem Hardware Requests**

Interrupt Requests				
--------------------	--	--	--	--



**Table 12-241. PCIe Subsystem Hardware Requests (continued)**

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
PCIE0	PCIE0_ASF_PULSE_INT	ESM_PLS_IN_648	ESM0	PCIE0 active internal diagnostics interrupt	Pulse
	PCIE0_DOWNSTREAM_PULSE_INT	GIC500_SPI_IN_345	COMPUTE_CLUSTER0	PCIE0 downstream interrupt	Pulse
		C66SS0_INTRTR0_IN_203	C66SS0_INTRTR0		Pulse
		C66SS1_INTRTR0_IN_203	C66SS1_INTRTR0		Pulse
		MAIN2MCU_PLS_INTRTR0_IN_32	MAIN2MCU_PLS_INTRTR0		Pulse
		R5FSS0_CORE0_INTR_IN_137	R5FSS0_CORE0		Pulse
		R5FSS0_CORE1_INTR_IN_137	R5FSS0_CORE1		Pulse
		R5FSS1_CORE0_INTR_IN_137	R5FSS1_CORE0		Pulse
		R5FSS1_CORE1_INTR_IN_137	R5FSS1_CORE1		Pulse
	PCIE0_ERROR_PULSE_INT	GIC500_SPI_IN_349	COMPUTE_CLUSTER0	PCIE0 error interrupt	Pulse
		C66SS0_INTRTR0_IN_207	C66SS0_INTRTR0		Pulse
		C66SS1_INTRTR0_IN_207	C66SS1_INTRTR0		Pulse
		MAIN2MCU_PLS_INTRTR0_IN_34	MAIN2MCU_PLS_INTRTR0		Pulse
		R5FSS0_CORE0_INTR_IN_141	R5FSS0_CORE0		Pulse
		R5FSS0_CORE1_INTR_IN_141	R5FSS0_CORE1		Pulse
		R5FSS1_CORE0_INTR_IN_141	R5FSS1_CORE0		Pulse
		R5FSS1_CORE1_INTR_IN_141	R5FSS1_CORE1		Pulse
	PCIE0_FLR_PULSE_INT	GIC500_SPI_IN_346	COMPUTE_CLUSTER0	PCIE0 function level interrupt	Pulse
		C66SS0_INTRTR0_IN_204	C66SS0_INTRTR0		Pulse
		C66SS1_INTRTR0_IN_204	C66SS1_INTRTR0		Pulse
		MAIN2MCU_PLS_INTRTR0_IN_33	MAIN2MCU_PLS_INTRTR0		Pulse
		R5FSS0_CORE0_INTR_IN_138	R5FSS0_CORE0		Pulse
		R5FSS0_CORE1_INTR_IN_138	R5FSS0_CORE1		Pulse
		R5FSS1_CORE0_INTR_IN_138	R5FSS1_CORE0		Pulse
		R5FSS1_CORE1_INTR_IN_138	R5FSS1_CORE1		Pulse

**Table 12-241. PCIe Subsystem Hardware Requests (continued)**

PCIE0_HOT_RESET_PULSE_INT	GIC500_SPI_IN_353	COMPUTE_CLUSTER0	PCIE0 hot reset interrupt	Pulse
	C66SS0_INTRTR0_IN_211	C66SS0_INTRTR0		Pulse
	C66SS1_INTRTR0_IN_211	C66SS1_INTRTR0		Pulse
	MAIN2MCU_PLS_INTRTR0_IN_38	MAIN2MCU_PLS_INTRTR0		Pulse
	R5FSS0_CORE0_INTR_IN_145	R5FSS0_CORE0		Pulse
	R5FSS0_CORE1_INTR_IN_145	R5FSS0_CORE1		Pulse
	R5FSS1_CORE0_INTR_IN_145	R5FSS1_CORE0		Pulse
PCIE0_LEGACY_PULSE_INT	GIC500_SPI_IN_344	COMPUTE_CLUSTER0	PCIE0 legacy interrupt	Pulse
	C66SS0_INTRTR0_IN_202	C66SS0_INTRTR0		Pulse
	C66SS1_INTRTR0_IN_202	C66SS1_INTRTR0		Pulse
	MAIN2MCU_PLS_INTRTR0_IN_31	MAIN2MCU_PLS_INTRTR0		Pulse
	R5FSS0_CORE0_INTR_IN_136	R5FSS0_CORE0		Pulse
	R5FSS0_CORE1_INTR_IN_136	R5FSS0_CORE1		Pulse
	R5FSS1_CORE0_INTR_IN_136	R5FSS1_CORE0		Pulse
PCIE0_LINK_STATE_PULSE_INT	GIC500_SPI_IN_350	COMPUTE_CLUSTER0	PCIE0 link state interrupt	Pulse
	C66SS0_INTRTR0_IN_208	C66SS0_INTRTR0		Pulse
	C66SS1_INTRTR0_IN_208	C66SS1_INTRTR0		Pulse
	MAIN2MCU_PLS_INTRTR0_IN_35	MAIN2MCU_PLS_INTRTR0		Pulse
	R5FSS0_CORE0_INTR_IN_142	R5FSS0_CORE0		Pulse
	R5FSS0_CORE1_INTR_IN_142	R5FSS0_CORE1		Pulse
	R5FSS1_CORE0_INTR_IN_142	R5FSS1_CORE0		Pulse
	R5FSS1_CORE1_INTR_IN_142	R5FSS1_CORE1		Pulse

**Table 12-241. PCIe Subsystem Hardware Requests (continued)**

PCIE0_LOCAL_LEVEL_INT	GIC500_SPI_IN_348	COMPUTE_CLUSTER0	PCIE0 local interrupt	Level
	C66SS0_INTRTR0_IN_206	C66SS0_INTRTR0		Level
	C66SS1_INTRTR0_IN_206	C66SS1_INTRTR0		Level
	MAIN2MCU_LVL_INTRTR0_IN_68	MAIN2MCU_LVL_INTRTR0		Level
	R5FSS0_CORE0_INTR_IN_140	R5FSS0_CORE0		Level
	R5FSS0_CORE1_INTR_IN_140	R5FSS0_CORE1		Level
	R5FSS1_CORE0_INTR_IN_140	R5FSS1_CORE0		Level
	R5FSS1_CORE1_INTR_IN_140	R5FSS1_CORE1		Level
PCIE0_PWR_STATE_PULSE_INT	GIC500_SPI_IN_351	COMPUTE_CLUSTER0	PCIE0 power state interrupt	Pulse
	C66SS0_INTRTR0_IN_209	C66SS0_INTRTR0		Pulse
	C66SS1_INTRTR0_IN_209	C66SS1_INTRTR0		Pulse
	MAIN2MCU_PLS_INTRTR0_IN_36	MAIN2MCU_PLS_INTRTR0		Pulse
	R5FSS0_CORE0_INTR_IN_143	R5FSS0_CORE0		Pulse
	R5FSS0_CORE1_INTR_IN_143	R5FSS0_CORE1		Pulse
	R5FSS1_CORE0_INTR_IN_143	R5FSS1_CORE0		Pulse
	R5FSS1_CORE1_INTR_IN_143	R5FSS1_CORE1		Pulse
PCIE0_PHY_LEVEL_INTERRUPT	GIC500_SPI_IN_347	COMPUTE_CLUSTER0	PCIE0 PHY interrupt	Level
	C66SS0_INTRTR0_IN_205	C66SS0_INTRTR0		Level
	C66SS1_INTRTR0_IN_205	C66SS1_INTRTR0		Level
	MAIN2MCU_LVL_INTRTR0_IN_67	MAIN2MCU_LVL_INTRTR0		Level
	R5FSS0_CORE0_INTR_IN_139	R5FSS0_CORE0		Level
	R5FSS0_CORE1_INTR_IN_139	R5FSS0_CORE1		Level
	R5FSS1_CORE0_INTR_IN_139	R5FSS1_CORE0		Level
	R5FSS1_CORE1_INTR_IN_139	R5FSS1_CORE1		Level

**Table 12-241. PCIe Subsystem Hardware Requests (continued)**

	PCIE0_PTM_VALID_P ULSE_INT	GIC500_SPI_IN_352	COMPUTE_CLUSTER0	PCIE0 PTM valid interrupt	Pulse
		C66SS0_INTRTR0_IN_210	C66SS0_INTRTR0		Pulse
		C66SS1_INTRTR0_IN_210	C66SS1_INTRTR0		Pulse
		MAIN2MCU_PLS_INTRTR0_I N_37	MAIN2MCU_PLS_INTRT R0		Pulse
		R5FSS0_CORE0_INTR_IN_14 4	R5FSS0_CORE0		Pulse
		R5FSS0_CORE1_INTR_IN_14 4	R5FSS0_CORE1		Pulse
		R5FSS1_CORE0_INTR_IN_14 4	R5FSS1_CORE0		Pulse
		R5FSS1_CORE1_INTR_IN_14 4	R5FSS1_CORE1		Pulse
		ESM_LVL_EVENT_369	ESM0	PCIE0 ECC AGGR 0 correctable error interrupt	Level
		ESM_LVL_EVENT_370	ESM0	PCIE0 ECC AGGR 0 uncorrectable error interrupt	Level
		ESM_LVL_EVENT_371	ESM0	PCIE0 ECC AGGR 1 uncorrectable error interrupt	Level
		GIC500_SPI_IN_354	COMPUTE_CLUSTER0	Timesync Interrupt	Level
		C66SS0_INTRTR0_IN_212	C66SS0_INTRTR0		Level
		C66SS1_INTRTR0_IN_212	C66SS1_INTRTR0		Level
		MAIN2MCU_LVL_INTRTR0_IN _69	MAIN2MCU_LVL_INTRT R0		Level
		R5FSS0_CORE0_INTR_IN_14 6	R5FSS0_CORE0		Level
		R5FSS0_CORE1_INTR_IN_14 6	R5FSS0_CORE1		Level
		R5FSS1_CORE0_INTR_IN_14 6	R5FSS1_CORE0		Level
		R5FSS1_CORE1_INTR_IN_14 6	R5FSS1_CORE1		Level
		ESM_PLS_IN_649	ESM0	PCIE1 active internal diagnostics interrupt	Pulse
		GIC500_SPI_IN_357	COMPUTE_CLUSTER0	PCIE1 downstream interrupt	Pulse
		C66SS0_INTRTR0_IN_215	C66SS0_INTRTR0		Pulse
		C66SS1_INTRTR0_IN_215	C66SS1_INTRTR0		Pulse
		MAIN2MCU_PLS_INTRTR0_I N_40	MAIN2MCU_PLS_INTRT R0		Pulse
		GIC500_SPI_IN_361	COMPUTE_CLUSTER0	PCIE1 error interrupt	Pulse
		C66SS0_INTRTR0_IN_219	C66SS0_INTRTR0		Pulse
		C66SS1_INTRTR0_IN_219	C66SS1_INTRTR0		Pulse
		MAIN2MCU_PLS_INTRTR0_I N_42	MAIN2MCU_PLS_INTRT R0		Pulse

**Table 12-241. PCIe Subsystem Hardware Requests (continued)**

PCIE1_FLR_PULSE_INT	GIC500_SPI_IN_358	COMPUTE_CLUSTER0	PCIE1 function level interrupt	Pulse
	C66SS0_INTRTR0_IN_216	C66SS0_INTRTR0		Pulse
	C66SS1_INTRTR0_IN_216	C66SS1_INTRTR0		Pulse
	MAIN2MCU_PLS_INTRTR0_IN_41	MAIN2MCU_PLS_INTRTR0		Pulse
PCIE1_HOT_RESET_PULSE_INT	GIC500_SPI_IN_365	COMPUTE_CLUSTER0	PCIE1 hot reset interrupt	Pulse
	C66SS0_INTRTR0_IN_223	C66SS0_INTRTR0		Pulse
	C66SS1_INTRTR0_IN_223	C66SS1_INTRTR0		Pulse
	MAIN2MCU_PLS_INTRTR0_IN_46	MAIN2MCU_PLS_INTRTR0		Pulse
PCIE1_LEGACY_PULSE_INT	GIC500_SPI_IN_356	COMPUTE_CLUSTER0	PCIE1 legacy interrupt	Pulse
	C66SS0_INTRTR0_IN_214	C66SS0_INTRTR0		Pulse
	C66SS1_INTRTR0_IN_214	C66SS1_INTRTR0		Pulse
	MAIN2MCU_PLS_INTRTR0_IN_39	MAIN2MCU_PLS_INTRTR0		Pulse
PCIE1_LINK_STATE_PULSE_INT	GIC500_SPI_IN_362	COMPUTE_CLUSTER0	PCIE1 link state interrupt	Pulse
	C66SS0_INTRTR0_IN_220	C66SS0_INTRTR0		Pulse
	C66SS1_INTRTR0_IN_220	C66SS1_INTRTR0		Pulse
	MAIN2MCU_PLS_INTRTR0_IN_43	MAIN2MCU_PLS_INTRTR0		Pulse
PCIE1_LOCAL_LEVEL_INT	GIC500_SPI_IN_360	COMPUTE_CLUSTER0	PCIE1 local interrupt	Level
	C66SS0_INTRTR0_IN_218	C66SS0_INTRTR0		Level
	C66SS1_INTRTR0_IN_218	C66SS1_INTRTR0		Level
	MAIN2MCU_LVL_INTRTR0_IN_74	MAIN2MCU_LVL_INTRTR0		Level
PCIE1_PWR_STATE_PULSE_INT	GIC500_SPI_IN_364	COMPUTE_CLUSTER0		Pulse
	C66SS0_INTRTR0_IN_222	C66SS0_INTRTR0	PCIE1 power state interrupt	Pulse
	C66SS1_INTRTR0_IN_222	C66SS1_INTRTR0		Pulse
	MAIN2MCU_PLS_INTRTR0_IN_45	MAIN2MCU_PLS_INTRTR0		Pulse
PCIE1_PHY_LEVEL_INT	GIC500_SPI_IN_359	COMPUTE_CLUSTER0	PCIE1 PHY interrupt	Level
	C66SS0_INTRTR0_IN_217	C66SS0_INTRTR0		Level
	C66SS1_INTRTR0_IN_217	C66SS1_INTRTR0		Level
	MAIN2MCU_LVL_INTRTR0_IN_73	MAIN2MCU_LVL_INTRTR0		Level
PCIE1_PTM_VALID_PULSE_INT	GIC500_SPI_IN_363	COMPUTE_CLUSTER0	PCIE1 PTM valid interrupt	Pulse
	C66SS0_INTRTR0_IN_221	C66SS0_INTRTR0		Pulse
	C66SS1_INTRTR0_IN_221	C66SS1_INTRTR0		Pulse
	MAIN2MCU_PLS_INTRTR0_IN_44	MAIN2MCU_PLS_INTRTR0		Pulse
PCIE1_ECC0_CORR_LEVEL_INT	ESM_LVL_EVENT_373	ESM0	PCIE1 ECC AGGR 0 correctable error interrupt	Level

**Table 12-241. PCIe Subsystem Hardware Requests (continued)**

PCIE1	PCIE1_ECC0_UNCORR_LEVEL_INT	ESM_LVL_EVENT_374	ESM0	PCIE1 ECC AGGR 0 uncorrectable error interrupt	Level
	PCIE1_ECC1_UNCORR_LEVEL_INT	ESM_LVL_EVENT_375	ESM0	PCIE1 ECC AGGR 1 uncorrectable error interrupt	Level
	PCIE1_CPTS_PEND_INT	GIC500_SPI_IN_366	COMPUTE_CLUSTER0	Timesync Interrupt	Level
		C66SS0_INTRTR0_IN_224	C66SS0_INTRTR0		Level
		C66SS1_INTRTR0_IN_224	C66SS1_INTRTR0		Level
		MAIN2MCU_LVL_INTRTR0_IN_75	MAIN2MCU_LVL_INTRTR0		Level
	PCIE2	PCIE2_ASF_PULSE_INT	ESM_PL5_IN_650	ESM0	PCIE2 active internal diagnostics interrupt
		PCIE2_DOWNSTREAM_PULSE_INT	GIC500_SPI_IN_369	COMPUTE_CLUSTER0	PCIE2 downstream interrupt
		C66SS0_INTRTR0_IN_227	C66SS0_INTRTR0		Pulse
		C66SS1_INTRTR0_IN_227	C66SS1_INTRTR0		Pulse
		R5FSS0_INTRTR0_IN_19	R5FSS0_INTRTR0		Pulse
		R5FSS1_INTRTR0_IN_19	R5FSS1_INTRTR0		Pulse
		MAIN2MCU_PL5_INTRTR0_IN_48	MAIN2MCU_PL5_INTRTR0		Pulse
		PCIE2_ERROR_PULSE_INT	GIC500_SPI_IN_373	COMPUTE_CLUSTER0	PCIE2 error interrupt
		C66SS0_INTRTR0_IN_231	C66SS0_INTRTR0		Pulse
		C66SS1_INTRTR0_IN_231	C66SS1_INTRTR0		Pulse
		R5FSS0_INTRTR0_IN_23	R5FSS0_INTRTR0		Pulse
		R5FSS1_INTRTR0_IN_23	R5FSS1_INTRTR0		Pulse
		MAIN2MCU_PL5_INTRTR0_IN_50	MAIN2MCU_PL5_INTRTR0		Pulse
PCIE2	PCIE2_FLR_PULSE_INT	GIC500_SPI_IN_370	COMPUTE_CLUSTER0	PCIE2 function level interrupt	Pulse
		C66SS0_INTRTR0_IN_228	C66SS0_INTRTR0		Pulse
		C66SS1_INTRTR0_IN_228	C66SS1_INTRTR0		Pulse
		R5FSS0_INTRTR0_IN_20	R5FSS0_INTRTR0		Pulse
		R5FSS1_INTRTR0_IN_20	R5FSS1_INTRTR0		Pulse
		MAIN2MCU_PL5_INTRTR0_IN_49	MAIN2MCU_PL5_INTRTR0		Pulse
	PCIE2_HOT_RESET_PULSE_INT	GIC500_SPI_IN_377	COMPUTE_CLUSTER0	PCIE2 hot reset interrupt	Pulse
		C66SS0_INTRTR0_IN_235	C66SS0_INTRTR0		Pulse
		C66SS1_INTRTR0_IN_235	C66SS1_INTRTR0		Pulse
		R5FSS0_INTRTR0_IN_27	R5FSS0_INTRTR0		Pulse
		R5FSS1_INTRTR0_IN_27	R5FSS1_INTRTR0		Pulse
		MAIN2MCU_PL5_INTRTR0_IN_54	MAIN2MCU_PL5_INTRTR0		Pulse
	PCIE2_LEGACY_PULSE_INT	GIC500_SPI_IN_368	COMPUTE_CLUSTER0	PCIE2 legacy interrupt	Pulse
		C66SS0_INTRTR0_IN_236	C66SS0_INTRTR0		Pulse
		C66SS1_INTRTR0_IN_236	C66SS1_INTRTR0		Pulse
		R5FSS0_INTRTR0_IN_18	R5FSS0_INTRTR0		Pulse

**Table 12-241. PCIe Subsystem Hardware Requests (continued)**

	R5FSS1_INTRTR0_IN_18	R5FSS1_INTRTR0		Pulse
	MAIN2MCU_PLS_INTRTR0_I N_47	MAIN2MCU_PLS_INTRTR0		Pulse
PCIE2_LINK_STATE_PULSE_INT	GIC500_SPI_IN_374	COMPUTE_CLUSTER0	PCIE2 link state interrupt	Pulse
	C66SS0_INTRTR0_IN_232	C66SS0_INTRTR0		Pulse
	C66SS1_INTRTR0_IN_232	C66SS1_INTRTR0		Pulse
	R5FSS0_INTRTR0_IN_24	R5FSS0_INTRTR0		Pulse
	R5FSS1_INTRTR0_IN_24	R5FSS1_INTRTR0		Pulse
	MAIN2MCU_PLS_INTRTR0_I N_51	MAIN2MCU_PLS_INTRTR0	PCIE2 local interrupt	Pulse
PCIE2_LOCAL_LEVEL_INT	GIC500_SPI_IN_372	COMPUTE_CLUSTER0		Level
	C66SS0_INTRTR0_IN_230	C66SS0_INTRTR0		Level
	C66SS1_INTRTR0_IN_230	C66SS1_INTRTR0		Level
	R5FSS0_INTRTR0_IN_22	R5FSS0_INTRTR0		Level
	R5FSS1_INTRTR0_IN_22	R5FSS1_INTRTR0		Level
	MAIN2MCU_LVL_INTRTR0_IN _80	MAIN2MCU_LVL_INTRTR0		Level
PCIE2_PWR_STATE_PULSE_INT	GIC500_SPI_IN_375	COMPUTE_CLUSTER0	PCIE2 power state interrupt	Pulse
	C66SS0_INTRTR0_IN_233	C66SS0_INTRTR0		Pulse
	C66SS1_INTRTR0_IN_233	C66SS1_INTRTR0		Pulse
	R5FSS0_INTRTR0_IN_25	R5FSS0_INTRTR0		Pulse
	R5FSS1_INTRTR0_IN_25	R5FSS1_INTRTR0		Pulse
	MAIN2MCU_PLS_INTRTR0_I N_52	MAIN2MCU_PLS_INTRTR0		Pulse
PCIE2_PHY_LEVEL_INTERRUPT	GIC500_SPI_IN_371	COMPUTE_CLUSTER0	PCIE2 PHY interrupt	Level
	C66SS0_INTRTR0_IN_229	C66SS0_INTRTR0		Level
	C66SS1_INTRTR0_IN_229	C66SS1_INTRTR0		Level
	R5FSS0_INTRTR0_IN_21	R5FSS0_INTRTR0		Level
	R5FSS1_INTRTR0_IN_21	R5FSS1_INTRTR0		Level
	MAIN2MCU_LVL_INTRTR0_IN _79	MAIN2MCU_LVL_INTRTR0		Level
PCIE2_PTM_VALID_PULSE_INT	GIC500_SPI_IN_376	COMPUTE_CLUSTER0	PCIE2 PTM valid interrupt	Pulse
	C66SS0_INTRTR0_IN_234	C66SS0_INTRTR0		Pulse
	C66SS1_INTRTR0_IN_234	C66SS1_INTRTR0		Pulse
	R5FSS0_INTRTR0_IN_26	R5FSS0_INTRTR0		Pulse
	R5FSS1_INTRTR0_IN_26	R5FSS1_INTRTR0		Pulse
	MAIN2MCU_PLS_INTRTR0_I N_53	MAIN2MCU_PLS_INTRTR0		Pulse
PCIE2_ECC0_CORR_LEVEL_INT	ESM_LVL_EVENT_377	ESM0	PCIE2 ECC AGGR 0 correctable error interrupt	Level
PCIE2_ECC0_UNCORR_LEVEL_INT	ESM_LVL_EVENT_378	ESM0	PCIE2 ECC AGGR 0 uncorrectable error interrupt	Level
PCIE2_ECC1_UNCORR_LEVEL_INT	ESM_LVL_EVENT_379	ESM0	PCIE2 ECC AGGR 1 uncorrectable error interrupt	Level

**Table 12-241. PCIe Subsystem Hardware Requests (continued)**

	PCIE2_CPTS_PEND_INT	GIC500_SPI_IN_378	COMPUTE_CLUSTER0	Timesync Interrupt	Level
		C66SS0_INTRTR0_IN_236	C66SS0_INTRTR0		Level
		C66SS1_INTRTR0_IN_236	C66SS1_INTRTR0		Level
		R5FSS0_INTRTR0_IN_28	R5FSS0_INTRTR0		Level
		R5FSS1_INTRTR0_IN_28	R5FSS1_INTRTR0		Level
		MAIN2MCU_LVL_INTRTR0_IN_81	MAIN2MCU_LVL_INTRTR0		Level
PCIE3	PCIE3_ASF_PULSE_INT	ESM_PLS_IN_651	ESM0	PCIE3 active internal diagnostics interrupt	Pulse
	PCIE3_DOWNSTREAM_PULSE_INT	GIC500_SPI_IN_381	COMPUTE_CLUSTER0	PCIE0 downstream interrupt	Pulse
		C66SS0_INTRTR0_IN_239	C66SS0_INTRTR0		Pulse
		C66SS1_INTRTR0_IN_239	C66SS1_INTRTR0		Pulse
		R5FSS0_INTRTR0_IN_54	R5FSS0_INTRTR0		Pulse
		R5FSS1_INTRTR0_IN_54	R5FSS1_INTRTR0		Pulse
		MAIN2MCU_PLS_INTRTR0_IN_56	MAIN2MCU_PLS_INTRTR0		Pulse
	PCIE3_ERROR_PULSE_INT	GIC500_SPI_IN_385	COMPUTE_CLUSTER0	PCIE3 error interrupt	Pulse
		C66SS0_INTRTR0_IN_243	C66SS0_INTRTR0		Pulse
		C66SS1_INTRTR0_IN_243	C66SS1_INTRTR0		Pulse
		R5FSS0_INTRTR0_IN_58	R5FSS0_INTRTR0		Pulse
		R5FSS1_INTRTR0_IN_58	R5FSS1_INTRTR0		Pulse
		MAIN2MCU_PLS_INTRTR0_IN_58	MAIN2MCU_PLS_INTRTR0		Pulse
	PCIE3_FLR_PULSE_INT	GIC500_SPI_IN_382	COMPUTE_CLUSTER0	PCIE3 function level interrupt	Pulse
		C66SS0_INTRTR0_IN_240	C66SS0_INTRTR0		Pulse
		C66SS1_INTRTR0_IN_240	C66SS1_INTRTR0		Pulse
		R5FSS0_INTRTR0_IN_55	R5FSS0_INTRTR0		Pulse
		R5FSS1_INTRTR0_IN_55	R5FSS1_INTRTR0		Pulse
		MAIN2MCU_PLS_INTRTR0_IN_57	MAIN2MCU_PLS_INTRTR0		Pulse
	PCIE3_HOT_RESET_PULSE_INT	GIC500_SPI_IN_389	COMPUTE_CLUSTER0	PCIE3 hot reset interrupt	Pulse
		C66SS0_INTRTR0_IN_247	C66SS0_INTRTR0		Pulse
		C66SS1_INTRTR0_IN_247	C66SS1_INTRTR0		Pulse
		R5FSS0_INTRTR0_IN_62	R5FSS0_INTRTR0		Pulse
		R5FSS1_INTRTR0_IN_62	R5FSS1_INTRTR0		Pulse
		MAIN2MCU_PLS_INTRTR0_IN_62	MAIN2MCU_PLS_INTRTR0		Pulse
	PCIE3_LEGACY_PULSE_INT	GIC500_SPI_IN_380	COMPUTE_CLUSTER0	PCIE3 legacy interrupt	Pulse
		C66SS0_INTRTR0_IN_238	C66SS0_INTRTR0		Pulse
		C66SS1_INTRTR0_IN_238	C66SS1_INTRTR0		Pulse
		R5FSS0_INTRTR0_IN_53	R5FSS0_INTRTR0		Pulse
		R5FSS1_INTRTR0_IN_53	R5FSS1_INTRTR0		Pulse
		MAIN2MCU_PLS_INTRTR0_IN_55	MAIN2MCU_PLS_INTRTR0		Pulse



**Table 12-241. PCIe Subsystem Hardware Requests (continued)**

PCIE3_LINK_STATE_PULSE_INT	GIC500_SPI_IN_386	COMPUTE_CLUSTER0	PCIE3 link state interrupt	Pulse
	C66SS0_INTRTR0_IN_244	C66SS0_INTRTR0		Pulse
	C66SS1_INTRTR0_IN_244	C66SS1_INTRTR0		Pulse
	R5FSS0_INTRTR0_IN_59	R5FSS0_INTRTR0		Pulse
	R5FSS1_INTRTR0_IN_59	R5FSS1_INTRTR0		Pulse
	MAIN2MCU_PLS_INTRTR0_IN_59	MAIN2MCU_PLS_INTRTR0		Pulse
PCIE3_LOCAL_LEVEL_INT	GIC500_SPI_IN_384	COMPUTE_CLUSTER0	PCIE3 local interrupt	Level
	C66SS0_INTRTR0_IN_242	C66SS0_INTRTR0		Level
	C66SS1_INTRTR0_IN_242	C66SS1_INTRTR0		Level
	R5FSS0_INTRTR0_IN_57	R5FSS0_INTRTR0		Level
	R5FSS1_INTRTR0_IN_57	R5FSS1_INTRTR0		Level
	MAIN2MCU_LVL_INTRTR0_IN_86	MAIN2MCU_LVL_INTRTR0		Level
PCIE3_PWR_STATE_PULSE_INT	GIC500_SPI_IN_387	COMPUTE_CLUSTER0	PCIE3 power state interrupt	Pulse
	C66SS0_INTRTR0_IN_245	C66SS0_INTRTR0		Pulse
	C66SS1_INTRTR0_IN_245	C66SS1_INTRTR0		Pulse
	R5FSS0_INTRTR0_IN_60	R5FSS0_INTRTR0		Pulse
	R5FSS1_INTRTR0_IN_60	R5FSS1_INTRTR0		Pulse
	MAIN2MCU_PLS_INTRTR0_IN_60	MAIN2MCU_PLS_INTRTR0		Pulse
PCIE3_PHY_LEVEL_INTERRUPT	GIC500_SPI_IN_383	COMPUTE_CLUSTER0	PCIE3 PHY interrupt	Level
	C66SS0_INTRTR0_IN_241	C66SS0_INTRTR0		Level
	C66SS1_INTRTR0_IN_241	C66SS1_INTRTR0		Level
	R5FSS0_INTRTR0_IN_56	R5FSS0_INTRTR0		Level
	R5FSS1_INTRTR0_IN_56	R5FSS1_INTRTR0		Level
	MAIN2MCU_LVL_INTRTR0_IN_85	MAIN2MCU_LVL_INTRTR0		Level
PCIE3_PTM_VALID_PULSE_INT	GIC500_SPI_IN_388	COMPUTE_CLUSTER0	PCIE3 PTM valid interrupt	Pulse
	C66SS0_INTRTR0_IN_246	C66SS0_INTRTR0		Pulse
	C66SS1_INTRTR0_IN_246	C66SS1_INTRTR0		Pulse
	R5FSS0_INTRTR0_IN_61	R5FSS0_INTRTR0		Pulse
	R5FSS1_INTRTR0_IN_61	R5FSS1_INTRTR0		Pulse
	MAIN2MCU_PLS_INTRTR0_IN_61	MAIN2MCU_PLS_INTRTR0		Pulse
PCIE3_ECC0_CORR_LEVEL_INT	ESM_LVL_EVENT_381	ESM0	PCIE3 ECC AGGR 0 correctable error interrupt	Level
PCIE3_ECC0_UNCORR_LEVEL_INT	ESM_LVL_EVENT_382	ESM0	PCIE3 ECC AGGR 0 uncorrectable error interrupt	Level
PCIE3_ECC1_UNCORR_LEVEL_INT	ESM_LVL_EVENT_383	ESM0	PCIE3 ECC AGGR 1 uncorrectable error interrupt	Level
PCIE3_CPTS_PENDING_INTERRUPT	GIC500_SPI_IN_390	COMPUTE_CLUSTER0	PCIE3 Timesync Interrupt	Level
	C66SS0_INTRTR0_IN_248	C66SS0_INTRTR0		Level

**Table 12-241. PCIe Subsystem Hardware Requests (continued)**

		C66SS1_INTRTR0_IN_248	C66SS1_INTRTR0		Level
		R5FSS0_INTRTR0_IN_63	R5FSS0_INTRTR0		Level
		R5FSS1_INTRTR0_IN_63	R5FSS1_INTRTR0		Level
		MAIN2MCU_LVL_INTRTR0_IN_87	MAIN2MCU_LVL_INTRTR0		Level
DMA Events					
Module Instance	Module DMA Event	Destination DMA Event Input	Destination	Description	Type
PCIE0, PCIE1, PCIE2 and PCIE3	-	-	-	PCIe subsystems do not provide built-in DMA capabilities	-
Time Sync and Compare Events (Output)					
Module Instance	Module Event	Destination Event Input	Destination	Description	Type
PCIE0	PCIE0_CPTS_HW1_P_USH_INT	TIMESYNC_INTRTR0_IN_20	TIMESYNC_INTRTR0	PCIE0 CPTS hardware push event (HW1_TS_PUSH)	Edge
	PCIE0_CPTS_COMP_INT	CMPEVENT_INTRTR0_IN_4	CMPEVENT_INTRTR0	PCIE0 CPTS compare output interrupt	
	PCIE0_CPTS_SYNC_INT	TIMESYNC_INTRTR0_IN_32	TIMESYNC_INTRTR0	PCIE0 CPTS sync output interrupt	
	PCIE0_CPTS_GENF0_INT	TIMESYNC_INTRTR0_IN_10	TIMESYNC_INTRTR0	PCIE0 CPTS GENF0 output interrupt	
PCIE1	PCIE1_CPTS_HW1_P_USH_INT	TIMESYNC_INTRTR0_IN_21	TIMESYNC_INTRTR0	PCIE1 CPTS hardware push event (HW1_TS_PUSH)	Edge
	PCIE1_CPTS_COMP_INT	CMPEVENT_INTRTR0_IN_5	CMPEVENT_INTRTR0	PCIE1 CPTS compare output interrupt	
	PCIE1_CPTS_SYNC_INT	TIMESYNC_INTRTR0_IN_33	TIMESYNC_INTRTR0	PCIE1 CPTS sync output interrupt	
	PCIE1_CPTS_GENF0_INT	TIMESYNC_INTRTR0_IN_11	TIMESYNC_INTRTR0	PCIE0 CPTS GENF0 output interrupt	
PCIE2	PCIE2_CPTS_HW1_P_USH_INT	TIMESYNC_INTRTR0_IN_22	TIMESYNC_INTRTR0	PCIE2 CPTS hardware push event (HW1_TS_PUSH)	Edge
	PCIE2_CPTS_COMP_INT	CMPEVENT_INTRTR0_IN_6	CMPEVENT_INTRTR0	PCIE2 CPTS compare output interrupt	
	PCIE2_CPTS_SYNC_INT	TIMESYNC_INTRTR0_IN_34	TIMESYNC_INTRTR0	PCIE2 CPTS sync output interrupt	
	PCIE2_CPTS_GENF0_INT	TIMESYNC_INTRTR0_IN_12	TIMESYNC_INTRTR0	PCIE2 CPTS GENF0 output interrupt	
PCIE3	PCIE3_CPTS_HW1_P_USH_INT	TIMESYNC_INTRTR0_IN_23	TIMESYNC_INTRTR0	PCIE3 CPTS hardware push event (HW1_TS_PUSH)	Edge
	PCIE3_CPTS_COMP_INT	CMPEVENT_INTRTR0_IN_7	CMPEVENT_INTRTR0	PCIE3 CPTS compare output interrupt	
	PCIE3_CPTS_SYNC_INT	TIMESYNC_INTRTR0_IN_35	TIMESYNC_INTRTR0	PCIE3 CPTS sync output interrupt	
	PCIE3_CPTS_GENF0_INT	TIMESYNC_INTRTR0_IN_13	TIMESYNC_INTRTR0	PCIE3 CPTS GENF0 output interrupt	
Time Sync Events (Input)					
Module Instance	Module Event	Source Event Output	Source	Description	Type

**Table 12-241. PCIe Subsystem Hardware Requests (continued)**

PCIE0	PCIE0_CPTS_HW2_P USH_INT	TIMESYNC_INTRTR0_OUTL_ 20	TIMESYNC_INTRTR0	PCIE0 CPTS hardware time stamp push event (HW2_TS_PUSH)	Edge
PCIE1	PCIE1_CPTS_HW2_P USH_INT	TIMESYNC_INTRTR0_OUTL_ 21	TIMESYNC_INTRTR0	PCIE0 CPTS hardware time stamp push event (HW2_TS_PUSH)	Edge
PCIE2	PCIE2_CPTS_HW2_P USH_INT	TIMESYNC_INTRTR0_OUTL_ 22	TIMESYNC_INTRTR0	PCIE2 CPTS hardware time stamp push event (HW2_TS_PUSH)	Edge
PCIE3	PCIE3_CPTS_HW2_P USH_INT	TIMESYNC_INTRTR0_OUTL_ 23	TIMESYNC_INTRTR0	PCIE3 CPTS hardware time stamp push event (HW2_TS_PUSH)	Edge

### Note

PCIe Subsystem interrupts are further described in [Section 12.2.3.4.4, PCIe Subsystem Interrupts](#).

For more information on the interconnects in device MAIN domain, see [Chapter 3, System Interconnect](#).

For more information on the power, reset and clock management in device MAIN domain, see the corresponding sections within [Chapter 5, Device Configuration](#).

For more information on the interrupt controllers in device MAIN domain, see [Chapter 9, Interrupts](#).

For more information on the time sync and compare events routers, see [Section 11.3, Time Sync and Compare Events](#).

### 12.2.3.4 PCIe Subsystem Functional Description

This chapter describes the functionality of the PCIe Gen4 controller module. It provides register description and a PCIe module configuration example.

#### 12.2.3.4.1 PCIe Subsystem Block Diagram

The block diagram of PCIe subsystem is shown in Figure 12-193. The subsystem comprises of these major components – the PCIe Core with AXI interfaces, bridges to connect to the system CBASS0 interconnect master and slave interfaces, bridges to connect the system CBASS0 configuration interfaces, additional logic to implement the Precision Time Measurement (PTM), user configuration and interrupt, and RAMs to support the controller FIFOs.

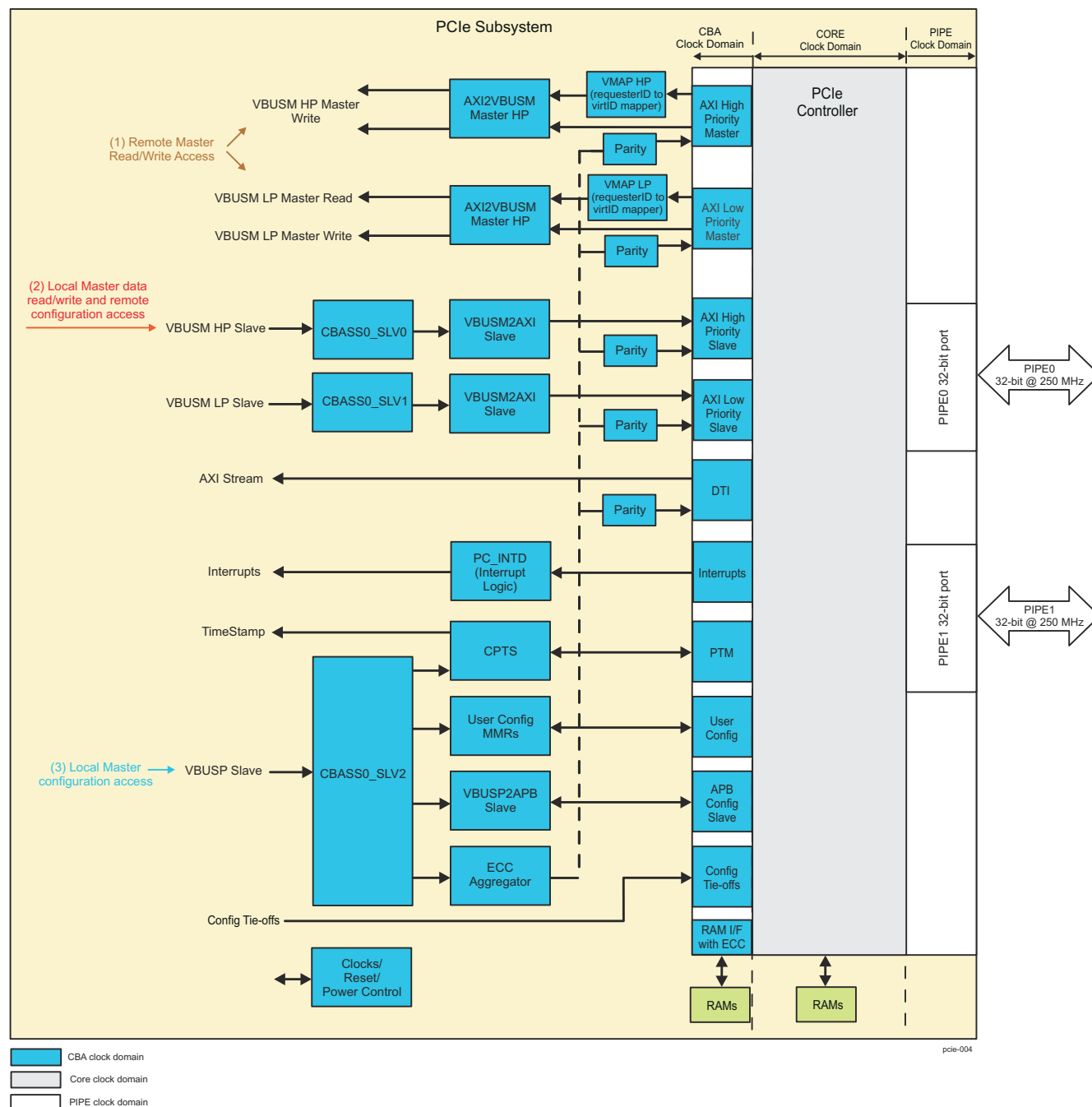


Figure 12-193 also shows example data flows in the PCIe subsystem, where:

1. A remote master issues read or write access over PCIe to the local device. This will create a command to be issued on the PCIe VBUSM master read or write interface.
2. A master in the local device wants to access a resource on the remote device over PCIe link or access the PCIe configuration registers (local or remote). This will create a transaction over the PCIe VBUSM slave interface.
3. A local master accesses registers in the PCIe Controller, ECC Aggregator or CPTS block. The PCIe VBUSM slave will be used for this transaction.

#### 12.2.3.4.1.1 PCIe Core Module

The PCIe Core module supports dual mode of operation - it can be configured as an End Point (EP) and also as a Root Port (RP). The operational mode is selected with PCIe\_MODE\_SELECT bit. Typically, it is expected that these bits be driven from a SoC level register that is programmed during initial power up based on settings in the SoC boot configuration or a non-volatile storage such as eFuse or Flash memory. The table below lists the encodings available.

It is not expected that pcie\_mode\_select would have to change during a full power cycle of the device. It is more likely that the operational mode of a SoC will stay as EP or RP for a particular end product's life cycle. The chip level circuitry is expected to provide the hardware input to set RP or EP mode and not switch back during operation and forth without a reset cycle.

The PCIe core module supports four virtual channels (VC) and four transfer classes (TC). The VCs can be used to implement Quality-of-Service (QoS) mechanism by enabling priority or round-robin arbitration. Typically, the highest numbered enabled VC is assigned the highest priority.

The PCIe core module is configured to support Single Root I/O virtualization (SR-IOV). It supports 6 Physical Functions (PF) and 16 Virtual Functions (VF).

There are two AXI4.0 master ports and two AXI4.0 slave port in the PCIe core. The two AXI master and slave ports are labeled high-priority and low-priority. Ingress data traffic that is assigned the highest priority (highest VC) will be delivered on the high-priority AXI master port. Data on the other VCs will be delivered on the low priority AXI master port. Similarly, egress data that is presented on the high-priority slave port will be assigned the highest VC by the PCIe core. Data presented on the low priority AXI slave port will be assigned lower VCs. This will enable the system to transfer the high and low priority data streams to different destinations based on the latency requirements. Having the two AXI master ports will also prevent low priority traffic from blocking the high priority traffic.

#### 12.2.3.4.1.2 PCIe PHY Interface

The PCIe subsystem incorporates a PCIe compliant PHY (PIPE) interface to connect to a SERDES-based PHY. The PCIe PHY module consist of a SERDES module and a PCIe PCS (Physical Coding Subblock) module. The SERDES module converts parallel data into PCIe serial signals and the PCIe PCS module provides an industry standard PIPE Interface to PCIe MAC. The frequency of the PIPE interface can be 62.5MHz, 125MHz, 250MHz or 500MHz depending on whether the system is operating in Gen1, Gen2, Gen3 or Gen4 modes. The width of the PIPE interface remains constant at 32-bits for all modes of operation. For more information on the SERDES module, see *Serializer/Deserializer (SerDes)*.

#### 12.2.3.4.1.3 CBA Infrastructure

The PCIe subsystem includes two CBASS modules (CBASS\_SLV0/1 and CBASS\_SLV2) to provide access to the various sub-components.

The CBASS\_SLV0/1 is used to access the PCIe egress data region through the VBUSM2AXI bridge and the AXI slave port of the PCIe controller. There are three data regions implemented in the CBASS\_SLV0/1. The PCIE\_DAT0 region provides a region with 27 bits of address and PCIE\_DAT1 region provides a region with 32 bits of address. These regions are provided to ensure separate memory spaces for 32-bit and 64-bit maps at the SoC level. The PCIE\_DAT2 region provides a region with 48 bits of address. This is used to route all requests with *case/* not equal to zero to the PCIe controller slave port. The destination for all these three regions is the

AXI slave port of the PCIe controller. CBASS\_SLV0 is used for high priority access; CBASS\_SLV1 is used for low priority access.

The CBASS\_SLV2 provides multiple regions to access the VBUSP configuration ports of the sub-components of the PCIe sub-system like the PCIe controller configuration port, USER\_CFG, VMAP, CPTS and the ECC aggregators. Having multiple regions on the VBUSP configuration port enables the SoC infrastructure to have hardware firewalls that can provide secure access to configuration registers.

The SERDES PLL should be initialized prior to accessing the PCIe core and ECC\_AGGR1 configuration registers. Any accesses to these two regions prior to initializing the SERDES PLL will complete with a read/write status error in CBASS\_SLV2.

#### **12.2.3.4.1.4 VBUSM to AXI Bridges**

A VBUSM2AXI protocol conversion bridge is attached to the each AXI slave port. This bridge translates all incoming data requests and sends them to the AXI slave port of the PCIe controller core. Wherever required, it also generates side band signals that the AXI bridge needs based upon the address and the address space of the transaction on the device's native bus.

#### **12.2.3.4.1.5 AXI to VBUSM Bridges**

Each of the two AXI master ports of the PCIe core has an AXI2VBUSM bridge that is used for high speed data transfer during data read/write transactions originated by remote devices over the PCIe link. This bridge provides the protocol translation between the AXI master interface of the PCIe controller core and the VBUSM interface used in this device.

#### **12.2.3.4.1.6 VBUSP to APB Bridge**

A VBUSP2APB bridge is used to convert the device VBUSP configuration access to the APB protocol supported by the PCIe controller core.

#### **12.2.3.4.1.7 Custom Logic**

The PCIe subsystem includes custom logic to implement Precision Time Management (PTM), interrupts and user configuration registers. The details of this logic are described in later sections of this document.

#### **12.2.3.4.2 PCIe Subsystem Reset Schemes**

This section describes the reset schemes supported by PCIe Base Specification and the way these resets are supported by the PCIe Subsystem. PCIe Base Specification specifies two mechanisms for performing reset – Conventional Reset mechanism and Functional Level Reset mechanism.

##### **12.2.3.4.2.1 PCIe Subsystem Conventional Reset**

There are three distinct types of Conventional Reset and each of these causes the hardware state machines, hardware logic, port states and configuration registers to be initialized to their default values.

1. Cold Reset – The reset occurring at power-up of the device is referred to as Cold Reset. Cold Reset is triggered by the PERSTn signal being asserted. PERSTn signal is an auxiliary signal in PCIe Specification and can be used at power on reset for the device that has PCIe as its primary bus interface to rest of the system. The device is allowed to generate its own power-on reset as long as the PCIe requirements for PERSTn are met. See PCIe Base Specifications for details.
2. Warm Reset – A reset can be triggered by the hardware without the removal of power from the device. This reset is referred to as Warm Reset. The hardware implementation is not specified by the PCIe Specification.
3. Hot Reset – The PCIe Specifications provides an in-band mechanism to propagate a Conventional Reset across a Link. This reset, called the Hot Reset, propagates via the transmission of TS1 Ordered Sets. In general, Hot Reset is software controlled procedure and can only be issued by Root Port in a PCIe network as the propagation is downstream only. PCIe subsystem translates the received in-band hot reset into an interrupt that can then be used by software to reset the PCIe subsystem.

After a conventional reset, the software must wait at least 100 ms before attempting any PCIe transaction on the device that has been reset. If the downstream device does not respond to transaction packets, it must not give up until 1 second plus an additional 50% (0.5 second) time is lapsed.

#### 12.2.3.4.2.2 PCIe Subsystem Function Level Reset

Function Level Reset (FLR) is an optional in-band reset mechanism that is used to reset one particular function in a PCIe device. The PCIe core will initiate the appropriate function level reset process when it receives the FLR message. The PCIE\_CORE\_EP\_I\_PCIE\_DEV\_CAP[28] FLRC bit indicates, if a function-level reset is active.

#### 12.2.3.4.2.3 Reset Isolation

It can be important to isolate reset of PCIe subsystem from reset of the rest of the device. The procedure to implement this depends upon what functionality the PCIe subsystem is providing – Root Port or End Point.

##### 12.2.3.4.2.3.1 Root Port Reset with Device Not Reset

When PCIe subsystem is operating as a Root Port, it is the master controller on the PCIe subsystem. When PCIe subsystem is to be reset in this operating mode, the link to the downstream device will get disconnected. To accomplish this reset, the PCIe subsystem RP should issue a hot reset to End Point or Switch downstream. It should also stop any ongoing transactions. Then, a PCIe subsystem reset can be issued. The sequence of events is outlined below:

1. Stop transactions at system and application level. This is recommended for graceful suspension of activity at system level. It is not required by PCIe subsystem though. Not choosing to gracefully stop transaction would cause some of the outstanding transactions to complete in error and it may be harmful from software stability standpoint.
2. Disable “Bus Master Enable” for End Points (see [Section 12.2.3.4.6.2, PCIe Transaction Limitations](#)). Note that a hand shake with EP Software may be required as not all End Point application software will automatically become aware of this disable action from RP.
3. Optionally, issue Hot Reset to End Point devices via PCIE\_USER\_RSTCMD[0] INIT\_HOT\_RESET bit . Note that the link will be getting disconnected. So, a reset may be happening regardless of the Hot Reset command from RP. It depends on the architecture of the other (external) device. In devices with PCIe subsystem, link disconnection automatically results in a reset request interrupt. A PCIe subsystem reset is required to get the memory buffers out of internal flush modes.
4. Initiate Clock Stop sequence and wait for Acknowledgement. For more information, see [Section 12.2.3.4.3.1, CBA Power Management](#).
5. Issue reset to PCIe subsystem (local reset).
6. Reinitialize PCIe subsystem and downstream devices. PCIe bus enumeration must be performed again.

Note that it is possible for Root Port to occasionally see the downstream device going down for some reason and disconnecting. Such events typically occur due to an internal error condition in the End Point. When such an event occurs, the sequence of events for the Root Port will be as follows:

1. The PCIe subsystem in RP mode will issue the “Reset Request” interrupt when it detects link getting disconnected unexpectedly.
2. The PCIe subsystem will flush all master transactions and any completion data from pending transactions. It will also start completing slave transactions with error completions.
3. After servicing the interrupt and disabling further interrupts, the system software must reset PCIe subsystem Root Port (see steps #4 and later specified previously in this section).

##### 12.2.3.4.2.3.2 Device Reset with Root Port Not Reset

As a Root Port, PCIe core is the master of PCIe subsystem. If the device hardware and the software is reset and re-initialized, the system does not get additional benefit of keeping the PCIe subsystem alive as all context information is lost. So, this mode is not supported.

##### 12.2.3.4.2.3.3 End Point Device Reset with Root Port Not Reset

It may be desirable to isolate the reset only to PCIe subsystem when it is operating as an End Point. Such resets may be implemented when a Hot Reset command is received from upstream device (Root Port or Switch). The sequence to be followed is as outlined below.

1. Hot Reset is received. The “Request Reset” interrupt is triggered.



2. The PCIe subsystem will automatically enter Flush Mode. The PCIe Link Training and Status State Machine (LTSSM) will be disabled automatically.
3. PCIe subsystem will complete all master transactions and any completion data from pending transactions will be accepted and discarded. It will also start issuing error response for new slave transactions.
4. Upon receipt of the reset interrupt from PCIe subsystem, the system software must immediately start preparing for shut down of PCIe subsystem. The DMA or software process accessing PCIe subsystem should gracefully suspend operations. Otherwise, step #3 may continue for unduly long time and Root Port may assume that the EP is non-responsive to its link re-training attempts.
5. Read the Reset Command register (PCIE\_USER\_RSTCMD) to check if the bridge activity flush bit is zero. This indicates that it is safe to issue warm reset to PCIe subsystem.
6. Initiate Clock Stop sequence. This will ensure no outstanding transactions exist.
7. Issue a PCIe subsystem Reset.
8. Resume initialization sequence.

#### **12.2.3.4.2.3.4 Device Reset with End Point Device Not Reset**

As an End Point, the PCIe subsystem does not need to stay operational when the device reset is happening. Upon detecting the link disconnection, the upstream port will re-train the link when Root Port issues a link retrain command to itself or to a switch port next to the EP.

If it is required that the PCIe subsystem be operational (without any transactions though), the End Point must negotiate with the other devices (primarily Root Port) to stop activity. Otherwise, if the End Point goes into force idle/standby or clock stop modes without properly managing the process, there will be timeouts or errors on incoming read transactions and writes will be completely lost. In such situation, the Root Port will encounter excessive errors and may issue a hot reset to the End Point anyway.

#### **12.2.3.4.2.4 PCIe Reset Limitations**

Once a reset situation occurs, the PCIe subsystem prepares for reset assertion from the device level reset controller. In this mode, all outbound write transactions are discarded and all outbound reads are returned with error. Similarly, all inbound reads/writes are discarded and any pending reads are completed but data is discarded. Additionally, any register reads on local or remote registers regions are returned in error even though data may be correct. Register writes on local registers are executed correctly.

Whenever the reset interrupt is asserted by PCIe subsystem, the host controller should clear the interrupt, perform a clock stop request/ack sequence and issue a local reset to PCIe subsystem.

#### **12.2.3.4.2.5 PCIe Reset Requirements**

Whenever link goes from connected to disconnected state, the PCIe subsystem requires re-initialization based on the events occurring in PCIe core. A request for reset is typically going to be issued and will manifest as a reset request interrupt. Since loopback requires transitions through link connect and disconnect, it will generate reset request interrupts. Typically, issuing the warm reset (module reset for CBA) would be sufficient.

#### **12.2.3.4.3 PCIe Subsystem Power Management**

PCIe has multiple power management protocols. Some of them are invoked by the hardware, such as Active State Power Management (ASPM), while others are activated at higher levels via software.

The PCIe core supports D0, D1 and D3-Hot power states.

Link power states L0, L0s, L1, L1s are supported.

L0s entry and exit is managed by the PCIe core if ASPM L0s is enabled.

L1 entry and exit is also managed by the PCIe core if ASPM L1 is enabled. Software can force the exit from L1 by writing to the PCIE\_USER\_PMCMD[0] CLIENT\_REQ\_EXIT\_L1 bit in the PCIe subsystem. Entry into L1 can also be blocked by setting the PCIE\_USER\_PMCMD[0] CLIENT\_REQ\_EXIT\_L1 bit.

L1s support requires that the CLKREQ pin be connected to the remote peer. L1s power state is entered when the link is in L1 and CLKREQ pin is de-asserted. Software can force the exit from



L1s by writing to either the PCIE\_USER\_PMCMD[0] CLIENT\_REQ\_EXIT\_L1 or PCIE\_USER\_PMCMD[1] CLIENT\_REQ\_EXIT\_L1\_SUBSTATE bits in the PCIe subsystem.

#### 12.2.3.4.3.1 CBA Power Management

The power management for PCIe subsystem with CBA interface is primarily accomplished through the clock stop protocol. Any time the clock stop protocol is initiated, the PCIe subsystem will acknowledge after making sure that there are no outstanding transactions. If there are any pending transactions in the subsystem, then the clock stop acknowledgement procedure cannot be completed. Therefore, it is necessary that the system software first suspend activity on the serial link as well as on the CBA interface. To do so, it is expected that the devices communicating with the device on which clock stop procedure is to be performed agree to stop transactions targeted to the device in question. Similarly, it is required that the outgoing transactions also be stopped to successfully complete the clock stop sequence. The PCIe subsystem will guarantee that in the event there are pending transactions that are still in process of draining will prevent the clock-stop acknowledgement to be issued to CBA subsystem power management logic.

#### Note

The PCIe subsystem does not have ability to terminate the clock stop state. Therefore, any wakeup sequence can only be initiated through other means such as software driven timers or software detected events occurring outside of PCIe subsystem.

#### 12.2.3.4.4 PCIe Subsystem Interrupts

The PCIe subsystem provides a number of interrupts, the PCIe controller interrupt, CPTS interrupt, and interrupts generated in the PCIe subsystem using device Interrupt Distributor (CP\_INTD).

The CP\_INTD module is used to generate and aggregate interrupts from various status signals of the PCIe core. Interrupts from the PCIe core and CPTS modules are ported out directly to the subsystem boundary without any aggregation. [Table 12-242](#) shows the details of the interrupt outputs

**Table 12-242. PCIe Core Interrupt Events**

Name	Width	Type	Polarity	Category	Clock	Reset	Description
PCIE_LEGACY_PULSE	1	Pulse	High	Func	PCIE_CBA_CLK	PCIE_RST_MOD_G_RST_N	PCIe legacy interrupt. INTA_OUT, INTB_OUT, INTC_OUT and INTD_OUT status outputs from the PCIe core are aggregated. Note: Valid in RP mode only.
PCIE_ERROR_PULSE	1	Pulse	High	Func	PCIE_CBA_CLK	PCIE_RST_MOD_G_RST_N	PCIe error interrupt. FATAL_ERROR_OUT, NON_FATAL_ERROR_OUT and CORRECTABLE_ERROR_OUT status outputs from the PCIe core are aggregated. Note: Valid in both RP and EP modes.
PCIE_ASF_PULSE	1	Pulse	High	Internal Diagnostics	PCIE_CBA_CLK	PCIE_RST_MOD_G_RST_N	PCIe active internal diagnostics interrupt. All active internal diagnostics status signals (asf_*) from the PCIe core are aggregated. Note: Valid in both RP and EP modes.
PCIE_FLR_PULSE	1	Pulse	High	Func	PCIE_CBA_CLK	PCIE_RST_MOD_G_RST_N	PCIe Function Level interrupt. FLR_IN_PROGRESS[5:0] and VF_FLR_IN_PROGRESS[15:0] status outputs from the PCIe core are aggregated. Note: Valid in EP mode only.

**Table 12-242. PCIe Core Interrupt Events (continued)**

Name	Width	Type	Polarity	Category	Clock	Reset	Description
PCIE_DOWNSTREAM_PULSE	1	Pulse	High	Func	PCIE_CBA_CLK	PCIE_RST_MOD_G_RST_N	<p>PCie downstream interrupt.</p> <ul style="list-style-type: none"> <li>F0_VSEC_INTERRUPT_OUT,</li> <li>F1_VSEC_INTERRUPT_OUT,</li> <li>F2_VSEC_INTERRUPT_OUT,</li> <li>F3_VSEC_INTERRUPT_OUT,</li> <li>F4_VSEC_INTERRUPT_OUT and</li> <li>F5_VSEC_INTERRUPT_OUT</li> </ul> <p>status outputs from the PCie controller are aggregated. Note: Valid in EP mode only</p>
PCIE_HOT_RESET_PULSE	1	Pulse	High	Func	PCIE_CBA_CLK	PCIE_RST_MOD_G_RST_N	<p>PCie hot reset interrupt. HOT_RESET_OUT status output from the PCie controller is aggregated. Note: Valid in EP mode only.</p>
PCIE_LINK_STATE_PULSE	1	Pulse	High	Func	PCIE_CBA_CLK	PCIE_RST_MOD_G_RST_N	<p>PCie link state interrupt. LINK_DOWN_RESET_OUT status from the PCie core is aggregated. Note: Valid in both RP and EP modes.</p>
PCIE_PWR_STATE_PULSE	1	Pulse	High	Func	PCIE_CBA_CLK	PCIE_RST_MOD_G_RST_N	<p>PCie power state interrupt. POWER_STATE_CHANGE and DPA_INTERRUPT[5:0] status output from the PCie core are aggregated. Note: Valid in both EP and RP modes.</p>
PCIE_LOCAL_LEVEL	1	Level	High	Func	PCIE_CBA_CLK	PCIE_RST_MOD_G_RST_N	<p>PCie local interrupt The LOCAL_INTERRUPT from the PCie controller is brought out directly without any aggregation. Note: Valid in both RP and EP modes.</p>
PCIE_PHY_LEVEL	1	Level	High	Func	PCIE_CBA_CLK	PCIE_RST_MOD_G_RST_N	<p>PCie PHY interrupt. The PHY_INTERRUPT_OUT from the PCie controller is brought out directly without any aggregation. Note: Valid in both RP and EP modes.</p>
PCIE_PTM_VALID_PULSE	1	Pulse	High	Func	PCIE_CBA_CLK	PCIE_RST_MOD_G_RST_N	<p>PCie PTM valid interrupt PTM_LOCAL_TIMER_OUT_VALID status output from the PCie controller is aggregated. Note: Valid only in EP mode.</p>
PCIE_ECC0_UNCORR_PULSE	1	Pulse	High	Internal Diagnostics	PCIE_CBA_CLK	PCIE_RST_MOD_G_RST_N	<p>PCie ECC Aggregator0 uncorrected pulse interrupt.</p>
PCIE_ECC0_UNCORR_LEVEL	1	Level	High	Internal Diagnostics	PCIE_CBA_CLK	PCIE_RST_MOD_G_RST_N	<p>PCie ECC Aggregator0 uncorrected level interrupt.</p>

**Table 12-242. PCIe Core Interrupt Events (continued)**

Name	Width	Type	Polarity	Category	Clock	Reset	Description
PCIE_ECC0_CORR_PULSE	1	Pulse	High	Internal Diagnostics	PCIE_CBA_CLK	PCIE_RST_MOD_G_RST_N	PCle ECC Aggregator0 corrected pulse interrupt.
PCIE_ECC0_CORR_LEVEL	1	Level	High	Internal Diagnostics	PCIE_CBA_CLK	PCIE_RST_MOD_G_RST_N	PCle ECC Aggregator0 corrected level interrupt.
PCIE_ECC1_UNCORR_PULSE	1	Pulse	High	Internal Diagnostics	PCIE_LANE0_TX_MCLK	PCIE_RST_MOD_G_RST_N	PCle ECC Aggregator1 uncorrected pulse interrupt.
PCIE_ECC1_UNCORR_LEVEL	1	Level	High	Internal Diagnostics	PCIE_LANE0_TX_MCLK	PCIE_RST_MOD_G_RST_N	PCle ECC Aggregator1 uncorrected level interrupt.
PCIE_CPTS_PEND_INTR	1	Level	High	Func	PCIE_CBA_CLK	PCIE_RST_MOD_G_RST_N	PCle CPTS level interrupt. The EVNT_PEND_INTR from the CPTS module is brought out directly without aggregation.

#### 12.2.3.4.4.1 Interrupt Generation in EP Mode

When PCIe subsystem is operating as an End Point (EP), either the legacy interrupts, MSI or MSI-X interrupts can be triggered to the upstream ports (eventually leading to an interrupt in RP device). As per PCIe Specifications, each PCIe function may generate only one of the Legacy or MSI interrupt types as decided during configuration period.

##### 12.2.3.4.4.1.1 Legacy Interrupt Generation in EP Mode

The End Point (EP) can trigger generation of a PCI Legacy Interrupt at the Root Port via an in-band Assert\_INTx / Deassert\_INTx messages (where x = A, B, C, or D). Software can write to the PCIE\_USER\_LEGACY\_INTR\_SET register to trigger the required INTx (where x = A, B, C, or D) assert and de-assert message from the PCIe core. Once an assert message has been generated, it cannot be generated again until a deassert message is generated. Thus, only one interrupt can be pending at a time.

There is no hardware input port provided that will allow generation of legacy interrupts on the EP port.

#### Note

The interrupt messaging mechanism makes it unfeasible to guarantee a time of delivery of the interrupt unlike in conventional designs where the interrupt line is often electrically connected to the final destination.

##### 12.2.3.4.4.1.2 MSI and MSI-X Interrupt Generation

In the case of MSI interrupts signaling, the interrupt conditions are communicated from the EP to the RP via messages. Thus, upon the occurrence of an interrupt condition, a message is sent by the EP with information that identifies the origin of the interrupt. Each message is in the form of a memory write request, containing an address and a data value to be written. Each PCI function supported by a device can be assigned a separate memory address, thus providing separate virtual channels for signaling interrupts generated by each function. In addition, the MSI mechanism allows a maximum of 32 distinct data patterns in the messages generated by each PCI function, and each pattern can be assigned to an interrupt condition within the function.

In the case of MSI-X interrupts signaling, the operation is similar to the MSI mode, except that the mechanism allows a much larger number of distinct interrupt conditions as many as 2048 per function to be communicated, and enables a distinct address to be defined for each of these conditions. This mechanism requires two tables to be stored in the EP memory. The MSI-X table contains the address/data patterns to be used for each interrupt condition (as many as 2048 per function) as well as individual enable/mask bits, and the Pending Bit Array (PBA) stores the status of each interrupt condition. Interrupt conditions are communicated from the EP to the RP via messages (write requests), as in the case of the MSI mode.

#### 12.2.3.4.4.2 PCIe Interrupt Reception in EP Mode

The PCIe specification does not have provision for End Points to receive legacy interrupts. As a result, only events other than these are used to map to interrupts.

##### 12.2.3.4.4.2.1 PCIe Core Downstream Interrupts

The Vendor Specific Capability signals of the PCIe core, F0\_VSEC\_INTERRUPT\_OUT, F1\_VSEC\_INTERRUPT\_OUT, F2\_VSEC\_INTERRUPT\_OUT, F3\_VSEC\_INTERRUPT\_OUT, F4\_VSEC\_INTERRUPT\_OUT and F5\_VSEC\_INTERRUPT\_OUT, are used to generate interrupts to the EP from the RP. F0\_VSEC\_INTERRUPT\_OUT represents the interrupt for the EP Physical Function 0, F1\_VSEC\_INTERRUPT\_OUT is the interrupt output for EP Physical Function 1 and so on. These signals are aggregated into the PCIE\_DOWNSTREAM\_PULSE interrupt to the local host.

The RP can write to the Vendor specific control registers to assert these signals at the EP and this will trigger the PCIE\_DOWNSTREAM\_PULSE interrupt to the EP host.

##### 12.2.3.4.4.2.2 PCIe Core Function Level Reset Interrupts

The PCIE\_FLR\_PULSE interrupt is asserted to indicate to the host that the PCIe controller has received a function-level reset request from the remote side. The PCIE\_FLR\_PULSE interrupt is an aggregation of the FLR\_IN\_PROGRESS[5:0] and VF\_FLR\_IN\_PROGRESS[15:0] signals from the PCIe core. Each bit of FLR\_IN\_PROGRESS[5:0] represents Physical Function0 through 5 and each bit of the VF\_FLR\_IN\_PROGRESS[15:0] represents Virtual Function0 through 15.

Upon assertion of the function level reset interrupt, software will need to write to the PCIE\_USER\_FLR\_DONE[5:0] FLR\_DONE bit field within 100ms to acknowledge to the PCIe core that all the application level processing related to the function level reset is complete. Similarly, software will need to acknowledge virtual function level reset completion by writing to the PCIE\_USER\_VF\_FLR\_DONE[15:0] VF\_FLR\_DONE bit field within 100ms of the virtual function level reset being asserted.

##### 12.2.3.4.4.2.3 PCIe Core Power Management Event Interrupts

The PCIE\_PWR\_STATE\_PULSE interrupt is generated to let the software know of the power management events. The PCIE\_PWR\_STATE\_PULSE interrupt is generated by the POWER\_STATE\_CHANGE\_INTERRUPT output of the PCIe core. This interrupt is asserted when the power state of a physical or virtual function is being changed to D1 or D3 state by writing into their Power Management Control register (PCIE\_CORE\_PFn\_I\_PWR\_MGMT\_CTRL\_STAT\_REP or PCIE\_CORE\_VFm\_I\_PWR\_MGMT\_CTRL\_STAT\_REP, respectively).

Software can check the PCIE\_USER\_LINKSTATUS[23-16] POWER\_STATE\_CHANGE\_FUNCTION\_NUM register field to determine the physical function for which power state change occurred. The PCIE\_USER\_PMCMD[2] POWER\_STATE\_CHANGE\_ACK register bit can be used to acknowledge the POWER\_STATE\_CHANGE\_INTERRUPT.

The PCIE\_DPA\_PULSE interrupt is generated by aggregating the PCIe controller DPA\_INTRs interrupt status outputs. This interrupt is asserted in EP mode when there is a configuration write to the dynamic power allocation control register (PCIE\_CORE\_PFn\_I\_DPA\_CTRL\_STATUS\_REG) to modify the DPA power state of the device. The DPA\_INTR0 is asserted for such an event for PF0, the DPA\_INTR1 for PF1 and so on.

##### 12.2.3.4.4.2.4 PCIe Core Hot Reset Request Interrupt

When the link is down, the Root Port may request reset of the End Point. This request is terminated as an PCIE\_HOT\_RESET\_PULSE interrupt to the End Point host software. All outstanding transactions are completed in error on slave port and further transactions are not generated on the master port. Once the transactions are completely stopped, the software should issue a local reset to PCIe subsystem. The re-initialization process may then be started.

The PCIE\_HOT\_RESET\_PULSE interrupt is generated from the HOT\_RESET\_OUT output of the PCIe core.

#### **12.2.3.4.4.2.5 PTM Valid Interrupt**

In EP mode, the PCIe subsystem will generate the `pcie_ptm_valid_pulse` interrupt to the local CPU after a PTM dialog between the PTM requester (EP) and the PTM responder (RP). The PTM valid interrupt indicates to the CPU in the EP that the local timers have been updated with the timestamp data obtained from the RP. The CPU can read the timer registers inside the EP to determine the current timebase.

The `pcie_ptm_valid_pulse` interrupt is generated from the `PTM_LOCAL_TIMER_OUT_VALID` signal from the PCIe core.

#### **12.2.3.4.4.3 PCIe Interrupt Generation in RP Mode**

As per PCIe Base Specifications, Root Port ports only receive interrupts. There is no mechanism to generate interrupts from RP port to EP mode as per PCIe specification.

However, PCIe subsystem does support generation of interrupts from RP to EP. The downstream interrupts described in [Section 12.2.3.4.4.2.1, PCIe Core Downstream Interrupts](#) provides a mechanism for the RP to generate software triggered interrupts to the EP.

#### **12.2.3.4.4.4 PCIe Interrupt Reception in RP Mode**

##### **12.2.3.4.4.4.1 PCIe Legacy Interrupt Reception in RP Mode**

The RP can receive any of the four legacy INTx interrupts from the EP. These will trigger the `PCIE_LEGACY_PULSE` interrupt to the host CPU.

The `PCIE_LEGACY_PULSE` interrupt is an aggregation `INTA_OUT`, `INTB_OUT`, `INTC_OUT` and `INTD_OUT` signals from the PCIe core.

##### **12.2.3.4.4.4.2 MSI/MSI-X Interrupt Reception in RP Mode**

The PCIe core decodes all MSI and MSI-X messages received from the link and forwards them on the low-priority AXI master interface. These messages must then be routed to the system interrupt controller by the SoC interconnect (CBASS0).

##### **12.2.3.4.4.4.3 Advanced Error Reporting Interrupt**

If enabled by software at the time of enumeration, PCIe subsystem will generate this interrupt to indicate occurrence of errors of various levels of severity. The software can choose not to enable Advanced Error Reporting but if it enables, it must process the interrupt as per PCIe Specifications.

The advanced error reporting interrupt is signaled by the `PCIE_ERROR_PULSE` interrupt. The `PCIE_ERROR_PULSE` is an aggregation of the `FATAL_ERROR_OUT`, `NON_FATAL_ERROR_OUT` and `CORRECTABLE_ERROR_OUT` from the PCIe core.

#### **12.2.3.4.4.5 PCIe Interrupt Reception in RP and EP Mode**

These interrupts are common to both RP and EP modes of operation.

##### **12.2.3.4.4.5.1 PCIe Local Interrupt**

The `LOCAL_INTERRUPT` from the PCIe controller is ported out directly to the PCIe subsystem boundary. For more details, see the PCIe specification

##### **12.2.3.4.4.5.2 PHY Interrupt**

The `PHY_INTERRUPT_OUT` from the PCIe controller is ported out directly to the PCIe subsystem boundary. For more details, see the PCIe specification

##### **12.2.3.4.4.5.3 Link down Interrupt**

If the PHY link is disconnected, the `PCIE_LINK_STATE_PULSE` interrupt will be generated. The expected course of action is to reset the entire PCIe subsystem and restart. All application states must also be initialized so that the operations can resume following the reset and renegotiation of the link.

The `PCIE_LINK_STATE_PULSE` interrupt is generated from the `LINK_DOWN_RESET_OUT` output of the PCIe core.

#### **12.2.3.4.4.5.4 Transaction Error Interrupts**

If there is a timeout on PCIe or an abort, PCIE\_PHY\_LOCAL\_LEVEL interrupt will be issued. The PCIE\_PHY\_LOCAL\_LEVEL interrupt is generated by the aggregation of the LOCAL\_INTERRUPT and PHY\_INTERRUPT\_OUT signals from the PCIe core. For more details, see the PCIe specification.

#### **12.2.3.4.4.5.5 Power Management Event Interrupt**

This PCIE\_PWR\_STATE\_PULSE interrupt is generated to let the software know of the power management events. The PCIE\_PWR\_STATE\_PULSE interrupt is generated by aggregating the POWER\_STATE\_CHANGE\_INTERRUPT and DPA\_INTERRUPT outputs of the PCIe core.

In EP mode, software can check the PCIE\_USER\_LINKSTATUS[23-16] POWER\_STATE\_CHANGE\_FUNCTION\_NUM bit field to determine the physical function for which power state change occurred.

The PCIE\_USER\_PMCMD[2] POWER\_STATE\_CHANGE\_ACK bit can be used to acknowledge the POWER\_STATE\_CHANGE\_INTERRUPT.

#### **12.2.3.4.4.5.6 Active Internal Diagnostics Interrupts**

The active internal diagnostics interrupt is signaled by the PCIE\_ASF\_PULSE interrupt. This is an aggregation of the ASF\_CSR\_ERR, ASF\_DAP\_ERR, ASF\_INTEGRITY\_ERR, ASF\_INT\_FATAL, ASF\_INT\_NONFATAL, ASF\_PROTOCOL\_ERR, ASF\_SRAM\_CORR\_ERR, ASF\_SRAM\_UNCORR\_ERR, and ASF\_TRANS\_TO\_ERR signals from the PCIe core.

#### **12.2.3.4.4.6 ECC Aggregator Interrupts**

The PCIE\_ECC0\_UNCORR\_PULSE/PCIE\_ECC0\_UNCORR\_LEVEL and PCIE\_ECC0\_CORR\_PULSE/PCIE\_ECC0\_CORR\_LEVEL interrupts are asserted by the CBA clock domain ECC Aggregator. The PCIE\_ECC1\_UNCORR\_PULSE/PCIE\_ECC1\_UNCORR\_LEVEL interrupts are asserted by the Core clock domain ECC Aggregator. The Core clock domain ECC Aggregator correctable interrupts are not exported since this aggregator is only connected to the parity injection logic and the correctable interrupts for this module will never fire.

#### **12.2.3.4.4.7 CPTS Interrupts**

The PCIE\_CPTS\_PEND\_INT interrupt is asserted by the CPTS module in the PCIe subsystem to signal a TimeStamp event. Please refer to the CPTS module specifications for details on these interrupts.

#### **12.2.3.4.5 PCIe Subsystem DMA Support**

The PCIe Subsystem has different requirements based upon whether it is operated in Root Port (RP) or End Point (EP) mode. The PCIe subsystem does not have DMA capabilities built into it. It has internal slave and master ports connected to the device-level interconnect.

An external DMA engine can make burst data read/writes on the slave port and the master port on PCIe subsystem can initiate reads/writes to memory on behalf of a remote PCIe device.

The PCIe subsystem does not specify the DMA protocol that is used for data transfers. The software implementations on the two ends of the PCIe link implement a data transfer protocol that is compatible with each other.

As a result, there will be several software drivers required – one driver that will manage the PCIe subsystem in RP mode and another set of drivers that will run on the RP side and will manage each of the End Points connected to the PCIe subsystem RP.

Similarly, when PCIe subsystem is operating as an End Point, there are two drivers – one to manage the PCIe subsystem from the device side and another driver that will run on the device operating as the Root Port.

##### **12.2.3.4.5.1 PCIe DMA Support in Root Port Mode**

When operating in Root Port mode, a DMA controller internal to the device (outside PCIe subsystem) can perform DMA to and from any remote device located on the PCIe subsystem. The memory address of such



devices is available to the software via the PCIe bus enumeration procedure. In addition, the PCIe subsystem has a provision to perform memory address translation on outbound requests. Thus, the software is able to map different memory regions in its memory map to correspond to different addresses (and different access types) on the PCIe side.

#### **12.2.3.4.5.2 PCIe DMA Support in End Point Mode**

When operating as a PCIe End Point, the device will be located in PCIe memory map at location programmed in the Base Address Registers by the PCIe Root device. Any PCIe transactions destined for the device from the upstream ports will get transferred to the master port on the PCIe subsystem. Similarly, any transactions originating from the software will be sent over to PCIe link.

In End Point mode, the PCIe subsystem provides address translation functionality. It is possible to map memory accesses originating on PCIe side to memory accesses with different address on the CBA bus side. These address ranges are configurable through application registers.

#### **12.2.3.4.6 PCIe Transactions**

##### **12.2.3.4.6.1 PCIe Supported Transactions**

PCIe subsystem supports the following transactions:

1. Memory Read/Write in inbound/outbound direction in Root Port (RP) and End Point (EP) modes.
2. I/O Read/Write in outbound direction in RP mode.
3. Configuration Read/Write in outbound direction in RP mode.
4. Configuration Read/Write in inbound direction in EP mode.
5. Message Transactions for interrupts and power management in RP and EP modes.

The following transactions are not supported:

1. Locked Read transactions and associated messages.
2. Inbound I/O Transaction Layer Packets (TLPs).
3. User defined messages.

##### **12.2.3.4.6.2 PCIe Transaction Limitations**

There are some limitations that must be adhered to while issuing transactions to PCIe subsystem internal bus interface.

- System Initialization

#### **CAUTION**

The PCIe subsystem operation is completely dependent upon the availability of the SERDES module for PCIe transactions. The SERDES module must be configured and corresponding PLLs must be locked before the PCIe subsystem is able to process any data transactions. Any data accesses to the PCIe subsystem prior to this initialization may result in an internal bus hang condition.

- Remote Configuration and I/O Requests

Since the remote configuration and I/O transaction windows are directly mapped to internal bus space, the software must care to not access these spaces when there is no operational PCIe link. No response may be generated for such transactions. It is recommended that checks be built into software to avoid remote accesses in the absence of an operational link.

- Byte Strobe Limitations

For any type of write transactions, the byte enables can only have a single unbroken string of 1s. In other words, in a transaction, if a byte's write strobe is set, then all following bytes must have write strobe set until the last byte with write enabled. "Holes" or "Zeros" in between the byte enables are not allowed.

Since the internal bus width is greater than 32-bit, the TLP (Transaction Layer Packets) size will not be 1 (PCIe counts in 32-bit units) and therefore, it is through the FBE/LBE (First/Last Byte Enable) that the actual data transfer size is controlled.

- Burst Type Limitations

The PCIe core and respectively the PCIe subsystem does not support 'fixed' or 'wrap' burst types on its Slave or Master port. Transactions that require any burst type other than incremental burst type will result in unspecified behavior, possibly bus lock up.

- Transaction Address Alignment

The PCIe subsystem imposes a limitation of a maximum of 128-byte outbound read/write command. However, if the starting address is not aligned to an 8-byte boundary, then the maximum transaction size is reduced to 120 bytes. This limitation is placed to avoid arithmetic overflow in computing transaction length from CBA to AXI. Unspecified behavior will occur if misaligned transactions in outbound direction are not limited to a maximum of 120 bytes.

- Read Interleaving

Read interleaving refers to the process of returning split read responses from multiple transactions. This implies that read data is not guaranteed to be sent in sequential order (data for one transaction to be sent completely before the next). The PCIe core will not interleave read responses, if the outbound read command/transaction size does not exceed the maximum transaction size configured in the PCIe core.

#### **12.2.3.4.7 PCIe Subsystem Address Translation**

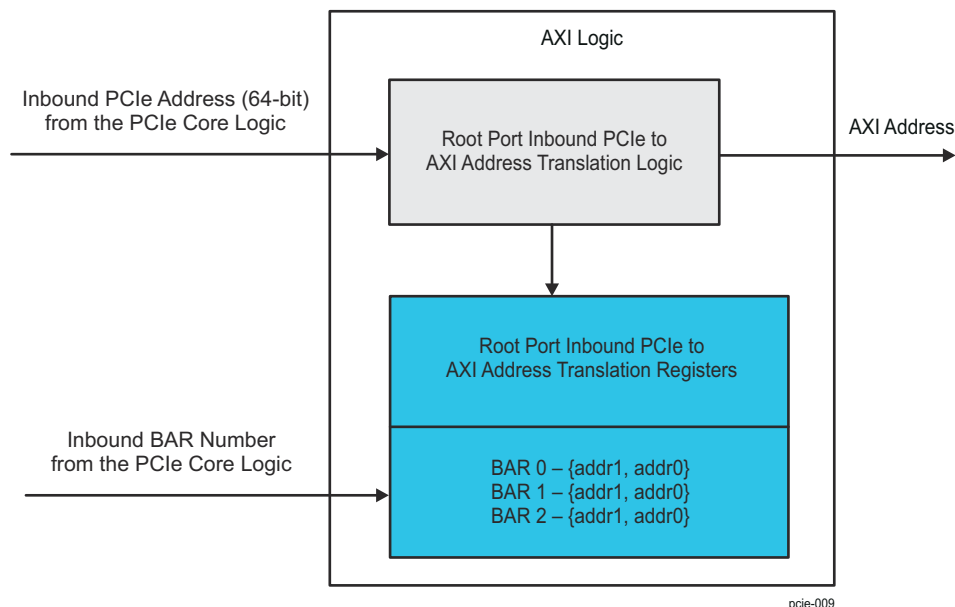
The PCIe subsystem uses the address translation registers in the PCIe core to translate outbound system addresses to PCIe space and inbound PCIe address to a valid system address. Note that internal address translation is different from the Address Translation Services (ATS).

##### **12.2.3.4.7.1 PCIe Inbound Address Translation**

###### **12.2.3.4.7.1.1 Root Port Inbound PCIe to AXI Address Translation**

The Root Port inbound PCIe to AXI address translation is performed on memory and IO TLPs. The selection of which address translation registers to use in the translation process is dependent on the BAR match of the incoming TLP. In Root Port mode there are 2 bars, so BAR 0 and BAR1 registers are implemented. There is a BAR7 register which is used as a no match BAR address translation register. In Root Port mode there are 2 bars but a BAR value of 7 will be indicated by HAL2AXI when BAR matching is disabled. Any address that does not match the Root Port BARs will be sent out as a BAR7 TLP. Each BAR register is implemented as two 32-bit registers which are named addr0 and addr1. The "Root Port Inbound PCIe to AXI Address Translation Logic" takes the upper bits from the "Root Port Inbound PCIe to AXI Address Translation Registers" and the lower bits are taken from the inbound PCIe address to form the AXI address. An addr0 [5:0] + 1 number of lower bits are passed from the inbound PCIe address to AXI address. In other words, the number of bits taken from inbound PCIe address is given by the addr0[5:0] + 1 value.





**Figure 12-194. PCIE Root Port Inbound PCIe to AXI Address Translation**

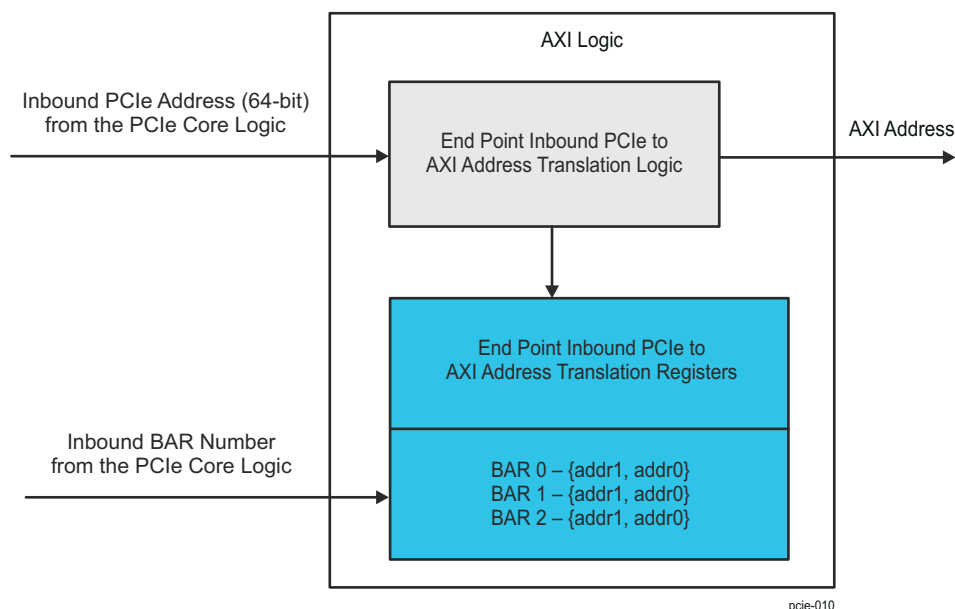
A set of registers corresponding to one Root Port BAR is shown in [Table 12-243](#).

**Table 12-243. PCIE Root Port Inbound PCIe to AXI Address Translation Registers for one BAR**

Register Name	Bits	Description	Default Value
addr1	31:0	Upper [63:32] bits of the AXI address.	32'd0
addr0	31:8	Lower [31:8] bits of the AXI address.	24'd0
	7:6	Reserved	2'd0
	5:0	Number of address bits passed through from PCIe to AXI. The PCIe controller passes the programmed value + 1 bits from PCIe to AXI. Minimum value to be programmed into this field is 7 as the lower 8 bits of the base address programmed in these registers (AXI) are replaced by zeros by the Root Port Inbound PCIe to AXI Address Translation Logic.	6'd0

#### 12.2.3.4.7.1.2 End Point Inbound PCIe to AXI Address Translation

The End Point Inbound PCIe to AXI address translation is performed on memory and IO TLPs. The selection of which address translation registers to use in the translation process is dependent on the function number and BAR match of the incoming TLP. In End Point mode there are 7 bars per function, so 7 sets of registers are implemented per function, each BAR having two 32-bit registers (addr0 and addr1). The "End Point Inbound PCIe to AXI Address Translation Logic" takes the upper bits from the "End Point Inbound PCIe to AXI Address Translation Registers" and the lower bits are taken from the Inbound PCIe Address to form the AXI address. The number of bits to pass from Inbound PCIe Address to AXI is decided by the Inbound BAR aperture.



**Figure 12-195. PCIE End Point Inbound PCIe to AXI Address Translation**

A set of registers corresponding to one End Point BAR is shown in [Table 12-244](#).

**Table 12-244. PCIE End Point Inbound PCIe to AXI Address Translation Registers for one BAR**

Register Name	Bits	Description	Default Value
addr1	31:0	Upper [63:32] bits of the AXI address.	32'd0
addr0	31:0	Lower [31:0] bits of the AXI address.	32'd0

#### 12.2.3.4.7.2 PCIe Outbound Address Translation

The PCIe Subsystem allows mapping of PCIe addresses to/from the internal bus addresses of the device. This is accomplished by using internal address translation unit (iATU) in the PCIe core. For each outbound read/write request, the address translation module within PCIe subsystem can convert a VBUSM address to a PCIe address of memory Read/Write type.

If a transaction is large enough that it goes past the address translation region, unspecified behavior may occur. The address translation only works at the time a command is issued. So, a memory write, for example, will not automatically go to next translation region, if it starts in the previous one and is bigger than the remaining size in the starting translation region.

The “Dynamic Method: Sideband Descriptor Based” outbound address translation mechanism is used to bypass the outbound address translation unit under certain conditions.

##### 12.2.3.4.7.2.1 PCIe Outbound Address Translation Bypass

The PCIe subsystem supports bypassing the outbound address translation in the PCIe controller when the casel value on the VBUSM high-priority or VBUSM low-priority interfaces is non-zero.

When the system DMA sets the casel value for any transaction to non-zero, the ATU bypass logic in the PCIe subsystem drives the AXI\_AWUSER or AXI\_ARUSER signals to disable the address translation unit in the PCIe controller. The PCIe TLP address for that transaction will be the same as the input CBA address and there will be no translation from the system address to the PCIe address.

The following values are driven on the AXI\_AWUSER/AXI\_ARUSER by the PCIe subsystem when casel is not equal to 0.

**Table 12-245. AXI\_A\*USER Sideband Signal Mapping**

AXI_A*USER field	Description	Value
AXI_A*USER[3:0]	Mem write	0h (for read) 2h (for write)
AXI_A*USER[4]	No snoop	0h
AXI_A*USER[5]	Relaxed ordering	0h
AXI_A*USER[6]	ID based ordering	0h
AXI_A*USER[8:7]	AT bits	2h
AXI_A*USER[15:9]	Reserved	0h
AXI_A*USER[16]	Reserved	0h
AXI_A*USER[19:17]	TC	0h (for low-priority I/F) 3h (for high-priority I/F)
AXI_A*USER[20]	Poison Write TLP	0h
AXI_A*USER[21]	Insert Read ECRC	0h
AXI_A*USER[22]	Reserved	0h
AXI_A*USER[23]	Requester ID Enable	0h
AXI_A*USER[31:24]	Requester ID – Function number + Device number	0h
AXI_A*USER[39:32]	Requester ID – Bus number	0h
AXI_A*USER[47:40]	Reserved	0h
AXI_A*USER[60:48]	Reserved	0h
AXI_A*USER[63:61]	Reserved	0h
AXI_A*USER[84:64]	Reserved	0h
AXI_A*USER[85]	PASID	0h
AXI_A*USER[105:86]	PASID value	0h
AXI_A*USER[106]	Privilege mode	0h
AXI_A*USER[107]	Execute mode	0h
AXI_A*USER[108]	Reserved	0h
AXI_A*USER[127:109]	Reserved	0h
AXI_A*USER[3:0]	Sideband valid	1h

#### 12.2.3.4.8 PCIe Subsystem Virtualization Support

The PCIe subsystem supports multiple virtualization related features as described below.

##### 12.2.3.4.8.1 EP SR-IOV support

In End Point (EP) mode, the PCIe subsystem supports Single Root I/O Virtualization (SR-IOV) with four physical functions (PF) and four virtual functions (VF) per physical function for PF0-PF3 only.

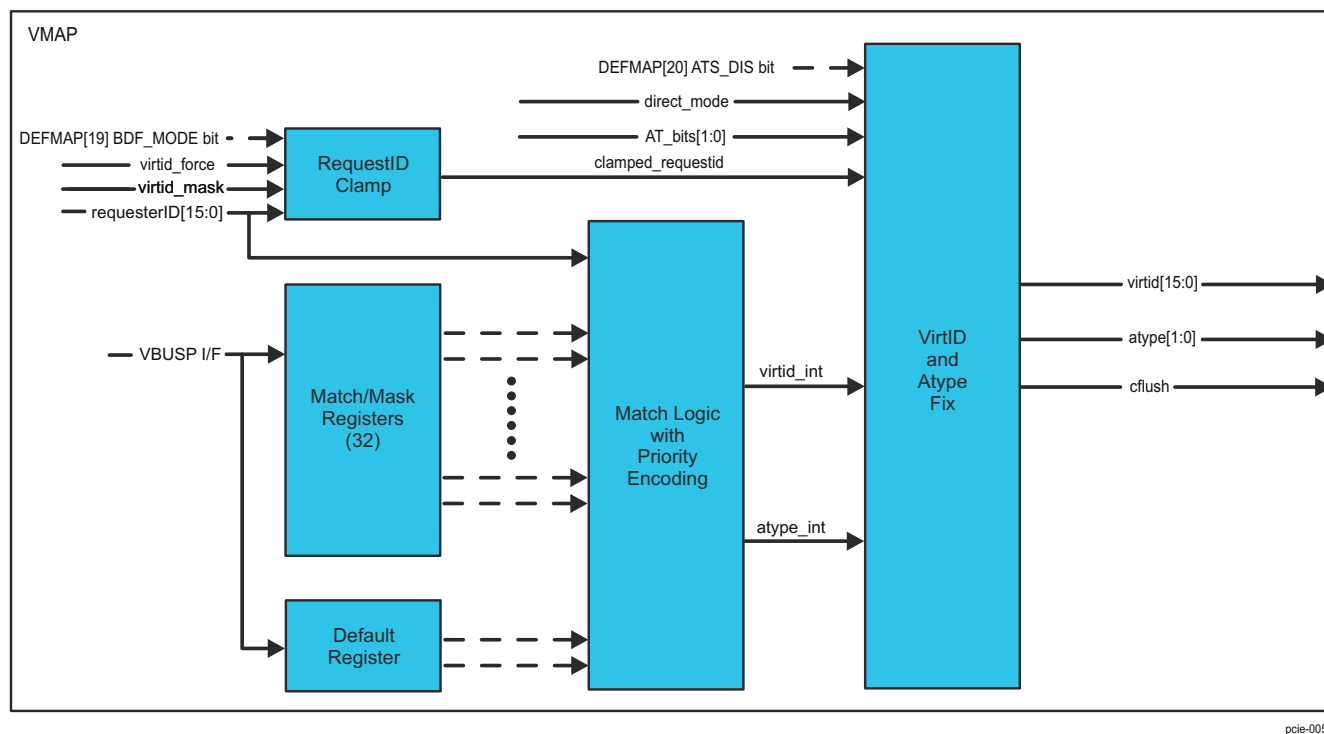
##### 12.2.3.4.8.2 RP ATS support

In Root Port (RP) mode, the PCIe subsystem has an AXI streaming interface (DTI) to connect to the system level SMMU. Address Translation Messages (ATS) from the End Point is routed to the SMMU over the DTI interface. The translated address and any invalidation requests from the SMMU, sent to the PCIe over the DTI interface, is converted to ATS messages and sent to the End Point.

##### 12.2.3.4.8.3 VirtID Mapping

The PCIe subsystem includes the VMAP module that can be programed to map (convert) the incoming 16-bit PCIe requestID + 2-bit PCIe AT bits to the 16-bit CBA virtID + 2-bit CBA atype attributes.

Figure 12-196 shows the block diagram of the VMAP module. The blocks shown in the diagram are replicated for the read and write AXI channels. However, there is only one set of configuration registers in the VMAP module and these are shared by both the read and write channel logic.



**Figure 12-196. PCIe Subsystem VirtID Mapper**

The RequestID Clamp block is used to clamp the incoming requestID to a maximum value. This block compares the incoming PCIe requestID[15:12] masked with the virtid\_mask[3:0] to an expected value. Bit [19] BDF\_MODE in PCIE\_VMAP\_DEFMAP is used to set the value to virtid\_force[3:0] or 4'h0. The clamped\_requestid is set to the maximum value of FFFFh if the comparison is false. The clamped\_requestid value is later used downstream by the PVU or SMMU in the SoC.

Each of the VMAP modules includes a set of 32 match and mask registers. Software can enable the registers to be used for the mapping by setting the [0] EN bit in PCIE\_VMAP\_CTRL\_j registers. Software can also program the [15-0] RID and [31-16] MASK bit fields in PCIE\_VMAP\_REQID\_j registers to indicate the incoming PCIe requestid that needs to be matched. The [11-0] VID and [17-16] ATYPE bit fields in PCIE\_VMAP\_VIRTID\_j registers are used to program the required CBA virtID and Atype fields in case of a match. In order to prevent a spurious match, the [15-0] RID and [31-16] MASK bit fields in PCIE\_VMAP\_REQID\_j registers should be set up before the [0] EN bit in PCIE\_VMAP\_CTRL\_j registers is set. The same sets of registers are used for both read and write transactions.

The Match logic with Priority Encoding block compares the incoming PCIe requestID with the values programmed in the VMAP control registers and sets the value of the virtid\_int and atype\_int variables. For every incoming PCIe transaction, the incoming PCIe requestID[11:0] bits are masked with the PCIE\_VMAP\_REQID\_j[31-16] MASK bit field and compared against each of the thirty two values indicated by PCIE\_VMAP\_REQID\_j[15-0] RID that are enabled. The location of the first match of this comparison operation is used to select the [11-0] VID and [17-16] ATYPE bit fields in PCIE\_VMAP\_VIRTID\_j registers. This is used as the virtid\_int and atype\_int values for the downstream logic. In case none of the enabled [15-0] RID and [31-16] MASK bit fields in PCIE\_VMAP\_REQID\_j registers pairs match the incoming requestID, the value in [11-0] DEF\_VID and [17-16] DEF\_ATYPE bit fields in PCIE\_VMAP\_DEFMAP registers is used.

The VirtID and Atype Fix block uses the incoming PCIe AT\_bits along with the matched and clamped values of the requestID from the downstream blocks to set the final values of the virtID, catype and cflush attributes for the particular CBA transaction. The values of the virtID, catype and cflush are determined based on the following criteria:

1. If the incoming PCIe TLP has a translated address, as indicated by the PCIe AT\_bits equal to 2 and if the match operation results in an expected atype value of 2 and Address Translation Services (ATS) is enabled in the system, as indicated by the PCIE\_VMAP\_DEFMAP[20] ATS\_DIS register bit, the virtID is set to the *clamped\_requestid* and the *catype* value is set to 2. This enables the PCIe subsystem to pass along the full incoming requestID[15:0] to the upstream SMMU/PVU. The incoming transaction is flushed if it fails this criteria by setting the cflush attribute to 1. The PCIe EP initiating this transaction will be sent a Completion Abort (CA) response for a read request, if the transaction is flushed.
2. If the incoming PCIe TLP does not have a translated address, as indicated by the PCIe AT\_bits equal to 0, the virtID can be set to the output of the mask/match registers or the *clamped\_requestid*. The *catype* attribute for this transaction can be set to the value from the mask/match registers.

### Note

The PCIE implementation in the device uses the *direct\_mode* workaround. This forces the *cvirtID* and *catype* to 0 when the incoming transaction matches the criteria outlined in (1) above.

The system Interconnect only supports 12-bits of *virtid*. As a result, only *virtid*[11:0] from the PCIe VMAP module will be used in the system and *virtid*[15:12] is not connected.

The pseudo-code for the virtID mapping algorithm implemented is described below.

```
// Requestid Clamp
// Note: only the low 4 bits of virtid_force & virtid_mask are used in the current PCIe wrapper
// virtid_mask[7:4] should always be 1
// virtid_mask[n] of 1 means use the virtid_force value for this bit
// virtid_mask[n] of 0 means use the requestID value for this bit, virtid_force[n] should be 0
// Bit [19] BDF_MODE in PCIE_VMAP_DEFMAP specifies if we are using
// 0 based bus numbers (0) or offset bus numbers (1)
expected_value[3:0] = PCIE_VMAP_DEFMAP[19] BDF_MODE ? virtid_force[3:0] : 4'h0;
if (requestID[15:12] & virtid_mask[3:0] == expected_value[3:0] )
{
    clamped_requestID[15:0] = requestID[15:0];
}
else
{
    clamped_requestID[15:0] = FFFFh;
}
// Match Logic with Priority Encoding
for n in 0 to 31 do
if (PCIE_VMAP_CTRL_j[0] EN == 1h &&
(requestID & PCIE_VMAP_REQID_j[31:16] MASK) == PCIE_VMAP_REQID_j[15:0] RID )
{
    found[n] = true;
}
```

```
else
{
found[n] = false;
}
done
// priority encoder for 32 results from above
// uses lowest numbered match
// also outputs a bool specifying if any match was found
all_miss = true;
for n in 0 to 31 do
if (found[n])
{
virtid_int[11:0] = PCIE_VMAP_VIRTID_j[11-0] VID;
atype_int[1:0] = PCIE_VMAP_VIRTID_j[17-16] ATYPE;
all_miss = false;
break;
}
done
// Use the default if no match is found
if (all_miss)
{
virtid_int[11:0] = PCIE_VMAP_DEFMAP[11-0] DEF_VID;
atype_int[1:0] = PCIE_VMAP_DEFMAP[17-16] DEF_ATYPE;
}
// VirtID and Atype Fix
if (AT_bits == 0x2) // Is this a pre-translated address?
{
// Is this EP using SMMU and is ATS enabled?
if (atype_int == 2h && PCIE_VMAP_DEFMAP[20] == 0h )
{
if (direct_mode == 1h) //
{
// yes, send directly to destination address
atype[1:0] = 2'h0; // Direct
virtid[15:0] = 16'h0; // should not matter
flush = 1'b0; // don't turn on the flush bit
```

```
at_cba = 1'b0; // Not used
}
else
{
// no, send to SMMU TBU for processing
atype[1:0] = 2'h2; // //SMMU
virtid[15:0] = clamped_requestID[15:0]; //Use the correct StreamID
flush = 1'b0; // don't turn on the flush bit
at_cba = 1'b1; // Indicate a pre-translated address
}
}
else
{
// All other cases should error the transaction for pre-translated transaction
// PVU & Direct should not see any pre-translated requests
// SMMU with ATS disables also errors
atype[1:0] = 2'h2; // use VirtSS to force the error
virtid[15:0] = 16'h0; // should not matter
flush = 1'b1; // force the error
at_cba = 1'b1; // should not matter
}
}
else
{
// This transaction is not pre-translated
atype[1:0] = atype_int //use the atype from the register
flush = 1'b0; // don't force an n error
at_cba = 1'b0; // this is not pre-e- translated
if (atype_int == 2'h2)
{
// this is SMMU, use the clamped RequestID
virtid[15:0] = clamped_requestID[15:0];
}
else
{
// this is not SMMU, use the value from the register
```

```

virtid[11:0] = virtid_int[11:0];
virtid[15:12] = 4'h0;
}
}

```

#### 12.2.3.4.9 PCIe Subsystem Quality-of-Service (QoS)

The Quality-of-Service (QoS) mechanism in the PCIe subsystem uses the Virtual Channel/Traffic Class (VC/TC) feature in the PCIe core in conjunction with the CBA QoS capabilities.

For ingress traffic, transactions that use the highest enabled virtual channel will be directed to the high priority master interface. Transactions using all other virtual channels will be directed to the low priority master interface.

In addition, the TC information from each transaction on the AXI master information is mapped to the CCHANID signal of the VBUSM master interface. The system level interconnect can use the CCHANID along with the orderid for QoS purposes. [Table 12-246](#) shows the TC mapping to the 12-bit CCHANID of each VBUSM master interface.

**Table 12-246. QoS Ingress CCHANID Mapping**

TC Value	CCHANID[11:0]
0	{9'd0, 3'b000}
1	{9'd0, 3'b001}
2	{9'd0, 3'b010}
3	{9'd0, 3'b011}
4	{9'd0, 3'b110}
5	{9'd0, 3'b101}
6	{9'd0, 3'b110}
7	{9'd0, 3'b111}

For egress traffic, the data presented on the VBUSM high priority port will be assigned the highest enabled VC. This will ensure that data on the high priority port will get priority on the PCIe link. All data presented on the VBUSM low priority port will be assigned lower VCs.

#### 12.2.3.4.10 PCIe Subsystem Precision Time Measurement (PTM)

Precision Time Measurement (PTM) enables precise coordination of events across multiple components with independent local time clocks. Ordinarily, such precise coordination would be difficult given that individual time clocks have differing notions of the value and rate of change of time. To work around this limitation, PTM enables components to calculate the relationship between their local times and a shared PTM Master Time: an independent time domain associated with a PTM Root.

PTM defines the following components:

- PTM Requester - A Function capable of using PTM as a consumer associated with an Endpoint.
- PTM Responder - A Function capable of using PTM to supply PTM Master Time associated with a RP.
- Time Source - A local clock associated with a PTM Responder.
- PTM Root – The source of PTM Master Time for a PTM hierarchy. A PTM Root must also be a Time Source and is typically also a PTM Responder.

When using PTM between two components on a Link, the EP sends PTM requests to the RP on the same link. During each dialog, the RP populates the PTM Response message based on timestamps stored during previous PTM dialogs. Once each component has historical timestamps from the preceding dialog, the EP can combine its timestamps with those passed in the PTM Response message to calculate the PTM Master Time.



The PCIe core implements all of the features required to handle the PTM conversation between the requestor and responder in hardware. In addition, the Timestamp module (CPTS) is connected to the timestamp interface of the PCIe core so that events can be logged.

#### 12.2.3.4.11 PCIe Subsystem Loopback

The PCIe subsystem provides loopback support through PIPE interface (Link Layer).

##### 12.2.3.4.11.1 PCIe PIPE Loopback

The procedure depends upon whether the device is operating in RP or EP Mode. In either case, the PCIe subsystem can be loopback master or loopback slave as outlined in PCIe specifications. A loopback master is the component requesting loopback and transmitting the data. A loopback slave is the component looping back the data.

#### Note

The PIPE loopback mode cannot be used for looping back transactions.

##### 12.2.3.4.11.1.1 PIPE Loopback Master Mode

The loopback path when PCIe subsystem is loopback master is:

PCIe subsystem (loopback master) -> PIPE (TX) -> PCIe Link -> Loopback -> PCIe Link -> PIPE (RX) -> PCIe subsystem

##### 12.2.3.4.11.1.2 PIPE Loopback Slave Mode

The loopback path when PCIe subsystem is loopback slave is:

Remote device -> PCIe Link -> PIPE (RX) -> Loopback -> PIPE (TX) -> PCIe Link -> Remote device

If the PCIe subsystem is a loopback slave, then the incoming serial data is routed back to the originating device from the PIPE interface as per PCIe loopback requirements. Typically, PCIe test equipment will be used as loopback master and it will transition PCIe subsystem into loopback slave state following which the inbound transactions will be loopback to the test equipment. There is no programming required on PCIe subsystem to enter loopback in slave mode. PHY support is not required to use this loopback mode.

##### 12.2.3.4.12 PCIe Subsystem Error Handling

As an End Point, PCIe subsystem reports errors to root complex so that corrective action may be taken. These messages become error interrupts on the Root Port side. In addition, the errors that are detected by the PCIe subsystem locally are also reported to software via interrupts.

Note that for several errors of the same type that are detected in close succession, the PCIe subsystem will not necessarily send as many error messages to Root Port. It is guaranteed to send at least one such error message.

##### 12.2.3.4.12.1 PCIe AXI to/from VBUSM Bus Error Mapping

[Table 12-247](#) and [Table 12-248](#) show the bus error mapping to/from the AXI interface on the PCIe controller to the system VBUSM interfaces.

The AXI2VBUSM bridge maps the bus errors on the ingress AXI interface to the VBUSM master interface as shown in the table below.

**Table 12-247. AXI to VBUSM Bus Error Mapping**

VBUSM		AXI	
Code	Error	Code	Error
0 (decimal)	Success	0h	OKAY
1 (decimal)	N/A	3h	N/A
2-3 (decimal) 7 (decimal)	All Others	2h	SLVERR

The VBUSM2AXI bridge maps the bus errors on the egress VBUSM slave interface to the AXI interface as shown in the table below.

**Table 12-248. VBUSM to AXI Bus Error Mapping**

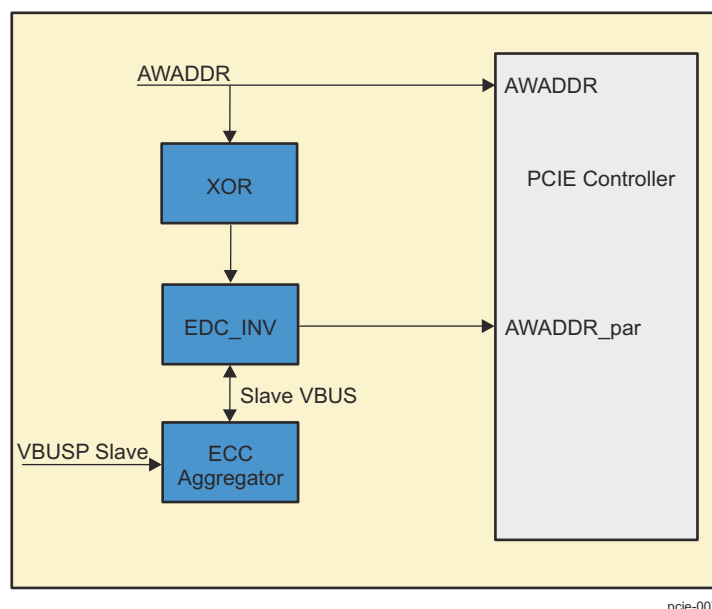
AXI		VBUSM	
Code	Error	Code	Error
N/A	N/A	2 (decimal)	Protection Error
0h	OKAY	0 (decimal)	Success
3h	DECERR	1 (decimal)	Addressing Error
2h	SLVERR	1 (decimal)	Addressing Error

### 12.2.3.4.13 PCIe Subsystem Internal Diagnostics Features

#### 12.2.3.4.13.1 PCIe Parity

All the parity inputs on the PCIe controller core are driven by the EDC\_INV module. This allows the parity input to the PCIe controller to be inverted and thus inject an error. [Figure 12-197](#) shows the parity logic on the AXI AWADDR port as an example. The parity of the AWADDR inputs to the PCIe controller are calculated as an XOR and this is fed into the PCIe controller using the EDC\_INV block. The input AWADDR\_par can be inverted using commands from the ECC Aggregator to enable error injection. Similar logic is present on all other parity inputs to the PCIe controller.

Checking input parity and output parity are handled inside the PCIe controller. Any errors in parity are signaled using the PCIe active internal diagnostics interrupt (PCIE\_ASF\_PEND).



**Figure 12-197. Parity Logic on AXI AWADDR Input**

#### 12.2.3.4.13.2 ECC Aggregators

The PCIe subsystem instantiates two ECC Aggregators: AXI\_ECC\_AGGR and CORE\_ECC\_AGGR.

AXI\_ECC\_AGGR is connected to the RAMs in PCIe CBA Clock Domain and CORE\_ECC\_AGGR is connected to the RAMs in the PIPE Clock Domain.

The ECC Aggregator modules integrated within the PCIe subsystem provide a mechanism to control and monitor the ECC RAMs via Single Error Correction (SEC) and Double Error Detection (DED) functions. Each ECC Aggregator module gathers level pending status from the ECC RAMs into two interrupts to system level. One interrupt is for correctable errors (SEC) and the other one for uncorrectable errors (DED). The ECC

Aggregator supports software readable status of ECC single/double-bit errors and associated information such as RAM address and data bit(s) that are in error. Write back correction for async read RAMs is not supported.

For more information on the ECC Aggregator operation, see [Section 12.11.4, ECC Aggregator](#).

#### 12.2.3.4.13.3 RAM ECC inversion

The RAM ECC logic inside the PCIe controller can be tested using the ECC inversion logic.

#### 12.2.3.4.14 LTSSM State Encoding

[Table 12-249](#) is used when the PCIe/M-PCIe core is configured to be operating in PCIe mode.

[Table 12-249](#) provides the encoding of the LTSSM states on the LTSSM\_STATE output of the core, as well the state read from the Physical Layer Configuration Register 0.

**Table 12-249. LTSSM State Encoding**

LTSSM State Name	Value (hex)
Detect.Quiet	00
Detect.Active	01
Polling.Active	02
Polling.Compliance	03
Polling.Configuration	04
Configuration.Linkwidth.Start	05
Configuration.Linkwidth.Accept	06
Configuration.Lanenum.Accept	07
Configuration.Lanenum.Wait	08
Configuration.Complete	09
Configuration.Idle	0A
Recovery.RcvrLock	0B
Recovery.Speed	0C
Recovery.RcvrCfg	0D
Recovery.Idle	0E
L0	10
Rx_L0s.Entry	11
Rx_L0s.Idle	12
Rx_L0s.FTS	13
Tx_L0s.Entry	14
Tx_L0s.Idle	15
Tx_L0s.FTS	16
L1.Entry	17
L1.Idle	18
L2.Idle	19
L2.TransmitWake	1A
Disabled	20
Loopback.Entry (Master)	21
Loopback.Active (Master)	22
Loopback.Exit (Master)	23
Loopback.Entry (Slave)	24
Loopback.Active (Slave)	25
Loopback.Exit (Slave)	26
Hot Reset	27

**Table 12-249. LTSSM State Encoding (continued)**

LTSSM State Name	Value (hex)
Recovery.Equalization, Phase 0	28
Recovery.Equalization, Phase 1	29
Recovery.Equalization, Phase 2	2A
Recovery.Equalization, Phase 3	2B

## 12.2.4 Universal Serial Bus (USB) Subsystem

This section describes the USB 3.0 Dual-Role-Device (DRD) interface subsystem of the device.

### Note

This chapter serves to describe the integration of the third-party USB controller and should not be considered sufficient for those wishing to modify the existing Linux or RTOS USB driver(s) or create a new driver to support this controller implementation. For those who do wish to substantially modify the existing Linux or RTOS USB driver(s), or create new drivers, contact TI for more information on how to obtain the third-party documentation under NDA.

### 12.2.4.1 USB Overview

Similar to earlier versions of USB bus, USB 3.0 is a general-purpose cable bus, supporting data exchange between a host device and a wide range of simultaneously accessible peripherals.

The device supports these USB subsystems:

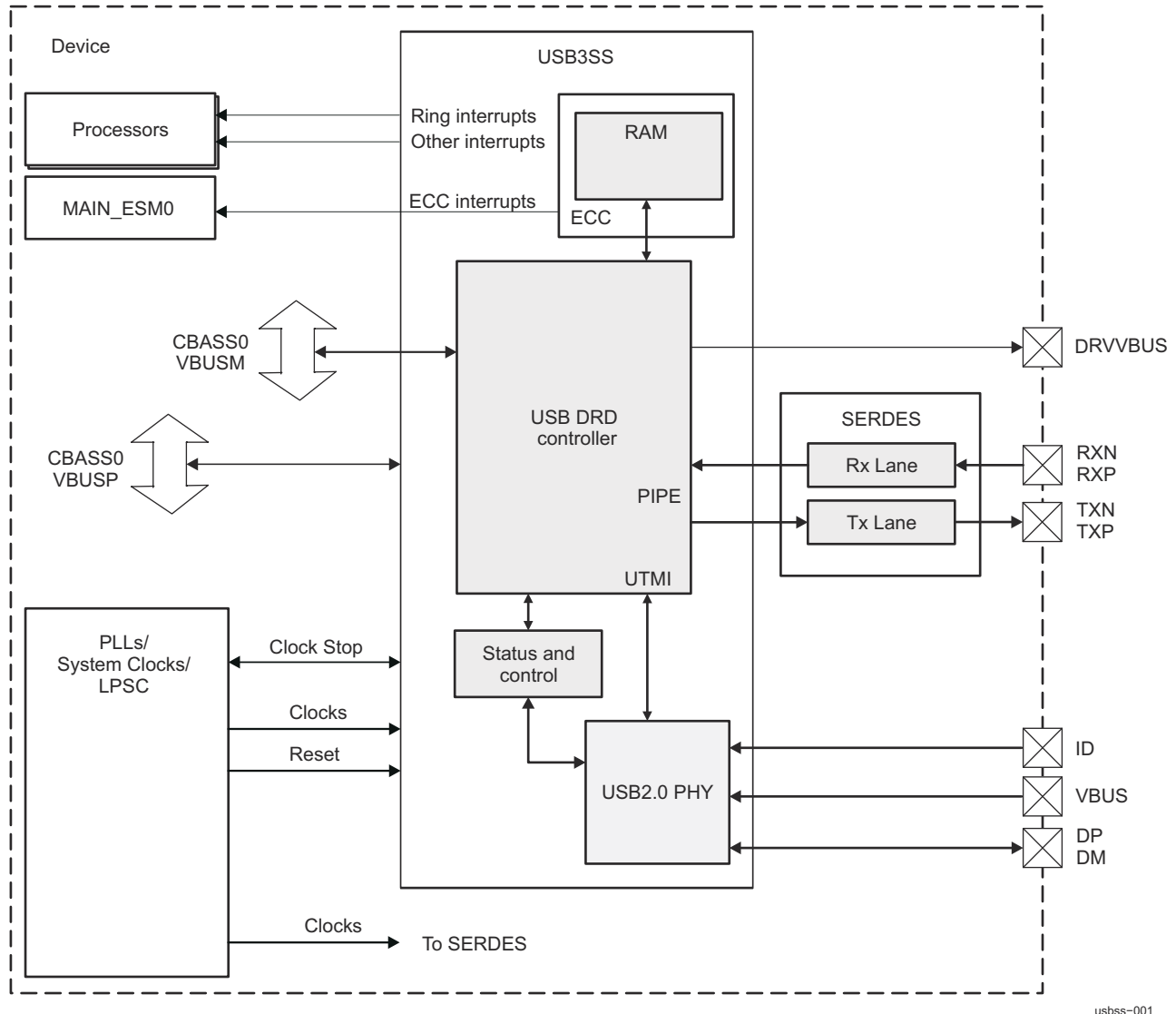
- USB3SS0 is SuperSpeed (SS) USB 3.0 Dual-Role-Device (DRD) subsystem with on-chip SS (USB3.0) PHY and HS/FS/LS <sup>1</sup>(USB2.0) PHY
- USB3SS1 is SuperSpeed (SS) USB 3.0 Dual-Role-Device (DRD) subsystem with on-chip SS (USB3.0) PHY and HS/FS/LS (USB2.0) PHY

**Table 12-250. USB Allocation Within Device Domains**

Module Instance	Domain		
	WKUP	MCU	MAIN
USB3SS0	-	-	✓
USB3SS1	-	-	✓

Figure 12-198 shows the USB subsystem highlights.

<sup>1</sup> LowSpeed (LS) is supported only in host mode, in both USB3SS0 and USB3SS1.



usbss-001

**Figure 12-198. USB Subsystem Overview**

#### 12.2.4.1.1 USB Features

The USB Dual-Role-Device (DRD) subsystem, supports the following USB features:

- General features:
  - Supports Peripheral (aka Device) mode at Superspeed (5 Gbps), Highspeed (480 Mbps), and Fullspeed (12 Mbps)
  - Supports Host mode at Superspeed (5 Gbps), Highspeed (480 Mbps), Fullspeed (12 Mbps), and Low-speed (1.5 Mbps)
  - Static peripheral operation
  - Static host operation
  - Compliant with USB 3.1 Gen1 Specification
  - Limited OTG 2.0 functionality
  - Supports VBUS and ID detection
  - Host Negotiation Protocol (HNP) support
  - Charger Downstream Port (CDP) as per Battery Charging Specification, Revision 1.2
- Each controller instance contains single xHCI with the following features:

- Compatible to the xHCI specification (revision 1.0)
- Supports 15 Transmit (TX), 15 Receive (RX) endpoints (EPs), and one EP0 endpoint which is bidirectional
- Internal scatter-gather DMA controller
- Dynamic data buffering
- USB3 power saving states (U1, U2, and U3) and USB2 L1/L2
- 64 slots supported with 32 endpoints per slot, for host operation
- Operation flexibility:
  - Uniform programming model for SS, HS, FS, and LS operation
  - Multiple interrupt lines:
    - 8 interrupts associated with 8 programmable Event Rings for multi-core support
    - Interrupt for OTG events
- Functional safety:
  - Internal RAM with ECC
  - Hardware transaction timeout monitor
- Supports VBUS and ID detection in the USB2.0 PHY
- External requirements and I/O features:
  - Requires an external charge pump or power switch for VBUS 5-V generation
  - Requires an external circuitry or PMIC to start the battery charging upon Battery Charger (BC) detected event
  - Requires external high-precision resistors for USB2.0 PHY and SERDES termination calibration
  - Supports internal data lane swapping for Type C connector. All other Type C features, like cable detection and CC configuration, have to be implemented using external circuitry
  - Supports short circuit protection to GND and short-term short circuit protection to VBUS on data pins

#### 12.2.4.1.2 USB Not Supported Features

The following are USB features which are not supported in the current device:

- OTG 3.0 functionality
- HSIC (High Speed Inter-Chip) and SSIC interface
- ULPI Interface for external PHY
- SRP and ADP protocols
- Hibernation
- Cache-line wrap addressing mode on both bus interfaces
- Debug trace interface
- xHCI IO virtualization
- MSI support

#### 12.2.4.1.3 USB Terminology

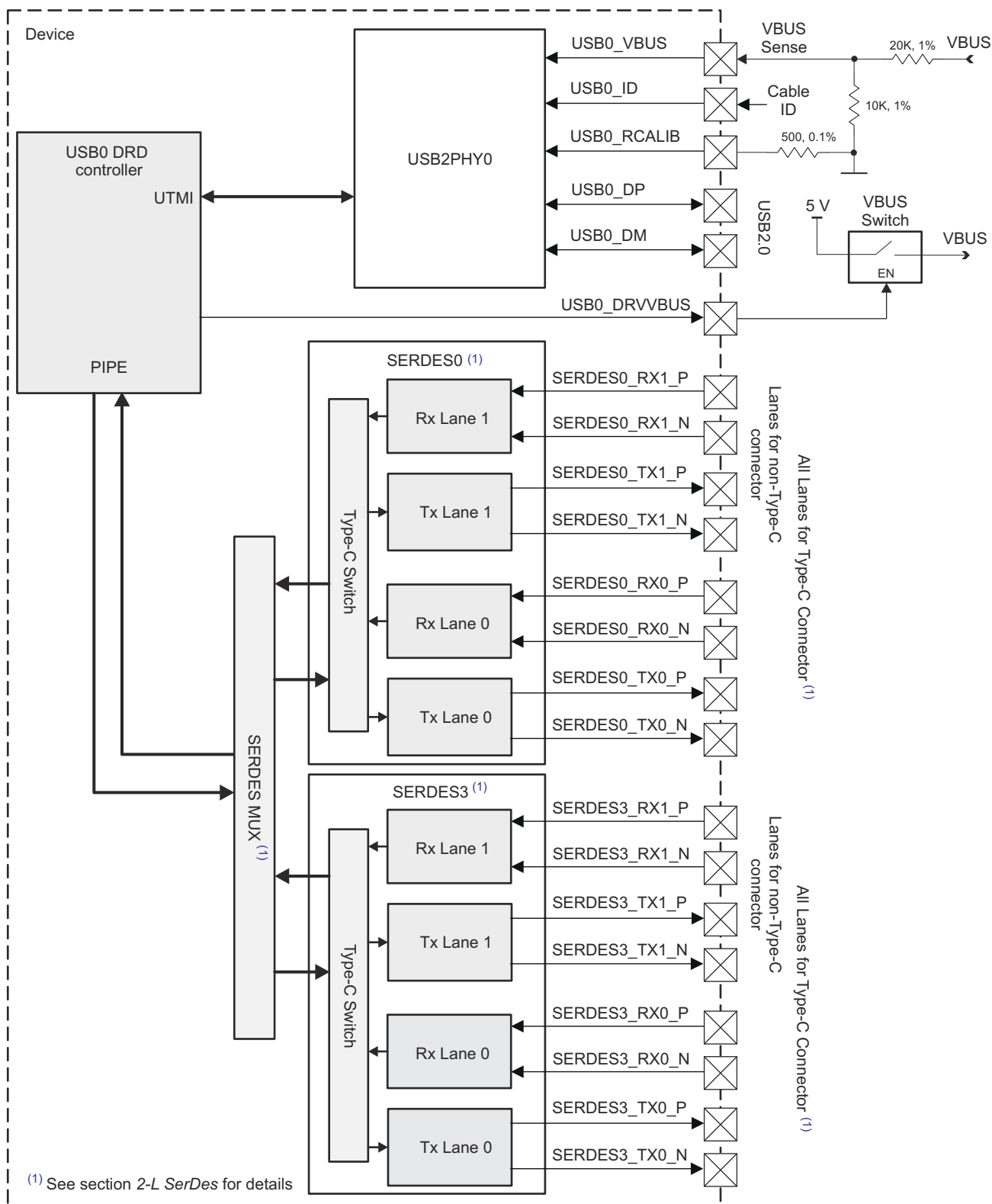
The following acronyms and abbreviations are related to USB.

Term	Definition
<b>ADP</b>	Attach Detection Protocol: detects USB OTG attach/detach events.
<b>DRD</b>	Dual Role Device: USB Host and Peripheral capable.
<b>DS</b>	Down Stream (A USB port facing from a Host or Hub to a Device/Peripheral).
<b>EP</b>	Endpoint: USB communication channel between the USB link partners carrying a single transfer type (BULK, ISOCH, INT, or CONTROL) and bus sharing arbitration scheme.
<b>FS</b>	Full-Speed USB data rate (12 Mbps).
<b>HNP</b>	Host Negotiation Protocol, OTG extension to swap USB host and peripheral roles.
<b>HS</b>	High-Speed USB data rate (480 Mbps).
<b>ITP</b>	Isochronous Timestamp Packet: USB SS micro-frame boundary packets.
<b>LMP</b>	Link Management Packet.
<b>LS</b>	Low-Speed USB data rate (1.5 Mbps).

Term	Definition
<b>OTG</b>	On-The-Go extension to USB protocol.
<b>PHY</b>	Physical Layer Device.
<b>SS</b>	Super-Speed USB data rate. 5 Gbps (USB3.0 or USB3.1 Gen1)
<b>US</b>	Upstream facing from a device (or hub) to the host (or a hub).
<b>USB IF</b>	USB Implementers Forum. The governing organization that develops and maintains the USB specifications and compliance standard. <a href="http://www.usb.org">http://www.usb.org</a>
<b>xHC</b>	Host Controller, designates the USB hardware that implements the xHCI specification.
<b>xHCI</b>	eXtensible Host Controller Interface. The specification that defines the register level interface for host controller.

## 12.2.4.2 USB Environment

Figure 12-199 shows the I/O interface signals of the USB3SS0 subsystem.

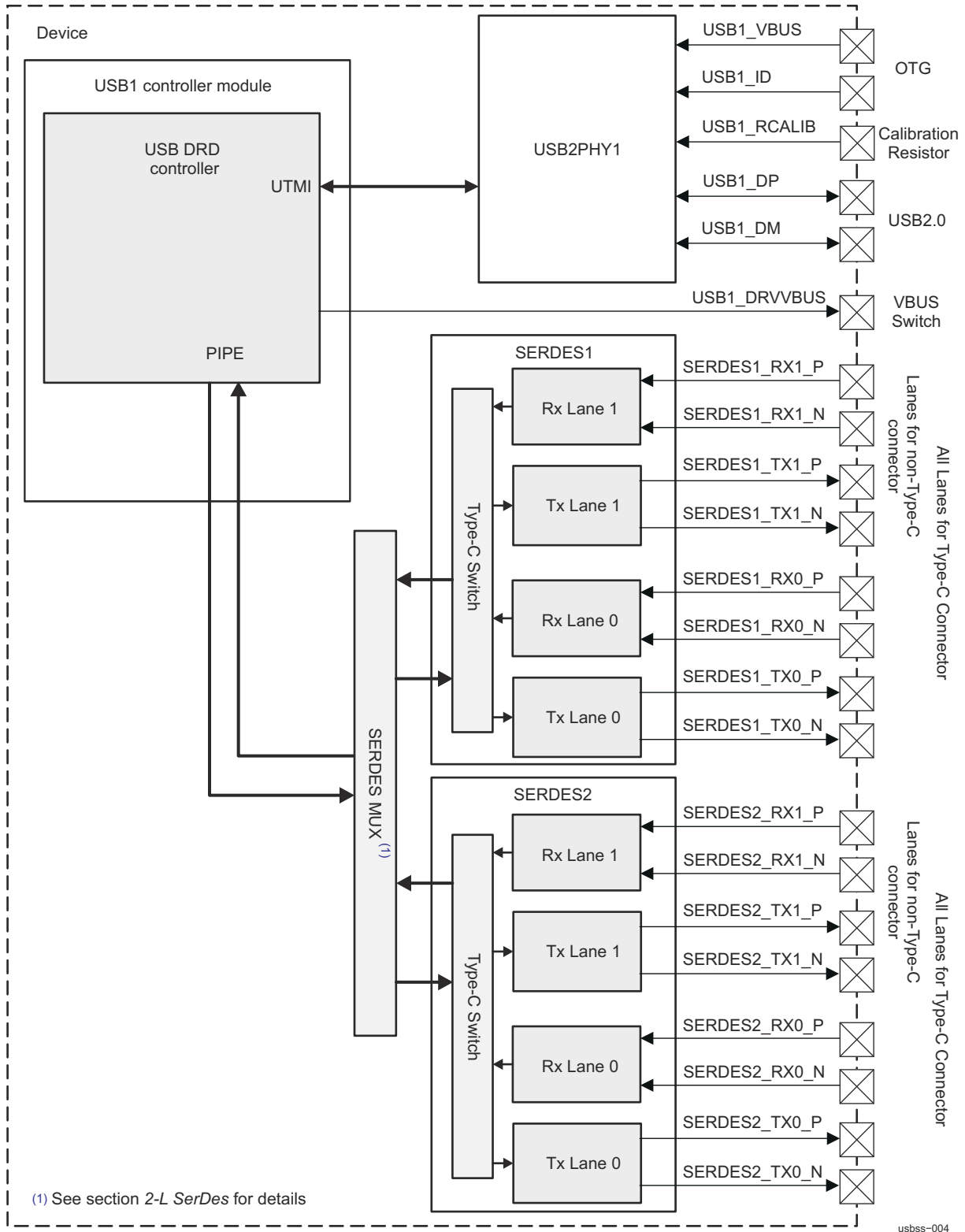


usbss-003

Figure 12-199. USB3SS0 Subsystem Environment



Figure 12-200 shows the I/O interface signals of the USB3SS1 subsystem.



**Figure 12-200. USB3SS1 Subsystem Environment**

Table 12-251 describes the external signals of the USB3SS0 subsystem.

**Table 12-251. USB3SS0 Input/Output Description**

Device Pin	Module or PHY Signal	I/O <sup>(1)</sup>	Description	Value at Reset
USB0_DP	DP	I/O	USB2.0 data pins. These are HS/FS/LS bidirectional differential data lane (D+/D-)	HiZ
USB0_DM	DM	I/O		HiZ
SERDES0_RX1_P (SERDES0_RX0_P) or <sup>(2)</sup> SERDES3_RX1_P (SERDES3_RX0_P)	RX_P_LN1 <sup>(3)</sup> (RX_P_LN0) <sup>(4)</sup>	I	USB3.0 differential data receive lane (RX+ pin)	HiZ
SERDES0_RX1_N (SERDES0_RX0_N) or <sup>(2)</sup> SERDES3_RX1_N (SERDES3_RX0_N)	RX_N_LN1 <sup>(3)</sup> (RX_N_LN0) <sup>(4)</sup>	I	USB3.0 differential data receive lane (RX- pin)	HiZ
SERDES0_TX1_P (SERDES0_TX0_P) or <sup>(2)</sup> SERDES3_TX1_P (SERDES3_TX0_P)	TX_P_LN1 <sup>(3)</sup> (TX_P_LN0) <sup>(4)</sup>	O	USB3.0 differential data transmit lane (TX+ pin)	HiZ
SERDES0_TX1_N (SERDES0_TX0_N) or <sup>(2)</sup> SERDES3_TX1_N (SERDES3_TX0_N)	TX_N_LN1 <sup>(3)</sup> (TX_N_LN0) <sup>(4)</sup>	O	USB3.0 differential data transmit lane (TX- pin)	HiZ
SERDES0_REXT or <sup>(2)</sup> SERDES3_REXT	CMN_REXT	A/I	USB3.0 SerDes external calibration resistor. Requires a 3.01 kOhm $\pm$ 1% accurate off-chip resistor connected from this pin to ground.	HiZ
USB0_ID	ID	A/I	USB cable identifier (A/B-device). Analog ID pin sense with internal pull-up	HiZ
USB0_VBUS	VBUS	A/I	An 3.3-V analog input for monitoring the voltage on VBUS (VBUS sense). 5-V VBUS must be applied via an external resistive divider /3. For example, R1 = 20 kOhm and R2 = 10 kOhm	HiZ
USB0_DRVVBUS	DRVVBUS	O	A digital output signal for VBUS Power Supply Enabling. Used to enable an external charge pump or power switch to supply +5V power to the VBUS port, when appropriate	0
USB0_RCALIB	RTRIM	A/I	External resistor for USB2.0 PHY calibration. Requires a 500 Ohm $\pm$ 1% off-chip resistor connected from this pin to ground	HiZ

(1) I = Input; O = Output; A = Analog

(2) USB3SS0 Superspeed can go to either SERDES0 or SERDES3. Refer to 2-L *Serializer/Deserializer (SerDes)* for more details.

(3) Lane 1 is active always, regardless if non-Type C or Type C connector is being used.

(4) Lane 0 is active only if Type C connector is being used.

Table 12-252 describes the external signals of the USB3SS1 subsystem.

**Table 12-252. USB3SS1 Input/Output Description**

Device Pin	Module or PHY Signal	I/O <sup>(1)</sup>	Description	Value at Reset
USB1_DP	DP	I/O	USB2.0 data pins. These are HS/FS/LS bidirectional differential data lane (D+/D-)	HiZ
USB1_DM	DM	I/O		HiZ
SERDES1_RX1_P (SERDES1_RX0_P) or <sup>(2)</sup> SERDES2_RX1_P (SERDES2_RX0_P)	RX_P_LN1 <sup>(3)</sup> (RX_P_LN0) <sup>(4)</sup>	I	USB3.0 differential data receive lane (RX+ pin)	HiZ

**Table 12-252. USB3SS1 Input/Output Description (continued)**

Device Pin	Module or PHY Signal	I/O <sup>(1)</sup>	Description	Value at Reset
SERDES1_RX1_N (SERDES1_RX0_N) or <sup>(2)</sup> SERDES2_RX1_N (SERDES2_RX0_N)	RX_N_LN1 <sup>(3)</sup> (RX_N_LN0) <sup>(4)</sup>	I	USB3.0 differential data receive lane (RX- pin)	HiZ
SERDES1_TX1_P (SERDES1_TX0_P) or <sup>(2)</sup> SERDES2_TX1_P (SERDES2_TX0_P)	TX_P_LN1 <sup>(3)</sup> (TX_P_LN0) <sup>(4)</sup>	O	USB3.0 differential data transmit lane (TX+ pin)	HiZ
SERDES1_TX1_N (SERDES1_TX0_N) or <sup>(2)</sup> SERDES2_TX1_N (SERDES2_TX0_N)	TX_N_LN1 <sup>(3)</sup> (TX_N_LN0) <sup>(4)</sup>	O	USB3.0 differential data transmit lane (TX- pin)	HiZ
SERDES1_REXT or <sup>(2)</sup> SERDES2_REXT	CMN_REXT	A/I	USB3.0 SerDes external calibration resistor. Requires a 3.01 kOhm $\pm 1\%$ accurate off-chip resistor connected from this pin to ground.	HiZ
USB1_ID	ID	A/I	USB cable identifier (host/device). Analog ID pin sense with internal pull-up	HiZ
USB1_VBUS	VBUS	A/I	An 3.3-V analog input for monitoring the voltage on VBUS (VBUS sense). 5-V VBUS must be applied via an external resistive divider /3. For example, R1 = 20 kOhm and R2 = 10 kOhm	HiZ
USB1_DRVVBUS	DRVVBUS	O	A digital output signal for VBUS Power Supply Enabling. Used to enable an external charge pump or power switch to supply +5V power to the VBUS port, when appropriate	0
USB1_RCALIB	RTRIM	A/I	External resistor for USB2.0 PHY calibration. Requires a 500 Ohm $\pm 1\%$ off-chip resistor connected from this pin to ground	HiZ

(1) I = Input; O = Output; A = Analog

(2) USB3SS1 SuperSpeed can go to either SERDES1 or SERDES2. Refer to 2-L *Serializer/Deserializer (SerDes)* for more details.

(3) Lane 1 is active always, regardless if non-Type C or Type C connector is being used.

(4) Lane 0 is active only if Type C connector is being used.

### 12.2.4.3 USB Integration

This section describes USB module integration in the device, including information about clocks, resets, and hardware requests.

[Figure 12-201](#) and [Figure 12-202](#) show the integration of the USB subsystem in the device.

For Superspeed SerDes integration details, please refer to in *Serializer/Deserializer (SerDes)*.

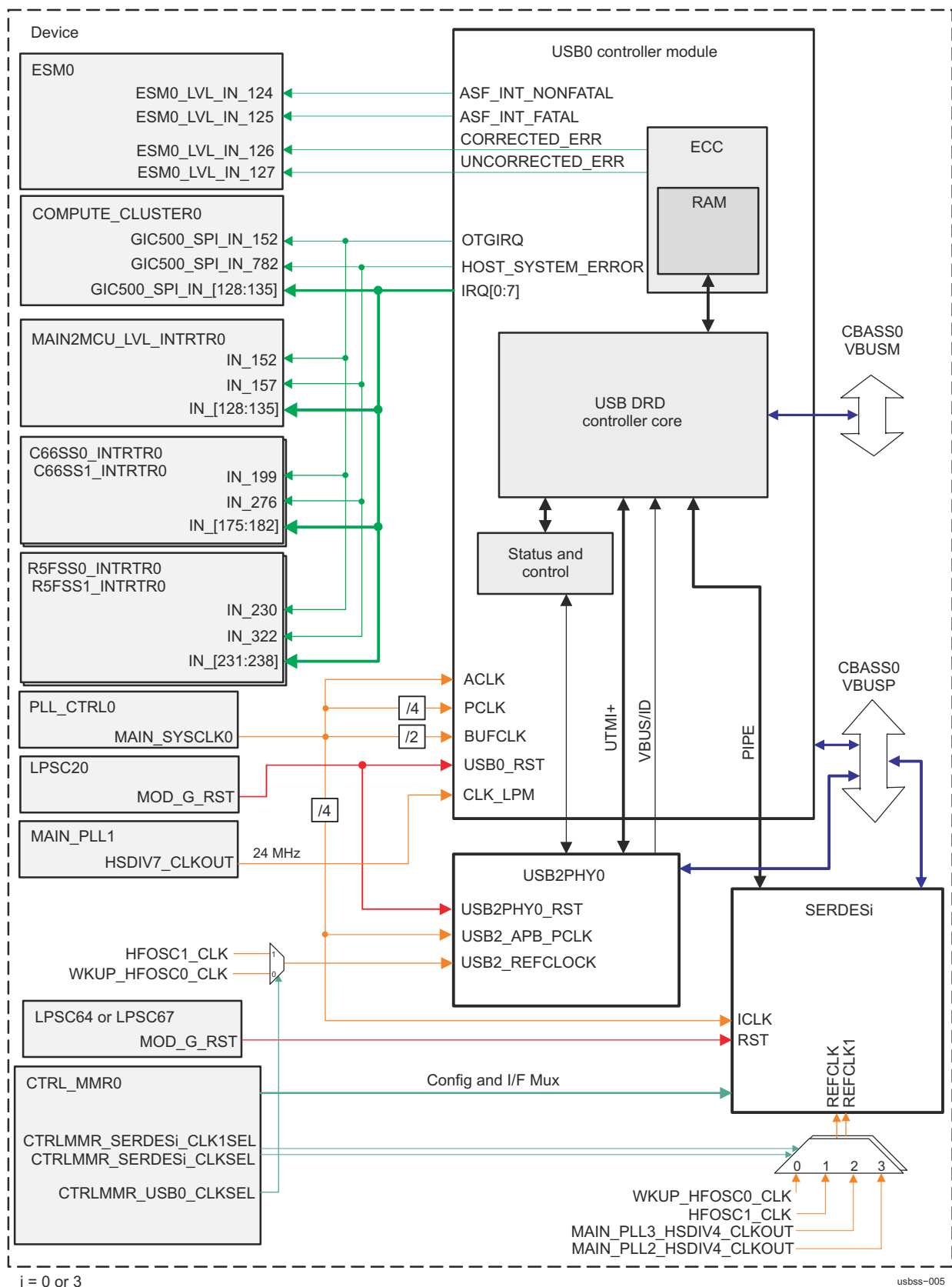


Figure 12-201. USB3SS0 Integration

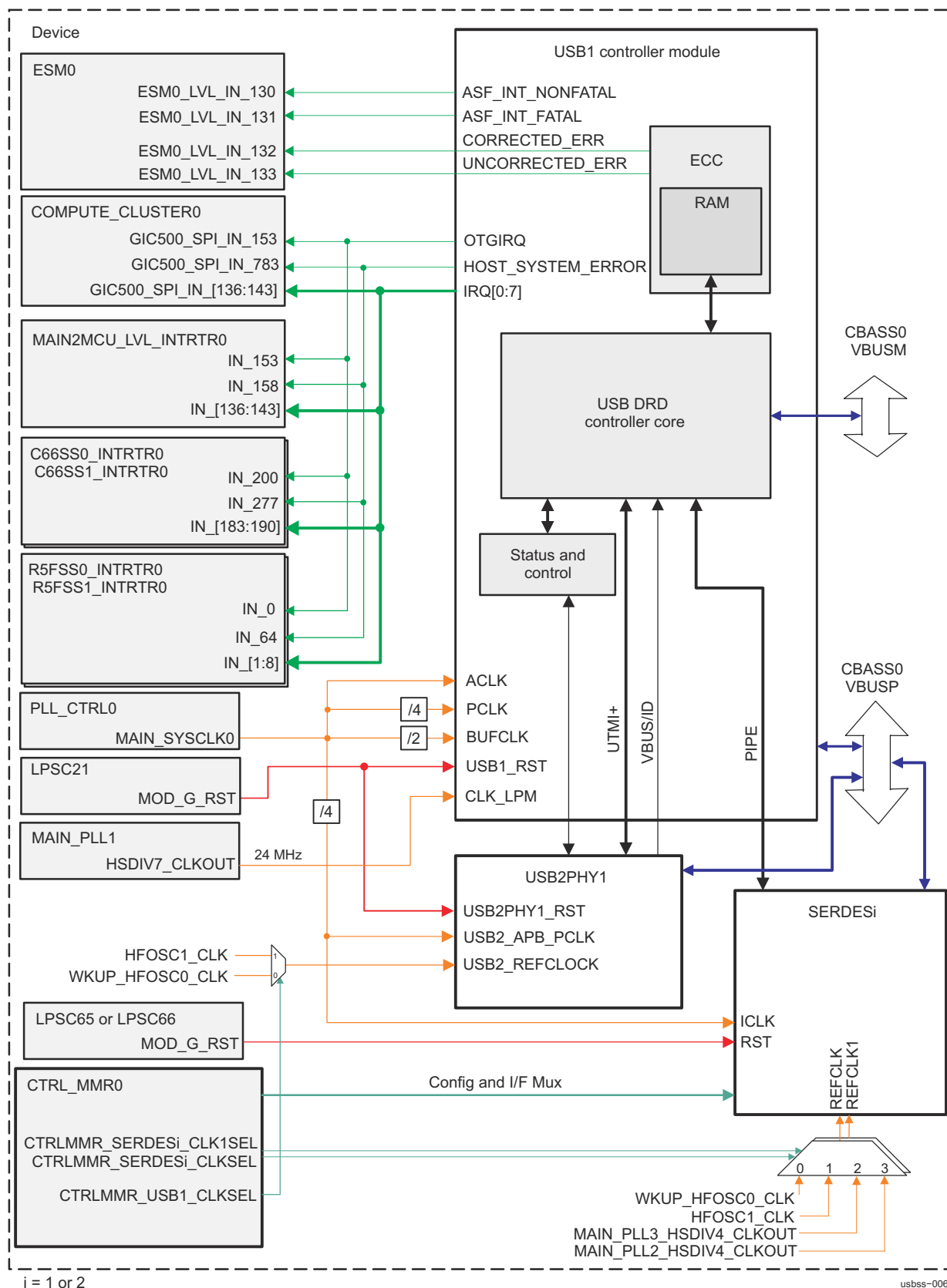


Figure 12-202. USB3SS1 Integration

Table 12-253 through Table 12-255 summarize the integration of USBSS.

**Table 12-253. USB Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
USB3SS0	PSC0	PD0	LPSC20	CBASS0
USB3SS1	PSC0	PD0	LPSC21	CBASS0

**Table 12-254. USB Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
USB3SS0	ACLK	MAIN_SYSCLK0	PLL_CTRL0	VBUSM interface clock
	PCLK	MAIN_SYSCLK0/4	PLL_CTRL0	VBUSP interface clock
	BUFCLK	MAIN_SYSCLK0/2	PLL_CTRL0	VBUSM bridge clock
	CLK_LPM	MAIN_PLL1_HSDIV7_C LKOUT	MAIN_PLL1	Low power clock (24 MHz). This clock has to be free running when USB is enabled.
USB2PHY0	USB2_REFCLOCK	WKUP_HFOSC0_CLK	WKUP_HFOSC0	USB2PHY reference clock. HFOSC selected via CTRLMMR_USB0_CLKSEL. Frequency value must be indicated to PHY via USB3P0SS_STATIC_CONFIG.
		HFOSC1_CLK	HFOSC1	
USB3SS1	USB2_APB_PCLK	MAIN_SYSCLK0/4	PLL_CTRL0	USB2PHY VBUSP interface clock
	ACLK	MAIN_SYSCLK0	PLL_CTRL0	VBUSM interface clock
	PCLK	MAIN_SYSCLK0/4	PLL_CTRL0	VBUSP interface clock
	BUFCLK	MAIN_SYSCLK0/2	PLL_CTRL0	VBUSM bridge clock
	CLK_LPM	MAIN_PLL1_HSDIV7_C LKOUT	MAIN_PLL1	Low power clock (24 MHz). This clock has to be free running when USB is enabled.
USB2PHY1	USB2_REFCLOCK	WKUP_HFOSC0_CLK	WKUP_HFOSC0	USB2PHY reference clock. HFOSC selected via CTRLMMR_USB1_CLKSEL. Frequency value must be indicated to PHY via USB3P0SS_STATIC_CONFIG.
		HFOSC1_CLK	HFOSC1	
	USB2_APB_PCLK	MAIN_SYSCLK0/4	PLL_CTRL0	USB2PHY VBUSP interface clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
USB3SS0	USB0_RST	MOD_G_RST	LPSC20	USB3SS0 hardware reset
USB3SS1	USB1_RST	MOD_G_RST	LPSC21	USB3SS1 hardware reset

**Table 12-255. USB Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt	Destination Interrupt Input	Destination	Description	Type
USB3SS0	IRQ_[0:7]	GIC500_SPI_IN_[128:135]	COMPUTE_CLUSTER0	8 event ring interrupts in host mode. In device mode, IRQ[6] is device USB interrupt and IRQ[7] is device wakeup request.	Level
		MAIN2MCU_LVL_INTRTR0_IN_[128:135]	MAIN2MCU_LVL_INTRTR0		
		C66SS0_INTRTR0_IN_[175:182]	C66SS0_INTRTR0		

**Table 12-255. USB Hardware Requests (continued)**

		C66SS1_INTRTR0_IN_[175:182]	C66SS1_INTRTR0		
		R5FSS0_INTRTR0_IN_[231:238]	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_[231:238]	R5FSS1_INTRTR0		
HOST_SYSTEM_ERROR		GIC500_SPI_IN_782	COMPUTE_CLUSTER0	USB host system error	Level
		MAIN2MCU_LVL_INTRTR0_IN_157	MAIN2MCU_LVL_INTRTR0		
		C66SS0_INTRTR0_IN_276	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_276	C66SS1_INTRTR0		
		R5FSS0_INTRTR0_IN_322	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_322	R5FSS1_INTRTR0		
OTGIRQ		GIC500_SPI_IN_152	COMPUTE_CLUSTER0	USB OTG events	Level
		MAIN2MCU_LVL_INTRTR0_IN_152	MAIN2MCU_LVL_INTRTR0		
		C66SS0_INTRTR0_IN_199	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_199	C66SS1_INTRTR0		
		R5FSS0_INTRTR0_IN_230	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_230	R5FSS1_INTRTR0		
ASF_INT_NONFATAL		ESM0_LVL_IN_124	ESM0	USB safety features nonfatal interrupt	Level
ASF_INT_FATAL		ESM0_LVL_IN_125	ESM0	USB safety features fatal interrupt	Level
A_ECC_AGGR_CORRECTED_ERROR_LEVEL		ESM0_LVL_IN_126	ESM0	ECC aggregator interrupt	Level
A_ECC_AGGR_UNCORRECTED_ERROR_LEVEL		ESM0_LVL_IN_127	ESM0	ECC aggregator interrupt	Level
USB3SS1	IRQ_[0:7]	GIC500_SPI_IN_[136:143]	COMPUTE_CLUSTER0	8 event ring interrupts in host mode. In device mode, IRQ[6] is device USB interrupt and IRQ[7] is device wakeup interrupt.	Level
		MAIN2MCU_LVL_INTRTR0_IN_[136:143]	MAIN2MCU_LVL_INTRTR0		
		C66SS0_INTRTR0_IN_[183:190]	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_[183:190]	C66SS1_INTRTR0		
		R5FSS0_INTRTR0_IN_[1:8]	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_[1:8]	R5FSS1_INTRTR0		
HOST_SYSTEM_ERROR		GIC500_SPI_IN_783	COMPUTE_CLUSTER0	USB host system error	Level
		MAIN2MCU_LVL_INTRTR0_IN_158	MAIN2MCU_LVL_INTRTR0		
		C66SS0_INTRTR0_IN_277	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_277	C66SS1_INTRTR0		
		R5FSS0_INTRTR0_IN_64	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_64	R5FSS1_INTRTR0		
OTGIRQ		GIC500_SPI_IN_153	COMPUTE_CLUSTER0	USB OTG events	Level
		MAIN2MCU_LVL_INTRTR0_IN_153	MAIN2MCU_LVL_INTRTR0		
		C66SS0_INTRTR0_IN_200	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_200	C66SS1_INTRTR0		



**Table 12-255. USB Hardware Requests (continued)**

	R5FSS0_INTRTR0_IN_0	R5FSS0_INTRTR0		
	R5FSS1_INTRTR0_IN_0	R5FSS1_INTRTR0		
ASF_INT_NONFATAL	ESM0_LVL_IN_130	ESM0	USB safety features nonfatal interrupt	Level
ASF_INT_FATAL	ESM0_LVL_IN_131	ESM0	USB safety features fatal interrupt	Level
A_ECC_AGGR_CORRECTED_ERR_LEVEL	ESM0_LVL_IN_132	ESM0	ECC aggregator interrupt	Level
A_ECC_AGGR_UNCORRECTED_ERR_LEVEL	ESM0_LVL_IN_133	ESM0	ECC aggregator interrupt	Level

DMA Events					
Module Instance	Module DMA Event	Destination DMA Event Input	Destination	Description	Type
USB3SS0 and USB3SS1	-	-	-	No PDMA channels to external DMA engines. USB has an internal DMA controller.	-

## 12.2.4.4 USB Functional Description

### Note

This chapter serves to describe the integration of the third-party USB controller and should not be considered sufficient for those wishing to modify the existing Linux or RTOS USB driver(s) or create a new driver to support this controller implementation. For those who do wish to substantially modify the existing Linux or RTOS USB driver(s), or create new drivers, contact TI for more information on how to obtain the third-party documentation under NDA.

The USB3SS subsystem contains the USB3.0 Dual Role Device (DRD) controller module and a USB2.0 PHY module. A wrapper module is controlling some top-level functions like Host and Device role switch and reset release.

The USB controller uses USB2PHY for USB2.0 operation and one of the device SERDESes for USB3.0 speeds. The SERDES also contains lane swap feature for USB Type-C plug flipping support.

### 12.2.4.4.1 USB Type-C Connector Support

Lane swapping for Type-C connector support is performed within the SERDES wrapper module. Type-C CC detection and configuration has to be performed by external ICs. Type-C device attachment/detachment and cable orientation have to be communicated using I2C, GPIO, or a similar method.

For USB Type-C support, the SERDES wrapper and SERDES must be programmed for USB protocol on both lane 0 and lane 1. Software must hold PHY in reset and then write to the LN10\_SWAP bit to mux the PIPE interface to the other lane. Also the SuperSpeed port in the controller has to be in disabled state when programming LN10\_SWAP.

### 12.2.4.4.2 USB Controller Reset

USB controller has three resets going in, called preset\_n, pwrup\_rst\_n, and aresetn.

- preset\_n comes from the mod\_g\_rst\_n input coming from LPSC
- pwrup\_rst\_n is PWRUP\_RST\_N register bit in the USB wrapper module (USB3P0SS\_W1). This reset has to be deasserted before accessing controller registers. The default value for pwrup\_rst\_n is 0 (reset asserted). This reset goes to the default asserted state when mod\_g\_rst\_n reset is asserted.
- areset\_n comes from PWRUP\_RST\_N register bit, but the deassertion is synchronized to ACLK.

### 12.2.4.4.3 Overcurrent Detection

The overcurrent detection circuit is external to the device. Software must specify an overcurrent condition to the controller whenever the overcurrent signal is received from the detection circuit. Software must set the OVERCURRENT\_N bit to 0 in the USB3P0SS\_W1 register when overcurrent was detected.

### 12.2.4.4.4 Top-Level Initialization Sequence

The following initialization sequence has to be followed before configuring the controller for USB operation:

1. Software must select and configure the SERDES for USB operation. Some of the registers are shown in [Table 12-256](#). Refer to *Serializer/Deserializer (SerDes)* for more details. USB2.0 PHY contains default configuration that normally does not need change.

**Table 12-256. USB3.0 Register Configuration**

Bitfields not listed should be left at default value

Register	Bit Field/ Programming Model	Value
RX_CREQ_FLTR_B_PREG__RX_CREQ_FLTR_A_MODE0_PREG_j	[7:6] CREQ_CRFLTR_GAIN1_MODE_0_PREG	2h
	[5:3] CREQ_CRFLTR_GAIN2_MODE0_PREG	3h
	[2:1] CREQ_CRFLTR_ACCUMSAT2_MODE0_PREG	1h
DEQ_PHALIGN_CTRL_j	[1:0] DEQ_PHALIGN_SAMPSIZE_PREG	3h
SDFILT_H2L_A_PREG	SIGDET_SUPPORT_PREG_j[15:0]	6013h (Default)

- Software must configure the pseudo-static settings in USB3P0SS\_W1 register
- Software must set PWRUP\_RST\_N to 1 in order to deassert controller reset
- Software must wait for controller to be ready in the respective mode selected by reading the respective status bits.

After the above sequence, software can access controller registers.

## 12.2.5 2-L Serializer/Deserializer (SerDes)

This section describes the 2-Lane Serializer/Deserializer (SerDeses) in the device.

### 12.2.5.1 2-L SerDes Overview

SerDes'es goal is to convert device (SoC) parallel data into serialized data that can be output over a high-speed electrical interface. In the opposite direction, SerDes converts high-speed serial data into parallel data that can be processed by the device. To this end, the SerDes contains a variety of functional blocks to handle both the external analog interface as well as the internal digital logic.

Most important building blocks of SerDes are:

- Physical Media Attachment (PMA):
  - Lanes: The lanes handle all inputs and outputs from the serial interface, and contain the Tx/Rx I/Os, serializer/deserializer, and Clock and Data Recovery (CDR) unit.
  - Common module (CMN): The CMN handles peripheral and Tx clocking of the SerDes. It consists of internal PLLs and external reference clock input buffer, reset, and startup circuitry.
- Physical Coding Sub-block (PCS): The PCS is responsible for translating data from/to the parallel interface, as well as data encoding/decoding and symbol alignment.
- WIZ: The WIZ acts as a wrapper for the SerDes, and can both send control signals to and report status signals from the SerDes, and muxes SerDes to peripherals.

The device contains four two-lane SerDeses: SERDES0 through SERDES3.

For four-lane SerDes (SERDES4), please see [Section 12.2.6, 4-L Serializer/Deserializer \(SerDes\)](#).

**Table 12-257. SerDes Allocation Within Device Domains**

Module Instance	Domain		
	WKUP	MCU	MAIN
SERDES0	-	-	✓
SERDES1	-	-	✓
SERDES2	-	-	✓
SERDES3	-	-	✓
SERDES4 (4-L)	-	-	✓

[Figure 12-203](#) through [Figure 12-206](#) show the SerDes highlights.

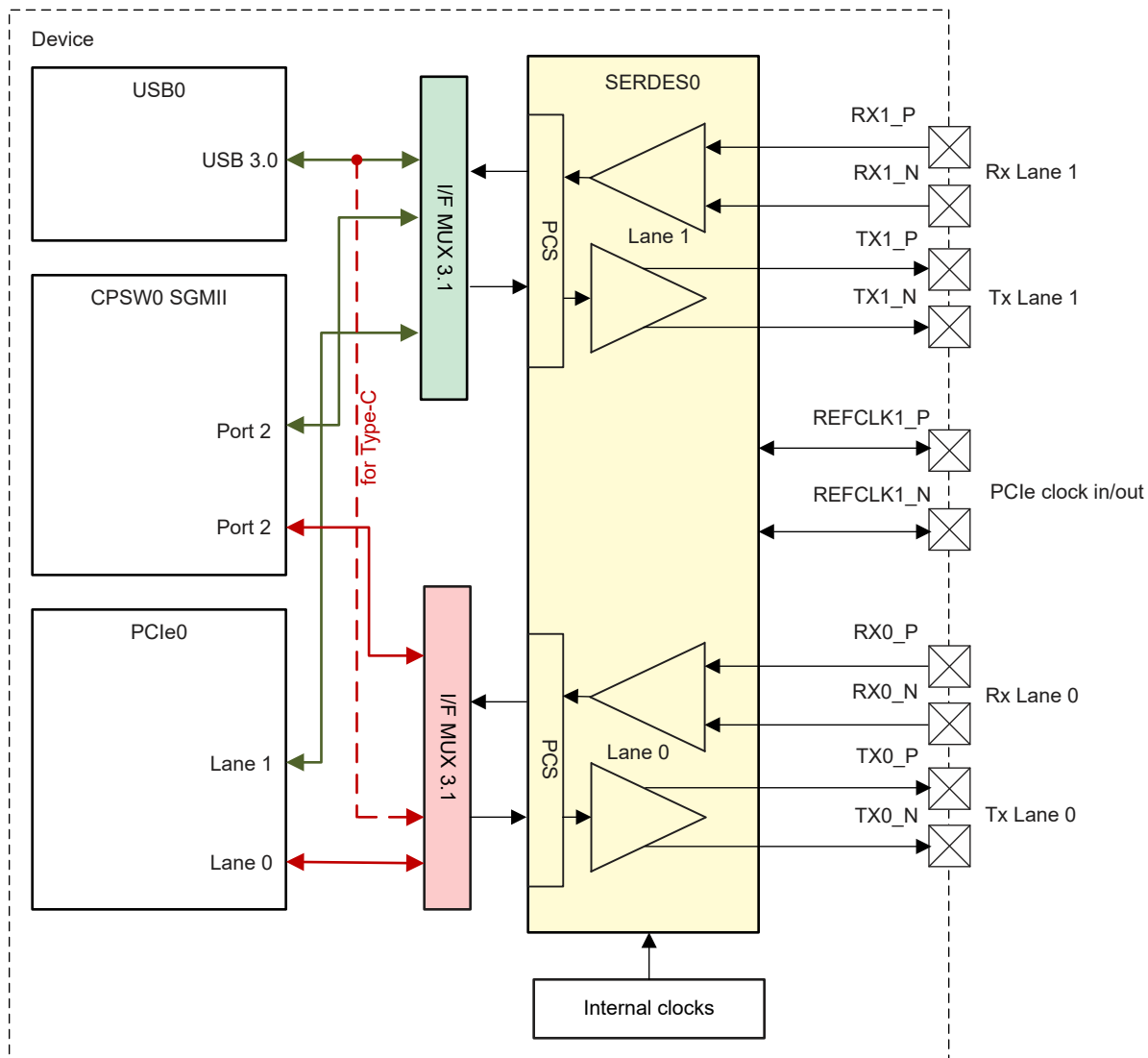


Figure 12-203. SERDES0 Overview

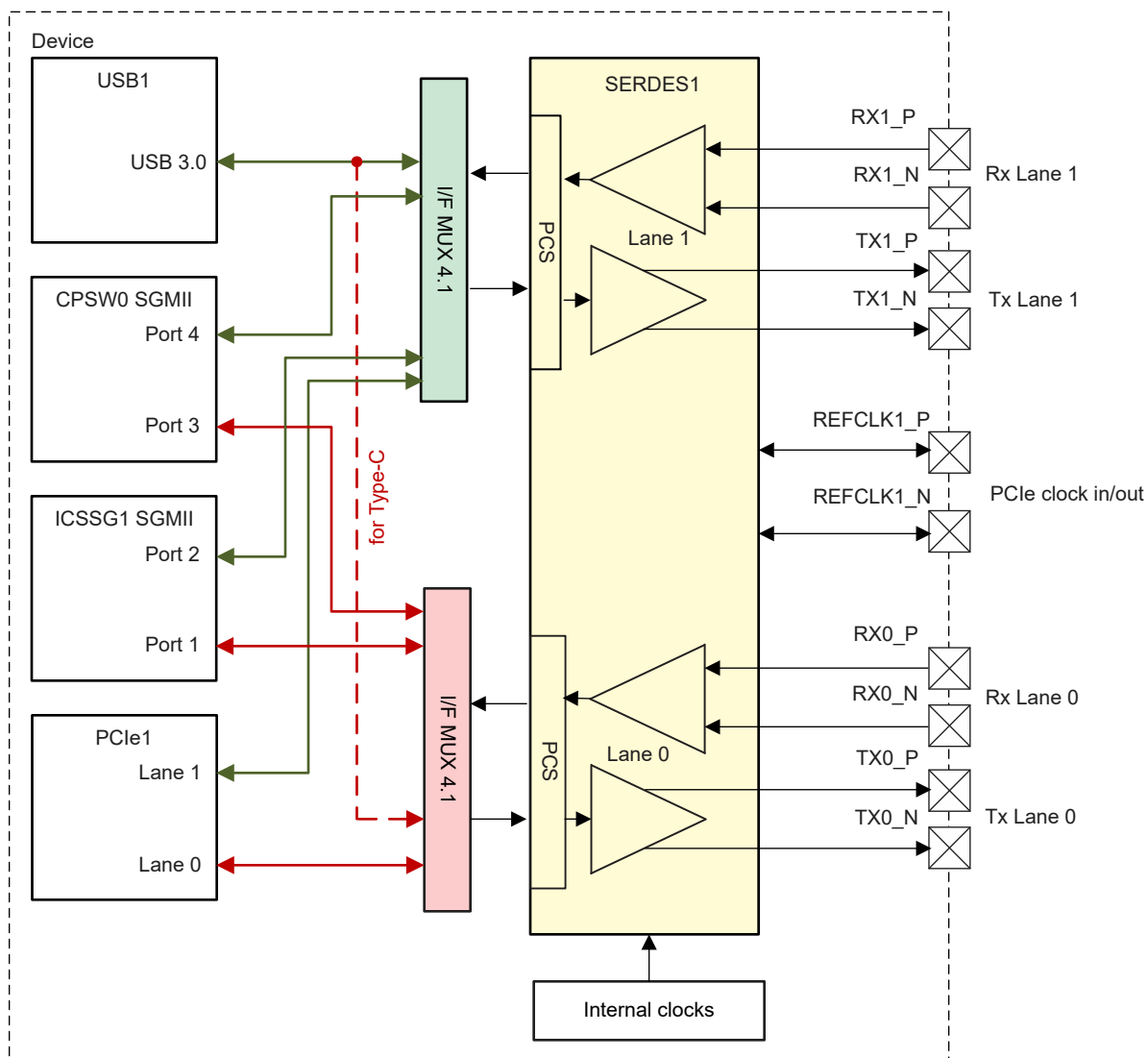
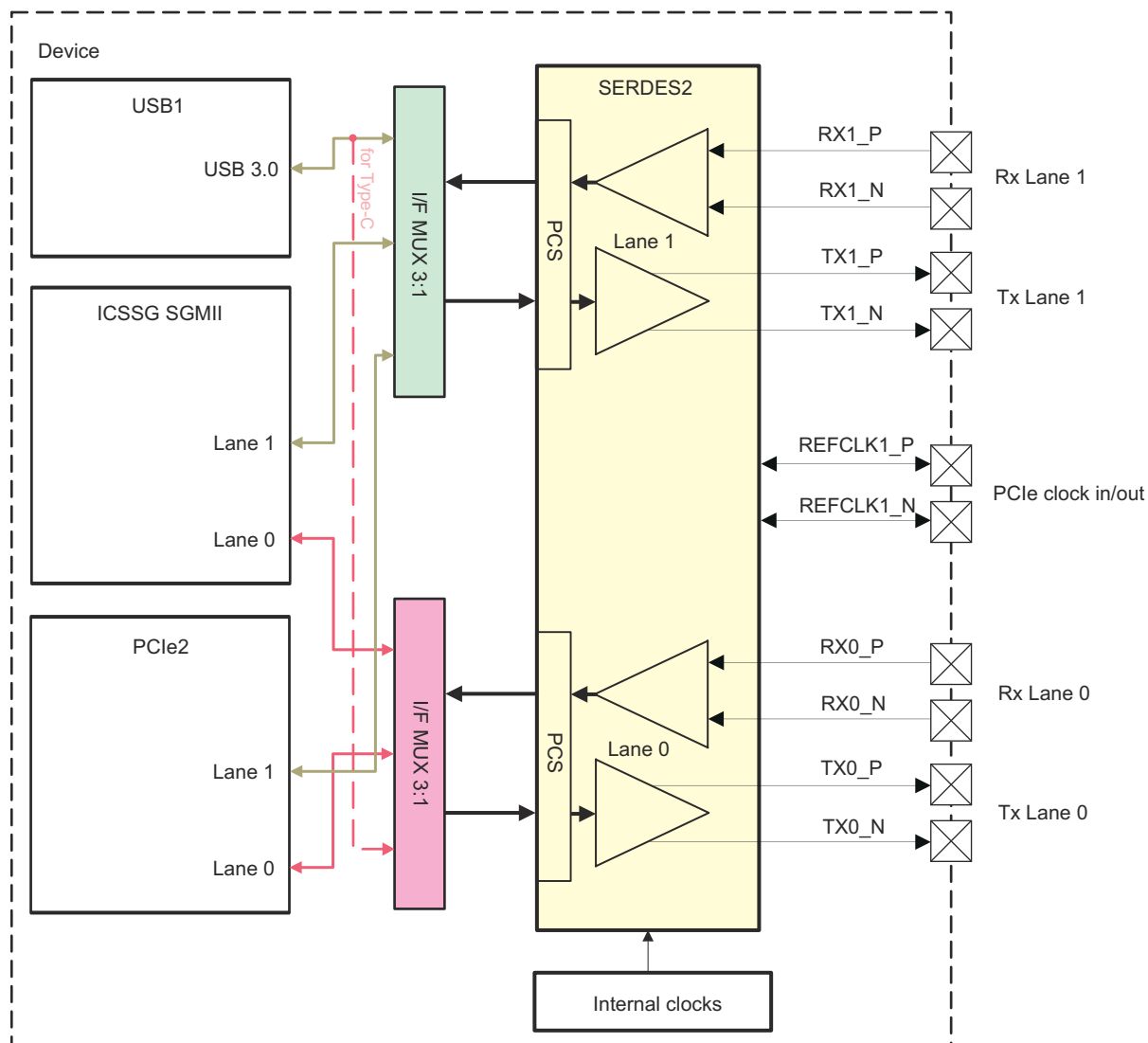
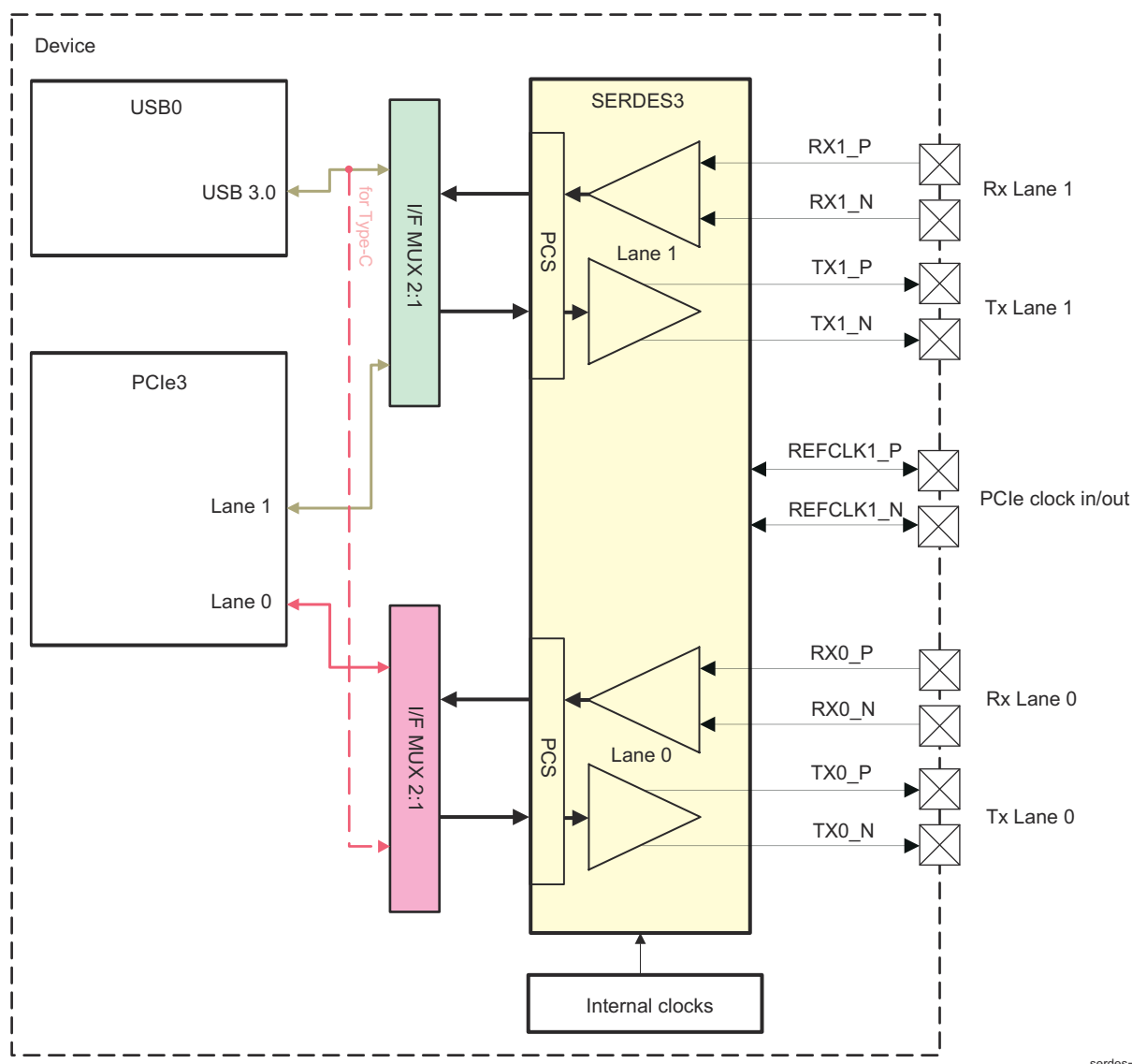


Figure 12-204. SERDES1 Overview



serdes-003

Figure 12-205. SERDES2 Overview



serdes-004

**Figure 12-206. SERDES3 Overview**

#### 12.2.5.1.1 2-L SerDes Features

The SERDES module features include:

- Dual lane PHY containing:
  - Transmit and Receive I/Os
  - Serializer
  - Deserializer
  - Clock and data recovery (CDR) unit
- Common Module (CMN)
  - PLLs
  - Master bias
  - Automatic calibration of pin termination resistors
  - Reference clock input buffers
  - Reset and startup management
- Physical Coding Sub-block (PCS)
  - PCIe per PCI Express Base Specification Revision 4.0, February 19, 2014



- ECNs included: L1 PM Substates CLKREQ; SRIS.
- USB3.1 per Universal Serial Bus 3.1 Specification, Revision 1.0, July 26, 2013
- QSGMII Specification revision 1.2
- Symbol alignment
- Selectable serial pin polarity reversal for both transmit and receive paths
- Bit stream reordering
- Physical Media Attachment (PMA) layer
  - Transmit equalization
  - Receive equalization
  - Data path BIST with programmable pattern generation and error detection
  - Serial bit stream and parallel word loopback for both line and parallel side
  - 8-bit ADC provides digitized ATB measurement results
  - Supports DC and AC JTAG (boundary scan) per IEEE 1149.6

The SERDES mux (WIZ) module supports the following features:

- Multiplexes device interfaces onto a single SERDES lane (one Tx and one Rx)
- Provides registers to implement SERDES control and status functions and alignment delays
- Clock generator block for providing MAC transmit clock
- Rx comma align block
  - Performs de-stuffing the Rx data stream in the event that the Rx rate is different from the Tx rate
  - Supports comma detection that is not sensitive to false commas using all 8B10B character combinations

#### 12.2.5.1.2 Industry Standards Compatibility

All SerDes interfaces are configured as point-to-point connections. It is assumed that the connection is made between the SoC and another device that is compliant to the appropriate industry standard. The list of supported standards is given below.

This chapter deals with the physical layer and, therefore, it is the electrical specifications in these standards that are relevant. For more information regarding protocol compliance <sup>2</sup>, see the device-specific Datasheet.

- SGMII: This is electrically compliant with SGMII revision 1.8 with the following clarifications:
  - It does not implement the separate clock signaling
  - Electrical compatibility does not guarantee interoperability with devices.
- QSGMII: This is electrically compliant with QSGMII revision 1.2 with the following clarifications:
  - It does not implement the separate clock signaling
  - Electrical compatibility does not guarantee interoperability with devices.
- PCIe: This is electrically compliant with version 4.0.
- USB: This is electrically compliant with USB 3.1.

**Table 12-258. SerDes Supported Standards**

Standard	Bit Rate (Gbps)	Reference Clock Frequency (MHz)	Bus Width (bits)
PCI Express 4.0 Gen1	2.5	19.2, 20, 24, 25, 26, 27, 31.25, 100	8
PCI Express 4.0 Gen2	5.0	19.2, 20, 24, 25, 26, 27, 31.25, 100	16
PCI Express 4.0 Gen3	8.0	19.2, 20, 24, 25, 26, 27, 31.25, 100	32
SGMII	1.25	19.2, 20, 24, 25, 26, 27, 62.5, 100	10
SGMII	2.5	19.2, 20, 24, 25, 26, 27, 62.5, 100	10
USB 3.1 Gen1 SuperSpeed	5.0	19.2, 20, 24, 25, 26, 27, 100	16

<sup>2</sup> Electrical compatibility does not guarantee interoperability with devices

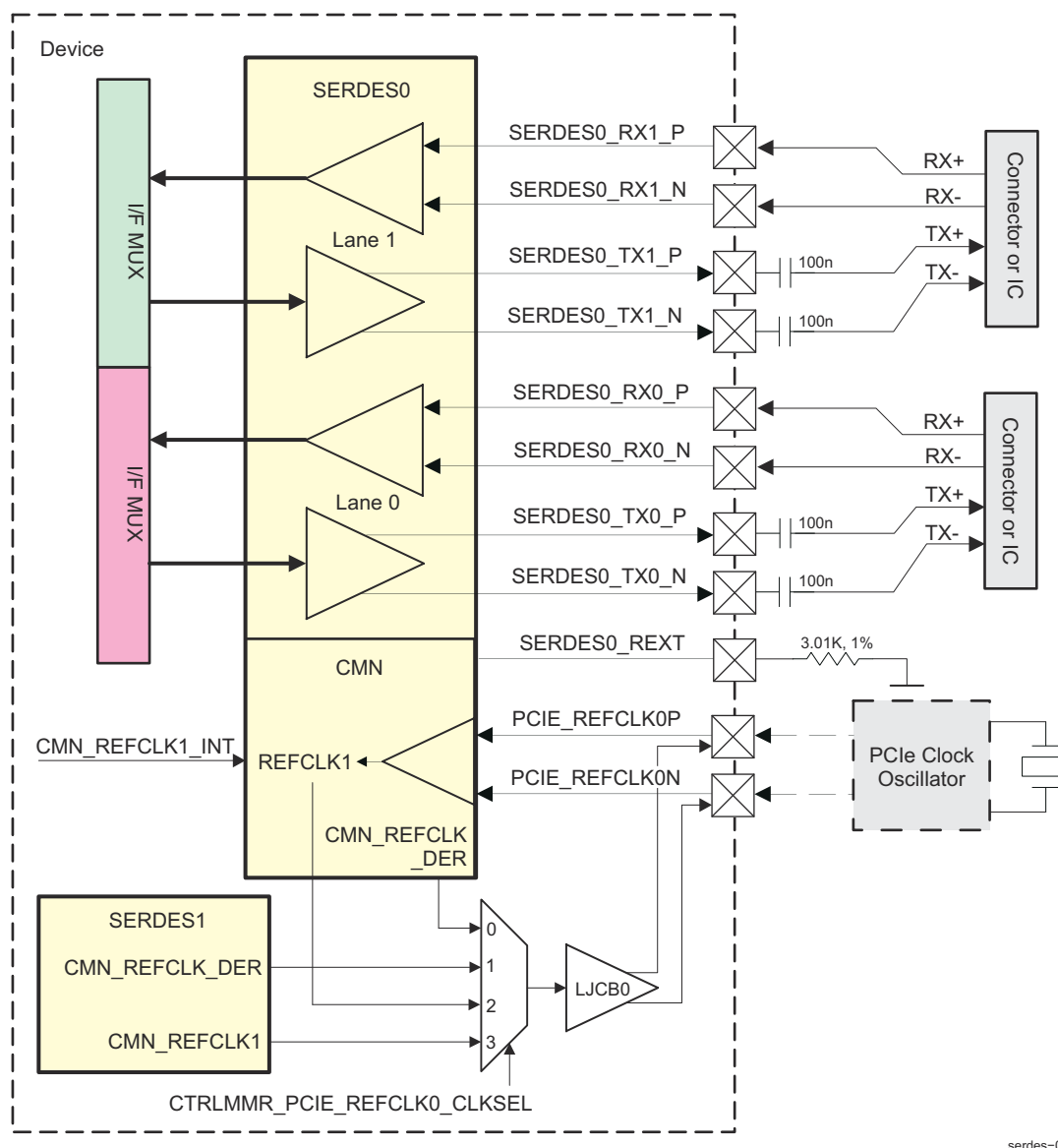
## 12.2.5.2 2-L SerDes Environment

### 12.2.5.2.1 2-L SerDes I/Os

#### Note

Although containing some of the basic external components, [Figure 12-207](#) to [Figure 12-210](#) must not be considered as an exhaustive guide for the PCB designer. TI provides additional documents for those who are willing to design PCBs and/or fine tune the SerDes.

[Figure 12-207](#) shows the I/O interface pins of SerDes.



serdes-006

**Figure 12-207. SERDES0 Environment**

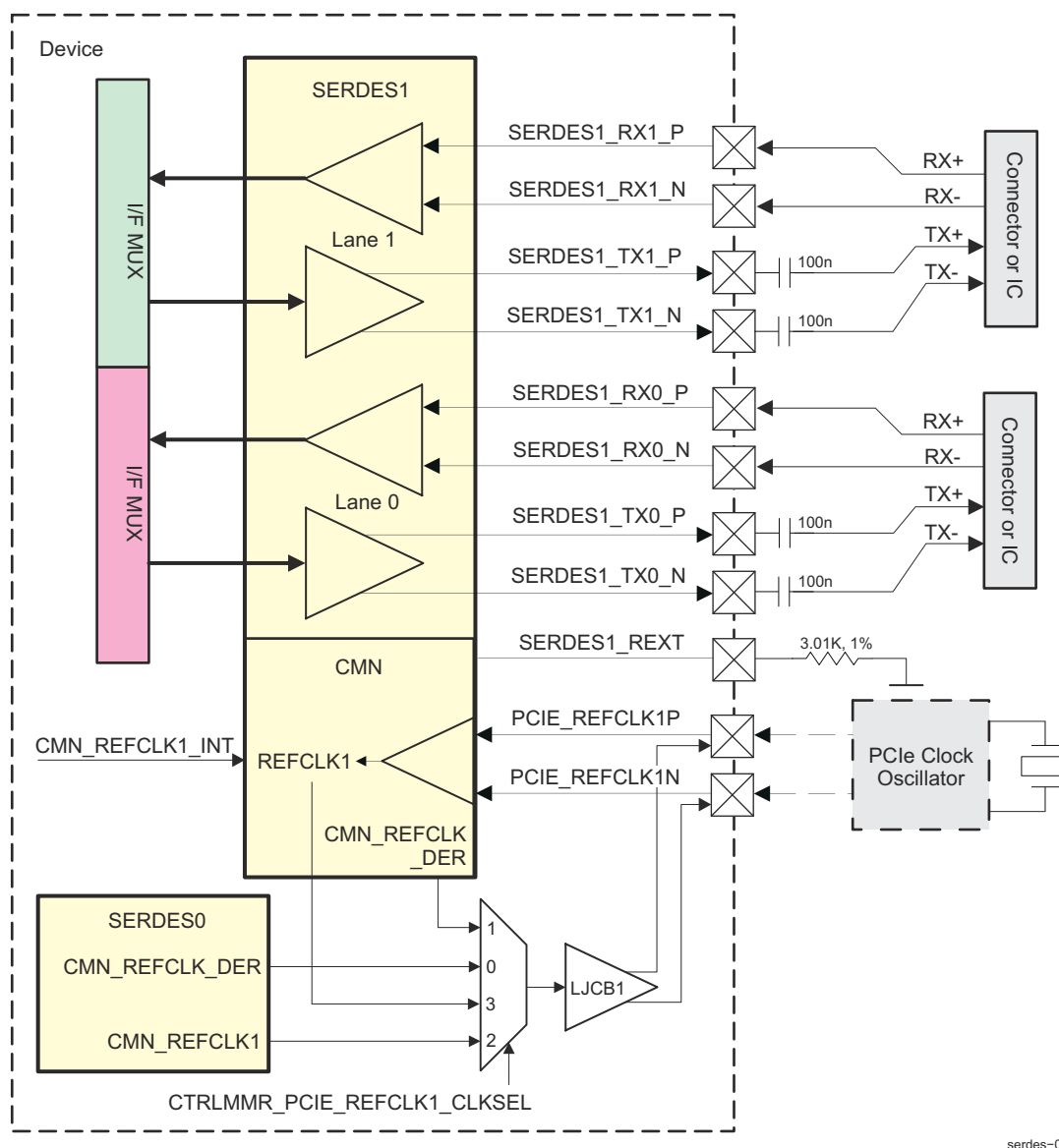
[Table 12-259](#) describes the external signals of SERDES0 module.

**Table 12-259. SERDES0 Input/Output Description**

Device Pin	Module Signal	I/O <sup>(1)</sup>	Description	Value at Reset
SERDES0_RX0_P	RX_P_LN0	I	SerDes differential data receive pins. Lane 0	HiZ
SERDES0_RX0_N	RX_M_LN0			
SERDES0_TX0_P	TX_P_LN0	O	SerDes differential data transmit pins. Lane 0	HiZ
SERDES0_TX0_N	TX_M_LN0			
SERDES0_RX1_P	RX_P_LN1	I	SerDes differential data receive pins. Lane 1	HiZ
SERDES0_RX1_N	RX_M_LN1			
SERDES0_TX1_P	TX_P_LN1	O	SerDes differential data transmit pins. Lane 1	HiZ
SERDES0_TX1_N	TX_M_LN1			
PCIE_REFCLK0P	CMN_REFCLK1_P	I	SerDes external system reference clock for PCIe	HiZ
PCIE_REFCLK0N	CMN_REFCLK1_M			
	SERDES0_CMN_REFCLK_DE R	O	SerDes system reference clock for external PCIe device. Selected via CTRLMMR_PCIE_REFCLK0_CLKSEL	
	SERDES1_CMN_REFCLK_DE R			
	SERDES0_CMN_REFCLK1			
	SERDES1_CMN_REFCLK1			
SERDES0_REXT	CMN_REXT	A/I	PMA external calibration resistor. Requires a 3.01 kOhm $\pm 1\%$ accurate off-chip resistor connected from this pin to ground.	HiZ

(1) I = Input; O = Output; A = Analog

Figure 12-208 shows the I/O interface pins of SerDes.



serdes-061

**Figure 12-208. SERDES1 Environment**

Table 12-260 describes the external signals of SERDES1 module.

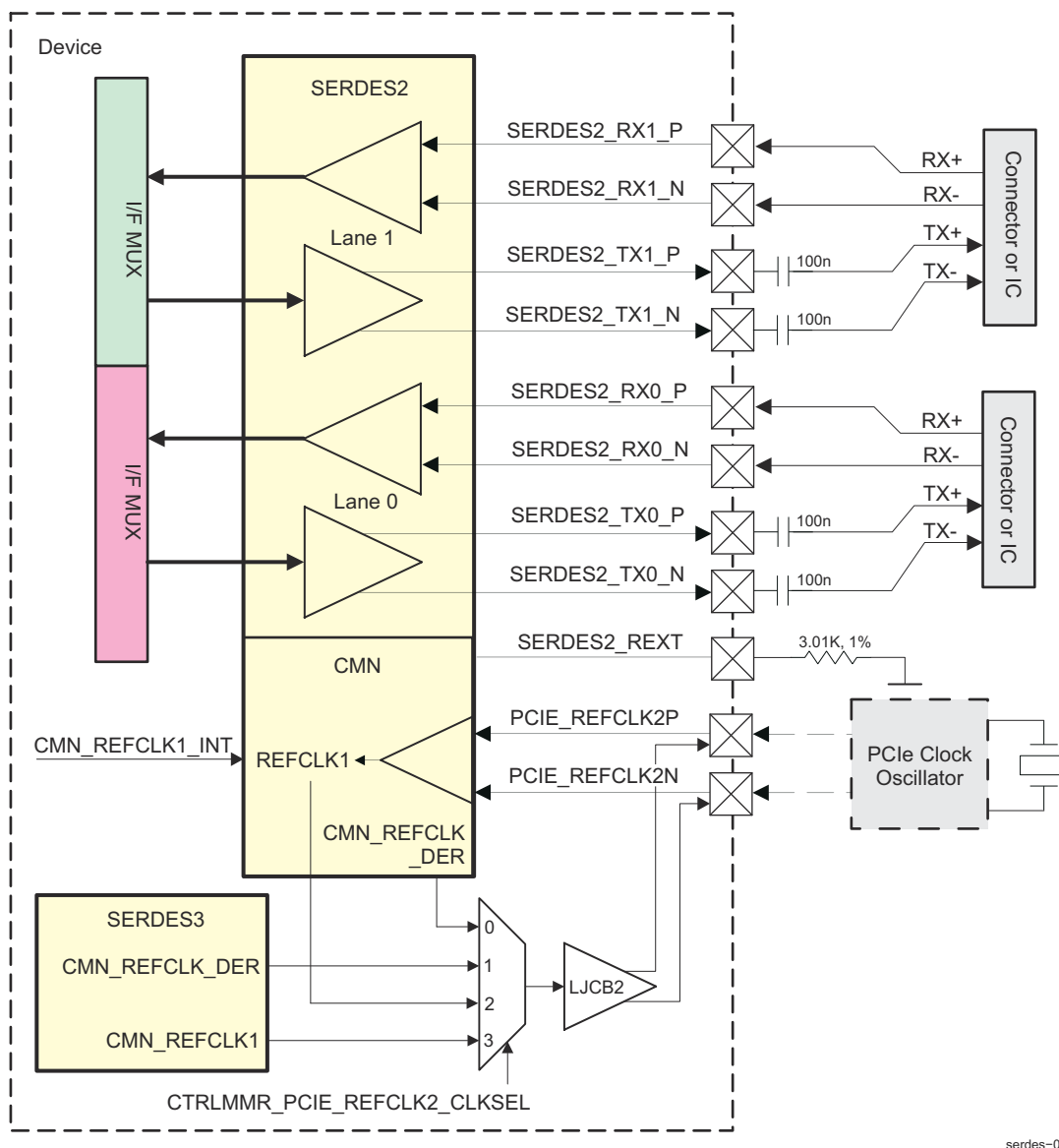
**Table 12-260. SERDES1 Input/Output Description**

Device Pin	Module Signal	I/O	Description	Value at Reset
SERDES1_RX0_P	RX_P_LN0	I	SerDes differential data receive pins. Lane 0	HiZ
SERDES1_RX0_N	RX_M_LN0			
SERDES1_TX0_P	TX_P_LN0	O	SerDes differential data transmit pins. Lane 0	HiZ
SERDES1_TX0_N	TX_M_LN0			
SERDES1_RX1_P	RX_P_LN1	I	SerDes differential data receive pins. Lane 1	HiZ
SERDES1_RX1_N	RX_M_LN1			
SERDES1_TX1_P	TX_P_LN1	O	SerDes differential data transmit pins. Lane 1	HiZ
SERDES1_TX1_N	TX_M_LN1			
PCIE_REFCLK1P	CMN_REFCLK1_P	I	SerDes external system reference clock for PCIe	HiZ
PCIE_REFCLK1N	CMN_REFCLK1_M			

**Table 12-260. SERDES1 Input/Output Description (continued)**

Device Pin	Module Signal	I/O	Description	Value at Reset
	SERDES0_CMN_REFCLK_DE R	O	SerDes system reference clock for external PCIe device. Selected via CTRLMMR_PCIE_REFCLK1_CLKSEL	
	SERDES1_CMN_REFCLK_DE R			
	SERDES0_CMN_REFCLK1			
	SERDES1_CMN_REFCLK1			
SERDES1_REXT	CMN_REXT	A/I	PMA external calibration resistor. Requires a 3.01 kOhm $\pm 1\%$ accurate off-chip resistor connected from this pin to ground.	HiZ

Figure 12-209 shows the I/O interface pins of SerDes.



serdes-062

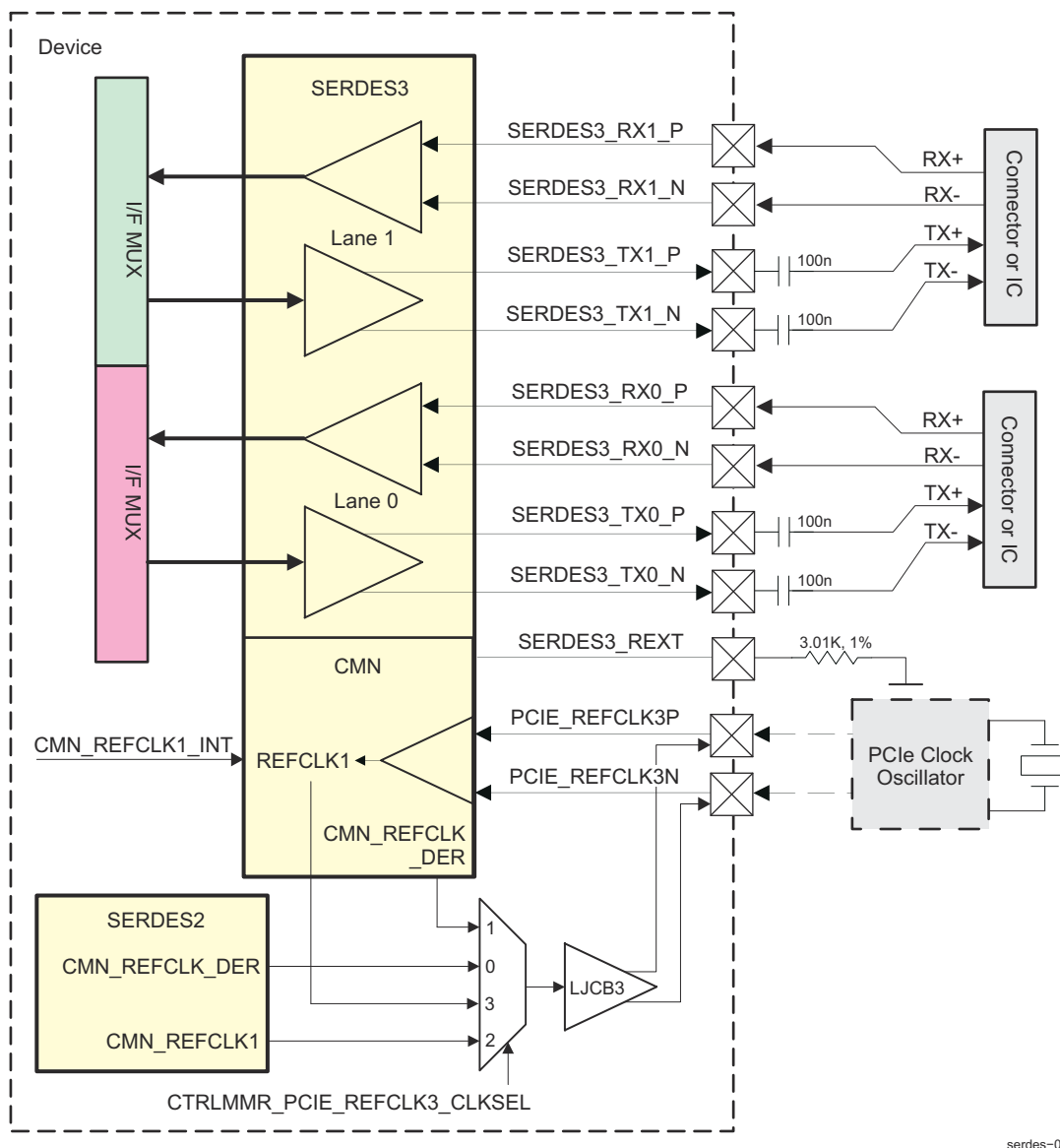
**Figure 12-209. SERDES2 Environment**

Table 12-261 describes the external signals of SERDES2 module.

**Table 12-261. SERDES2 Input/Output Description**

Device Pin	Module Signal	I/O	Description	Value at Reset
SERDES2_RX0_P	RX_P_LN0	I	SerDes differential data receive pins. Lane 0	HiZ
SERDES2_RX0_N	RX_M_LN0			
SERDES2_TX0_P	TX_P_LN0	O	SerDes differential data transmit pins. Lane 0	HiZ
SERDES2_TX0_N	TX_M_LN0			
SERDES2_RX1_P	RX_P_LN1	I	SerDes differential data receive pins. Lane 1	HiZ
SERDES2_RX1_N	RX_M_LN1			
SERDES2_TX1_P	TX_P_LN1	O	SerDes differential data transmit pins. Lane 1	HiZ
SERDES2_TX1_N	TX_M_LN1			
PCIE_REFCLK2P	CMN_REFCLK1_P	I	SerDes external system reference clock for PCIe	HiZ
PCIE_REFCLK2N	CMN_REFCLK1_M			
	SERDES2_CMN_REFCLK_DE R	O	SerDes system reference clock for external PCIe device. Selected via CTRLMMR_PCIE_REFCLK2_CLKSEL	
	SERDES3_CMN_REFCLK_DE R			
	SERDES2_CMN_REFCLK1			
	SERDES3_CMN_REFCLK1			
SERDES2_REXT	CMN_REXT	A/I	PMA external calibration resistor. Requires a 3.01 kOhm $\pm 1\%$ accurate off-chip resistor connected from this pin to ground.	HiZ

Figure 12-210 shows the I/O interface pins of SerDes.



serdes-063

**Figure 12-210. SERDES3 Environment**

Table 12-262 describes the external signals of SERDES3 module.

**Table 12-262. SERDES3 Input/Output Description**

Device Pin	Module Signal	I/O	Description	Value at Reset
SERDES3_RX0_P	RX_P_LN0	I	SerDes differential data receive pins. Lane 0	HiZ
SERDES3_RX0_N	RX_M_LN0			
SERDES3_TX0_P	TX_P_LN0	O	SerDes differential data transmit pins. Lane 0	HiZ
SERDES3_TX0_N	TX_M_LN0			
SERDES3_RX1_P	RX_P_LN1	I	SerDes differential data receive pins. Lane 1	HiZ
SERDES3_RX1_N	RX_M_LN1			
SERDES3_TX1_P	TX_P_LN1	O	SerDes differential data transmit pins. Lane 1	HiZ
SERDES3_TX1_N	TX_M_LN1			
PCIE_REFCLK3P	CMN_REFCLK1_P	I	SerDes external system reference clock for PCIe	HiZ
PCIE_REFCLK3N	CMN_REFCLK1_M			

**Table 12-262. SERDES3 Input/Output Description (continued)**

Device Pin	Module Signal	I/O	Description	Value at Reset
	SERDES2_CMN_REFCLK_DE R	O	SerDes system reference clock for external PCIe device. Selected via CTRLMMR_PCIE_REFCLK3_CLKSEL	
	SERDES3_CMN_REFCLK_DE R			
	SERDES2_CMN_REFCLK1			
	SERDES3_CMN_REFCLK1			
SERDES3_REXT	CMN_REXT	A/I	PMA external calibration resistor. Requires a 3.01 kOhm $\pm 1\%$ accurate off-chip resistor connected from this pin to ground.	HiZ



### 12.2.5.3 2-L SerDes Integration

This section describes SerDes module integration in the device, including information about clocks, resets, and hardware requests.

Figure 12-211 shows the integration of the SerDes modules in the device.

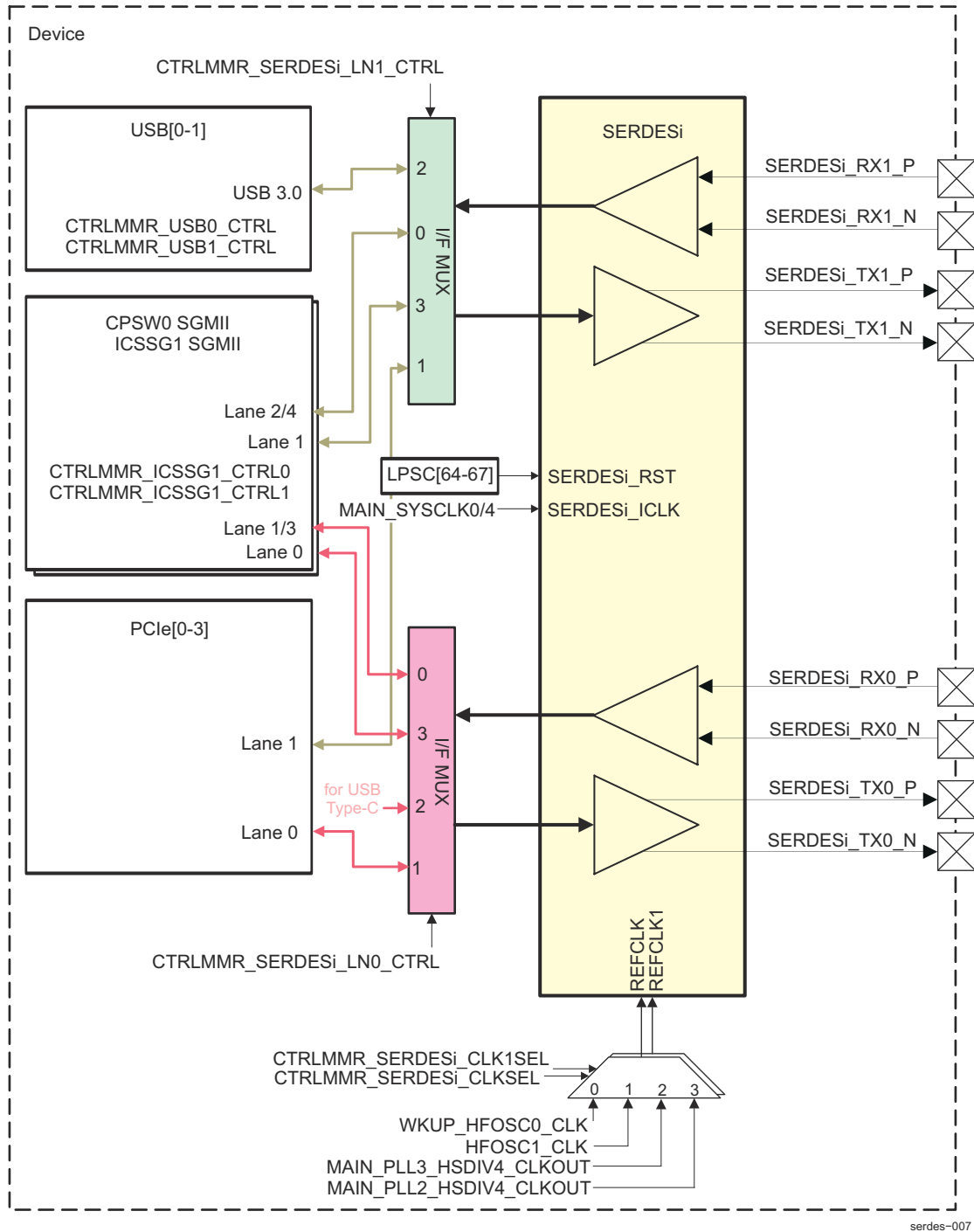


Figure 12-211. 2-L SerDes Integration

i = 0 to 3

Table 12-263 through Table 12-265 summarize the integration of SerDes in device MAIN domain.

**Table 12-263. 2-L SerDes Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
SERDES0	PSC0	PD5	LPSC64	CBASS0
SERDES1	PSC0	PD6	LPSC65	CBASS0
SERDES2	PSC0	PD7	LPSC66	CBASS0
SERDES3	PSC0	PD8	LPSC67	CBASS0

**Table 12-264. 2-L SerDes Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
SERDES0	SERDES0_ICLK	MAIN_SYSCCLK0/4	PLL_CTRL0	VBUS interface clock
	CMN_REFCLK_INT	WKUP_HFOSC0_CLK	WKUP_HFOSC0	Internal reference clock from device sources. Software selectable. See <a href="#">Section 12.2.5.3.1.2</a>
		HFOSC1_CLK	HFOSC1	
		MAIN_PLL3_HSDIV4_C LKOUT	PLL3	
		MAIN_PLL2_HSDIV4_C LKOUT	PLL2	
	CMN_REFCLK1_INT	WKUP_HFOSC0_CLK	WKUP_HFOSC0	Internal reference clock 1 from device sources. Software selectable. See <a href="#">Section 12.2.5.3.1.2</a>
		HFOSC1_CLK	HFOSC1	
		MAIN_PLL3_HSDIV4_C LKOUT	PLL3	
		MAIN_PLL2_HSDIV4_C LKOUT	PLL2	
SERDES1	SERDES1_ICLK	MAIN_SYSCCLK0/4	PLL_CTRL0	VBUS interface clock
	CMN_REFCLK_INT	WKUP_HFOSC0_CLK	WKUP_HFOSC0	Internal reference clock from device sources. Software selectable. See <a href="#">Section 12.2.5.3.1.2</a>
		HFOSC1_CLK	HFOSC1	
		MAIN_PLL3_HSDIV4_C LKOUT	PLL3	
		MAIN_PLL2_HSDIV4_C LKOUT	PLL2	
	CMN_REFCLK1_INT	WKUP_HFOSC0_CLK	WKUP_HFOSC0	Internal reference clock 1 from device sources. Software selectable. See <a href="#">Section 12.2.5.3.1.2</a>
		HFOSC1_CLK	HFOSC1	
		MAIN_PLL3_HSDIV4_C LKOUT	PLL3	
		MAIN_PLL2_HSDIV4_C LKOUT	PLL2	
SERDES2	SERDES2_ICLK	MAIN_SYSCCLK0/4	PLL_CTRL0	VBUS interface clock
	CMN_REFCLK_INT	WKUP_HFOSC0_CLK	WKUP_HFOSC0	Internal reference clock from device sources. Software selectable. See <a href="#">Section 12.2.5.3.1.2</a>
		HFOSC1_CLK	HFOSC1	
		MAIN_PLL3_HSDIV4_C LKOUT	PLL3	
		MAIN_PLL2_HSDIV4_C LKOUT	PLL2	

**Table 12-264. 2-L SerDes Clocks and Resets (continued)**

SERDES3	CMN_REFCLK1_INT	WKUP_HFOSC0_CLK	WKUP_HFOSC0	Internal reference clock 1 from device sources. Software selectable. See <a href="#">Section 12.2.5.3.1.2</a>
		HFOSC1_CLK	HFOSC1	
		MAIN_PLL3_HSDIV4_C LKOUT	PLL3	
		MAIN_PLL2_HSDIV4_C LKOUT	PLL2	
	SERDES3_ICLK	MAIN_SYSCLK0/4	PLL_CTRL0	VBUS interface clock
	CMN_REFCLK_INT	WKUP_HFOSC0_CLK	WKUP_HFOSC0	Internal reference clock from device sources. Software selectable. See <a href="#">Section 12.2.5.3.1.2</a>
		HFOSC1_CLK	HFOSC1	
		MAIN_PLL3_HSDIV4_C LKOUT	PLL3	
		MAIN_PLL2_HSDIV4_C LKOUT	PLL2	
	CMN_REFCLK1_INT	WKUP_HFOSC0_CLK	WKUP_HFOSC0	Internal reference clock 1 from device sources. Software selectable. See <a href="#">Section 12.2.5.3.1.2</a>
		HFOSC1_CLK	HFOSC1	
		MAIN_PLL3_HSDIV4_C LKOUT	PLL3	
		MAIN_PLL2_HSDIV4_C LKOUT	PLL2	

Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
SERDES0	SERDES0_RST	MOD_G_RST	LPSC64	SERDES0 LPSC reset
SERDES1	SERDES1_RST	MOD_G_RST	LPSC65	SERDES1 LPSC reset
SERDES2	SERDES2_RST	MOD_G_RST	LPSC66	SERDES2 LPSC reset
SERDES3	SERDES3_RST	MOD_G_RST	LPSC67	SERDES3 LPSC reset

**Table 12-265. 2-L SerDes Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
SERDES0 to SERDES3	-	-	-	No interrupt requests to interrupt controllers	-

DMA Events					
Module Instance	Module DMA Event	Destination DMA Event Input	Destination	Description	Type
SERDES0 to SERDES3	-	-	-	No PDMA channels to external DMA engines	-

### Note

For more information on the interconnects in device, see [Chapter 3, System Interconnects](#).

For more information on the power, reset and clock management in device MAIN domain, see the corresponding sections within [Chapter 5, Device Configuration](#).

### 12.2.5.3.1 2-L WIZ Settings

This section describes the quazi-static settings that software must perform after global resets. These settings are made via registers in the CTRL\_MMR0 module. For CTRL\_MMR0 description, please refer to [Section 5.1, Control Module \(CTRL\\_MMR\)](#).

#### 12.2.5.3.1.1 Interface Selection

SerDeses provide PHY functions for the following high-speed interfaces:

- USB0 (USB 3.1 Gen1)
- USB1 (USB 3.1 Gen1)
- PCIe0 (one- or two-lane)
- PCIe1 (one- or two-lane)
- PCIe2 (one- or two-lane)
- PCIe3 (one- or two-lane)
- CPSW0 SGMII (one- to four-lane)
- CPSW0 QSGMII
- ICSSG1 SGMII (one- or two-lane)

[Table 12-266](#) describes the interface combinations supported by SERDES0.

**Table 12-266. SERDES0 Supported Configurations**

Interface Alias	CTRLMMR_SERDES0_LN0_CTRL		CTRLMMR_SERDES0_LN1_CTRL	
	[1:0] LANE_FUNC_SEL	Interface on Lane 0	[1:0] LANE_FUNC_SEL	Interface on Lane 1
IP1	0x0	CPSW0 Q/SGMII Lane 1	0x0	CPSW0 Q/SGMII Lane 2
IP2	0x1	PCIe0 Lane 0	0x1	PCIe0 Lane 1
IP3	0x2	-(1)	0x2	USB0
IP4	0x3	-	0x3	-

(1) Reserved for USB0 for type-C connector lane swap. That is, select this function if Type C was implemented in the design. See [Section 12.2.4, USB](#).

[Table 12-267](#) describes the interface combinations supported by SERDES1.

**Table 12-267. SERDES1 Supported Configurations**

Interface Alias	CTRLMMR_SERDES1_LN0_CTRL		CTRLMMR_SERDES1_LN1_CTRL	
	[1:0] LANE_FUNC_SEL	Interface on Lane 0	[1:0] LANE_FUNC_SEL	Interface on Lane 1
IP1	0x0	CPSW0 Q/SGMII Lane 3	0x0	CPSW0 Q/SGMII Lane 4
IP2	0x1	PCIe1 Lane 0	0x1	PCIe1 Lane 1
IP3	0x2	-(1)	0x2	USB1
IP4	0x3	ICSSG1 SGMII Lane 0	0x3	ICSSG1 SGMII Lane 1

(1) Reserved for USB1 for type-C connector lane swap. That is, select this function if Type C was implemented in the design. See [Section 12.2.4, USB](#).

[Table 12-268](#) describes the interface combinations supported by SERDES2.

**Table 12-268. SERDES2 Supported Configurations**

Interface Alias	CTRLMMR_SERDES2_LN0_CTRL		CTRLMMR_SERDES2_LN1_CTRL	
	[1:0] LANE_FUNC_SEL	Interface on Lane 0	[1:0] LANE_FUNC_SEL	Interface on Lane 1
IP1	0x0	-	0x0	-
IP2	0x1	PCIe2 Lane 0	0x1	PCIe2 Lane 1
IP3	0x2	-(1)	0x2	USB1
IP4	0x3	ICSSG1 SGMII Lane 0	0x3	ICSSG1 SGMII Lane 1

(1) Reserved for USB1 for type-C connector lane swap. That is, select this function if Type C was implemented in the design. See [Section 12.2.4, USB](#).

Table 12-269 describes the interface combinations supported by SERDES3.

**Table 12-269. SERDES3 Supported Configurations**

Interface Alias	CTRLMMR_SERDES3_LN0_CTRL		CTRLMMR_SERDES3_LN1_CTRL	
	[1:0] LANE_FUNC_SEL	Interface on Lane 0	[1:0] LANE_FUNC_SEL	Interface on Lane 1
IP1	0x0	-	0x0	-
IP2	0x1	PCIe3 Lane 0	0x1	PCIe3 Lane 1
IP3	0x2	-(1)	0x2	USB0
IP4	0x3	-	0x3	-

(1) Reserved for USB0 for type-C connector lane swap. That is, select this function if Type C was implemented in the design. See [Section 12.2.4, USB](#).

As seen in [Table 12-266](#) to [Table 12-269](#), USB0, USB1, and ICSSG1 SGMII can be routed to two different lanes. To avoid routing to two lanes at the same time, an additional muxing exists.

[Table 12-270](#) to [Table 12-273](#) describes the additional muxing for USB0, USB1, and ICSSG1 SGMII lane 0, and lane 1.

#### Note

Settings in [Table 12-270](#) to [Table 12-273](#) must be aligned with the settings made in [Table 12-266](#) to [Table 12-269](#).

**Table 12-270. USB0 Muxing to Serdeses**

CTRLMMR_USB0_CTRL	
[27] SERDES_SEL	Serdes Selected
0	SERDES0
1	SERDES3

**Table 12-271. USB1 Muxing to Serdeses**

CTRLMMR_USB1_CTRL	
[27] SERDES_SEL	Serdes Selected
0	SERDES1
1	SERDES2

**Table 12-272. ICSSG1 SGMII Lane 0 Muxing to Serdeses**

CTRLMMR_ICSSG1_CTRL0	
[28] SGMII_SERDES_SEL	Serdes Selected
0	SERDES1 Lane 0
1	SERDES2 Lane 0

**Table 12-273. ICSSG1 SGMII Lane 1 Muxing to Serdeses**

CTRLMMR_ICSSG1_CTRL1	
[28] SGMII_SERDES_SEL	Serdes Selected
0	SERDES1 Lane 1
1	SERDES2 Lane 1

#### 12.2.5.3.1.2 Internal Reference Clock Selection

The device clock tree provides the SerDeses with identical clock options but with independent selection control for each SerDes.

Table 12-274 and Table 12-275 describe the internal reference clock options for SERDES0 through SERDES3, REFCLK and REFCLK1 inputs.

**Table 12-274. 2-L SerDes Internal Reference Clock Selection (REFCLK)**

SERDES0: CTRLMMR_SERDES0_CLKSEL SERDES1: CTRLMMR_SERDES1_CLKSEL SERDES2: CTRLMMR_SERDES2_CLKSEL SERDES3: CTRLMMR_SERDES3_CLKSEL	
[1:0] CORE_REFCLK_SEL	Internal Clock to SERDES Core
0x0	WKUP_HFOSC0_CLK
0x1	HFOSC1_CLK
0x2	MAIN_PLL3_HSDIV4_CLKOUT
0x3	MAIN_PLL2_HSDIV4_CLKOUT

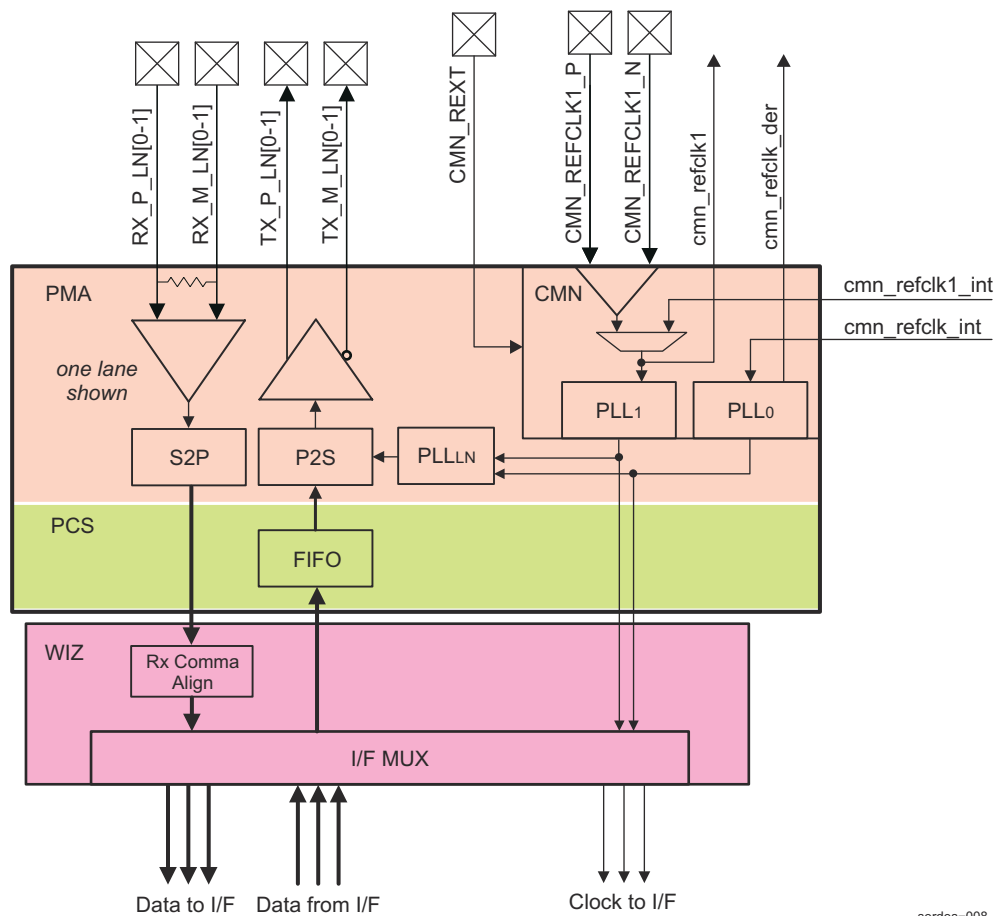
**Table 12-275. 2-L SerDes Internal Reference Clock Selection (REFCLK1)**

SERDES0: CTRLMMR_SERDES0_CLK1SEL SERDES1: CTRLMMR_SERDES1_CLK1SEL SERDES2: CTRLMMR_SERDES2_CLK1SEL SERDES3: CTRLMMR_SERDES3_CLK1SEL	
[1:0] CORE_REFCLK_SEL	Internal Clock to SERDES Core
0x0	WKUP_HFOSC0_CLK
0x1	HFOSC1_CLK
0x2	MAIN_PLL3_HSDIV4_CLKOUT
0x3	MAIN_PLL2_HSDIV4_CLKOUT

## 12.2.5.4 2-L SerDes Functional Description

### 12.2.5.4.1 2-L SerDes Block Diagram

Figure 12-212 is a top-level, non-exhaustive diagram of SerDes and WIZ wrapper. Note that only one Tx/Rx lane is shown.



**Figure 12-212. SerDes and WIZ Block Diagram**

Building blocks of SerDes include:

- Lanes: The lanes handle all inputs and outputs from the serial interface, and contain the Tx/Rx I/Os, serializer/deserializer (P2S/S2P), and Clock and Data Recovery (CDR) unit. Each SerDes contains one Tx and Rx lane.
- Common module (CMN): The CMN handles peripheral and Tx clocking of the SerDes. It consists of internal PLLs and external reference clock input buffer, reset, and startup circuitry.
- Lanes and CMN are parts of the Physical Media Attachment (PMA) layer.
- Physical Coding Sub-block (PCS): The PCS is responsible for translating data from/to the parallel interface, as well as data encoding/decoding and symbol alignment.
- WIZ: The WIZ acts as a wrapper for the SerDes, and can both send control signals to and report status signals from the SerDes (register interface), and muxes SerDes to peripherals (PCIe, USB3.0, and SGMII).

## 12.2.6 4-L Serializer/Deserializer (SerDes)

This section describes the 4-Lane Serializer/Deserializer (SerDes) in the device.

### 12.2.6.1 4-L SerDes Overview

SerDes'es goal is to convert device (SoC) parallel data into serialized data that can be output over a high-speed electrical interface. In the opposite direction, SerDes converts high-speed serial data into parallel data that can be processed by the device. To this end, the SerDes contains a variety of functional blocks to handle both the external analog interface as well as the internal digital logic.

Most important building blocks of SerDes are:

- Physical Media Attachment (PMA):
  - Lanes: The lanes handle all inputs and outputs from the serial interface, and contain the Tx/Rx I/Os, serializer/deserializer, and Clock and Data Recovery (CDR) unit.
  - Common module (CMN): The CMN handles peripheral and Tx clocking of the SerDes. It consists of internal PLLs and external reference clock input buffer, reset, and startup circuitry.
- Physical Coding Sub-block (PCS): The PCS is responsible for translating data from/to the parallel interface, as well as data encoding/decoding and symbol alignment.
- WIZ: The WIZ acts as a wrapper for the SerDes, and can both send control signals to and report status signals from the SerDes, and muxes SerDes to peripherals.

The device contains one quad-lane SerDes: SERDES4.

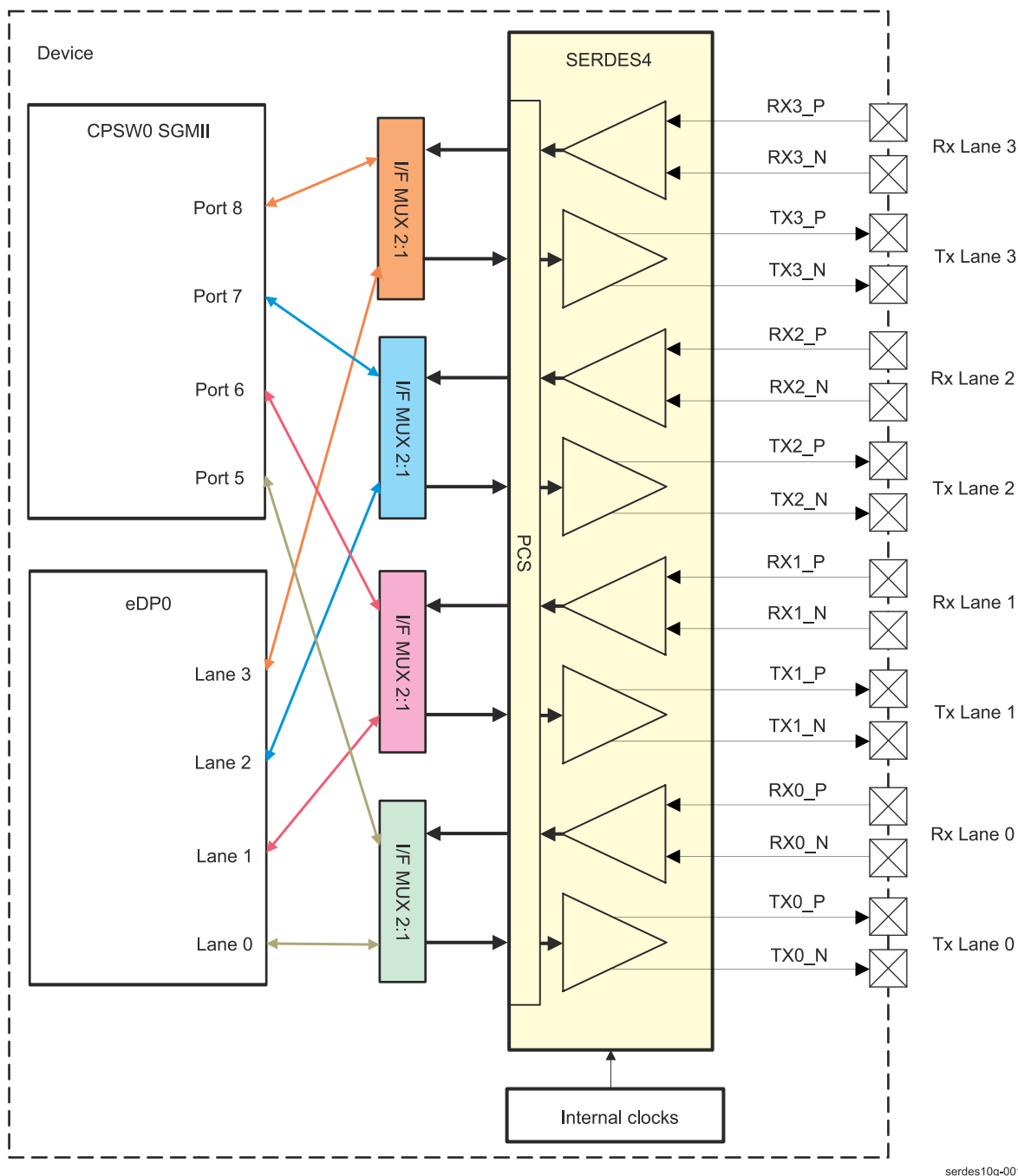
For two-lane SerDeses (SERDES[0-3]), please refer to [Section 12.2.5, 2-L Serializer/Deserializer \(SerDes\)](#).

**Table 12-276. SerDes Allocation Within Device Domains**

Module Instance	Domain		
	WKUP	MCU	MAIN
SERDES0 (2-L)	-	-	✓
SERDES1 (2-L)	-	-	✓
SERDES2 (2-L)	-	-	✓
SERDES3 (2-L)	-	-	✓
SERDES4	-	-	✓

[Figure 12-213](#) shows 4-Lane SerDes highlights.





serdes10g-001

Figure 12-213. SERDES4 Overview

#### 12.2.6.1.1 4-L SerDes Features

The SERDES module features include:

- Quad lane PHY containing:
  - Transmit and Receive I/Os
  - Serializer
  - Deserializer
  - Clock and data recovery (CDR) unit
- Common Module (CMN)
  - PLLs

- Master bias
- Automatic calibration of pin termination resistors
- Reference clock input buffers
- Reset and startup management
- Physical Coding Sub-block (PCS)
  - Embedded DisplayPort (eDP1.4) Tx (UI\_Rate\_1 - UI\_Rate\_8)
    -
  - QSGMII Specification revision 1.2
  - Symbol alignment
  - Selectable serial pin polarity reversal for both transmit and receive paths
  - Bit stream reordering
- Physical Media Attachment (PMA) layer
  - Transmit equalization
  - Receive equalization
  - Data path BIST with programmable pattern generation and error detection
  - Serial bit stream and parallel word loopback for both line and parallel side
  - 8-bit ADC provides digitized ATB measurement results
  - Supports DC and AC JTAG (boundary scan) per IEEE 1149.6

The SERDES mux (WIZ) module supports the following features:

- Multiplexes device interfaces onto a single SERDES lane (one Tx and one Rx)
- Provides registers to implement SERDES control and status functions and alignment delays
- Clock generator block for providing MAC transmit clock
- Rx comma align block
  - Performs de-stuffing the Rx data stream in the event that the Rx rate is different from the Tx rate
  - Supports comma detection that is not sensitive to false commas using all 8B10B character combinations

#### 12.2.6.1.2 Industry Standards Compatibility

All SerDes interfaces are configured as point-to-point connections. It is assumed that the connection is made between the SoC and another device that is compliant to the appropriate industry standard. The list of supported standards is given below.

This chapter deals with the physical layer and, therefore, it is the electrical specifications in these standards that are relevant. For more information regarding protocol compliance <sup>3</sup>, see the device-specific Datasheet.

- SGMII: This is electrically compliant with SGMII revision 1.8 with the following clarifications:
  - It does not implement the separate clock signaling
  - Electrical compatibility does not guarantee interoperability with devices.
- QSGMII: This is electrically compliant with QSGMII revision 1.2 with the following clarifications:
  - It does not implement the separate clock signaling
  - Electrical compatibility does not guarantee interoperability with devices.
- Embedded DisplayPort Tx (UI\_Rate\_1 - UI\_Rate\_8)

**Table 12-277. 4-L SerDes Supported Standards**

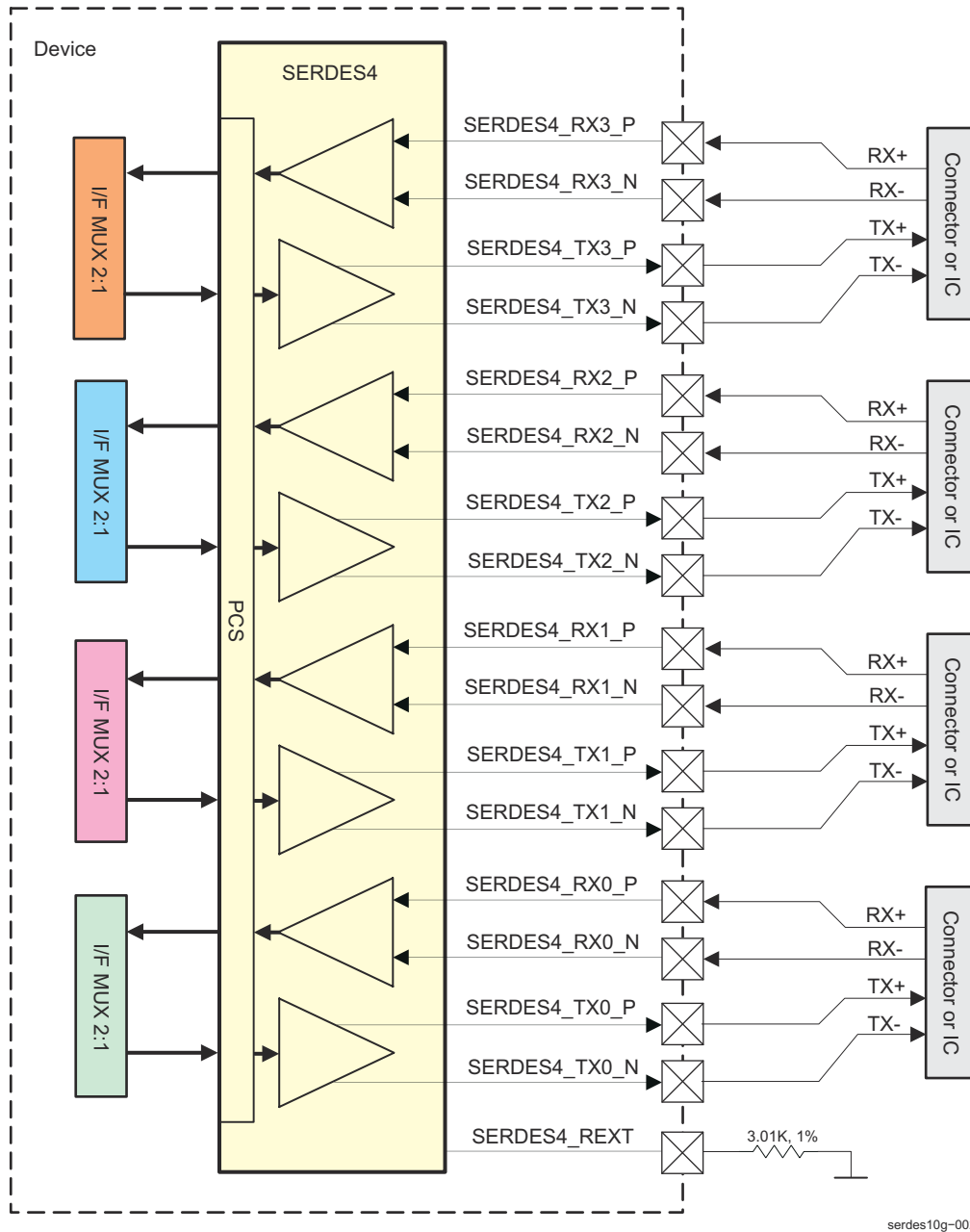
Standard	Bit Rate (Gbps)	Reference Clock Frequency (MHz)	Bus Width (bits)
SGMII	1.25	19.2, 20, 24, 25, 26, 27, 100, 125	10
SGMII	2.5	19.2, 20, 24, 25, 26, 27, 100, 125	10
eDP	25.92 (total)	19.2, 20, 24, 25, 26, 27, 100	10

<sup>3</sup> Electrical compatibility does not guarantee interoperability with devices

## 12.2.6.2 4-L SerDes Environment

### 12.2.6.2.1 4-L SerDes I/Os

Figure 12-214 shows the I/O interface signals of SERDES.



serdes10g-002

**Figure 12-214. 4-L SerDes Environment**

#### Note

Although containing some of the basic external components, [Figure 12-214](#) must not be considered as an exhaustive guide for the PCB designer. TI provides additional documents for those who are willing to design PCBs and/or fine tune the SerDes.

[Table 12-278](#) describes the external signals of SERDES4 module.

**Table 12-278. 4-L SerDes Input/Output Description**

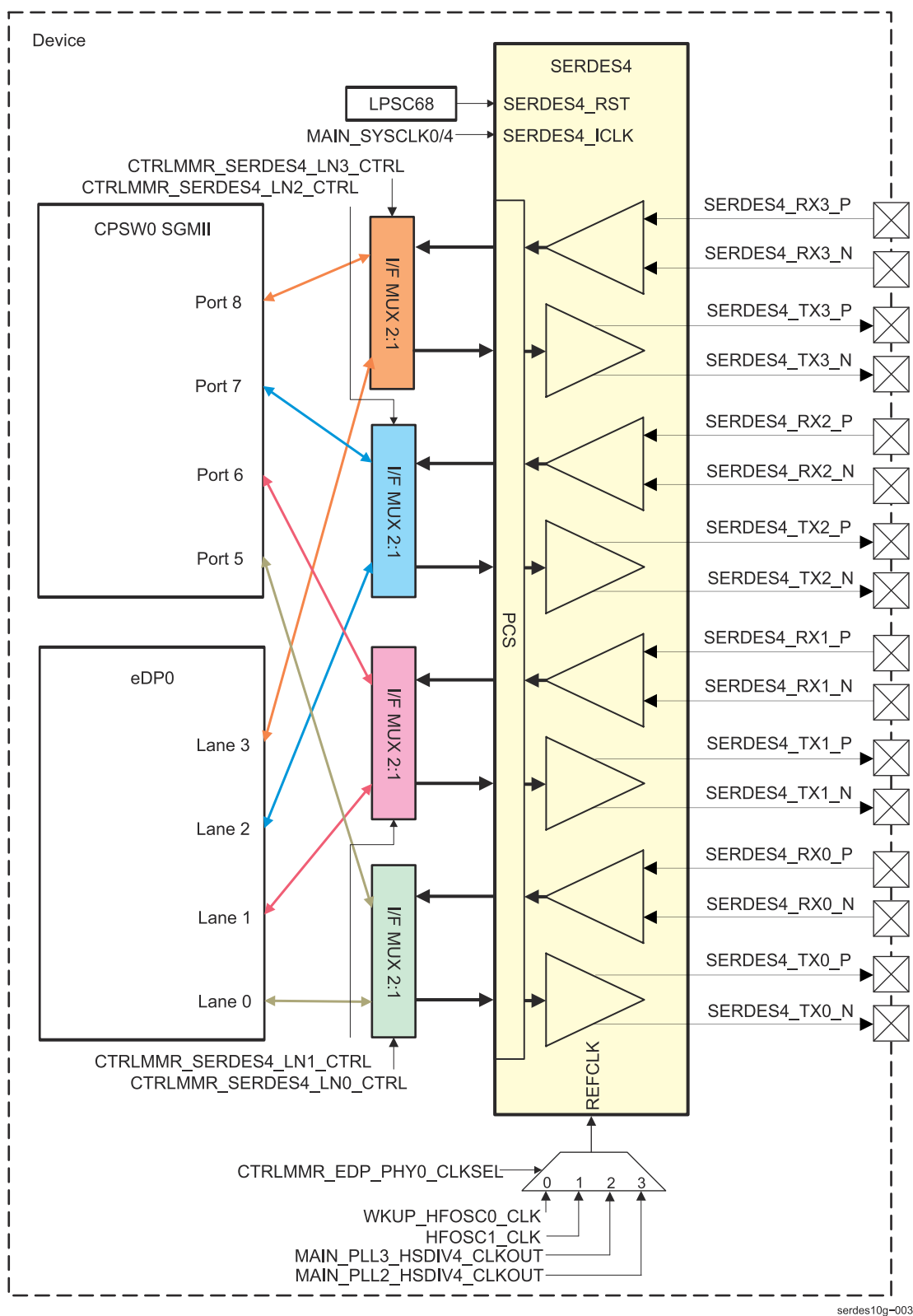
Device Pin	Module Signal	I/O <sup>(1)</sup>	Description	Value at Reset
SERDES4_RX0_P	RX_P_LN0	I	SerDes differential data receive pins. Lane 0	HiZ
SERDES4_RX0_N	RX_M_LN0	I		HiZ
SERDES4_TX0_P	TX_P_LN0	O	SerDes differential data transmit pins. Lane 0	HiZ
SERDES4_TX0_N	TX_M_LN0	O		HiZ
SERDES4_RX1_P	RX_P_LN1	I	SerDes differential data receive pins. Lane 1	HiZ
SERDES4_RX1_N	RX_M_LN1	I		HiZ
SERDES4_TX1_P	TX_P_LN1	O	SerDes differential data transmit pins. Lane 1	HiZ
SERDES4_TX1_N	TX_M_LN1	O		HiZ
SERDES4_RX2_P	RX_P_LN2	I	SerDes differential data receive pins. Lane 2	HiZ
SERDES4_RX2_N	RX_M_LN2	I		HiZ
SERDES4_TX2_P	TX_P_LN2	O	SerDes differential data transmit pins. Lane 2	HiZ
SERDES4_TX2_N	TX_M_LN2	O		HiZ
SERDES4_RX3_P	RX_P_LN3	I	SerDes differential data receive pins. Lane 3	HiZ
SERDES4_RX3_N	RX_M_LN3	I		HiZ
SERDES4_TX3_P	TX_P_LN3	O	SerDes differential data transmit pins. Lane 3	HiZ
SERDES4_TX3_N	TX_M_LN3	O		HiZ
SERDES4_REFCLK_P	CMN_REFCLK_P	I/O	SerDes external system reference clock for debug	HiZ
SERDES4_REFCLK_N	CMN_REFCLK_N	I/O		HiZ
SERDES4_REXT	CMN_REXT	A/I	PMA external calibration resistor. Requires a 3.01 kOhm $\pm 1\%$ accurate off-chip resistor connected from this pin to ground.	HiZ

(1) I = Input; O = Output; A = Analog

### 12.2.6.3 4-L SerDes Integration

This section describes SerDes module integration in the device, including information about clocks, resets, and hardware requests.

[Figure 12-215](#) shows the integration of the SerDes modules in the device.



**Figure 12-215. 4-L SerDes Integration**

Table 12-279 through Table 12-281 summarize the integration of SerDes in device MAIN domain.

**Table 12-279. 4-L SerDes Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
SERDES4	PSC0	PD9	LPSC68	CBASS0

**Table 12-280. 4-L SerDes Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
SERDES4	SERDES4_ICLK	MAIN_SYSCLK0/4	PLL_CTRL0	VBUS interface clock
	CMN_REFCLK_INT	WKUP_HFOSC0_CLKO UT	WKUP_HFOSC0	Internal reference clock from device sources. Software selectable. See <a href="#">Section 12.2.6.3.1.2</a>
		HFOSC1_CLKOUT	HFOSC1	
		MAIN_PLL3_HSDIV4_C LKOUT	PLL3	
		MAIN_PLL2_HSDIV4_C LKOUT	PLL2	
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
SERDES4	SERDES4_RST	MOD_G_RST	LPSC68	Serdes LPSC reset

**Table 12-281. 4-L SerDes Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
SERDES4	-	-	-	No interrupt requests to interrupt controllers	-
DMA Events					
Module Instance	Module DMA Event	Destination DMA Event Input	Destination	Description	Type
SERDES4	-	-	-	No PDMA channels to external DMA engines	-

### Note

For more information on the interconnects in device, see [Chapter 3, System Interconnects](#).

For more information on the power, reset and clock management in device MAIN domain, see the corresponding sections within [Chapter 5, Device Configuration](#).

### 12.2.6.3.1 4-L WIZ Settings

This section describes the quazi-static settings that software must perform after global resets. These settings are made via registers in the CTRL\_MMR0 module. For CTRL\_MMR0 description, please refer to [Section 5.1, Control Module \(CTRL\\_MMR\)](#).

#### 12.2.6.3.1.1 Interface Selection

SerDeses provide PHY functions for the following high-speed interfaces:

- CPSW0 SGMII (one- to four-lane)
- CPSW0 QSGMII
- eDP (two- or four-lane)

[Table 12-282](#) describes the interface combinations supported by SERDES4.

**Table 12-282. SERDES4 Supported Configurations**

CTRLMMR_SERDES4_LN0_CT RL [1:0]		CTRLMMR_SERDES4_LN1_CT RL [1:0]		CTRLMMR_SERDES4_LN2_CT RL [1:0]		CTRLMMR_SERDES4_LN3_CT RL [1:0]	
LANE_FUNC_SEL	Interface on Lane 0	LANE_FUNC_S EL	Interface on Lane 1	LANE_FUNC_S EL	Interface on Lane 2	LANE_FUNC_S EL	Interface on Lane 3
0x0	eDP Lane 0	0x0	eDP Lane 1	0x0	eDP Lane 2	0x0	eDP Lane 3
0x1	-	0x1	-	0x1	-	0x1	-
0x2	SGMII Lane 5	0x2	SGMII Lane 6	0x2	SGMII Lane 7	0x2	SGMII Lane 8
0x3	-	0x3	-	0x3	-	0x3	-

#### 12.2.6.3.1.2 Internal Reference Clock Selection

The device clock tree provides the SerDeses with identical clock options but with independent selection control for each SerDes.

[Table 12-283](#) describes the internal reference clock options for SERDES4.

**Table 12-283. 4-L SerDes Internal Reference Clock Selection**

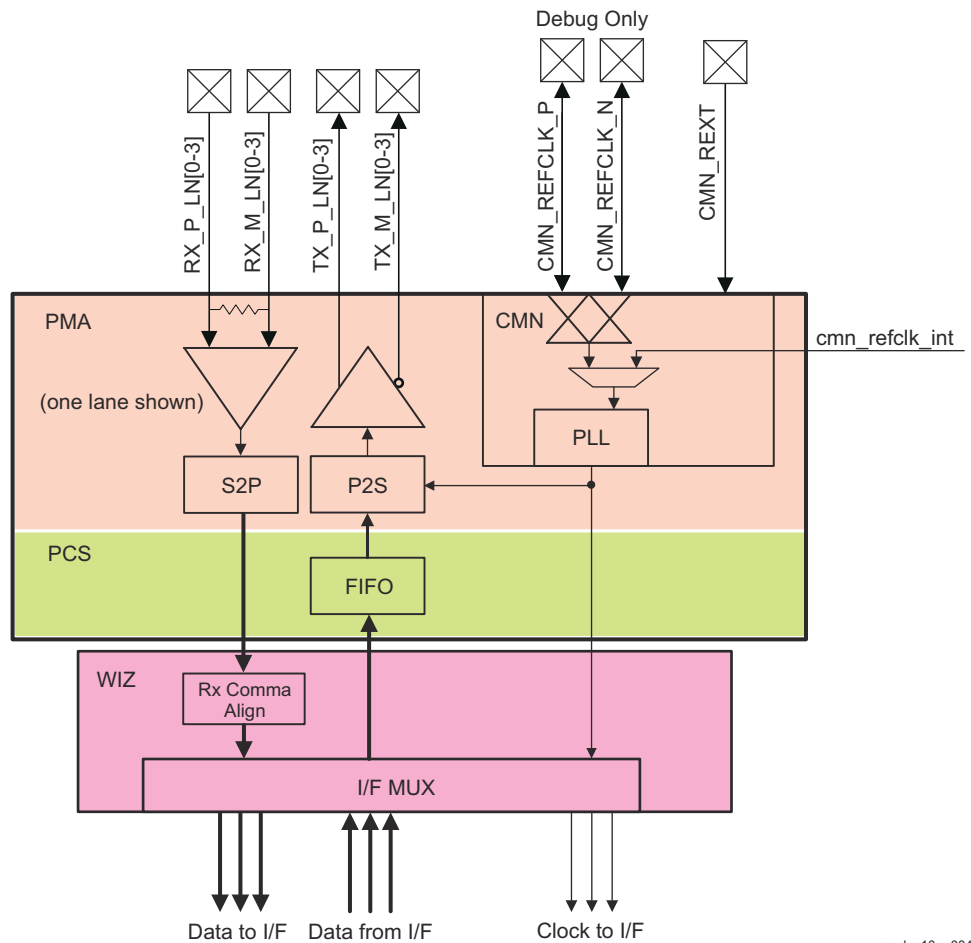
CTRLMMR_EDP_PHY0_CLKSEL	
[1:0] CLK_SEL	Internal Clock to SERDES Core
0x0	WKUP_HFOSC0_CLKOUT
0x1	HFOSC1_CLKOUT
0x2	MAIN_PLL3_HSDIV4_CLKOUT
0x3	MAIN_PLL2_HSDIV4_CLKOUT



## 12.2.6.4 4-L SerDes Functional Description

### 12.2.6.4.1 4-L SerDes Block Diagram

Figure 12-216 is a top-level, non-exhaustive diagram of SerDes and WIZ wrapper. Note that only one (Tx + Rx) lane is shown.



**Figure 12-216. 4-L SerDes and WIZ Block Diagram**

Building blocks of SerDes include:

- Lanes: The lanes handle all inputs and outputs from the serial interface, and contain the Tx/Rx I/Os, serializer/deserializer (P2S/S2P), and Clock and Data Recovery (CDR) unit. Each SerDes contains one Tx and Rx lane.
- Common module (CMN): The CMN handles peripheral and Tx clocking of the SerDes. It consists of internal PLLs and external reference clock input buffer, reset, and startup circuitry.
- Lanes and CMN are parts of the Physical Media Attachment (PMA) layer.
- Physical Coding Sub-block (PCS): The PCS is responsible for translating data from/to the parallel interface, as well as data encoding/decoding and symbol alignment.
- WIZ: The WIZ acts as a wrapper for the SerDes, and can both send control signals to and report status signals from the SerDes (register interface), and muxes SerDes to peripherals (eDP and SGMII).

## 12.3 Memory Interfaces

This section describes the memory interfaces in the device.

### 12.3.1 Flash Subsystem (FSS)

This section describes the Flash Subsystem (FSS) in the device.

#### 12.3.1.1 FSS Overview

The Flash Subsystem (FSS) provides access to external flash devices via Octal SPI (OSPI) and HyperBus™ interface.

The FSS includes two OSPIs and one HyperBus interface. For more information, see *Octal Serial Peripheral Interface (OSPI)* and [Section 12.3.3, HyperBus Interface](#).

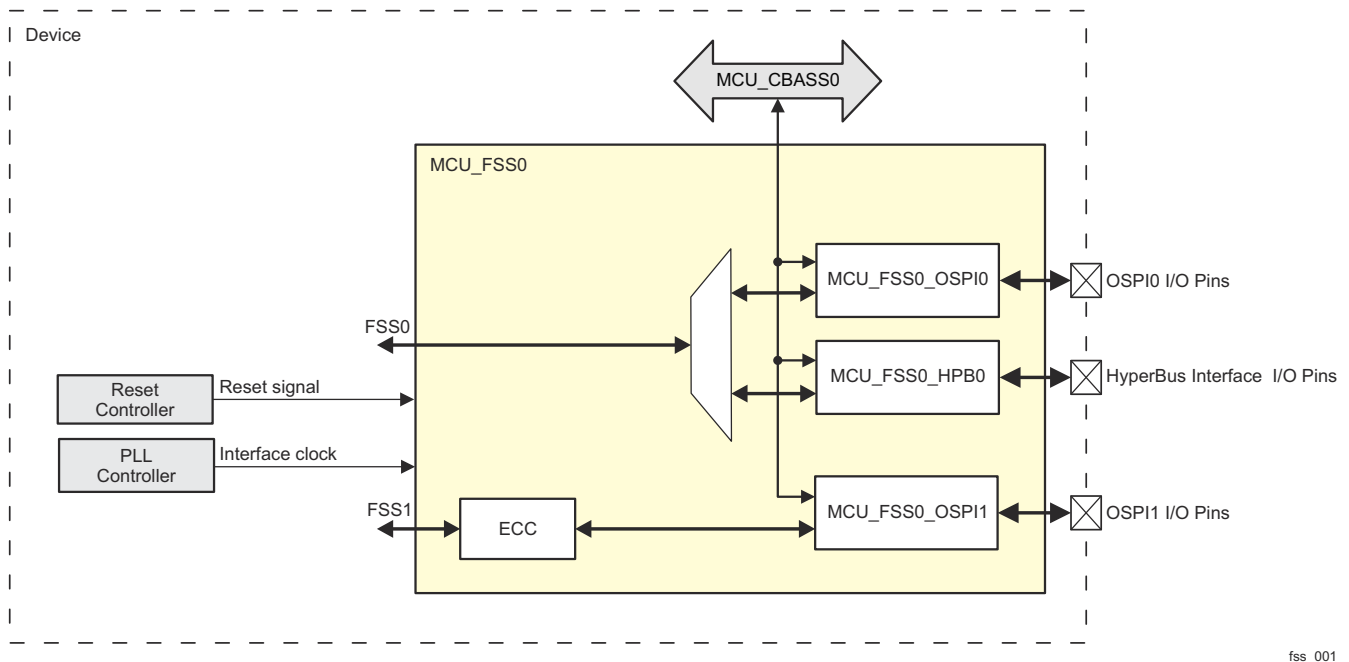
[Table 12-284](#) shows FSS allocation across device domains.

**Table 12-284. FSS Allocation Across Device Domains**

Instance	Domain		
	WKUP	MCU	MAIN
MCU_FSS0	-	✓	-

[Figure 12-217](#) shows the FSS overview.

The first FSS path (FSS0) has access to either OSPI0 or HyperBus interface. The second FSS path (FSS1) has access only to OSPI1 and can be configured with or without ECC.



fss\_001

**Figure 12-217. FSS Overview**

#### 12.3.1.1.1 FSS Features

The FSS has the following features:

- Two simultaneous flash interfaces:
  - Two OSPIs (OSPI0 and OSPI1) OR
  - One HyperBus interface and one OSPI (OSPI1)
- Primary OSPI0/HyperBus interface supports:
  - Execute in place (XIP) operation
  - 32-byte block copy (BC) operation

- Secondary OSPI1 interface supports:
  - 32-byte block copy (BC) operation
  - ECC
- OSPIs support single, dual, quad, or octal SPI devices
- OSPIs support up to 4 devices
- HyperBus interface supports up to 2 devices
- The OSPIs and HyperBus interface have independent power management for low power operations

**12.3.1.1.2 FSS Not Supported Features**

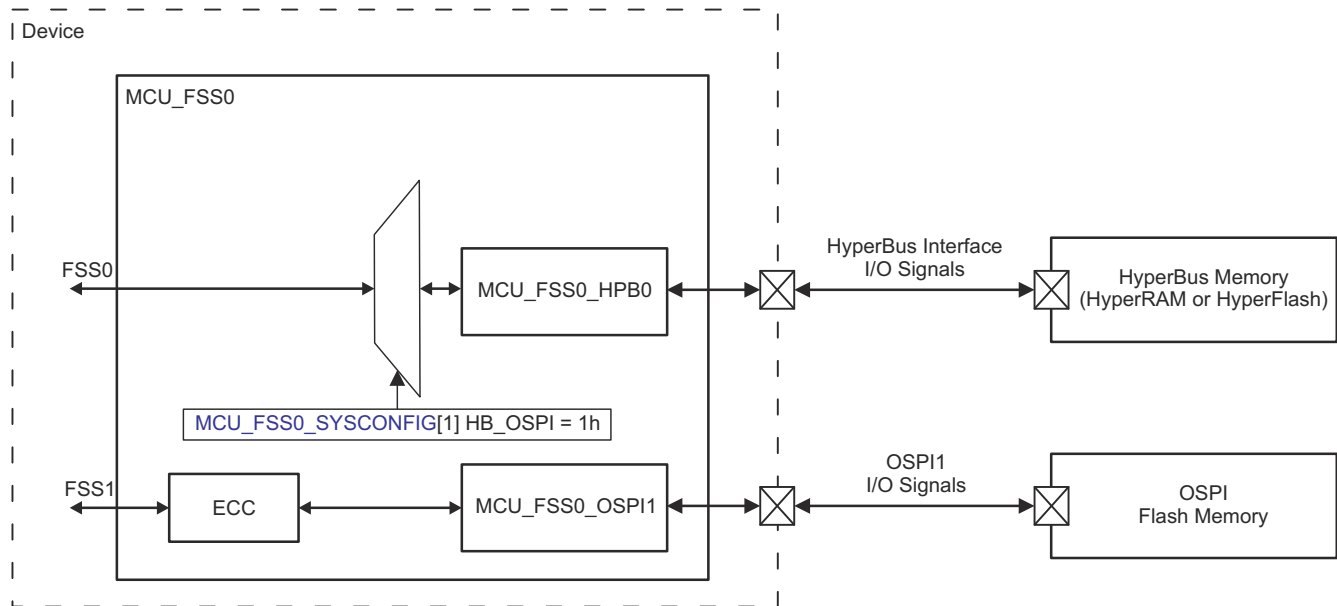
- DMA is not supported for OSPI0 and OSPI1 in INDAC mode

### 12.3.1.2 FSS Environment

The FSS is hereinafter also referred to as MCU\_FSS0.

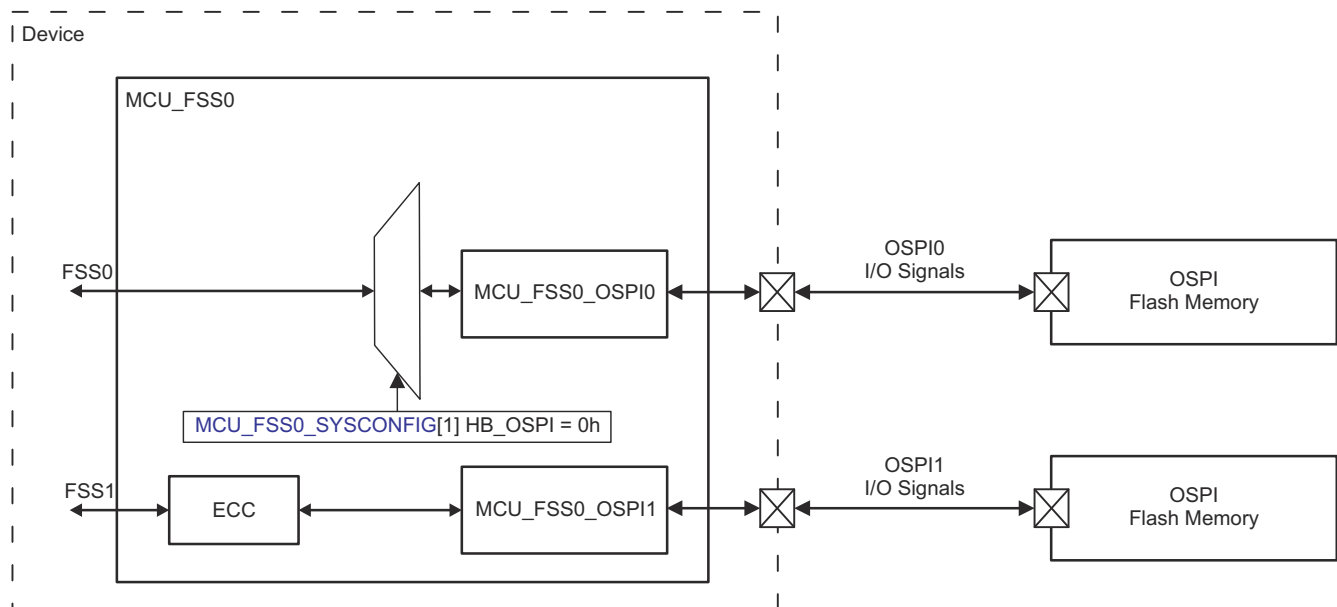
#### 12.3.1.2.1 FSS Typical Application

Figure 12-218 and Figure 12-219 show typical MCU\_FSS0 applications (two simultaneous FSS interfaces).



fss\_002

**Figure 12-218. MCU\_FSS0 Typical Application - HyperBus Interface and OSPI1**



fss\_002

**Figure 12-219. MCU\_FSS0 Typical Application - OSPI0 and OSPI1**

[Table 12-285](#) describes the MCU\_FSS0 I/O signals.

**Table 12-285. MCU\_FSS0 I/O Signals**

FSS Interface	I/O Signals
OSPIs (OSPI0 and OSPI1)	For more information about OSPIs I/O signals, see <i>OSPI I/O Signals</i> .
HyperBus interface	For more information about HyperBus interface I/O signals, see <a href="#">Table 12-305</a> .

---

**Note**

For more information on the OSPI environment, see *OSPI Environment*.

For more information on the HyperBus environment, see *HyperBus Environment*.

---



---

**Note**

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

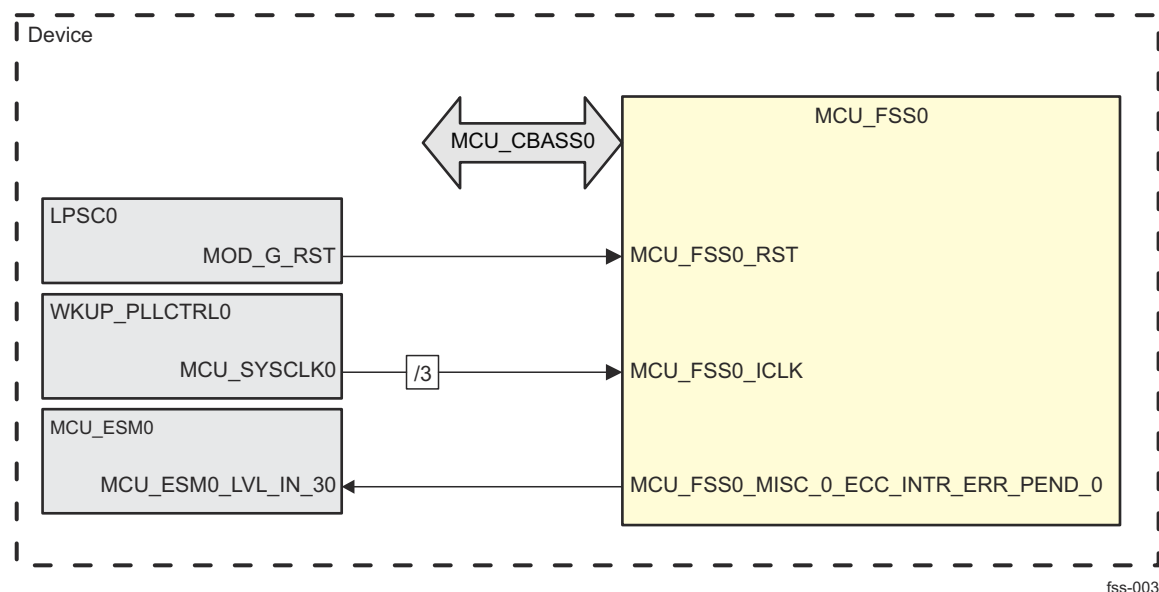
---

### 12.3.1.3 FSS Integration

This section describes the FSS integration in the device, including information about clocks, resets, and hardware requests.

#### 12.3.1.3.1 FSS Integration in MCU Domain

There is one FSS integrated in the device MCU domain - MCU\_FSS0. [Figure 12-220](#) shows the integration of MCU\_FSS0.



**Figure 12-220. MCU\_FSS0 Integration**

[Table 12-286](#) through [Table 12-288](#) summarize the integration of MCU\_FSS0 in the device MCU domain.

**Table 12-286. MCU\_FSS0 Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
MCU_FSS0	WKUP_PSC0	PD0	LPSC0	MCU_CBASS0

**Table 12-287. MCU\_FSS0 Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
MCU_FSS0	MCU_FSS0_ICLK	MCU_SYSCLK0/3	WKUP_PLLCTRL0	MCU_FSS0 Interface Clock

**Table 12-287. MCU\_FSS0 Clocks and Resets (continued)**

Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
MCU_FSS0	MCU_FSS0_RST	MOD_G_RST	LPSC0	MCU_FSS0 System Reset

**Table 12-288. MCU\_FSS0 Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCU_FSS0	MCU_FSS0_MISC_0_ECC_INTR_ERR_PEND_0	MCU_ESM0_LVL_IN_30	MCU_ESM0	MCU_FSS0 ECC Error Interrupt	Level



### 12.3.1.4 FSS Functional Description

#### 12.3.1.4.1 FSS Block Diagram

The FSS provides access to external Flash and RAM devices. It supports XIP (Execute-in-Place) and BC (Block Copy) operations.

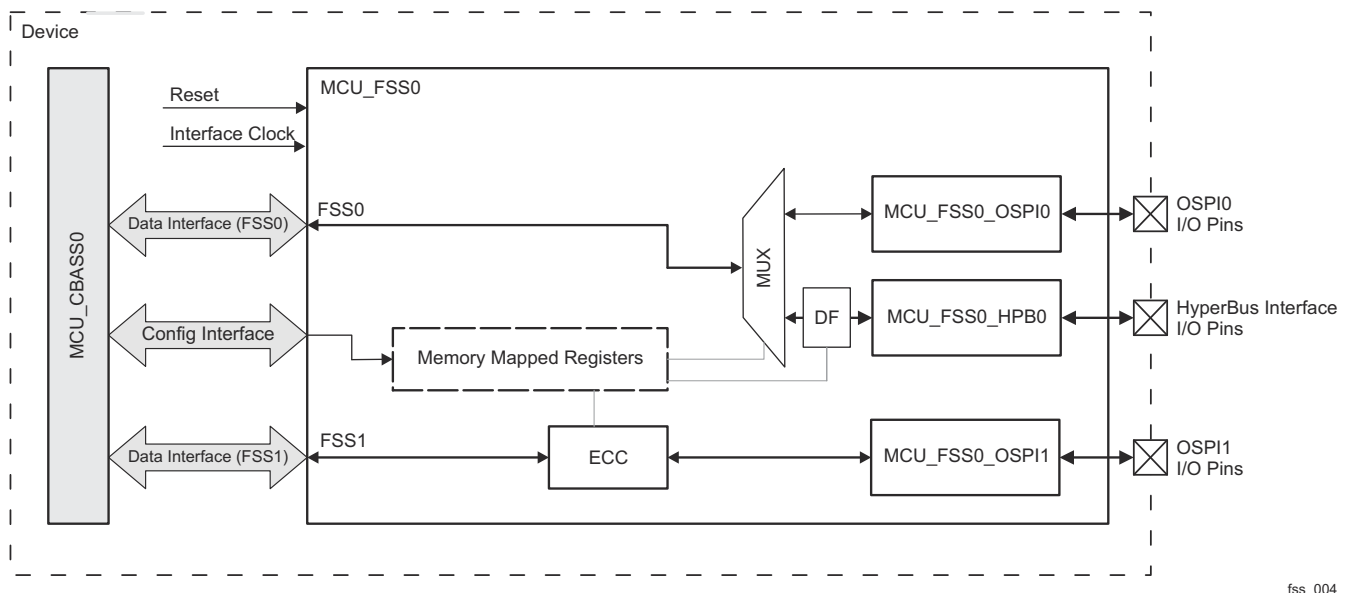
The FSS consists of two OSPIs and one HyperBus interface. There are two FSS paths - FSS0 and FSS1 (see [Figure 12-221](#)). The first path (FSS0) includes OSPI0 and HyperBus interface. The second path (FSS1) includes OSPI1. These two FSS paths can be used simultaneously (see *FSS Typical Application*).

[Table 12-289](#) shows the FSS allowed interface combinations.

**Table 12-289. FSS Allowed Interface Combinations**

Combination	Primary Interface (FSS0)	Secondary Interface (FSS1)
1.	OSPI0 (MCU_FSS0_OSPI0)	Not Used
2.	HyperBus Interface (MCU_FSS0_HPB0)	Not Used
3.	OSPI0 (MCU_FSS0_OSPI0)	OSPI1 (MCU_FSS0_OSPI1)
4.	HyperBus Interface (MCU_FSS0_HPB0)	OSPI1 (MCU_FSS0_OSPI1)
5.	Not Used	OSPI1 (MCU_FSS0_OSPI1)

[Figure 12-221](#) shows the FSS block diagram.



**Figure 12-221. FSS Block Diagram**

#### FSS Blocks:

- **MCU\_CBASS0:** The MCU\_CBASS0 interconnect allows FSS to communicate with the device modules and subsystems.
- **Data Interface (FSS0):** It is 64-bit data/32-bit address multi issue data interface with coherent in-band bypass. It provides accessibility to either the OSPI0 or HyperBus interface.
- **Data Interface (FSS1):** It is 64-bit data/32-bit address multi issue data interface with coherent in-band bypass. It provides accessibility to the OSPI1.
- **Config Interface:** It is used for configuration of the memory mapped registers within the FSS.
- **Interface Clock and Reset:**
  - For more information, see *MCU\_FSS0 Clocks and Resets*.
  - For more information, see *MCU\_FSS0\_OSPI Clocks and Resets*.

- For more information, see *MCU\_FSS0\_HPBO Clocks and Resets*.
- Memory Mapped Registers: This block conditionally includes registers from the following blocks: FSS, ECC, MUX, and DF. The configuration of these registers defines which combination of FSS interfaces is selected (see [Table 12-289](#)) and also which FSS features and operation modes are used. For more information, see *MCU\_FSS0\_SYSCONFIG*.
- MUX: The Muxing (MUX) block is used as a software controlled switch in the primary FSS (FSS0) interface. It defines which interface to be used (OSPI0 or HyperBus interface). The MUX block can switch only when the traffic is idle. The software has responsibility to cause the traffic to be stopped.
- DF: The Dynamic Fragmenter (DF) module is responsible for fragmenting write data to the flash region so that all writes to the flash region are done in 16-bits chunks (a requirement for HyperFlash). It passes all other transaction through unaffected.
- ECC: The Single Error Correction and Double Error Detection (SECEDED) mechanism is used. Each 32-byte block is protected by four SECEDED bytes. The secondary FSS1 interface includes a stand-alone ECC module which can be used for OSPI1 only. For more information, see *ECC Support*.
- FSS Interfaces:
  - *MCU\_FSS0\_OSPI0*: The OSPI0 is part of the primary FSS interface (FSS0). It can be configured to use ECC and/or OTFE module.
  - *MCU\_FSS0\_OSPI1*: The OSPI1 is part of the secondary FSS interface (FSS1). It can be configured with or without ECC.
  - *MCU\_FSS0\_HPBO*: The HyperBus interface is part of the primary FSS interface (FSS0). It can be configured to use ECC and/or OTFE module.
- FSS I/O Pins:
  - OSPI1 I/O Pins: All used *MCU\_FSS0\_OSPI1* interface pins (for more information, see *OSPI I/O Signals*).
  - HyperBus I/O Pins: All used *MCU\_FSS0\_HPBO* interface pins (for more information, see *HyperBus I/O Signals*).

#### 12.3.1.4.2 FSS ECC Support

The Error Correcting Code (ECC) is a mechanism for providing increased system reliability via reduction of memory software errors by allowing single bit errors to be detected and corrected (SEC) and double bit errors to be detected (DED).

For more information about ECC, refer to *ECC Aggregator*.

For FSS1 path when the stand-alone ECC module is used:

- The input for the ECC module is 32-byte data block.
- The ECC module generates ECC codes for the 32-byte data block.
- The ECC module packs data + ECC, reformats the address and sends this to the flash controller (OSPI1 flash controller).

For more information, see *ECC Calculation*.

##### 12.3.1.4.2.1 FSS ECC Calculation

The ECC calculation is generated or checked on the concatenated structure {BIT('1'),BLOCK\_ADDRESS[26-0],WORD1[127-0],WORD0[127-0]} broken into four 71-bit chunks stored in the ECC word. The ECC word has four 8-bits values. Each value checks a 71-bit segment of the concatenated structure.

Details:

- The 71-bit portion uses a standard 7-bit ECC calculation plus a parity bit (8-bit value).
- The WORD0[127-0] is the first 128-bit word of a block and the WORD1[127-0] is the second 128-bit word of the data block.
- The BLOCK\_ADDRESS[26-0] is the upper 27 bits of the requested byte address known as the block address.

- If the MCU\_FSS0\_SYSCONFIG[3] ECC\_DISABLE\_ADR bit is set, the BLOCK\_ADDRESS[26-0] is assumed zero so that it does not affect the ECC calculation.
- If the ECC detects a double error, the returned data is cleared to zero.
- Both SEC and DED are reported to the ECC registers.

For more information about ECC registers, see *FSS Registers*.

#### 12.3.1.4.3 FSS Modes of Operation

Table 12-289 presents the possible combinations of FSS interfaces.

Both paths (FSS0 and FSS1) support XIP and BC modes.

#### 12.3.1.4.4 FSS Read Operations

The ECC module can read any byte or block of bytes. If the region is ECC protected, the ECC module reads the block and returns the appropriate bytes requested. If not in an ECC protected region, only the requested bytes are read.

##### 12.3.1.4.4.1 OSPI Read Pipeline Mode

The OSPIs support a pipeline mode. In this mode if requests are supplied back to back, the OSPIs provide data at a higher rate. When consecutive blocks are read whether in or out of regions with ECC, the ECC module reads the entire block including the unused ECC word and only return the necessary portion so that the OSPI maintains the pipelining operation. Reading a single non ECC protected and non 32-byte request or non consecutive block causes the OSPI pipeline to flush. For more information, see *PHY Pipeline Mode*.

##### 12.3.1.4.5 FSS Memory Address Translation

The ECC module stores ECC in the target memory. This creates the need to translate the address from the requestor to the memory. The translation is fixed for a given memory. If the memory contains ECC, the entire memory will reserve an ECC word for each 32-byte block.

Table 12-290 defines the memory address equation for a given input address.

The configuration mode for a given memory is not ever expected to change once setup. Although regions could be changed, the fundamental modes are considered to be static.

**Table 12-290. Memory Address Equation**

ECC Mode	Authentication Mode	Memory Block Address	64 MB Blocks Available
No	No	Input Address/32×32	64
Yes	No	Input Address/32×36	56.8

If the memory is configured to have ECC, the total blocks are reduced by 11 %. That is 11 % of the flash is reserved for ECC data regardless if any regions for ECC are used.

#### 12.3.1.4.6 FSS0 and FSS1 Regions

Both FSS interfaces (from FSS0 and FSS1 paths) have three regions:

Both FSS interfaces (from FSS0 and FSS1 paths) have three regions:

- Region 0 is used for XIP, DMA read or DMA write of flash or RAM which contains ECC protected data.
- Region 1 is used for a smaller boot region of 64 MB or 128 MB. It is similar to Region 0, but can select any 64 MB or 128 MB block from Region 0.
- Region 3 is used to enable HyperBus programming (by writing flash key patterns). Region 3 can also be used to program the flash with ECC protected data and to clear ECC errors when the flash takes a bit hit (to clear a single bit error, it may be necessary to re-write an entire block of the flash). Region 3 can also be used to validate ECC block programming.

#### 12.3.1.4.6.1 FSS0 and FSS1 Regions Boot Size Configuration

The boot size for FSS0 (OSPI0/HyperBus interface) and FSS1 (OSPI1) defaults to 64 MB but can be configured to be 128 MB. Selection of boot block which will be used is also configurable. For more information see CTRLMMR\_MCU\_FSS\_CTRL register in *Control Module (CTRL\_MMR)*.

#### 12.3.1.4.7 FSS Memory Regions

Table 12-291 shows the FSS memory regions.

**Table 12-291. FSS Memory Regions**

Address Range		Size	Description
FSS0 (HyperBus interface or OSPI0)	FSS1 (OSPI1)		
0x04 0000 0000 to 0x04 FFFF FFFF	0x06 0000 0000 to 0x06 FFFF FFFF	4 GB	External Memory Space (Region 0)
0x00 5000 0000 to 0x00 57FF FFFF	0x00 5800 0000 to 0x00 5FFF FFFF	128 MB	Boot Space (Region 1)
0x05 0000 0000 to 0x05 FFFF FFFF	0x07 0000 0000 to 0x07 FFFF FFFF	4 GB	External Memory Space (Region 3)

### 12.3.1.5 FSS Programming Guide

#### 12.3.1.5.1 FSS Initialization Sequence

Initialization steps:

- Configure the main boot parameters for FSS0 or FSS1):
  - Select the boot block to be used.
  - Select the size of the boot block to be used.
- Enable FSS in PSC.
- Configure the FSS interface which will be used:
  - For FSS0 - OSPI0 or HyperBus interface.
  - For FSS1 - OSPI1.
- Enable the selected FSS interface in PSC.
- Configure the FSS interface:
  - For more information about OSPI configuration, please see *Octal Serial Peripheral Interface (OSPI)*.
  - For more information about HyperBus interface configuration, please see *HyperBus Interface*.
- Configure the ECC region start address and size:
  - MCU\_FSS0\_ECC\_RGSTRT\_j
  - MCU\_FSS0\_ECC\_RGSIZ\_j
- Enable the ECC option if the target flash device has ECC encoded within.
- If the target device is HyperFlash/SRAM configure fragmentation address boundary:
  - Flash device requires fragmentation.
  - SRAM device does not require fragmentation.
- Enable error interrupts (see MCU\_FSS0\_ENABLE\_SET).

#### 12.3.1.5.2 FSS Real-Time Operation

The CPU or DMA can read any location in the memory map and ECC will occur based on the region selection and size configuration.

In the event of an ECC single error detect, the 32-byte block address and associated error bits are stored and an interrupt is generated (if enabled). The CPU can then service the interrupt and determine the error type. If a single error occurs, the CPU can scrub the flash block to determine if the error is permanent and requires reprogramming.

In the event of an ECC double error detect, the 32-byte block address and associated error bits are stored and an interrupt is generated (if enabled). The CPU can then service the interrupt and determine the error type. If the double error detect is within the flash, the region is corrupt and have to be treated as unused.

For ECC double error detect the bus status will also be set. This prevents the CPU from executing the erroneous data.

#### **12.3.1.5.3 FSS Power Up/Down Sequence**

There are four PSC controls for the FSS: the FSS itself, OSPI0, HyperBus interface, and OSPI1. The CPU enables the appropriate interfaces before using FSS0 or FSS1 of the FSS.

The FSS0 can be used for HyperBus interface or OSPI0 and only the interface in use has to be enabled. Software should ensure the selected interface is enabled prior to FSS0 transactions.

The FSS1 can only be used to access OSPI1. The FSS1 transaction may be blocked depending on the power state of the OSPI1.

Normal Power Down Sequence:

- Block any new transaction to the particular interface - FSS0 or FSS1.
- Power down the target FSS interface.
- In the event when all targets are powered down, the FSS can then be powered down.

### 12.3.2 Octal Serial Peripheral Interface (OSPI)

This section describes the Octal Serial Peripheral Interface (OSPI) module for the device.

#### 12.3.2.1 OSPI Overview

The Octal Serial Peripheral Interface (OSPI) module is a kind of Serial Peripheral Interface (SPI) module which allows single, dual, quad, or octal read and write access to external flash devices.

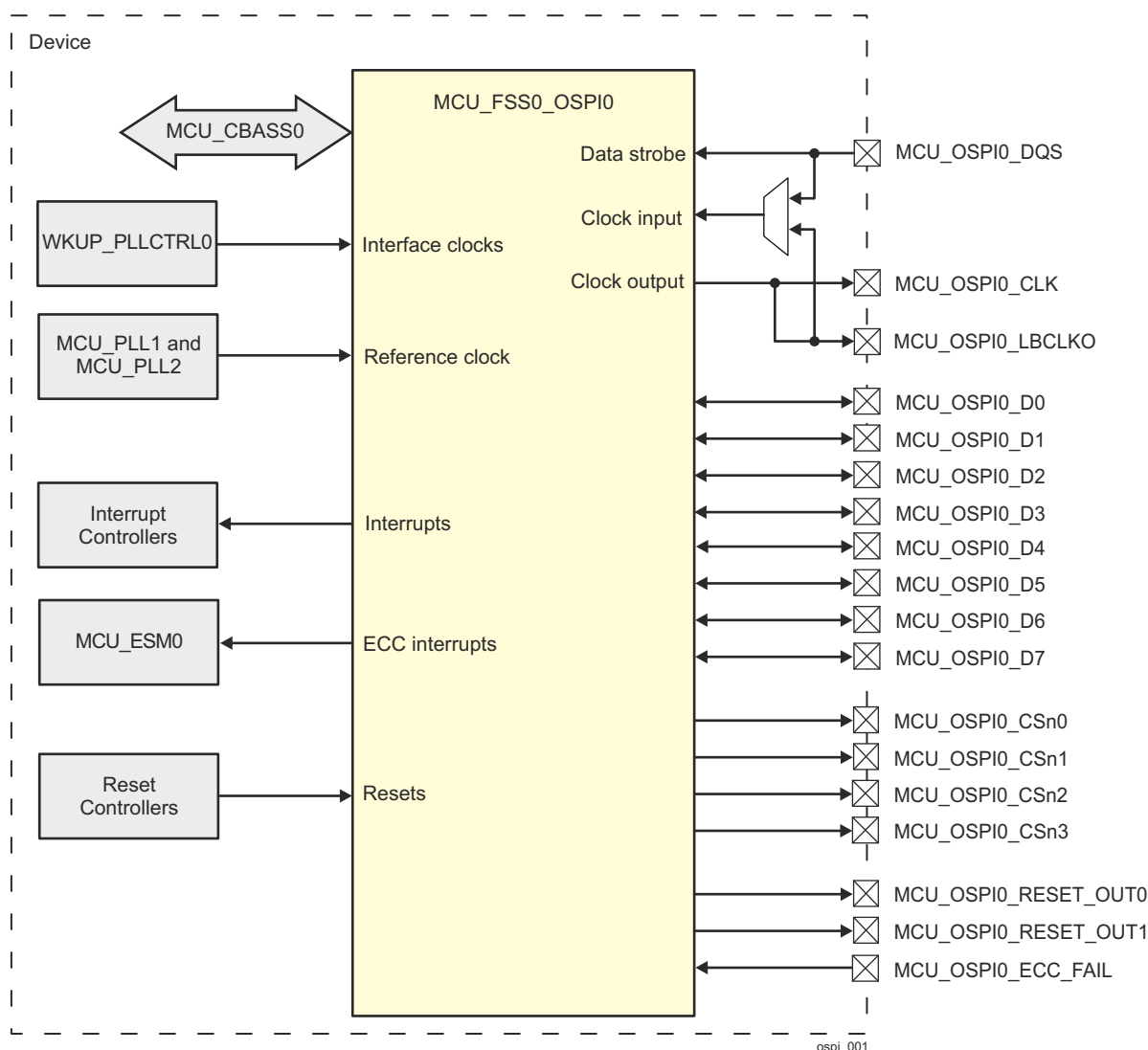
The OSPI module is used to transfer data, either in a memory mapped direct mode (for example a processor wishing to execute code directly from external flash memory), or in an indirect mode where the module is set-up to silently perform some requested operation, signaling completion via interrupts or status registers. For indirect operations, data is transferred between system memory and external flash memory via an internal SRAM which is loaded for writes and unloaded for reads by a device controller at low latency system speeds. Interrupts or status registers are used to identify the specific times at which this SRAM is accessed using user programmable configuration registers.

[Table 12-292](#) shows the OSPI allocation across device domains.

**Table 12-292. OSPI Allocation Across Device Domains**

Instance	Domain		
	WKUP	MCU	MAIN
MCU_FSS0_OSPI0	-	✓	-

[Figure 12-222](#) shows the OSPI module overview.



**Figure 12-222. OSPI Overview**

### 12.3.2.1.1 OSPI Features

The OSPI module has the following features:

- Support for single, dual, quad (QSPI mode) or octal I/O bus widths.
- Memory mapped 'direct' mode of operation for performing flash data transfers and executing code from flash memory.
- Software triggered 'indirect' mode of operation for performing low latency and non-processor intensive flash data transfers.
- Local SRAM of configurable size to reduce advanced high-performance bus overhead and buffer flash data during indirect transfers.
- Set of software advanced peripheral bus accessible flash control registers to perform any flash command, including data transfers up to 8-bytes at a time.
- Additional addressable memory bank to accommodate more than 8-bytes at a time.
- Support for XIP, sometimes referred to as continuous mode.
- Support for DDR Mode and DTR protocol (including Octal DDR protocol with DQS for Octal-SPI devices)
- Programmable device sizes.
- Programmable write protected regions to block system writes from taking effect.



- Programmable delays between transactions.
- Legacy mode allowing software direct access to low level transmit and receive FIFOs, bypassing the higher layer processes.
- An independent reference clock to decouple bus clock from SPI clock – allows slow system clocks.
- Programmable baud rate generator to generate OSPI clocks.
- Features included to improve high speed read data capture mechanism.
- Option to use adapted clocks or DQS to further improve read data capturing.
- Programmable interrupt generation.
- Up to four external device selects - OSPI and QSPI devices can be mixed
- Programmable data decoder, enables continuous addressing mode for each of the connected devices and auto-detection of boundaries between devices.
- Supports BOOT mode.
- Bidirectional CRC on Multiple-SPI interface.
- Handling ECC errors for flash devices with embedded correction engine.
- Full integration with PHY module dedicated to more flexible and power efficient transfers.
- Supports RESET\_OUT[1-0] and ECC\_FAIL pins for external flash devices where ECC is checked on the flash.
- Automatic Flash device status polling for programming operation (Auto HW Polling)

#### **12.3.2.1.2 OSPI Not Supported Features**

The following features are not supported on this family of devices:

- Pulse events not used.
- In Octal-SPI and Quad-SPI mode, Mode 1, 2, and 3 are not supported.
- Flash device status polling for programming operation using STIG

### 12.3.2.2 OSPI Environment

The FSS0\_OSPI0 module is hereinafter referred to as OSPI module.

This section describes the OSPI external connections (environment).

The OSPI module is primarily intended for fast booting from Octal- and Quad-SPI flash memories. [Figure 12-223](#) shows a typical connection of the OSPI module to an external Octal-SPI flash memory.

**Figure 12-223. OSPI Connected to an External Octal-SPI Flash Memory**

[Table 12-293](#) lists and describes the FSS0\_OSPI I/O signals.

**Table 12-293. OSPI I/O Signals**

Module Pin	Device Level Signal	I/O <sup>(1)</sup>	Description	Module Pin Reset Value <sup>(2)</sup>
<b>FSS0_OSPI0</b>				
DQ0	OSPI0_D0	IO	FSS0_OSPI0 data input/output 0	HiZ
DQ1	OSPI0_D1	IO	FSS0_OSPI0 data input/output 1	HiZ
DQ2	OSPI0_D2	IO	FSS0_OSPI0 data input/output 2	HiZ
DQ3	OSPI0_D3	IO	FSS0_OSPI0 data input/output 3	HiZ
DQ4	OSPI0_D4	IO	FSS0_OSPI0 data input/output 4	HiZ
DQ5	OSPI0_D5	IO	FSS0_OSPI0 data input/output 5	HiZ
DQ6	OSPI0_D6	IO	FSS0_OSPI0 data input/output 6	HiZ
DQ7	OSPI0_D7	IO	FSS0_OSPI0 data input/output 7	HiZ
N_SS_OUT0	OSPI0_CS <sub>n</sub> 0	O	FSS0_OSPI0 external flash device chip select 0	0x1
N_SS_OUT1	OSPI0_CS <sub>n</sub> 1	O	FSS0_OSPI0 external flash device chip select 1	0x1
N_SS_OUT2	OSPI0_CS <sub>n</sub> 2	O	FSS0_OSPI0 external flash device chip select 2	0x1
N_SS_OUT3	OSPI0_CS <sub>n</sub> 3	O	FSS0_OSPI0 external flash device chip select 3	0x1
OCLK	OSPI0_CLK	O	FSS0_OSPI0 clock output for the external flash device	0x0
	OSPI0_LBCLKO	O	FSS0_OSPI0 external loopback output	0x0
DQS	OSPI0_DQS	I <sup>(3)</sup>	FSS0_OSPI0 data strobe / external loopback input	Don't care
RESET_OUT0	OSPI0_RESET_OUT0	O <sup>(4)</sup>	FSS0_OSPI0 reset output 0 for the external flash device. Pin is active low.	0x1
RESET_OUT1	OSPI0_RESET_OUT1	O <sup>(4)</sup>	FSS0_OSPI0 reset output 1 for the external flash device. Pin is active low.	0x1
ECC_FAIL	OSPI0_ECC_FAIL	I	FSS0_OSPI0 ECC status from the external flash device	0x1

(1) I = Input; O = Output

(2) HiZ = High Impedance

(3) When used as an external loopback input, the DQS signal can alternatively be referred to as LBCLKI. The LBCLKI clock input signal is a looped back version of the LBCLKO clock output signal and facilitates easier timing closure at higher speeds. The loopback has to be at board level in order to support higher OSPI speeds. The source of the loopback clock is defined by CTRLMMR\_OSPI0\_CLKSEL[4] LOOPCLK\_SEL bit in *Control Module (CTRL\_MMR)*.

(4) When OSPI flash memory is not used for Boot, the RESET\_OUT[1:0] signals can be used to reset the OSPI flash. If OSPI flash is used for Boot, the flash needs to be reset in tandem with SoC PORz, thus the OSPI0\_RESET\_OUT[0:1] signal should not be used.

[Table 12-294](#) describes the OSPI I/O connectivity to external SPI devices.

**Table 12-294. OSPI I/O Connectivity to External SPI Devices**

Module Pin	I/O <sup>(1)</sup>	Description			
		4-pin <sup>(1)</sup> SPI - Single Read/Write (SIO) (DATA_XFER_TYPE_EXT_MODE_FLD=0x0)	4-pin <sup>(1)</sup> SPI - Dual Read/Write (DATA_XFER_TYPE_EXT_MODE_FLD=0x1)	6-pin <sup>(1)</sup> SPI - Quad Read/Write (DATA_XFER_TYPE_EXT_MODE_FLD=0x2)	11-pin <sup>(1)</sup> SPI - Octal Read/Write (DATA_XFER_TYPE_EXT_MODE_FLD=0x3)
DQ0	IO	Used as SPI data output	Used as SPI data input 0 Used as SPI data output 0	Used as SPI data input 0 Used as SPI data output 0	Used as SPI data input 0 Used as SPI data output 0

**Table 12-294. OSPI I/O Connectivity to External SPI Devices (continued)**

Module Pin	I/O <sup>(1)</sup>	Description			
		4-pin <sup>(1)</sup> SPI - Single Read/Write (SIO) (DATA_XFER_TYPE_EXT_MODE_FLD=0x0)	4-pin <sup>(1)</sup> SPI - Dual Read/Write (DATA_XFER_TYPE_EXT_MODE_FLD=0x1)	6-pin <sup>(1)</sup> SPI - Quad Read/Write (DATA_XFER_TYPE_EXT_MODE_FLD=0x2)	11-pin <sup>(1)</sup> SPI - Octal Read/Write (DATA_XFER_TYPE_EXT_MODE_FLD=0x3)
DQ1	IO	Used as SPI data input	Used as SPI data input 1 Used as SPI data output 1	Used as SPI data input 1 Used as SPI data output 1	Used as SPI data input 1 Used as SPI data output 1
DQ2	IO	Not used	Not used	Used as SPI data input 2 Used as SPI data output 2	Used as SPI data input 2 Used as SPI data output 2
DQ3	IO	Not used	Not used	Used as SPI data input 3 Used as SPI data output 3	Used as SPI data input 3 Used as SPI data output 3
DQ4	IO	Not used	Not used	Not used	Used as SPI data input 4 Used as SPI data output 4
DQ5	IO	Not used	Not used	Not used	Used as SPI data input 5 Used as SPI data output 5
DQ6	IO	Not used	Not used	Not used	Used as SPI data input 6 Used as SPI data output 6
DQ7	IO	Not used	Not used	Not used	Used as SPI data input 7 Used as SPI data output 7
DQS	I <sup>(2)</sup>	Not used	Not used	Not used	Data strobe or loopback clock input
OCLK	O	Output clock or loopback clock output. For more information, see <a href="#">Table 12-293</a> .			
N_SS_OUT0	O	External SPI device chip-select 0			
N_SS_OUT1	O	External SPI device chip-select 1			
N_SS_OUT2	O	External SPI device chip-select 2			
N_SS_OUT3	O	External SPI device chip-select 3			
RESET_OUT0	O	External SPI device reset 0. Pin is active low.			
RESET_OUT1	O	External SPI device reset 1. Pin is active low.			
ECC_FAIL	I	External SPI device ECC failure indication			

(1) This is the pin count at the external SPI flash memory side.

(2) When used as an external loopback input, the DQS signal can alternatively be referred to as LBCLKI. The LBCLKI clock input signal is a looped back version of the LBCLKO clock output signal and facilitates easier timing closure at higher speeds. The loopback has to be at board level in order to support higher OSPI speeds. The source of the loopback clock is defined by CTRLMMR\_OSPI0\_CLKSEL[4] LOOPCLK\_SEL bit in *Control Module (CTRL\_MMR)*.

#### Note

For OSPI0\_CLK, OSPI0\_LBCLKO, and OSPI0\_DQS signals to work properly, the RXACTIVE bit of the appropriate registers should be set to 0x1 because of retiming purposes.

#### Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

### 12.3.2.3 OSPI Integration

This section describes module integration in the device, including information about clocks, resets, and hardware requests.

### 12.3.2.3.1 OSPI Integration in MCU Domain

There is one OSPI module integrated in the device domain - FSS0\_OSPI0. [Figure 12-224](#) shows its integration in the device.

**Figure 12-224. FSS0\_OSPI Integration**

[Table 12-295](#) through [Table 12-297](#) summarize the integration of FSS0\_OSPI0 in device domain.

**Table 12-295. FSS0\_OSPI Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
FSS0_OSPI0	PSC0	PD0		CBASS0

**Table 12-296. MCU\_FSS0\_OSPI Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
MCU_FSS0_OSPI0	OSPI0_HCLK	MCU_SYSCLK0/3	WKUP_PLLCTRL0	MCU_FSS0_OSPI0 data transfer clock
	OSPI0_PCLK	MCU_SYSCLK0/3	WKUP_PLLCTRL0	MCU_FSS0_OSPI0 configuration clock
	OSPI0_RCLK	MCU_PLL1_HSDIV4_C LKOUT	MCU_PLL1_HSDIV4	MCU_FSS0_OSPI0 Reference clock. Mux controlled by CTRLMMR_MCU_OSPI0_CLKSEL[0] CLK_SEL in <i>Control Module (CTRL_MMR)</i>
		MCU_PLL2_HSDIV4_C LKOUT	MCU_PLL2_HSDIV4	
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
MCU_FSS0_OSPI0	MCU_FSS0_OSPI0_RST	MOD_G_RST	LPSC10	MCU_FSS0_OSPI0 reset

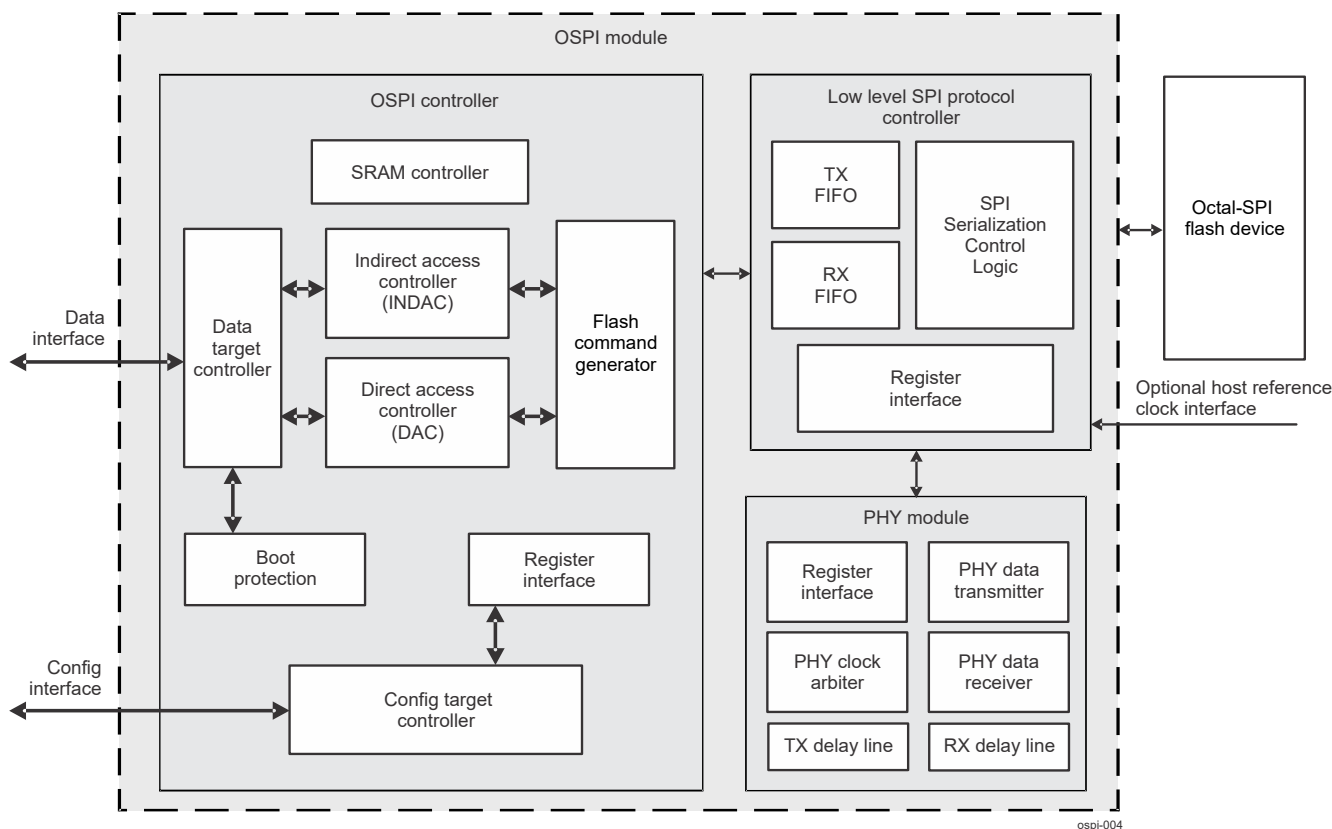
**Table 12-297. FSS0\_OSPI Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
	FSS0_OSPI0_OSPI_ECC_CORR_LVL_INTR_0		ESM0	FSS0_OSPI0 ECC Aggregator correctable error interrupt	Level
	FSS0_OSPI0_OSPI_ECC_UNCORR_LVL_INTR_0		ESM0	FSS0_OSPI0 ECC Aggregator uncorrectable error interrupt	Level

### 12.3.2.4 OSPI Functional Description

#### 12.3.2.4.1 OSPI Block Diagram

Figure 12-225 shows the OSPI module block diagram.



**Figure 12-225. OSPI Block Diagram**

The OSPI module is composed of three main blocks. The first one is the OSPI controller, the second one is the low level SPI protocol controller, and the third one is the integrated PHY.

The OSPI module has the following two target interfaces:

- Data target interface intended for data transfer.
- Configuration target interface intended for accessing the programmable set of registers.

##### 12.3.2.4.1.1 Data Target Interface

The data interface is used for data transfer to external flash devices in direct and indirect mode of operation. The data target controller validates incoming data accesses, responds to invalid requests, performs any required byte and halfword reordering, blocks writes that violate the programmed write protection rules (only for direct access) and forwards the transfer request to either the direct access controller (DAC) or the indirect access controller (INDAC).

The data interface bus is 32-bits wide. Therefore only byte, halfword and word accesses are permitted. When the controller is configured to work in SPI Octal DDR Mode or Octal DDR Protocol (where 2 bytes are collected within single SPI clock cycle what exceeds the size of 1 byte transfer request), 8 bit transfer size is not allowed.

#### Note

Cache line wrap accesses over the data target port should be word aligned.

Data target port doesn't support cache line wrap bursts of 128 bytes.

#### 12.3.2.4.1.2 Configuration Target Interface

The configuration interface is used to configure the OSPI module and perform software controlled flash accesses using the OSPI\_FLASH\_CMD\_CTRL\_REG register (for more information refer to [Section 12.3.2.4.11, Software Triggered Instruction Generator \(STIG\)](#)). Depending on the address it routes the incoming interconnect transfer to the Low level SPI protocol controller or to the ECC aggregator. The configuration port is also used to interact with the OSPI configuration and SRAM ECC registers.

#### Note

The configuration interface supports only 32-bit accesses. For single byte or halfword manipulations software should perform read-modify-write operations.

#### 12.3.2.4.1.3 OSPI Clock Domains

The OSPI module has two main clock sources for the Octal-SPI controller.

- For interface clocks
- For reference clock

The source for the interface clocks corresponds to the configuration and data buses. The data bus clock (OSPI\_HCLK) is the main system clock used to transfer data over the data bus between a controller on the system interconnect and the OSPI module. The data bus clock also drives the internal OSPI SRAM. The configuration bus clock (OSPI\_PCLK) is used to access the OSPI configuration register and perform basic configuration and for interrupt handling. The OSPI reference clock (OSPI\_RCLK) drives the SPI transmit and receive logic in the OSPI module. It is also used to generate the output SPI protocol clock (OSPI\_OCLK) and for oversampling of the input data. Using the reference clock (OSPI\_RCLK) allows the OSPI module to decouple the frequency of the SPI flash device from the device system clocks, thereby providing more flexible clocking solution.

#### Note

There is no particular clock ratio requirement between configuration (OSPI\_PCLK) and data bus (OSPI\_HCLK) clocks.

#### 12.3.2.4.2 OSPI Modes

#### Note

Some of the OSPI features described in this section may not be supported on this family of devices. For more information, see *OSPI Not Supported Features*.

The OSPI module supports four SPI modes. These modes are defined through the OSPI\_CONFIG\_REG[1] SEL\_CLK\_POL\_FLD and OSPI\_CONFIG\_REG[2] SEL\_CLK\_PHASE\_FLD bits. The SEL\_CLK\_POL\_FLD bit defines the clock polarity and the SEL\_CLK\_PHASE\_FLD bit defines the data launch and data capture relation to the OSPI clock edges. [Table 12-298](#) gives a brief description of these modes.

**Table 12-298. OSPI Modes**

SPI Mode	SEL_CLK_POL_FLD	SEL_CLK_PHASE_FLD	Description
0	0	0	Clock inactive state: low Data launch edge: clock falling edge Data capture edge: clock rising edge
1	0	1	Clock inactive state: low Data launch edge: clock rising edge Data capture edge: clock falling edge

**Table 12-298. OSPI Modes (continued)**

SPI Mode	SEL_CLK_POL_FLD	SEL_CLK_PHASE_FLD	Description
2	1	0	Clock inactive state: high
			Data launch edge: clock rising edge
			Data capture edge: clock falling edge
3	1	1	Clock inactive state: high
			Data launch edge: clock falling edge
			Data capture edge: clock rising edge

Octal flash devices provide DQS signal which allows source synchronous capture, but for Quad flash devices the OSPI module has a loopback mode. In this loopback mode the clock, looped back at board level, is used for registering the input data, and the edge used is same as the launch edge, thus giving a full cycle path (for more information, see [Section 12.3.2.4.2.1, Read Data Capture](#)).

#### 12.3.2.4.2.1 Read Data Capture

[Figure 12-226](#) shows the Read Data Capture Logic in the OSPI module.

**Figure 12-226. Read Data Capture Logic**

The PHY module includes a DLL which allows adjustment of the sampling edge with respect to the incoming data to achieve maximum frequency. There are three sources for the sampling signal:

- The reference clock
- Output SPI clock external loopback
- The DQS (only available in Octal Flash devices)

The loopback mode (only for Quad flash devices) can work in two cases. The first one is when OSPI\_CONFIG\_REG[2] SEL\_CLK\_PHASE\_FLD=0. When SEL\_CLK\_PHASE\_FLD=1 there aren't enough clock falling edges for the register pipeline to catch the last data driven, thus causing a functional failure. Additionally, since the capture edge is falling edge, it gives a full cycle input path only in SPI mode 0, that is when SEL\_CLK\_POL\_FLD=0 and SEL\_CLK\_PHASE\_FLD=0. Thus SPI mode 0 is the first of two modes that support high MHz operation (greater than 50 MHz). The second mode is when SEL\_CLK\_PHASE\_FLD=1 and SEL\_CLK\_PHASE\_FLD=1 (SPI mode 3). In this case the missing clock falling edge is compensated inside the OSPI controller when using the incorporated PHY module by inverting the loopback clock.

The loopback mode is enabled by writing 0x0 to OSPI\_RD\_DATA\_CAPTURE\_REG[0] BYPASS\_FLD. The taps are selected by programming OSPI\_RD\_DATA\_CAPTURE\_REG[4-1] DELAY\_FLD field. The taps delay the read data capturing logic by the programmed number of OSPI\_RCLK cycles.

#### 12.3.2.4.2.1.1 Mechanisms of Data Capturing

There are two mechanisms of data capturing in the OSPI module. They can be combined in some parts to ensure reliable sampling solution independent on the system requirements and the controller configuration. The mechanisms are as follows:

- Data capturing mechanism using taps
- Data capturing mechanism using PHY module.

#### 12.3.2.4.2.1.2 Data Capturing Mechanism Using Taps

This section describes the data capturing mechanism where sampling point is adjusted for one of the reference clock edges inside divided OSPI clock.

After POR, the adapted loopback clock circuit and the OSPI\_RCLK delay register line both wake in a disabled state. The OSPI\_RD\_DATA\_CAPTURE\_REG register provides the control for the mechanism using taps.

OSPI\_RD\_DATA\_CAPTURE\_REG[5] SAMPLE\_EDGE\_SEL\_FLD bit selects the edge of the reference clock, on which data outputs from flash memory are sampled.



OSPI\_RD\_DATA\_CAPTURE\_REG[4-1] DELAY\_FLD bit field controls the additional number of read data capture cycles (this is the fast reference clock, running at least x4 of the device clock) that should be applied to the internal read data capture circuit. The large clock-to-out delay of the flash memory together with trace delays as well as other device delays may impose a maximum flash clock frequency which is less than the flash memory device itself can operate at. To compensate, software shall set this register to a value that guarantees robust data captures.

#### **12.3.2.4.2.1.3 Data Capturing Mechanism Using PHY Module**

PHY module is responsible for data capturing. More detailed description of all internal PHY sampling mechanisms is included in [Section 12.3.2.4.16.2, Read Data Capturing by the PHY Module](#).

#### **12.3.2.4.2.2 External Pull Down on DQS**

Per the OSPI protocol, the FLASH device drives DQS while CS is asserted. When CS is not asserted the FLASH device presents HiZ on DQS. When configured to use DQS, the controller uses the DQS as a clock, which samples the incoming data into a FIFO. Noise on the DQS when it is HiZ can cause spurious false triggering of the FIFO and filling it with invalid data. There is no way to clear this data except to reset the OSPI module.

To avoid this issue, it is recommended to add a pull down on the DQS line.

During device wakeup, before the IO ring is configured properly, the CS to the FLASH device is HiZ. Depending on the actual level of the CS line the FLASH device might drive the DQS High, Low or HiZ. A pull down on DQS forces the DQS input to Low, but the DQS might still be High or in the presence of noise there might be transitions between Low and High. This again can cause the same issue of capturing garbage data in the Controller FIFO.

To avoid this issue it is recommended to release the OSPI from reset only after the IO ring is configured properly.

#### **12.3.2.4.3 OSPI Power Management**

##### **Note**

The OSPI module does not provide any hardware signal for busy or idle status. Software need to ensure that the OSPI module is idle before clocks can be shut off by reading the OSPI\_CONFIG\_REG[31] IDLE\_FLD bit.

OSPI\_PCLK and OSPI\_HCLK share the same clock stop request/acknowledge and clock enable/acknowledge interface.

#### **12.3.2.4.4 Auto HW Polling**

The OSPI controller is capable of automatically testing the Flash device busy bit to guarantee no reads or writes are ignored by the flash when it is busy burning in programmed data.

At the end of a programming transaction, the Flash device goes into a burn-in state and becomes busy.

When Auto HW Polling is enabled, the OSPI controller keeps track of programming transactions and will initiate a Flash status read polling transactions automatically, until Flash indicates it is not busy, before any additional data read or programming operations are sent to the flash device. See OSPI\_WRITE\_COMPLETION\_CTRL\_REG register and the associated registers.

The OSPI controller requires that the OSPI\_WRITE\_COMPLETION\_CTRL\_REG[23-16] POLL\_COUNT\_FLD field should always be set with values greater or equal to 3 ( $\geq 3$ ).

#### **12.3.2.4.5 Flash Reset**

OSPI provides Flash reset out ports. These ports are active low and controlled thru OSPI\_CONFIG\_REG register.

The controller provide Flash reset out ports. These ports are active low and controlled thru the Config\_reg(0x0)[5].reset\_cfg\_fld and cofnig\_reg(0x0)[6].reset\_pin\_fld.

The register fields control the `ospi_reset_out[3:0]` mentioned in section 7.1.24 of the functional spec.

#### 12.3.2.4.6 OSPI Memory Regions

Table 12-299 shows the OSPI memory map in MCU domain.

**Table 12-299. OSPI Memory Map**

Address Range	Size	Description
<b>MCU_FSS0_OSPI0</b>		
0x00 4704 0000 to 0x00 4704 0100	256 B	Configuration registers space
0x00 4704 4000 to 0x00 4704 4200	512 B	Global control registers space
0x00 4706 8000 to 0x00 4706 8400	1 KB	OSPI_ECC_AGGR registers space

#### Note

For more information about the memory space, see *FSS Memory Regions*.

#### 12.3.2.4.7 OSPI Interrupt Requests

The OSPI module generates three interrupts. The ECC interrupts (`FSS0_OSPI_0_OSPI_ECC_CORR_LVL_INTR_0` and `FSS0_OSPI_0_OSPI_ECC_UNCORR_LVL_INTR_0`) are generated by the OSPI ECC aggregator.

The other interrupt (`FSS0_OSPI_0_OSPI_LVL_INTR_0`) is generated by the OSPI module.

Table 12-300 lists the event flags and the corresponding mask bits of the sources which can cause interrupts.

**Table 12-300. OSPI Events**

Event Flag	Event Mask	Description
OSPI_IRQ_STATUS_REG[0] MODE_M_FAIL_FLD	OSPI_IRQ_MASK_REG[0] MODE_M_FAIL_MASK_FLD	Event Flag and Event Mask for the OSPI Interrupts.
OSPI_IRQ_STATUS_REG[1] UNDERFLOW_DET_FLD	OSPI_IRQ_MASK_REG[1] UNDERFLOW_DET_MASK_FLD	
OSPI_IRQ_STATUS_REG[2] INDIRECT_OP_DONE_FLD	OSPI_IRQ_MASK_REG[2] INDIRECT_OP_DONE_MASK_FLD	
OSPI_IRQ_STATUS_REG[3] INDIRECT_READ_REJECT_FLD	OSPI_IRQ_MASK_REG[3] INDIRECT_READ_REJECT_MASK_FLD	
OSPI_IRQ_STATUS_REG[4] PROT_WR_ATTEMPT_FLD	OSPI_IRQ_MASK_REG[4] PROT_WR_ATTEMPT_MASK_FLD	
OSPI_IRQ_STATUS_REG[5] ILLEGAL_ACCESS_DET_FLD	OSPI_IRQ_MASK_REG[5] ILLEGAL_ACCESS_DET_MASK_FLD	
OSPI_IRQ_STATUS_REG[6] INDIRECT_XFER_LEVEL_BREACH_FLD	OSPI_IRQ_MASK_REG[6] INDIRECT_XFER_LEVEL_BREACH_MASK_FLD	
OSPI_IRQ_STATUS_REG[7] RCV_OVERFLOW_FLD	OSPI_IRQ_MASK_REG[7] RCV_OVERFLOW_MASK_FLD	
OSPI_IRQ_STATUS_REG[8] TX_FIFO_NOT_FULL_FLD	OSPI_IRQ_MASK_REG[8] TX_FIFO_NOT_FULL_MASK_FLD	
OSPI_IRQ_STATUS_REG[9] TX_FIFO_FULL_FLD	OSPI_IRQ_MASK_REG[9] TX_FIFO_FULL_MASK_FLD	
OSPI_IRQ_STATUS_REG[10] RX_FIFO_NOT_EMPTY_FLD	OSPI_IRQ_MASK_REG[10] RX_FIFO_NOT_EMPTY_MASK_FLD	
OSPI_IRQ_STATUS_REG[11] RX_FIFO_FULL_FLD	OSPI_IRQ_MASK_REG[11] RX_FIFO_FULL_MASK_FLD	
OSPI_IRQ_STATUS_REG[12] INDRD_SRAM_FULL_FLD	OSPI_IRQ_MASK_REG[12] INDRD_SRAM_FULL_MASK_FLD	
OSPI_IRQ_STATUS_REG[13] POLL_EXP_INT_FLD	OSPI_IRQ_MASK_REG[13] POLL_EXP_INT_MASK_FLD	
OSPI_IRQ_STATUS_REG[14] STIG_REQ_INT_FLD	OSPI_IRQ_MASK_REG[14] STIG_REQ_MASK_FLD	
OSPI_IRQ_STATUS_REG[16] RX_CRC_DATA_ERR_FLD	OSPI_IRQ_MASK_REG[16] RX_CRC_DATA_ERR_MASK_FLD	
OSPI_IRQ_STATUS_REG[17] RX_CRC_DATA_VAL_FLD	OSPI_IRQ_MASK_REG[17] RX_CRC_DATA_VAL_MASK_FLD	
OSPI_IRQ_STATUS_REG[18] TX_CRC_CHUNK_BRK_FLD	OSPI_IRQ_MASK_REG[18] TX_CRC_CHUNK_BRK_MASK_FLD	
OSPI_IRQ_STATUS_REG[19] ECC_FAIL_FLD	OSPI_IRQ_MASK_REG[19] ECC_FAIL_MASK_FLD	

**Table 12-300. OSPI Events (continued)**

Event Flag	Event Mask	Description
OSPI_ECC_DED_STATUS_REG0[0] SRAM_PEND	OSPI_ECC_DED_ENABLE_SET_REG0[0] SRAM_ENABLE_SET OSPI_ECC_SEC_ENABLE_CLR_REG0[0] SRAM_ENABLE_CLR	Event Flag and Event Mask for the ECC Interrupts.
OSPI_ECC_DED_STATUS_REG0[0] SRAM_PEND	OSPI_ECC_DED_ENABLE_SET_REG0[0] SRAM_ENABLE_SET OSPI_ECC_SEC_ENABLE_CLR_REG0[0] SRAM_ENABLE_CLR	
OSPI_ECC_AGGR_STATUS_SET[1-0] PARITY	OSPI_ECC_AGGR_ENABLE_SET[0] PARITY	
OSPI_ECC_AGGR_STATUS_SET[3-2] TIMEOUT	OSPI_ECC_AGGR_ENABLE_SET[1] TIMEOUT	
OSPI_ECC_AGGR_STATUS_CLR[1-0] PARITY	OSPI_ECC_AGGR_ENABLE_CLR[0] PARITY	
OSPI_ECC_AGGR_STATUS_CLR[3-2] TIMEOUT	OSPI_ECC_AGGR_ENABLE_CLR[1] TIMEOUT	

#### 12.3.2.4.8 OSPI Data Interface

##### 12.3.2.4.8.1 Data Interface Address Remapping

The incoming data interface address, by default, maps directly to the address sent serially to the FLASH device. If the FLASH device has a 24-bit address, then the 24 LSB's of the data address is forwarded. A remap feature is available to remap all incoming data addresses to ADDRESS + N, where N is the value stored in the OSPI\_REMAP\_ADDR\_REG[31-0] VALUE\_FLD bit field. It is enabled via the OSPI\_CONFIG\_REG[16] ENB\_AHB\_ADDR\_REMAP\_FLD bit. This feature could be used when software needs to move boot code to another FLASH region.

##### 12.3.2.4.8.2 Write Protection

In order to protect the FLASH device, a software controlled write protection feature is supported. Any data write detected (by using DAC), pointing to an area of the FLASH that is protected, is not permitted.

A programmable region of the FLASH device, defined as a number of FLASH 'blocks' starting from a particular block number can be protected. Three programmable registers are provided. The first OSPI\_LOWER\_WR\_PROT\_REG register defines the FLASH block that is located at the bottom of the region to be protected. The second OSPI\_UPPER\_WR\_PROT\_REG register defines the FLASH block that is located at the top of the region to be protected. The third OSPI\_WR\_PROT\_CTRL\_REG register is a control register consisting of 2 bits. The OSPI\_WR\_PROT\_CTRL\_REG[0] INV\_FLD bit allows software to invert the region that is being protected, causing the programmed region to become the only areas of FLASH memory that is not protected from writes. The OSPI\_WR\_PROT\_CTRL\_REG[1] ENB\_FLD bit is the write protection enable bit. When this bit is set to 0, the FLASH device is unprotected.

For implementation, the data interface must map the incoming address into its associated FLASH block. A block can be between 1 and 65 KB, programmed via the OSPI\_DEV\_SIZE\_CONFIG\_REG register.

##### 12.3.2.4.8.3 Access Forwarding

For legal accesses, the data interface will forward all accesses to one of two access controllers - the direct access and the indirect access controllers. Assuming DAC has been enabled via the OSPI\_CONFIG\_REG[7] ENB\_DIR\_ACC\_CTRL\_FLD bit, then by default all accesses will be forwarded to this controller. Before any accesses can be forwarded to INDAC, it must first be configured by software. This process is fully explained in [Section 12.3.2.4.10, Indirect Controller \(INDAC\)](#). If DAC is disabled, any incoming access that cannot be

forwarded to INDAC will be completed immediately with an error. If DAC is enabled, the same access will be forwarded and serviced by DAC.

#### **12.3.2.4.9 OSPI Direct Access Controller (DAC)**

Direct access refers to the operation where data interface accesses directly trigger a read or write to FLASH memory. It is memory mapped and can be used to both access and directly execute code from external FLASH memory. Any incoming access that is not recognized as being within the programmable indirect trigger region is assumed to be a direct access and will be serviced by the DAC. Note that accesses that use DAC do not use the embedded SRAM. The data transfer stops when read or write burst is carried out. The amount of wait states applied will be dependent on the latency through the controller. Latency is kept to a minimum when the use of XIP read instructions are enabled (see OSPI\_CONFIG\_REG[18] ENTER\_XIP\_MODE\_IMM\_FLD and OSPI\_CONFIG\_REG[17] ENTER\_XIP\_MODE\_FLD bits).

#### **12.3.2.4.10 OSPI Indirect Access Controller (INDAC)**

##### **12.3.2.4.10.1 Indirect Read Controller**

The aim of the indirect mode of operation is to read significant numbers of bytes from FLASH memory without requiring a data interface access to trigger it. Instead indirect operations are controlled and triggered by software via specific control/configuration Indirect Read Transfer registers (OSPI\_INDIRECT\_READ\_XFER\_CTRL\_REG, OSPI\_INDIRECT\_READ\_XFER\_WATERMARK\_REG, OSPI\_INDIRECT\_READ\_XFER\_START\_REG, and OSPI\_INDIRECT\_READ\_XFER\_NUM\_BYTES\_REG). This block will communicate with an embedded low level SPI protocol state machine module to perform an efficient and optimized FLASH read burst, placing the read data into the local SRAM module ready for fast and low latency delivery to any external controller.

By default, the Indirect Read controller is disabled. Before enabling it, software must configure how much data is required and the start address. The start address and total number of bytes to be fetched is defined in OSPI\_INDIRECT\_READ\_XFER\_START\_REG and OSPI\_INDIRECT\_READ\_XFER\_NUM\_BYTES\_REG registers, respectively. Up to two indirect operations can be programmed at any one time. The second operation can be triggered while the first is in progress. Supporting two indirect operations allows a short turnaround time between the completion of one indirect operation and the start of the second. For more information refer to [Section 12.3.2.4.10.3, Indirect Access Queuing](#).

The total number of bytes to read in an indirect operation is not limited by the size of the SRAM. The size of SRAM will only limit the size of requests. In the case of SRAM overrun, the controller will back pressure FLASH reads until space becomes available in the SRAM. Back pressuring the reads on the SPI interface is handled by completing any current read burst, waiting until space in the SRAM becomes available and then issuing a new read burst at the address where the previous terminated burst ended.

An external controller will be able to fetch the data that the controller has read from external FLASH memory by issuing data interface reads to the OSPI module. The address of the incoming read access must be in the range of indirect trigger address programmed via the OSPI\_IND\_AHB\_ADDR\_TRIGGER\_REG register to indirect trigger address +  $2^{**}(\text{indirect trigger address range}) - 1$ . Default value of the range is equal to 16 locations. This allows a 16-beat burst to be applied starting from the indirect trigger address. The smaller bursts are possible to handle effectively as well with this approach. Furthermore it is not strict requirement to push consecutive address sequence. Actual address just has to be in the indirect range to grant SRAM as source. Each valid Indirect Read will cause the internal SRAM to be popped, thereby decoupling the incoming read access address from the FLASH address – that is not direct mapped. Therefore the indirect trigger address does not have any relationship with the FLASH address. It is just to indicate that data should take the SRAM as source instead of the FLASH memory array after triggering of any valid Indirect Read. The FLASH address for Indirect Read is taken from the OSPI\_INDIRECT\_READ\_XFER\_START\_REG register. Assuming the requested data is present in the SRAM at the point the data interface access is received by the OSPI module, then the data will be fetched from the SRAM and the response to the read burst will be achieved with minimum latency. Once the data has been read from the SRAM, the OSPI module will free up the associated resource in the SRAM.

If a read access is received whose address is not within the range described above then that access will not be completed using the indirect controller. It will instead be serviced by the direct access controller.

If a read access is received whose address is within the range described above but the requested data is not immediately present in the SRAM then wait states will be applied until the data has been read from FLASH and pushed to the SRAM.

If a read burst is received whose access elements traverse the Indirect trigger range, then the accesses within the Indirect trigger range will be processed by the indirect controller and the rest will be taken by the direct access controller. This is likely to be a software configuration error.

The external controller is only permitted to issue 32-bit data interface reads until the last word of an indirect transfer. This helps keep the SRAM control logic less complex. On the final read, the external controller may issue a 16-bit (Halfword) or byte access to complete the transfer. It is also permitted for the external controller to always issue a 32-bit Word read on the last indirect access. The controller will pad the upper bits of the response with zero. The current expectation is that the SRAM will be kept fairly full while the read operation is carried out. The fill level of the SRAM is directly readable by software reading the OSPI\_SRAM\_FILL\_REG register.

An indirect operation may be cancelled at any time by setting 1 to OSPI\_INDIRECT\_READ\_XFER\_CTRL\_REG[1] CANCEL\_FLD bit.

Any bus controller should be allowed to initiate an indirect access. The OSPI module provide software access mechanism to the SRAM fill-level directly via configuration registers and then decide for itself when the data should be fetched from the local SRAM. The fill level watermark register (see OSPI\_INDIRECT\_READ\_XFER\_WATERMARK\_REG register) is provided. When the SRAM fill level passes this watermark, an interrupt is generated. If the watermark value is > 0, the watermark interrupt is also generated when the final byte of data has been read by the OSPI module and placed in the SRAM, even if the actual SRAM fill level has not risen above the watermark. This last feature is useful to avoid software tracking how much data has been read and resetting the watermark value for the last few bytes of an indirect read transfer.

Two further interrupt sources are provided to help understand the status of an indirect operation. Firstly, an interrupt is generated when an indirect operation has completed. Secondly, an interrupt is generated if an Indirect Read operation was requested but could not be accepted due to the fact 2 indirect operations have already been buffered by the OSPI module.

Setting the OSPI\_INDIRECT\_READ\_XFER\_CTRL\_REG[0] START\_FLD bit starts an indirect read operation. OSPI\_INDIRECT\_READ\_XFER\_CTRL\_REG[2] RD\_STATUS\_FLD bit is available to check the status.

#### 12.3.2.4.10.1 Indirect Read Transfer Process

The following sequence can be followed:

1. Setup OSPI\_CONFIG\_REG register.
2. Setup the indirect transfer's FLASH start address in the OSPI\_INDIRECT\_READ\_XFER\_START\_REG register.
3. Setup the number of bytes to be transferred in the OSPI\_INDIRECT\_READ\_XFER\_NUM\_BYTES\_REG register.
4. Setup the indirect transfer's trigger address in the OSPI\_IND\_AHB\_ADDR\_TRIGGER\_REG register.
5. Setup the indirect transfer's trigger address range in the OSPI\_INDIRECT\_TRIGGER\_ADDR\_RANGE\_REG register.
6. If the watermark interrupt feature is to be used, set the OSPI\_INDIRECT\_READ\_XFER\_WATERMARK\_REG register which will cause an interrupt to be generated when the fill level increases beyond the watermark level. Setting the watermark can be useful indication to software when to read the next part of the indirect read transfer. Note that if the watermark is set to a value other than zero, the watermark interrupt will always trigger once the final byte of indirect transfer has been fetched and placed in the embedded SRAM, even if the watermark value is higher than the actual completed fill level.
7. Trigger Indirect Read access by setting the OSPI\_INDIRECT\_READ\_XFER\_CTRL\_REG[0] START\_FLD bit to 1.
8. If the watermark interrupt feature is to be used, wait for watermark interrupt. Else poll the SRAM fill level via the OSPI\_SRAM\_FILL\_REG register to decide when sufficient data is in the SRAM to trigger data fetches.



9. Read the expected amount of data from SRAM. If there is still more data to fetch in order to complete the indirect read transfer, then loop back to step 8. Otherwise continue to step 10.
10. The completion status of the Indirect Read operation can be polled via the OSPI\_INDIRECT\_READ\_XFER\_CTRL\_REG[5] IND\_OPS\_DONE\_STATUS\_FLD bit.
11. An Indirect Complete interrupt will be generated when the Indirect read operation has completed.

#### 12.3.2.4.10.2 Indirect Write Controller

The aim of the indirect mode of operation is to perform bulk transfer of data from the processor into a FLASH memory in the most efficient manner. The fewest possible write cycles inside the FLASH device will be carried out for the indirect transfer, thus maximizing the life of the device. Indirect write operation can be thought of from a software perspective as the inverse of the indirect read. It is controlled and triggered by software via specific control/configuration Indirect Write Transfer registers (for more information see the following registers: OSPI\_INDIRECT\_WRITE\_XFER\_CTRL\_REG, OSPI\_INDIRECT\_WRITE\_XFER\_WATERMARK\_REG, OSPI\_INDIRECT\_WRITE\_XFER\_START\_REG, and OSPI\_INDIRECT\_WRITE\_XFER\_NUM\_BYTES\_REG). This block will await delivery of the write data via the external data interface controller, placing it in the local SRAM before communicating with the existing legacy SPI core to perform an efficient and optimized FLASH write burst.

By default, the indirect write controller is disabled. Before enabling it, the software must configure how much data is required and the start address. The start address and total number of bytes to be written is defined in OSPI\_INDIRECT\_WRITE\_XFER\_START\_REG and OSPI\_INDIRECT\_WRITE\_XFER\_NUM\_BYTES\_REG registers, respectively. Up to two indirect operations can be programmed at any one time. The second operation can be triggered while the first is in progress. Supporting two indirect operations allows a short turnaround time between the completion of one indirect operation and the start of the second. The Indirect write queuing is very similar to indirect read queuing. For more information refer to [Section 12.3.2.4.10.3, Indirect Access Queuing](#).

The total number of bytes to write in an indirect operation is not limited by the size of the SRAM. The size of SRAM will only limit the amount of data that can be accepted from the external controller. In the case of an SRAM overrun, the controller will back pressure the data interface with wait states. Note the fill level of the SRAM is readable via programmable OSPI\_SRAM\_FILL\_REG register and this can be used to avoid this situation.

An external controller will provide the write data and will transfer this to the OSPI module by issuing data interface writes. The address of the incoming write access must be in the range of Indirect trigger address programmed via the OSPI\_IND\_AHB\_ADDR\_TRIGGER\_REG register to Indirect trigger address +  $2^{**}(\text{Indirect trigger address range}) - 1$ . Default value of the range is equal to 16 locations. This allows a 16-beat burst to be applied starting from the Indirect trigger address. The smaller bursts are possible to handle effectively as well with this approach. Furthermore it is not strict requirement to push consecutive address sequence. Actual address just has to be in the Indirect Range to grant SRAM as source. Each write will cause the internal SRAM to be pushed, thereby decoupling the incoming write access address from the FLASH address – that is not direct mapped. Therefore Indirect trigger address does not have any relationship with FLASH address. It is just to indicate that data should take SRAM as source instead of FLASH Memory array after triggering of any valid Indirect Write. The FLASH address for Indirect Write is taken from the OSPI\_INDIRECT\_WRITE\_XFER\_START\_REG register. Assuming the SRAM is not full at the point the data interface access is received by the OSPI module, then the data will be pushed to the SRAM with minimum latency.

If a write access is received whose address is not within the range described above then that access will not be completed using the indirect controller. It will instead be serviced by the direct access controller.

If a write access is received whose address is within the range described above but the SRAM is full then wait states will be applied until some or all of the data has been pushed from the SRAM to the FLASH.

If a write burst is received whose access elements traverse the Indirect trigger range, then the accesses within the Indirect trigger range will be processed by the indirect controller, and the rest will be taken by the direct access controller. This is likely to be a software configuration error.

The external controller is only permitted to issue 32-bit data interface writes until the last word of an indirect transfer. This helps keep the SRAM control logic less complex. On the final write, the external controller may issue a 32-bit word, 16-bit (halfword) or a byte access to complete the transfer. If the number of bytes to write is less than 4 on the last transfer, the controller is still permitted to issue a 32-bit transfer. In these cases, the extra bytes are discarded by the controller.

When the SRAM holds a number of bytes equal to or greater than the size of a FLASH page (which itself is programmed into the OSPI module, with a default of 256 bytes) or when the SRAM holds all remaining bytes of the currently executing indirect transfer, the OSPI module will initiate a write burst to the flash command generator.

An indirect operation may be cancelled at any time by setting 1 to the OSPI\_INDIRECT\_WRITE\_XFER\_CTRL\_REG[1] CANCEL\_FLD bit.

Any bus controller should be allowed to initiate an indirect access. The OSPI module provide software access mechanism to the SRAM fill-level directly via the configuration registers and then decide for itself when the data should be written to the local SRAM. The fill level watermark register (see OSPI\_INDIRECT\_WRITE\_XFER\_WATERMARK\_REG register) is provided. When the SRAM fill level falls below this watermark, an interrupt is generated.

Two further interrupt sources are provided to help understand the status of an indirect operation. Firstly, an interrupt is generated when an indirect operation has completed. Secondly, an interrupt is generated if an indirect write operation was requested but could not be accepted due to the fact 2 indirect operations have already been buffered by the OSPI module.

Setting the OSPI\_INDIRECT\_WRITE\_XFER\_CTRL\_REG[0] START\_FLD bit starts an indirect write operation. The OSPI\_INDIRECT\_WRITE\_XFER\_CTRL\_REG[2] WR\_STATUS\_FLD bit is available to check the status.

#### 12.3.2.4.10.2.1 Indirect Write Transfer Process

The following sequence can be followed:

1. Setup OSPI\_CONFIG\_REG register.
2. Setup the indirect transfer's FLASH start address in the OSPI\_INDIRECT\_WRITE\_XFER\_START\_REG register.
3. Setup the number of bytes to be transferred in the OSPI\_INDIRECT\_WRITE\_XFER\_NUM\_BYTES\_REG register.
4. Setup the indirect transfer's trigger address in the OSPI\_IND\_AHB\_ADDR\_TRIGGER\_REG register.
5. Setup the indirect transfer's trigger address range in the OSPI\_INDIRECT\_TRIGGER\_ADDR\_RANGE\_REG register.
6. It is functionally valid for software to simply write all the data to the SRAM in one block transfer. However, if the total number of bytes to write is greater than the size of the partitioned SRAM, then it is quite likely the SRAM will become full causing the OSPI to back-pressure the system data bus for a considerable time. This time is based on the FLASH data-rate and the page-write time of the device. To avoid sending all the write data in one block transfer, software can make use of the watermark interrupt to identify a convenient time to send data a page at a time to the SRAM module. Alternatively, software can poll the SRAM fill level register directly to identify how empty the SRAM is at any one time in order to make a judgment as to when the most practical time to send the next part of the transfer.
7. If the watermark interrupt feature is to be used, set the OSPI\_INDIRECT\_WRITE\_XFER\_WATERMARK\_REG register which will cause an interrupt to be generated when the fill level falls below the watermark. The watermark should be set to a number between zero and a page size. That is if the page size is 256 bytes, then setting the watermark to a value between 10 and 250 is reasonable and will cause the interrupt to trigger when the fill level drops below the programmed number. Setting the watermark can be useful to provide an indication to software when to write the next page of data to the SRAM.
8. Trigger Indirect Write access by setting OSPI\_INDIRECT\_WRITE\_XFER\_CTRL\_REG[0] START\_FLD bit.



9. If the remaining number of bytes still to be transferred into the SRAM for the current indirect transfer is greater than a FLASH page, then write 1 FLASH page worth of data to the SRAM. Otherwise send the remaining data from the indirect transfer to SRAM.
10. If all the data in the indirect transfer has now been sent to the SRAM, then go to 12 and await indirect complete status. Otherwise if there is more data still to be transferred then either:
  - If the watermark interrupt feature is being used, then wait for watermark interrupt.
  - Alternatively the SRAM fill level can be interrogated to identify a convenient time to send more data.
11. Loop back to 9.
12. Optional: The completion status of the Indirect write operation can be polled via OSPI\_INDIRECT\_WRITE\_XFER\_CTRL\_REG[5] IND\_OPS\_DONE\_STATUS\_FLD.
13. An Indirect Complete interrupt will be generated when the Indirect write operation has completed.

#### **12.3.2.4.10.3 Indirect Access Queuing**

Software is permitted to queue up to two indirect transfers for both the indirect write controller and the indirect read controller. Supporting two indirect operations allows a short turnaround time between the completion of one indirect operation and the start of the second. Any attempt to queue more than two operations will cause an interrupt to be generated. To take advantage of this feature, software should attempt to keep both indirect programming slots full at all times.

From the software perspective, indirect access queuing is achieved by triggering bit 0 of the indirect transfer control register (OSPI\_INDIRECT\_READ\_XFER\_CTRL\_REG[0] START\_FLD bit or OSPI\_INDIRECT\_WRITE\_XFER\_CTRL\_REG[0] START\_FLD bit) twice in short succession. The indirect number of bytes register (OSPI\_INDIRECT\_READ\_XFER\_NUM\_BYTES\_REG or OSPI\_INDIRECT\_WRITE\_XFER\_NUM\_BYTES\_REG register) and the indirect FLASH start address register (OSPI\_INDIRECT\_READ\_XFER\_START\_REG or OSPI\_INDIRECT\_WRITE\_XFER\_START\_REG register) must be setup with the relevant transfer data before START\_FLD bit can be triggered for each transfer. Since these registers will change regularly, the hardware must keep sampled versions of these registers for the duration of the indirect transfer.

The internal register block will only issue an indirect start trigger to the key underlying datapath blocks one at a time. There are 2 independent datapath blocks in the indirect access controller that will receive and independently sample this information. The first is the datapath block on the data bus side of the SRAM. For indirect reads, this is a read interface, for indirect writes, it is a write interface. The second is the datapath block on the FLASH side of the SRAM. For indirect reads, this is a write interface, for indirect writes, it is a read interface. Both blocks will process the indirect transfers at different times. For example, for an indirect read operation, the datapath block on the FLASH side of the SRAM will be able to start processing the second queued transfer as soon as the last byte of the first transfer has been written to the SRAM. Before commencing the second transfer, this block must resample the OSPI\_INDIRECT\_READ\_XFER\_NUM\_BYTES\_REG and OSPI\_INDIRECT\_READ\_XFER\_START\_REG registers. Similarly, the datapath block on the bus side will resample the same registers locally when it has forwarded all the FLASH data associated with the first indirect transfer from the SRAM onto the data bus.

#### **12.3.2.4.10.4 Consecutive Writes and Reads Using Indirect Transfers**

It is permitted for software to trigger an indirect read operation while an indirect write operation is in progress. Similarly it is permitted to trigger an indirect write while an indirect read operation is in progress. Indirect write operations will take overall precedence.

#### **12.3.2.4.10.5 Accessing the SRAM**

The SRAM depth is separated in two segments. The lower segment is reserved for indirect read use. The upper segment is for indirect write use only. The size of each segment is programmable via the OSPI\_SRAM\_PARTITION\_CFG\_REG register. This feature allows to allocate how many bits of the SRAM address bus are allocated to indirect read. By default, this is set so that exactly half of the SRAM is portioned for use by the indirect read controller. To ensure the read data bus is not directly fed by the SRAM read data through combinatorial logic, an extra bank of holding registers is included in the indirect read data path. These registers act as an extra location to be added to the allocated number of SRAM locations for indirect read.

To illustrate how the SRAM (and the extra bank of holding registers) can be allocated between indirect read and write, the following example is provided. The depth of the SRAM in this example is configured to be 8 bits. This is equal to 256 locations.

- If the OSPI\_SRAM\_PARTITION\_CFG\_REG[7-0] ADDR\_FLD field is set to 0x00, then 256 locations are allocated to indirect writes and 1 location to indirect reads.
- If the OSPI\_SRAM\_PARTITION\_CFG\_REG[7-0] ADDR\_FLD field is set to 0x01, then 255 locations are allocated to indirect writes and 2 locations to indirect reads.
- If the OSPI\_SRAM\_PARTITION\_CFG\_REG[7-0] ADDR\_FLD field is set to 0x02, then 254 locations are allocated to indirect writes and 3 locations to indirect reads.
- And so on until.
- If the OSPI\_SRAM\_PARTITION\_CFG\_REG[7-0] ADDR\_FLD field is set to 0xFD, then 3 locations are allocated to indirect writes and 254 locations to indirect reads.
- If the OSPI\_SRAM\_PARTITION\_CFG\_REG[7-0] ADDR\_FLD field is set to 0xFE, then 2 locations are allocated to indirect writes and 255 locations to indirect reads.
- If the OSPI\_SRAM\_PARTITION\_CFG\_REG[7-0] ADDR\_FLD field is set to 0xFF, then 1 location is allocated to indirect writes and 256 locations to indirect reads.

### Note

A value of 0xFF or 0x00 in the OSPI\_SRAM\_PARTITION\_CFG\_REG register should be avoided by software, as only the bottom 8 bits of the SRAM fill level are accessible through software (up to 255 limit) via the OSPI\_SRAM\_FILL\_REG register. If the fill level reaches 256 on either the indirect read or write side, it will appear when reading the Fill Level to be 0.

There are four SRAM sources that are arbitrated and muxed onto the single SRAM port. Up to three sources can access this port at any one time. The sources are described as follows:

- Indirect Write, Write source. This is located on the data bus side of the SRAM.
- Indirect Write, Read source. This is located on the FLASH side of the SRAM.
- Indirect Read, Write source. This is located on the FLASH side of the SRAM.
- Indirect Read, Read source. This is located on the data bus side of the SRAM.

A fixed priority arbitration scheme is implemented. [Table 12-301](#) shows priority allocated to these sources.

**Table 12-301. SRAM Access Priority**

SRAM Access Priority		
Indirect Write	Write to SRAM (from System Data Bus)	3rd (exclusive with Data Bus Read Request)
	Read from SRAM (from OSPI Module)	2nd
Indirect Read	Write to SRAM (from OSPI Module)	1st
	Read from SRAM (from System Data Bus)	3rd (exclusive with Data Bus Write Request)

### Note

With the exception of the write port during an Indirect Read operation (on the FLASH side of the SRAM), the logic driving all four sources must not assume single cycle completion. Writes to the SRAM during an indirect read must be allowed to complete immediately to avoid data loss. Therefore this port is given maximum priority.

#### 12.3.2.4.11 OSPI Software-Triggered Instruction Generator (STIG)

The DAC and INDAC are used to transfer data. In order to access the volatile and non-volatile configuration registers, the legacy SPI Status register, other status/protection registers as well as to perform ERASE functions, a separate software controller is required. The software triggered instruction generator (STIG) is controlled using the OSPI\_FLASH\_CMD\_CTRL\_REG register by setting

up the command to issue to the FLASH device. This is a generic controller and can be used to perform any instruction that the FLASH device supports from the extended SPI protocol. Configuring of instructions which are not compliant with the specification of the FLASH devices could cause unpredicted behavior of the controller. OSPI\_FLASH\_CMD\_CTRL\_REG[31-24] CMD\_OPCODE\_FLD bits should be set different than OSPI\_DEV\_INSTR\_RD\_CONFIG\_REG[7-0] RD\_OPCODE\_NON\_XIP\_FLD and OSPI\_DEV\_INSTR\_WR\_CONFIG\_REG[7-0] WR\_OPCODE\_FLD. The OSPI\_FLASH\_CMD\_CTRL\_REG[0] CMD\_EXEC\_FLD bit is used to trigger the command. The OSPI\_FLASH\_CMD\_CTRL\_REG[1] CMD\_EXEC\_STATUS\_FLD bit is used by software to poll the status of the command execution. For reads, when the command has been serviced (OSPI\_FLASH\_CMD\_CTRL\_REG[1] CMD\_EXEC\_STATUS\_FLD bit toggles from '1' to '0'), up to 8 bytes of read data will be placed in the OSPI\_FLASH\_RD\_DATA\_LOWER\_REG and OSPI\_FLASH\_RD\_DATA\_UPPER\_REG registers. For writes, the write data should be placed in the OSPI\_FLASH\_WR\_DATA\_LOWER\_REG and OSPI\_FLASH\_WR\_DATA\_UPPER\_REG registers.

The completion of the STIG request could be also checked by the corresponding interrupt. The occurrence of the interrupt indicates that the controller is ready for accepting a new STIG request. It is important to notice that completion of the STIG request is not equivalent to completion it on SPI side. For example, if STIG is configured to the command composed of data to transmit only, the data is taken from the corresponding STIG register fields and put into TX FIFO. Since all bytes to write are known, another STIG can be queued before serialization of the current one is completed.

There are some commands which require more data to read than 8 bytes (for example READ ID command). The additional STIG Memory Bank is implemented in order to accommodate these data if needed. The STIG Memory Bank (internal component of the controller) is controlled by the OSPI\_FLASH\_CMD\_CTRL\_REG[2] STIG\_MEM\_BANK\_EN\_FLD bit. If enabled, the number of bytes to read in the STIG is extended to 16 as defined in OSPI\_FLASH\_COMMAND\_CTRL\_MEM\_REG[18-16] NB\_OF\_STIG\_READ\_BYTES\_FLD bit field. It should be noticed that there are very few commands (excluding Read Array ones which are not intended to handle effectively in STIG Mode but in Direct/Indirect Modes) which return more than 8 bytes to the controller. If the maximum number of bytes to Read using STIG in target application is less than 16, the depth of the STIG Memory Bank can be set smaller what will result in saving noticeable part of the area.

If number of bytes to Read in the STIG as defined in OSPI\_FLASH\_COMMAND\_CTRL\_MEM\_REG[18-16] NB\_OF\_STIG\_READ\_BYTES\_FLD bit field exceeds the Memory Bank Depth, remaining data will overwrite the STIG Memory Bank locations starting from its first address. OSPI\_FLASH\_RD\_DATA\_LOWER\_REG and OSPI\_FLASH\_RD\_DATA\_UPPER\_REG keep the last 8 bytes read from the Flash Device by STIG when Memory Bank is enabled. Therefore, for example if the user wants to get just a single byte from the last eight bytes from long continuous read SPI data chain, there is no need to access the STIG Memory Bank since data can be taken from suitable Flash Command Read Data register. In order to access more data, STIG Memory Bank data request should be triggered. It is controlled by the OSPI\_FLASH\_COMMAND\_CTRL\_MEM\_REG and works analogously for triggering STIG from the functional standpoint.

OSPI\_FLASH\_COMMAND\_CTRL\_MEM\_REG[0] TRIGGER\_MEM\_BANK\_REQ\_FLD bit is used to trigger the command, bit OSPI\_FLASH\_COMMAND\_CTRL\_MEM\_REG[1] MEM\_BANK\_REQ\_IN\_PROGRESS\_FLD is used by software to poll the status of the command execution. When MEM\_BANK\_REQ\_IN\_PROGRESS\_FLD bit toggles from "1" to "0", the byte of data (OSPI\_FLASH\_COMMAND\_CTRL\_MEM\_REG[15-8] MEM\_BANK\_READ\_DATA\_FLD) from corresponding address (OSPI\_FLASH\_COMMAND\_CTRL\_MEM\_REG[28-20] MEM\_BANK\_ADDR\_FLD bit field) is valid. The address should be set before triggering the STIG Memory Bank access. Each consecutive STIG access overwrites the previous one so that the data in the Bank always fit into byte index fetched by the last STIG access configured to use the Memory Bank (first incoming byte equals first address of the Memory Bank, second one equals the second address and so on).

#### 12.3.2.4.11.1 Servicing a STIG Request

A STIG request will cause the OSPI Flash controller to interrogate the OSPI\_FLASH\_CMD\_CTRL\_REG register to determine what and how many bytes it should send to the FLASH device. The OSPI\_FLASH\_CMD\_CTRL\_REG[31-24] CMD\_OPCODE\_FLD field of this register indicate the instruction to be sent and is always pushed first. If there is an address to send, then the address (the size

of which is also programmed in the same register) is sent next. The address itself is stored in the OSPI\_FLASH\_CMD\_ADDR\_REG register. If Mode bits are enabled by OSPI\_FLASH\_CMD\_CTRL\_REG[18] ENB\_MODE\_BIT\_FLD bit. OSPI\_MODE\_BIT\_CONFIG\_REG[7-0] MODE\_FLD bit field are being sent right after address. If OSPI\_FLASH\_CMD\_CTRL\_REG[18] ENB\_MODE\_BIT\_FLD and OSPI\_CONFIG\_REG[29] CRC\_ENABLE\_FLD are both enabled, STIG will replace XIP Mode bits (not applicable for CRC aware SPI interface) for automatically calculated address CRC byte. Therefore, to execute CRC aware STIGs (meaning the commands requiring sending address CRC byte), ENB\_MODE\_BIT\_FLD bit should always be set. If there are any dummy cycles to send (the size of which is also programmed in OSPI\_FLASH\_CMD\_CTRL\_REG register) then those are sent next. If there is data to write or read (the size of which is also programmed in OSPI\_FLASH\_CMD\_CTRL\_REG register) then for the case of writes, up to 8 bytes can be sent (as stored in the Flash Command Write Data registers, OSPI\_FLASH\_WR\_DATA\_LOWER\_REG and OSPI\_FLASH\_WR\_DATA\_UPPER\_REG registers) next. In the read case, when the read data has been collected from the FLASH device, the OSPI Flash Controller stores that in the Flash Command Read Data Registers (OSPI\_FLASH\_RD\_DATA\_LOWER\_REG and OSPI\_FLASH\_RD\_DATA\_UPPER\_REG registers). Up to 8 bytes can be get if OSPI\_FLASH\_CMD\_CTRL\_REG[2] STIG\_MEM\_BANK\_EN\_FLD bit is disabled or up to 512 when enabled. When the OSPI Flash controller starts to service a STIG request, it sets the OSPI\_FLASH\_CMD\_CTRL\_REG[1] CMD\_EXEC\_STATUS\_FLD bit to indicate a command execution is in progress. When the OSPI Flash controller is in the auto-polling state, servicing a STIG request is slightly different. Most of devices are largely inaccessible after a program operation until the device has completed that write. Some group of them has a possibility to suspend programming page. It can be controlled by the OSPI\_POLLING\_FLASH\_STATUS\_REG[8] DEVICE\_STATUS\_VALID\_FLD bit, which indicate active auto-polling phase. After requesting a STIG, the OSPI Flash Controller immediately issues appropriate OPCODE to Memory. During servicing a STIG (in auto-polling phase) the status bit of command execution remains steady and other parts of transfer such as ADDRESS or DUMMY BITS, and so forth, are disabled (to issued Program Suspend Command is needed OPCODE only). There is a programmable option to add delay between every repetitive poll operation (delay is defined by OSPI\_WRITE\_COMPLETION\_CTRL\_REG[31-24] POLL\_REP\_DELAY\_FLD bit field). This feature is implemented to free up SPI bandwidth if needed.

---

#### Note

The OSPI data is sent LSB first, while address is sent MSB first.

---



---

#### Note

The STIG complete status bit gets cleared before the actual flash access completes. Software should wait for about 700 ns if there is any dependency on actual access completion.

---

### 12.3.2.4.11.2

---

#### Note

The OSPI data is sent LSB first, while address is sent MSB first.

---



---

#### Note

The STIG complete status bit gets cleared before the actual flash access completes. Software should wait for about 700 ns if there is any dependency on actual access completion.

---

### 12.3.2.4.12 OSPI Arbitration Between Direct / Indirect Access Controller and STIG

When multiple controllers are active simultaneously, a simple fixed-priority arbitration scheme is used to arbitrate between each interface and access the external FLASH. The fixed priority is defined as follows, highest priority first.

- The Indirect Access Write
- The Direct Access Write

- The STIG
- The Direct Access Read
- The Indirect Access Read

#### 12.3.2.4.13 OSPI Command Translation

Requests issued by the direct access controller, the indirect access controller or the STIG will be translated into a sequence of byte transfers to send downstream (before serialization to the FLASH device). These sequences depend on the requested transfer but an example of a typical 1-byte non sequential READ is shown below:

INSTRUCTION OPCODE -> ADDRESS -> Mode Byte -> Dummy Bytes -> 1 byte of don't care

For sequential accesses, an extra byte of data per read is pushed to the FLASH device on the back of the above sequence assuming it can be done so with no gap between each transferred byte.

When PHY mode is enabled and consequently no clock divider is configured, latency caused by multi domain synchronization may make an extra byte insufficient to avoid the transfer gap. To ensure the sequential access non-interrupted and keep the maximum performance of the controller, PHY Pipeline Mode is implemented. When enabled, number of don't care bytes is calculated based on the configuration.

The actual sequence sent to the FLASH device depends on the requested transfers, whether the transfer is non-sequential or sequential, whether the device has been configured in XIP mode and the state of the main Device Instruction Type programmable registers (OSPI\_DEV\_INSTR\_RD\_CONFIG\_REG and OSPI\_DEV\_INSTR\_WR\_CONFIG\_REG).

For writes, the write enable latch (or WEL) within the FLASH device itself must be high before a write sequence can be issued. The OSPI Flash Controller will automatically issue the write enable latch command before triggering a write command via the direct or indirect access controllers (DAC/INDAC) – that is the user does not need to perform this operation. For increasing flexibility and performance user can turn off this feature by setting the OSPI\_DEV\_INSTR\_WR\_CONFIG\_REG[8] WEL\_DIS\_FLD bit. The opcode for WREN is typically 0x06 and is common between devices.

When write requests from the direct or indirect access controllers are no longer being received and all outstanding requests have been sent, the FLASH device will automatically start the page program write cycle. Any incoming request at this time will be held in wait states until the cycle has completed. The OSPI Flash Controller will automatically poll the FLASH device legacy SPI status register to identify when the write cycle has completed. This is achieved by sending the RDSR opcode to the FLASH device and waiting until the device itself has indicated the write cycle has completed (until the Write in Progress bit has cleared to zero and the write enable latch bit has also cleared to zero or device is ready bit has set to one). The WREN and the RDSR device instructions are the only ones that are sent by the controller under the hood. For any other specific instruction that the user determines should be sent to the device (for example if the device needs to be unprotected before a write command is issued), these should be handled separately by issuing FLASH commands via the STIG.

There is an option to trigger HOLD or RESET feature on I/Os of the Flash Device. The HOLD one is generally common across the devices and takes an alternative function of DQ3 pin (applicable when device operates neither in Quad SPI mode nor DDR). The transfer can be hold and then resumed by dedicated software trigger field (OSPI\_CONFIG\_REG[4] HOLD\_PIN\_FLD). The devices which have the HOLD feature on DQ3 usually need another dedicated pin for hardware reset and ones without HOLD feature usually have alternative reset on DQ3 what makes the additional reset pin being redundant. The controller supports both variants and reset selection register field (OSPI\_CONFIG\_REG[6] RESET\_CFG\_FLD) allows the user to configure which hardware reset solution is implemented in the device under usage.

After configuration is done, it is possible to trigger HOLD or RESET features using I/Os (OSPI\_CONFIG\_REG[4] HOLD\_PIN\_FLD or OSPI\_CONFIG\_REG[5] RESET\_PIN\_FLD bits). After HOLD activation the controller is introduced into waiting state and any other operations should not be requested before de-asserting of HOLD configuration bit. The HOLD feature is useful when any SPI transaction needs to be prolonged in order to adjust it into specific point in time. Note that any HOLD trigger issued during active SPI transaction may be synchronized into reference clock domain at the time the SPI transfer turns to be finished. In this case, there is nothing to hold so the low level SPI logic will not activate HOLD on DQ3. To check if HOLD



request suspended the transfer OSPI\_CONFIG\_REG[31] IDLE\_FLD bit can be polled for. If SPI is not in the IDLE state, the transfer was successfully suspended. It is important for the software to take care of resetting OSPI\_CONFIG\_REG[4] HOLD\_PIN\_FLD bit before newly triggered SPI transaction. In case HOLD request is set before the beginning of the transfer it will be HOLD right after it starts what may not always be a goal. The hardware RESET needs to be activated when CS is high (no valid transaction is present on SPI bus). It can be checked by polling of OSPI\_CONFIG\_REG[31]. If the controller is in the IDLE state and no other transfer requests are queued to perform, the hardware RESET can be triggered. The RESET feature is useful when any write, program or erase operation needs to be cancelled. No transfer request is permitted before driving the reset back to being inactive. Triggering HOLD or RESET on DQ3 at the time the device is configured to work in Quad SPI mode or DDR will overwrite transfer data on DQ3 with '0'. This behavior is considered as a software error so it is advisable for the system to make sure that the flash device was introduced to suitable SPI mode (that is by polling its configuration register) before triggering alternative DQ3 function. There are four independent reset outputs implemented to separate between multiple devices connected to the controller (up to 4 are supported). The decision which reset output is to be activated after triggering OSPI\_CONFIG\_REG[5] RESET\_PIN\_FLD bit is made based on OSPI\_CONFIG\_REG[9] PERIPH\_SEL\_DEC\_FLD and OSPI\_CONFIG\_REG[13-10] PERIPH\_CS\_LINES\_FLD bits. Reset output OSPI\_ECC\_VECTOR is to be directly driven into corresponding dedicated RESET pins of the devices with separated RESET pin and alternatively, Reset output OSPI\_ECC\_VECTOR is to be control OSPI\_ECC\_VECTOR of the DQ3 RESET devices enabling separating of DQ3 Controller Outputs on SoC integration level.

The controller supports all combinations of CPHA and CPOL for Serial Clock. It allows the controller to support any SPI target devices not limited to Flash Memories. Multiple-SPI flash devices use just a subset of these combinations depending on the Transfer Mode as defined in [Table 12-302](#).

**Table 12-302. Flash SPI Modes**

(SEL_CLK_POL_FLD, SEL_CLK_PHASE_FLD)	Edge Mode	Support
0x00 (SPI MODE 0)	SDR	Yes
0x01 (SPI MODE 1)	SDR	No
0x10 (SPI MODE 2)	SDR	No
0x11 (SPI MODE 3)	SDR	No
0x00 (SPI MODE 0)	DDR	Yes
0x01 (SPI MODE 1)	DDR	No
0x10 (SPI MODE 2)	DDR	No
0x11 (SPI MODE 3)	DDR	No

#### 12.3.2.4.14 Selecting the Flash Instruction Type

In order to send the correct READ and WRITE opcodes, software should initialize the OSPI\_DEV\_INSTR\_RD\_CONFIG\_REG and the OSPI\_DEV\_INSTR\_WR\_CONFIG\_REG registers. These registers include fields to setup the required instruction opcodes that is intended to be used to access the FLASH (default is basic READ and basic page program) as well as the instruction type, edge mode (DDR or SDR) and whether the instruction uses single, dual, quad or octal pins for address and data transfer. Providing this level of control to the user provides a future proofed generic solution. To ensure the controller can operate from a reset state, the registers will be reset to an opcode compatible with SIO devices what can be modified using BOOT feature.

Despite being applicable for both READs and WRITEs, the OSPI\_DEV\_INSTR\_RD\_CONFIG\_REG[9-8] INSTR\_TYPE\_FLD field only appears once – it is not included in the OSPI\_DEV\_INSTR\_WR\_CONFIG\_REG register. If software sets this to anything other than '0', then the address transfer type and the data transfer type bits of both OSPI\_DEV\_INSTR\_RD\_CONFIG\_REG and OSPI\_DEV\_INSTR\_WR\_CONFIG\_REG registers become don't care. It is made available to allow software to support the less common FLASH instructions where the opcode, address and data are sent on 2 or 4 lanes (the opcode from most instructions are sent serially to the FLASH device, even for dual/quad instructions).

There are devices capable of handling Read Operations in Dual Data Rate Mode (DDR) (it is also called Dual Transfer Rate Mode (DTR)). That means they can issue and capture the data on both rising and falling edges during working with dedicated command type. This enables the controller to maintain throughput at twice lower frequency of OSPI clock. The Device Read Instruction Register has DDR enable bit which informs Octal-SPI Flash Controller that opcode written into Read Opcode field is capable with DDR command type. The other field defined in OSPI\_RD\_DATA\_CAPTURE\_REG[19-16] DDR\_READ\_DELAY\_FLD which enables the controller to shift the transmitted data in DDR mode. By default, data are shifted by 1 clock cycle to ensure hold timing greater than 0 during DDR transactions. It may not be sufficient for high reference clock frequency in accordance with the high dividers.

Table 12-303 shows how software should configure the OSPI module for selected specific READ and WRITE instruction supported by the abovementioned device.

**Table 12-303. READ and WRITE Instruction Configuration**

READ							
OPCODE	OPCODE sent over how many lanes / edge mode?	ADDRESS / DUMMY / MODE sent over how many lanes / edge mode?	DATA bytes sent over how many lanes / edge mode?	Instruction Type (OSPI_DEV_INSTR_RD_CONFIG_REG[9-8] INSTR_TYPE_FLD)	Address transfer type (OSPI_DEV_INSTR_RD_CONFIG_REG[13-12] ADDR_XFER_TYPE_STD_MODE_FLD)	Data transfer type (OSPI_DEV_INSTR_RD_CONFIG_REG[17-16] DATA_XFER_TYPE_EXT_MODE_FLD)	DDR bit enable (OSPI_DEV_INSTR_RD_CONFIG_REG[10] DDR_EN_FLD)
READ	1/SDR	1/SDR	1/SDR	0	0	0	0
FAST_READ	1/SDR	1/SDR	1/SDR	0	0	0	0
DTR_FAST_READ	1/SDR	1/DDR	1/DDR	0	0	0	1
DOFR (Dual O/p Fast Read)	1/SDR	1/SDR	2/SDR	0	0	1	0
DIOFR (Dual I/O Fast Read)	1/SDR	2/SDR	2/SDR	0	1	1	0
DDIOFR (DTR Dual I/O Fast Read)	1/SDR	2/DDR	2/DDR	0	1	1	1
QOFR (Quad O/p Fast Read)	1/SDR	1/SDR	4/SDR	0	0	2	0
QIOFR (Quad I/O Fast Read)	1/SDR	4/SDR	4/SDR	0	2	2	0
DQIOFR (DTR Quad I/O Fast Read)	1/SDR	4/DDR	4/DDR	0	2	2	1
OOFR (Octal O/p Fast Read)	1/SDR	1/SDR	8/SDR	0	0	3	0
OIOFR (Octal I/O Fast Read)	1/SDR	8/SDR	8/SDR	0	3	3	0
DOIOFR (DTR Octal O/p Fast Read)	1/SDR	1/DDR	8/DDR	0	0	3	1
4DOIOFR (4-byte DTR Octal I/O Fast Read)	1/SDR	8/DDR	8/DDR	0	3	3	1
DCFR (Dual Command Fast Read)	2/SDR	2/SDR	2/SDR	1	Don't care	Don't care	0
DDCFR (DTR Dual Command Fast Read)	2/SDR	2/DDR	2/DDR	1	Don't care	Don't care	1

**Table 12-303. READ and WRITE Instruction Configuration (continued)**

QCFR (Quad Command Fast Read)	4/SDR	4/SRD	4/SDR	2	Don't care	Don't care	0
DQCFR (DTR Quad Command Fast Read)	4/SDR	4/DDR	4/DDR	2	Don't care	Don't care	1
OCFR (Octal Command Fast Read)	8/SDR	8/SDR	8/SDR	3	Don't care	Don't care	0
4DOCFR (4-byte DTR Octal Command Fast Read)	8/SDR	8/DDR	8/DDR	3	Don't care	Don't care	1

**WRITE**

OPCODE	OPCODE sent over how many lanes?	ADDRESS / DUMMY / MODE sent over how many lanes?	DATA bytes sent over how many lanes?	Instruction Type (OSPI_DEV_INSTR_RD_CONFIG_REG[9-8] INSTR_TYPE_FLD)	Address transfer type (OSPI_DEV_INSTR_WR_CONFIG_REG[13-12] ADDR_XFER_TYPE_STD_MODE_FLD)	Data transfer type (OSPI_DEV_INSTR_WR_CONFIG_REG[17-16] DATA_XFER_TYPE_EXT_MODE_FLD)
PP	1	1	1	0	0	0
DIFP (Dual Input Fast Program)	1	1	2	0	0	1
DIEFP (Dual Input Extended Fast Program)	1	2	2	0	1	1
QIFP (Quad Input Fast Program)	1	1	4	0	0	2
QIEFP (Quad Input Extended Fast Program)	1	4	4	0	2	2
OIFP (Octal Input Fast Program)	1	1	8	0	0	3
OIEFP (Octal Input Extended Fast Program)	1	8	8	0	3	3
DCPP (Dual Command Fast Program)	2	2	2	1	Don't care	Don't care
QCPP (Quad Command Fast Program)	4	4	4	2	Don't care	Don't care
OCPP (Octal Command Fast Program)	8	8	8	3	Don't care	Don't care

**Note**

This data are applicable for both 3-byte or 4-byte address variants of the commands if did not indicate otherwise.



### Note

In DTR protocol all transfer phases (including opcode) take DDR edge mode independently on the command under execution. DTR protocol is to be enabled by OSPI\_CONFIG\_REG[24] ENABLE\_DTR\_PROTOCOL\_FLD bit. It has higher priority than DDR Mode enable bit from OSPI\_DEV\_INSTR\_RD\_CONFIG\_REG[10] DDR\_EN\_FLD.

#### 12.3.2.4.15 OSPI Data Integrity

The CRC aware SPI transfer can be performed when both controller and device are configured to work in the Octal DDR Protocol.

For write transactions (the controller transmits data throughout all transfer), the controller is responsible for sending address CRC byte (XOR of all address bytes) following address bytes and TX data CRC byte (XOR of all data bytes to write) following data chunk with size as defined in OSPI\_MODE\_BIT\_CONFIG\_REG[10-8] CHUNK\_SIZE\_FLD bit field. All CRC data are being calculated and sent automatically by the controller and the external device is responsible for reacting accordingly on any possible interpolation on the Flash interface. For read transactions, the controller is also responsible for sending address CRC byte (like for write) and for getting and progressing RX data CRC byte returning by the Flash Device after each chunk with size as defined in OSPI\_MODE\_BIT\_CONFIG\_REG[10-8] CHUNK\_SIZE\_FLD bit field. At the time when the Flash Device is returning data back to the controller, the controller dynamically calculates checksum byte by byte. Once the chunk is completed, CRC returned from Flash Device should fit to dynamically calculated CRC by the controller. In case of any deviation, controller reports CRC error to the system by corresponding interrupt (CRC error interrupt). The controller also provides the last captured CRC data in RX data chunk (defined in OSPI\_MODE\_BIT\_CONFIG\_REG[31-24] RX\_CRC\_DATA\_LOW\_FLD and OSPI\_MODE\_BIT\_CONFIG\_REG[23-16] RX\_CRC\_DATA\_UP\_FLD bit fields) to give the software driver the opportunity to further detecting any data corruption on system interfaces. The CRC data valid interrupt informs the system about the accessibility of the new RX CRC data in the registers. Once the system gets the full data word, it can calculate CRC by itself. At the time it collects all data words in chunk and then gets the CRC data valid interrupt, it can compare these data and react accordingly.

Some devices also have embedded ECC mechanism allowing them to report data abnormal conditions on their ECC Correction Signal output. At the time this output turns low, the device expect the OSPI controller to read status register of the device in order to get more details about the source of detected abnormal situation. The OSPI controller investigates ECC status on its ECC\_FAIL input and generates an interrupt when detecting this signal being low.

#### 12.3.2.4.16 OSPI PHY Module

OSPI module fully integrates PHY module dedicated to more flexible and power efficient transfers.

The PHY module communicates with the OSPI Flash controller via the aforementioned PHY Interface and handles data transfer on low-level stage of design hierarchy. However, when the OSPI\_RCLK is configured to be equal to the SPI clock instead of alternative approach using clock divider, there is just one OSPI\_RCLK cycle (not 4 or more) within single SPI period or half period for DDR Mode (SPI Control Module works on reference clock). Given that OSPI\_RCLK is the input clock for RX FIFO and the output one for TX FIFO, the PHY solution incurs more restrictive requirement for value of system clock in order to synchronize data without SPI transfer interruption. For example, when the controller operates in DDR 1× octal Mode, 2 bytes of data (equivalent to one RX FIFO location) is gathered within just single OSPI\_RCLK cycle. The controller cannot predict next data access while operating in the Direct Mode (meaning its size or whether it is sequential to the previous one or not). As a result, if the OSPI\_HCLK is not significantly greater than OSPI\_RCLK, the SPI transfer has to be suspended until the Flash Command Generator forwards new data to TX FIFO.

An optional PHY Pipeline Mode is implemented to avoid the necessity of stable clocking of the system clock for the Direct Mode when the PHY mode is enabled and to keep maximum performance while ensuring correct operation of the OSPI controller with the PHY using low frequencies from all its domains. This mode is a trade-off between large software overhead when operating in the Indirect Mode and the described limitations

present in the Direct Mode. For more information about PHY Pipeline Mode, see [Section 12.3.2.4.16.1, PHY Pipeline Mode](#).

When DDR 2× Mode is granted based on configuration – SPI transfer is automatically performed using the PHY module even if the OSPI\_CONFIG\_REG[3] PHY\_MODE\_ENABLE\_FLD is de-asserted. SDR 2× commands are handled with PHY module paths being bypassed. Nevertheless, dividers of 2, 4 or 6 for DDR and divider of 2 for SDR should not be configured based on controller requirements and these configurations are perceived as a software error.

The following steps are an example of software algorithm of adapting the OSPI controller with the PHY module incorporated to work in octal 1× clock DDR Protocol. Note that all necessary configuration steps described in [Section 12.3.2.5.2, Configuring the OSPI Controller for Optimal Use](#) shall be completed before the algorithm.

1. Set PHY mode enable (OSPI\_CONFIG\_REG[3] PHY\_MODE\_ENABLE\_FLD bit) and DDR protocol (OSPI\_CONFIG\_REG[24] ENABLE\_DTR\_PROTOCOL\_FLD bit). It is assumed that device is configured to work in DDR Protocol.
2. Before setting the DLL parameters, software calibration could be needed.  
OSPI\_PHY\_MASTER\_CONTROL\_REG[23] PHY\_MASTER\_BYPASS\_MODE\_FLD bit controls the bypass mode of the controller and target DLLs. If this bit is set, the DLL bypass mode is enabled. This mode is intended to be used only for debug. When set to 0, a Controller operational mode is selected, when set to 1 the Bypass mode is selected.

DLL works in normal mode of operation where the target delay line settings are used as fractional delay of the controller delay line encoder reading of the number of delays in one cycle.

Controller DLL is disabled with only 1 delay element in its delay line. The target delay lines decode delays in absolute delay elements rather than as fractional delays.

- DLL Bypass Mode (follow only if operating in this mode):
  - Depending on frequency of reference clock, calculate how many delay elements should be used to shift this clock by 25% of its period (best case for DDR transfers from setup/hold timings standpoint). Note that delay could be slightly different in a real design. TX Delay is configured in OSPI\_PHY\_CONFIGURATION\_REG[22-16] PHY\_CONFIG\_TX\_DLL\_DELAY\_FLD bit field.
  - Re-synchronize DLLs by asserting OSPI\_PHY\_CONFIGURATION\_REG[31] PHY\_CONFIG\_RESYNC\_FLD bit (If this bit is already set by previous re-synchronization, toggle sequence from "0" to "1" must be generated in order to trigger re-synchronization DLL logic) and set PHY bypass mode enable through OSPI\_PHY\_MASTER\_CONTROL\_REG[23] PHY\_MASTER\_BYPASS\_MODE\_FLD bit.
- DLL Controller Mode (follow only if operating in this mode):
  - Drive DLL reset bit OSPI\_PHY\_CONFIGURATION\_REG[30] PHY\_CONFIG\_RESET\_FLD into low.
  - Calculate initial delay value for the Controller DLL according to the OSPI\_PHY\_MASTER\_CONTROL\_REG[6-0] PHY\_MASTER\_INITIAL\_DELAY\_FLD bit field.
  - Depending on frequency of reference clock, calculate how many delay elements should be used to shift this clock by 25% of its period (best case for DDR transfers from setup/hold timings standpoint). Note that delay could be slightly different in a real design. TX Delay is configured in OSPI\_PHY\_CONFIGURATION\_REG[22-16] PHY\_CONFIG\_TX\_DLL\_DELAY\_FLD bit field.
  - Re-synchronize DLLs by asserting OSPI\_PHY\_CONFIGURATION\_REG[31] PHY\_CONFIG\_RESYNC\_FLD (If this bit is already set by previous re-synchronization, toggle sequence from "0" to "1" must be generated in order to trigger re-synchronization DLL logic) and set DLL reset bit back to high (since both bits are within the same register, it is acceptable to set both bits simultaneously).
  - Poll OSPI\_DLL\_OBSERVABLE\_LOWER\_REG[15] DLL\_OBSERVABLE\_LOWER\_LOOPBACK\_LOCK\_FLD bit. When set – lock is done.
  - Re-synchronize Target DLLs by asserting OSPI\_PHY\_CONFIGURATION\_REG[31] PHY\_CONFIG\_RESYNC\_FLD bit (If this bit is already set by previous re-synchronization, toggle sequence from "0" to "1" must be generated in order to trigger re-synchronization DLL logic) and set TX DLL Delay (OSPI\_PHY\_CONFIGURATION\_REG[22-16] PHY\_CONFIG\_TX\_DLL\_DELAY\_FLD)

- and RX DLL Delay (OSPI\_PHY\_CONFIGURATION\_REG[6-0] PHY\_CONFIG\_RX\_DLL\_DELAY\_FLD) fields which are equivalent to percentage clock offsets now. It is recommended to wait for the new configuration being propagated by 20 reference clock cycles before triggering the next SPI transfer.
- Consider Read Data from location where its value is predictable. This step can be performed in different ways, depending on the device. Parameter Page, ID, Status, Data from OTP region or Data from location of Flash Array the value of which is known can act as the pattern.
  - Trigger Read request chosen from above options.
  - Check correctness of data and store that information:
    - Increment value of RX clock delay – it is configurable in the OSPI\_PHY\_CONFIGURATION\_REG[6-0] PHY\_CONFIG\_RX\_DLL\_DELAY\_FLD bit field.
    - Re-synchronize DLLs.
    - Trigger valid Read request.
    - Check correctness of data and store information.
    - If range boundary of RX clock delay is achieved, go to step 3. Otherwise go back to step "Increment value of RX clock delay".
  - 3. Set RX clock delay value for one from the middle of valid range based on information in storage.
  - 4. Re-synchronize DLLs.
  - 5. Set OSPI\_DEV\_INSTR\_RD\_CONFIG\_REG for Octal Read DDR Configuration (each transfer phase should be configured to work in Octal mode, Number of Dummy cycles should be set as specified in the documentation of the device or more when because of additional read paths delays of actual systems data is predicted to be flopped by PHY module with delay excesses actual cycle of SPI clock generated by the controller).
  - 6. Enable Pipeline mode in the OSPI\_CONFIG\_REG[25] PIPELINE\_PHY\_FLD bit.
  - 7. Perform Sequential Read of Data consistent with conditions indicated within [Section 12.3.2.4.16.1, PHY Pipeline Mode](#).
  - 8. After de-asserting the data target select signal by software – poll OSPI\_CONFIG\_REG[31] IDLE\_FLD bit.
  - 9. When it is asserted to high – next transfer request can be triggered.

#### 12.3.2.4.16.1 PHY Pipeline Mode

This mode is used for Direct Read Mode of operation. If any other operations are intended to be executed, it is recommended to disable PHY Pipeline Mode and re-enable for subsequent Direct Reads in PHY mode. Since there is comprehensive software mechanism controlling Read data transfers in Indirect Mode, pipeline of data interface accesses is not effective for this mode. Enable PHY Pipeline feature when at least four 4-byte-sized data words are predicted to be read in sequentially. The Flash Command Generator pipelines and puts them into TX FIFO which causes CS to remain active because low level SPI protocol controller controls TX FIFO fill level. In order to correctly trigger Direct Read in Pipeline Mode TX FIFO must be empty. Therefore first polling of OSPI\_CONFIG\_REG[31] IDLE\_FLD bit needs to be done. The sequential data transfer will be interrupted when the data target select signal of the data interface is asserted to low. This information is also detected by Data Target Module which informs the Flash Command Generator that the next access is invalid and a TX FIFO locations can be flushed transparently for the system.

In PHY Pipeline Mode it is recommended for Data Controller not to introduce wait states in between consecutive occurrences of the data interface signal that indicates transfer has finished. It will ensure regular transfer rate on data side. Introducing wait states gradually slows data transfer rate down and may finally cause SPI transfer interruption because of TX FIFO data starvation. The system, however, may need to introduce some number of wait states after completion of sequential transfer (composed of 4-byte sized data words) for progressing the data. The dedicated buffer is implemented in Data Target Controller which collects all incoming data during wait states injection. In order to keep SPI transfer uninterrupted, number of wait states should be as little as possible. The higher the OSPI\_HCLK/ OSPI\_RCLK ratio the more wait states can be introduced without SPI transfer interruption. In case the system is able to launch a new transfer before wait states overflow, buffered data transfer to the host will continue. It compensates slowed down transfer by introducing wait states. In case the system is not able to launch a new transfer before wait states overflow, next incoming transfer is considered non-sequential and is executed after all pipelined data is flushed.

This mode can be enabled when following conditions are met:

- OSPI\_HCLK > OSPI\_RCLK (Comparing the slow data clock with the fast reference one makes Pipeline Mode ineffective – Suspend of SPI Transfer would be possible. Consequently, this condition has to be met to operate in this mode.)
- Only 4-byte sized Data Words are permitted (This ensures more data clock cycles for synchronization of FIFOs between consecutive pulses of the signal indicating transfer has finished.)
- The transfer with introduced wait states or non-sequential transfers can only be triggered in between at least four 4-byte sized Data Bursts sequential accesses (16 Bytes) to be sure that Data Controller can trust buffered incoming data during wait states injection.
- Do not use Pipeline Mode along with Continuous Mode (XIP). Benefit of XIP is limited for bulk data transfers intended to execute in Pipeline Mode.

#### **12.3.2.4.16.2 Read Data Capturing by the PHY Module**

Read Data Capturing by the PHY module is useful, as the user is not responsible for the design dedicated DLL being compatible with the Octal-SPI Flash Controller. Another benefit is an option to adjust both SPI clock and sampling clock in a very wide range to fit them into individual requirements of any system. If loopback clock (OSPI\_RD\_DATA\_CAPTURE\_REG[0] BYPASS\_FLD) and PHY mode (OSPI\_CONFIG\_REG[3] PHY\_MODE\_ENABLE\_FLD) are both enabled, the loopback clock is driven into RX DLL instead of gated reference clock. Because of the architecture of DLL, loopback clock needs to be provided in SPI Mode 0. If DQS (OSPI\_RD\_DATA\_CAPTURE\_REG[8] DQS\_ENABLE\_FLD) and PHY mode (OSPI\_CONFIG\_REG[3] PHY\_MODE\_ENABLE\_FLD) are both enabled, the DQS is driven into RX DLL instead of gated reference clock.

### 12.3.2.5 OSPI Programming Guide

#### 12.3.2.5.1 Configuring the OSPI Controller for Use After Reset

The OSPI controller has been designed to wake up in a state that is suitable for performing basic reads and writes using the direct access controller. The BASIC read (opcode 0x03) and BASIC write (opcode 0x02) instructions are operations supported by all target devices. The controller also wakes up with a baud rate divider setting of divide-by-32. Assuming the reference clock is operating at 400 MHz after reset, then this means the effective SPI clock is just 12.5 MHz. This should be slow enough to meet all timing requirements of all target devices without any further device programming.

If the target device does not use 3 address bytes, the device size configuration register must be modified to the appropriate size.

If software plans to write to the device, and the number of bytes per device page is not equal to 256, then the device size configuration register must also be modified.

While not a requirement, it is prudent for software to enable the write protect feature prior to enabling the OSPI controller. This will block any data writes from taking effect. To do so, the protection registers (OSPI\_LOWER\_WR\_PROT\_REG, OSPI\_UPPER\_WR\_PROT\_REG and OSPI\_WR\_PROT\_CTRL\_REG) should be setup and the number of bytes per device block in the device size configuration register should also be setup.

After Power-on Reset (POR), software can read from and write to the FLASH device (albeit slowly). Enabling/Disabling the controller and DAC is achieved with just one write to corresponding fields of the OSPI\_CONFIG\_REG register. User shall take note to maintain the default values of the baud rate divisor and the default state of SEL\_CLK\_POL\_FLD/ SEL\_CLK\_PHASE\_FLD bits of this register. A write data value of 0x00780081 is recommended.

#### 12.3.2.5.2 Configuring the OSPI Controller for Optimal Use

##### Note

When using the OSPI Controller, the opcodes in OSPI\_DEV\_INSTR\_RD\_CONFIG\_REG[7-0] RD\_OPCODE\_NON\_XIP\_FLD, OSPI\_DEV\_INSTR\_WR\_CONFIG\_REG[7-0] WR\_OPCODE\_FLD and OSPI\_WRITE\_COMPLETION\_CTRL\_REG[7-0] OPCODE\_FLD bit fields shall not match the opcode in the OSPI\_FLASH\_CMD\_CTRL\_REG[31-24] CMD\_OPCODE\_FLD bit field.

For high speed transfers PHY mode can be enabled and for optimal configuration PHY Pipeline mode is recommended. For more information, see [Section 12.3.2.4.16.1, PHY Pipeline Mode](#).

To access the flash optimally, software must configure the controller accurately:

1. Wait until any pending STIG or INDAC operation has completed or poll OSPI\_CONFIG\_REG[31] IDLE\_FLD bit.
2. Disable the DAC through OSPI\_CONFIG\_REG[7] ENB\_DIR\_ACC\_CTLR\_FLD bit. It is permitted, but not necessary to also disable the OSPI controller completely via OSPI\_CONFIG\_REG[0] ENB\_SPI\_FLD bit.
3. Update the OSPI\_DEV\_INSTR\_RD\_CONFIG\_REG and OSPI\_DEV\_INSTR\_WR\_CONFIG\_REG registers for the instruction type you wish to use for indirect and direct writes and reads.
4. Update the OSPI\_MODE\_BIT\_CONFIG\_REG[7-0] MODE\_FLD bit field if mode bits have been enabled in the OSPI\_DEV\_INSTR\_RD\_CONFIG\_REG[20] MODE\_BIT\_ENABLE\_FLD bit.
5. Update the OSPI\_DEV\_SIZE\_CONFIG\_REG if the contents are incorrect. Note parts or all of this register may have been updated after initialization. The number of address bytes is a key configuration setting required for performing reads and writes. The number of bytes per page is required for performing any write. The number of bytes per device block is only required if the write protect feature is used. If the default values are correct for the target device, or if some of the values (not including the number address bytes) were incorrect but device writes were not permitted.
6. Update the OSPI\_DEV\_DELAY\_REG. This register allows the user to tweak how the chip select is driven after each FLASH access. This is required as each device may have different timing requirements. As



the serial clock frequency is increased, these timing requirements become more important. Note the numbers programmed in this register are based on the period of reference clock. Example: A device needs 50ns minimum time before CS can be re-asserted after it has been de-asserted. By default, the controller will only provide a minimum of 1 SCLK period. When the device is operating at 100 MHz, the SCLK period is only 10ns, so 40ns extra is required. Since the register defines the number of reference clock cycles to add, and reference clock is running at 400 MHz (2.5ns period), then the user should program a value of at least 16 to the OSPI\_DEV\_DELAY\_REG[31-24] D\_NSS\_FLD. This delay can be extended during auto-polling phase. There is possibility to define the polling repetition delay in the OSPI\_WRITE\_COMPLETION\_CTRL\_REG[31-24] POLL\_REP\_DELAY\_FLD bit field.

7. Update the OSPI\_REMAP\_ADDR\_REG register, if required. Affects DAC path only.
8. Setup and enable write protection registers (OSPI\_LOWER\_WR\_PROT\_REG, OSPI\_UPPER\_WR\_PROT\_REG and OSPI\_WR\_PROT\_CTRL\_REG) if they are required and if they have not already been setup from post initialization.
9. Enable required interrupts via the OSPI\_IRQ\_MASK\_REG register.
10. Setup the baud rate divisor in the OSPI\_CONFIG\_REG[22-19] MSTR\_BAUD\_DIV\_FLD to define the required clock frequency of the target device.
11. Update the OSPI\_RD\_DATA\_CAPTURE\_REG register. This register will delay when the read data is captured and can help when the read data path from the device to the controller is long and the device clock frequency is high. An update to this register may not be necessary.
12. Enable the OSPI controller and the DAC via the OSPI\_CONFIG\_REG.

#### 12.3.2.5.3 Using the Flash Command Control Register (STIG Operation)

The OSPI\_FLASH\_CMD\_CTRL\_REG register provides software means to access the FLASH device in a flexible and programmable manner. This is known as a STIG operation (Software Triggered Instruction Generator). The instruction opcode, number of address bytes (if any), the address itself, number of dummy cycles (if any), number of write data bytes (if any), the write data itself and the number of read data bytes (if any) can be programmed. Once these have been programmed, software can trigger the command via OSPI\_FLASH\_CMD\_CTRL\_REG[0] CMD\_EXEC\_FLD bit and wait for its acceptance by polling OSPI\_FLASH\_CMD\_CTRL\_REG[1] CMD\_EXEC\_STATUS\_FLD bit. When CMD\_EXEC\_STATUS\_FLD bit turns de-asserted, another STIG can be triggered. This method of accessing the FLASH is the typical mechanism that software would use to access the FLASH device's registers, as well as for performing ERASE operations. It can also be used to access the FLASH array itself, although the maximum of 8 data bytes may be read or written at any one time, defined in the Flash Command Write and Read Data registers (OSPI\_FLASH\_RD\_DATA\_LOWER\_REG, OSPI\_FLASH\_RD\_DATA\_UPPER\_REG, OSPI\_FLASH\_WR\_DATA\_LOWER\_REG and OSPI\_FLASH\_WR\_DATA\_UPPER\_REG). This number of bytes can be extended for Read Data commands using additional STIG Memory Bank controlled by OSPI\_FLASH\_CMD\_CTRL\_REG[2] STIG\_MEM\_BANK\_EN\_FLD and OSPI\_FLASH\_COMMAND\_CTRL\_MEM\_REG.

Commands issued using this interface have a higher priority than all other READ accesses coming from data interface, and will therefore interrupt any READ commands being requested by the indirect or direct controllers.

#### 12.3.2.5.4 Using SPI Legacy Mode

SPI legacy mode allows software to access the internal TX-FIFO and RX-FIFO directly, thus bypassing the direct, indirect and STIG controllers.

Legacy mode allows the user to issue any FLASH instruction to the device, but does place a heavy software overhead in order to manage the fill levels of the FIFO's effectively. This is because the legacy SPI core is bi-directional in nature, with data continuously being transferred in either direction while the chip select is enabled. Even if the driver only wishes to read data from the FLASH device, dummy data must be written out to ensure the chip select stays active, and vice versa for write transactions.

Since the TX-FIFO and RX-FIFO are of limited depth, software has a responsibility to maintain the FIFO levels to ensure the TX-FIFO does not become exhausted during the instruction execution and the RX-FIFO doesn't overflow. This can place a lot of overhead on software. Interrupts are provided to indicate when the fill levels

pass programmable watermarks, which are themselves programmable registers `OSPI_TX_THRESH_REG` and `OSPI_RX_THRESH_REG`.

The limited depth may impose the limitation over execution of some specific SPI commands in legacy mode. Note that the controller interprets all transmitted bytes as valid. For example, if the Flash Device was configured to return valid data after many dummy cycles, the TX FIFO could become full before the controller sends all of dummy data.

#### **12.3.2.5.5 Entering XIP Mode from POR**

XIP is a mode that can be entered in a non-volatile way if the device has XIP enabled as a non-volatile configuration setting. Software will not be able to discover the state of XIP from POR via FLASH status register reads as the only operation a FLASH device will recognize when XIP mode is enabled is an XIP read operation.

If it is already known that the device will enter XIP from POR, then the `OSPI_MODE_BIT_CONFIG_REG[7-0] MODE_FLD` and `OSPI_CONFIG_REG[18] ENTER_XIP_MODE_IMM_FLD` bits should be set in initial boot.

If it is not already known that the device will enter XIP from POR, and XIP from POR may be supported by the attached FLASH device, then software can attempt to exit XIP mode by issuing an XIP exit command using a STIG command (via the `OSPI_FLASH_CMD_CTRL_REG` register). To do this, software must be aware of the mode bit requirements of that device, as XIP entry and exit changes per device.

#### **12.3.2.5.6 Entering XIP Mode Otherwise**

XIP mode is supported in most FLASH devices. Some of them use signature bits that are sent to the device immediately following the address bytes, other use signature bits and also require a FLASH device configuration register write to enable XIP. For the FLASH devices that must be compliant to the OSPI controller, the following steps can be taken by software to enter XIP mode:

1. Disable the DAC and INDAC (`OSPI_CONFIG_REG[7] ENB_DIR_ACC_CTLR_FLD`) to ensure no new data read accesses will be sent to the FLASH device.
2. (Optional) Configure the `OSPI_FLASH_CMD_CTRL_REG` to issue a VCR write to FLASH memory, because XIP mode must first be enabled for some devices.
3. Configure the XIP mode bits in the `OSPI_MODE_BIT_CONFIG_REG[7-0] MODE_FLD` bit field.
4. Enable the local controllers XIP mode by setting `OSPI_CONFIG_REG[17] ENTER_XIP_MODE_FLD` bit.
5. Re-enable the DAC and, if required, the INDAC.

#### **12.3.2.5.7 Exiting XIP Mode**

To exit XIP mode, software should first disable the DAC and INDAC to ensure no new data read accesses will be sent to the FLASH device. It should then set mode bits to other than the established in the corresponding Flash Device specifications. These are dependent on the FLASH device and manufacturer. Software should then reset `OSPI_CONFIG_REG[17] ENTER_XIP_MODE_FLD`.

Note the FLASH device must see a READ instruction before it can disable its internal XIP mode state, so this means XIP mode will internally stay active until the next READ instruction is serviced. User must take care to ensure that XIP mode is disabled before the end of any READ sequence.

### 12.3.3 HyperBus Interface

This section describes the HyperBus module in the device.

#### 12.3.3.1 HyperBus Overview

The HyperBus module is a part of the device Flash Subsystem (FSS). For more information about FSS, see [Section 12.3.1, Flash Subsystem \(FSS\)](#).

The HyperBus module is a low pin count memory interface that provides high read/write performance. The HyperBus module connects to HyperBus memory (HyperFlash or HyperRAM) and uses simple HyperBus protocol for read and write transactions.

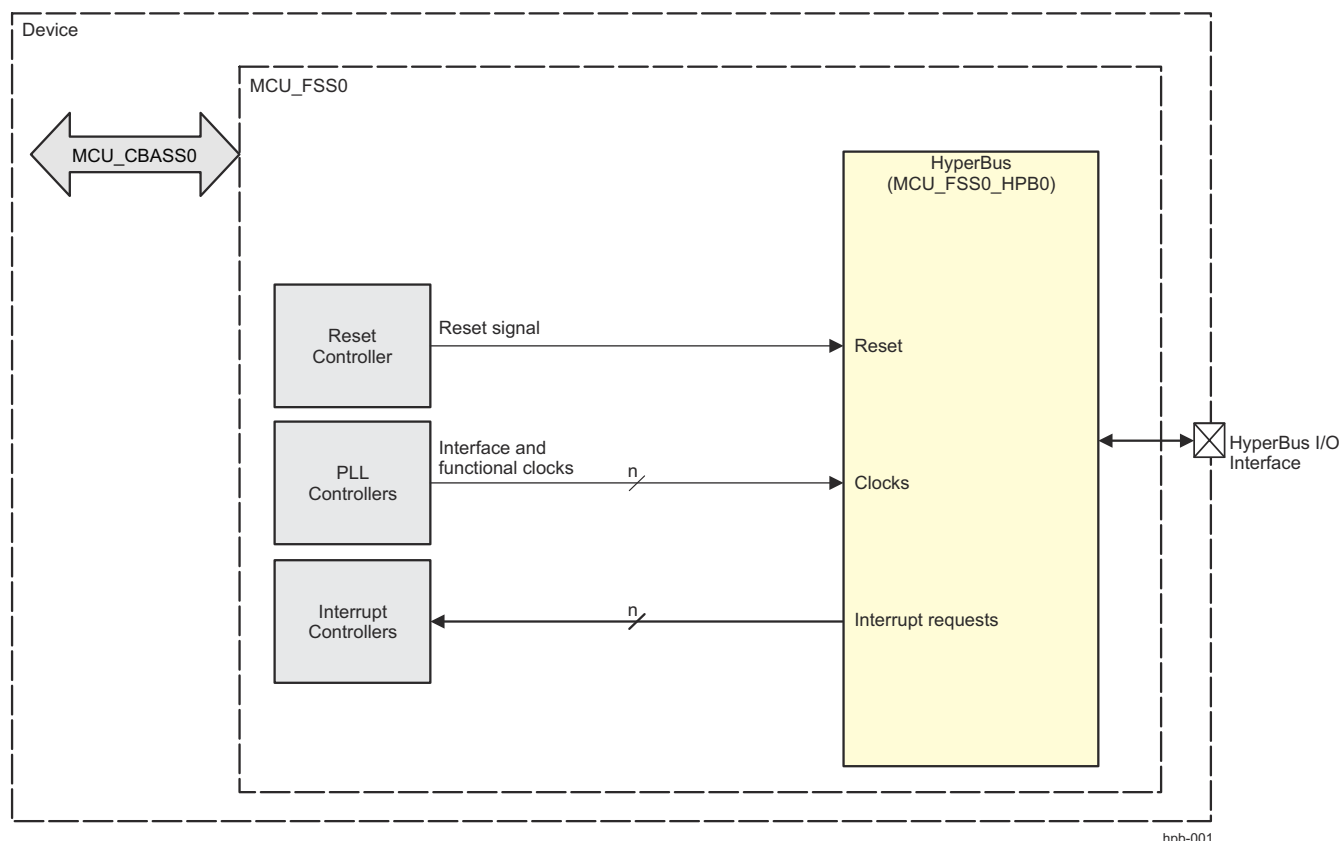
There is one HyperBus™ module inside the device. The HyperBus module includes one HyperBus Memory Controller (HBMC).

[Table 12-304](#) shows HyperBus allocation across device domains.

**Table 12-304. HyperBus Allocation Across Device Domains**

Instance	Domain		
	WKUP	MCU	MAIN
MCU_FSS0_HPBO	-	✓	-

[Figure 12-227](#) shows the HyperBus module overview.



**Figure 12-227. HyperBus Module Overview**

#### 12.3.3.1.1 HyperBus Features

- Support for Cypress® HyperFlash and HyperRAM
- Up to 166 MHz maximum memory bus operation for reads



- Supports up to 166 MHz dual data rate (333 MBps) flash devices for system requiring rapid boot or instant-on displays
- Supports up to 333 MBps external pseudo-RAM (HyperRAM) for systems

**Note:** For more information, see section Timing and Switching Characteristics in the device-specific Datasheet.

- Low pin count interface with LVCMOS I/O pins (can be muxed with other FSS interfaces (OSPIS))
- Two memory chip selects
- Linear incrementing mode for reads and writes
- Up to 16 outstanding read transactions (see [Section 12.3.3.4.5](#), *HyperBus True Continuous Read (TCR) Mode*)
- Asynchronous bus clock

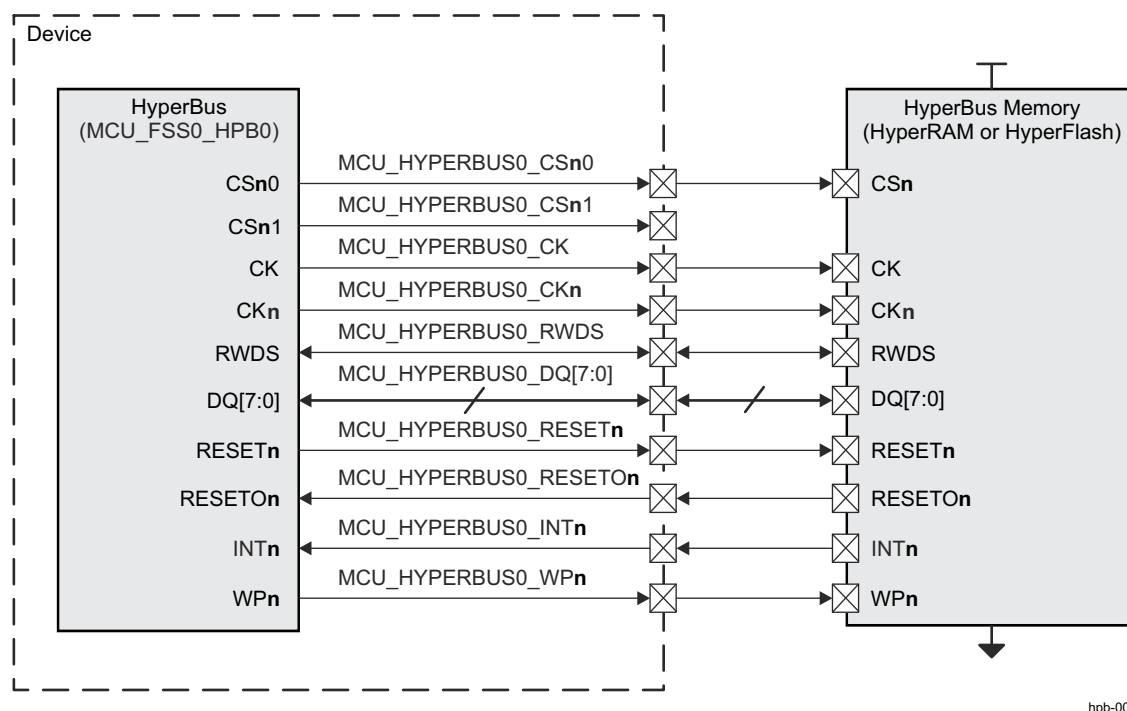
#### **12.3.3.1.2 HyperBus Not Supported Features**

- Cache-line wrap and fixed address modes for reads or writes
- General Purpose Output register (MCU\_FSS0\_HPBO\_MC\_GPOR) of the HBMC is not used

### 12.3.3.2 HyperBus Environment

This section describes the HyperBus external connections (environment).

Figure 12-228 shows the HyperBus connected to HyperRAM or HyperFlash.



**Figure 12-228. HyperBus Connected to HyperRAM or HyperFlash**

Table 12-305 describes the HyperBus I/O signals.

**Table 12-305. HyperBus I/O Signals**

Module Pin	Device Level Signal <sup>(1)</sup>	I/O <sup>(2)</sup>	Description	Module Pin Reset Value
<b>MCU_FSS0_HPB0</b>				
CK	MCU_HYPERBUS0_CK	O	HyperBus Clock Output (differential)	0x0
CKn	MCU_HYPERBUS0_CKn	O	HyperBus Inverted Clock Output (differential)	0x1
RWDS	MCU_HYPERBUS0_RWDS	I/O/HiZ	HyperBus Read/Write Data Strobe	0x1
DQ[7:0]	MCU_HYPERBUS0_DQ[7:0]	I/O/HiZ	HyperBus Data	0x0
CSn0	MCU_HYPERBUS0_CSn0	O	HyperBus Chip Select 0	0x1
CSn1	MCU_HYPERBUS0_CSn1	O	HyperBus Chip Select 1	0x1
RESETn	MCU_HYPERBUS0_RESETn	O	HyperBus Reset Output	0x1
RESETOn	MCU_HYPERBUS0_RESETOn	I	Reset State Indicator (output from HyperBus memory)	0x1
INTn	MCU_HYPERBUS0_INTn	I	Memory Interrupt (output from HyperBus memory)	0x1
WPn <sup>(3)</sup>	MCU_HYPERBUS0_WPn	O	HyperBus Write Protect (output to HyperBus memory)	0x1

(1) n - Active Low

(2) I = Input; O = Output; HiZ = High Impedance

(3) WPn pin is not used on Cypress flash devices.

---

**Note**

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables Pin Attributes and Pin Multiplexing in the device-specific Datasheet.

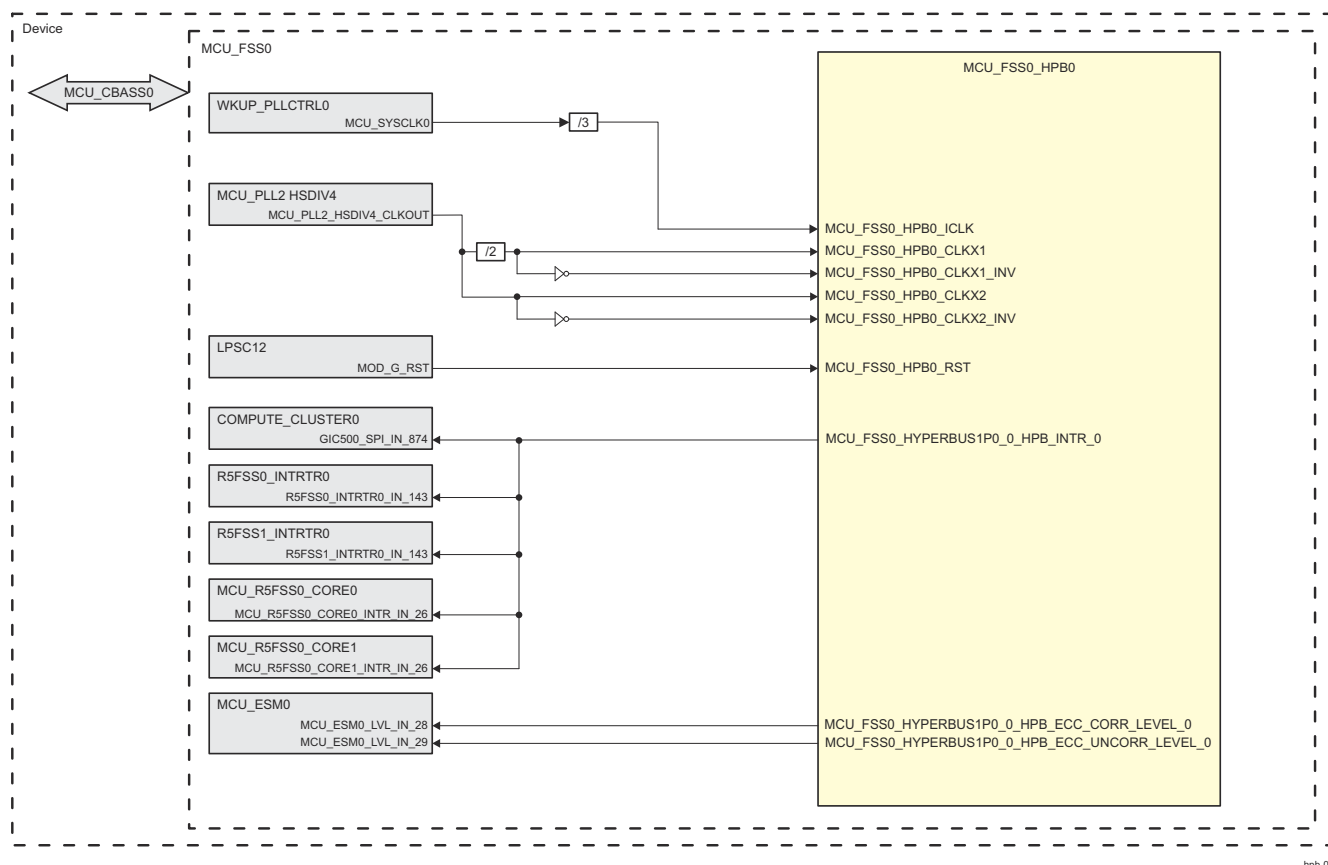
---

### 12.3.3.3 HyperBus Integration

This section describes the HyperBus integration in the device, including information about clocks, resets, and hardware requests.

#### 12.3.3.3.1 HyperBus Integration in MCU Domain

There is one HyperBus module integrated in the device MCU domain - MCU\_FSS0\_HPBO. Figure 12-229 shows the integration of MCU\_FSS0\_HPBO.



**Figure 12-229. MCU\_FSS0\_HPBO Integration**

Table 12-306 through Table 12-308 summarize the integration of MCU\_FSS0\_HPBO in the device MCU domain.

**Table 12-306. MCU\_FSS0\_HPBO Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
MCU_FSS0_HPBO	WKUP_PSC0	PD0	LPSC12	MCU_CBASS0

**Table 12-307. MCU\_FSS0\_HPBO Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
MCU_FSS0_HPBO	MCU_FSS0_HPBO_ICLK	MCU_SYCLK0/3	WKUP_PLLC TRLO	Interface Clock

**Table 12-307. MCU\_FSS0\_HPBO Clocks and Resets (continued)**

MCU_FSS0_HPBO_CLKX1	MCU_PLL2_HSDIV4_CLKO UT/2	MCU_PLL2 HSDIV4	Functional Clock 1 (main functional clock)
MCU_FSS0_HPBO_CLKX1 _INV	INV(MCU_PLL2_HSDIV4_C LKOUT/2)		Functional Clock 2 (inverted Functional Clock 1 to create 180 degree phase shift)
MCU_FSS0_HPBO_CLKX2	MCU_PLL2_HSDIV4_CLKO UT		Functional Clock 3 (2 × Functional Clock 1 for DDR data and command)
MCU_FSS0_HPBO_CLKX2 _INV	INV(MCU_PLL2_HSDIV4_C LKOUT)		Functional Clock 4 (inverted Functional Clock 3 to create 180 degree phase shift)

Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
MCU_FSS0_HPBO	MCU_FSS0_HPBO_RST	MOD_G_RST	LPSC12	HyperBus Reset

**Table 12-308. MCU\_FSS0\_HPBO Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCU_FSS0_HPBO	MCU_FSS0_HYPERBUS1P0_0_HPBO_INTR_0	GIC500_SPI_IN_874	COMPUTE_CLUSTER0	HyperBus Interrupt Request	Level
		R5FSS0_INTRTR0_IN_143	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_143	R5FSS1_INTRTR0		
		MCU_R5FSS0_CORE0_INT R_IN_26	MCU_R5FSS0_CORE0		
		MCU_R5FSS0_CORE1_INT R_IN_26	MCU_R5FSS0_CORE1		
	MCU_FSS0_HYPERBUS1P0_0_HPBO_ECC_CORR_LEVEL_0	MCU_ESM0_LVL_IN_28	MCU_ESM0	HyperBus ECC Correctable Error (SEC) Interrupt Request	Level
	MCU_FSS0_HYPERBUS1P0_0_HPBO_ECC_UNCORR_LEVEL_0	MCU_ESM0_LVL_IN_29	MCU_ESM0	HyperBus ECC Uncorrectable Error (DEC) Interrupt Request	Level

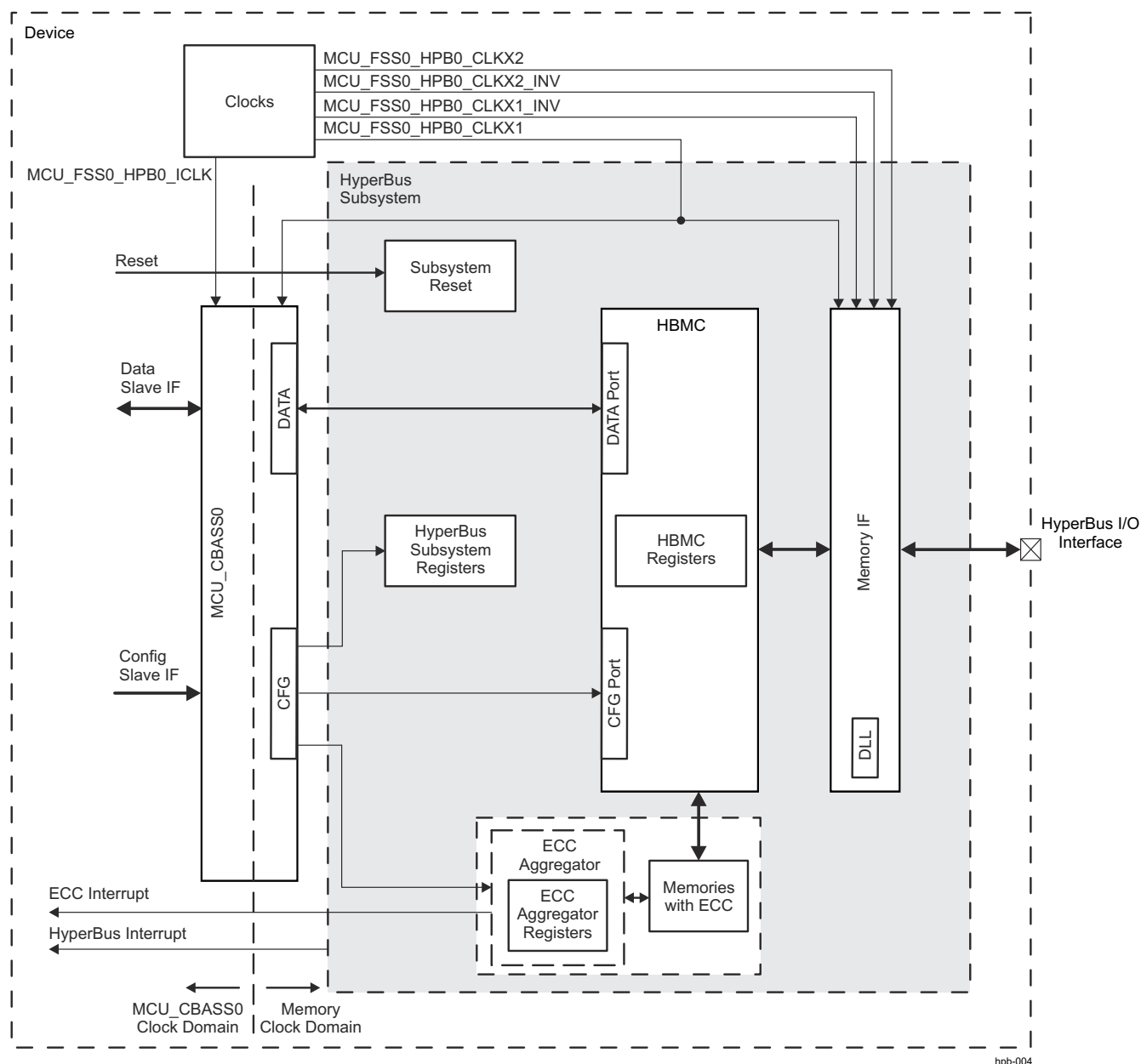
### 12.3.3.4 HyperBus Functional Description

The HyperBus module is a high throughput memory interface. It is a part of the device FSS and provides execute in place (XIP) operation and block copy access to HyperBus memory devices (HyperRAM and HyperFlash). The FSS enables in-line ECC protection and authentication features for the HyperBus module.

For more information about FSS, see *Flash Subsystem (FSS)*.

The HyperBus module is compliant to the *Cypress HyperBus™ Specification v1.2*.

Figure 12-230 shows the HyperBus module block diagram.



**Figure 12-230. HyperBus Module Block Diagram**

Basic Blocks:

- HBMC: The HyperBus Memory Controller (HBMC) is the main part (the core) of the HyperBus Subsystem and provides accessibility to external HyperBus memory (HyperFlash or HyperRAM) using simple HyperBus protocol for data read and write transfers.
  - The Data Port on the HBMC is used for CBASS data slave interface to access the external HyperBus memory.
  - The CFG Port on the HBMC is used for configuration of the HBMC registers.
- HyperBus I/O Interface: The HyperBus I/O Interface includes all used interface pins (for more information, see *HyperBus I/O Signals*).
- Memory IF: The Memory IF block implements the following functionality:
  - Convert single-rate command and data output from the HBMC to double-rate (DDR)
  - Use DLL to delay the incoming read data strobe from the memory
- FIFO Memories with ECC: There are internal HyperBus FIFOs implemented with memories that are used during read and write transactions. These FIFO memories are ECC protected (for more information about the FIFOs, see [Section 12.3.3.4.3, HyperBus Internal FIFOs](#)).
- ECC Aggregator: The ECC Aggregator facilitates aggregating and reporting internal HyperBus FIFO memory errors (for more information, see [Section 12.3.3.4.2, HyperBus ECC Support](#)).
- HBMC Registers: This block includes set of all used HBMC registers.
- HyperBus Subsystem Registers: This block implements memory-mapped registers at the HyperBus Subsystem level.
- ECC Aggregator Registers: This block includes all used ECC Aggregator registers.

For more information about all HyperBus registers, see *HyperBus Registers*.

- MCU\_CBASS0: The MCU\_CBASS0 interconnect implements the clock domain crossing bridges needed to separate the two main clock domains: MCU\_CBASS0 clock domain and Memory clock domain.
- Data Slave IF: The Data Slave IF on the interconnect is a 32-bit wide interface and is used to access an external HyperBus memory via the Data Port on the HBMC. The data interface supports linear incrementing addresses only. Cache-line wrap and fixed addressing modes are not supported.
- Config Slave IF: The Config Slave IF on the interconnect is a 32-bit wide interface and is used to access the HBMC registers (via the CFG Port on the HBMC), HyperBus Subsystem registers, and ECC Aggregator registers. The config interface also supports linear incrementing address mode only.

For more information about supported and not supported HyperBus features, see *HyperBus Overview*.

- Clocks: For more information about HyperBus Clocks, see *MCU\_FSS0\_HBP0 Clocks and Resets*.
- Subsystem Reset: The reset signal to the Subsystem Reset block provides reset to the all HyperBus Subsystem parts (for more information about HyperBus Resets, see *MCU\_FSS0\_HBP0 Clocks and Resets*).
- Interrupts: For more information, see *MCU\_FSS0\_HBP0 Clocks and Resets* and *HyperBus Interrupts*.

#### 12.3.3.4.1 HyperBus Interrupts

The HyperBus module sources one active high level HyperBus interrupt and two active high level ECC Aggregator interrupts (see *MCU\_FSS0\_HBP0 Hardware Requests*).

The HyperBus interrupt is generated based on the HyperBus memory's active low level interrupt. In the HyperBus Subsystem this active low level interrupt is converted to active high level HyperBus interrupt (see RESETOn pin in *Hyperbus I/O Signals*).

The ECC Aggregator interrupts are generated based on the ECC errors (single (correctable) and double (uncorrectable) bit errors) in the embedded HyperBus memories (for more information, see *HyperBus ECC Support*).

#### 12.3.3.4.2 HyperBus ECC Support

The Error Correcting Code (ECC) is a mechanism for providing increased system reliability via reduction of memory software errors by allowing single bit errors to be detected and corrected (SEC) and double bit errors to be detected (DED).

The SEC logic detects and corrects a single bit error (single bit error per ECC word or per ECC data segment). The DED logic only detects (does not correct) double errors (double bit errors per ECC word or per ECC data segment).

The embedded HyperBus memories are ECC protected.

#### 12.3.3.4.2.1 ECC Aggregator

##### Note

For more information about ECC Aggregator, refer to [Section 12.11.4, ECC Aggregator](#).

For more information about ECC Aggregator Registers, refer to *HyperBus Registers*.

#### 12.3.3.4.3 HyperBus Internal FIFOs

There are 10 FIFO buffers in the module that help improve performance. [Table 12-309](#) shows the HyperBus FIFOs details.

**Table 12-309. HyperBus FIFOs**

Buffer Name	Size (Width x Depth)
Address (ADR) FIFO	Two port RAM – 16 x 46-bit
Write data (WDAT) FIFO	Two port RAM – 256 x 40-bit
Write status (BDAT) FIFO	Two port RAM – 16 x 2-bit
Read data (RDAT) FIFO	Two port RAM – 128 x 40-bit
Receive (RX) FIFO	Two port RAM – 256 x 20-bit
Address write (AW) FIFO	Two port RAM – 16 x 44-bit
Write ID (WID) FIFO	Two port RAM – 16 x 2-bit
Address write ID (AWID) FIFO	Two port RAM – 16 x 14-bit
Address read (AR) FIFO	Two port RAM – 16 x 44-bit
Address read ID (ARID) FIFO	Two port RAM – 16 x 18-bit

#### 12.3.3.4.4 HyperBus Data Regions

For more information about HyperBus data regions, refer to [Section 12.3.1.4.7, FSS Memory Regions](#).

#### 12.3.3.4.5 HyperBus True Continuous Read (TCR) Mode

The HBMC supports True Continuous Read (TCR) operation for HyperFlash. This feature is able to improve HyperFlash read performance.

When this feature is enabled, the HBMC can merge multiple up to 16 max burst read commands that are present in its command FIFO into a single memory read transaction to the HyperFlash memory. As a result, the command input and initial latency for the latter burst read commands are saved, which leads to improved throughput.



### 12.3.3.5 HyperBus Programming Guide

#### 12.3.3.5.1 HyperBus Initialization Sequence

The HBMC provides two sets of Memory Base Address, Memory Configuration and Memory Timing registers to access devices connected to the two external chip selects. The following programming guidelines need to use the correct set of registers depending on which chip select is accessed.

The HBMC compares the input address bits [31-24] to the value programmed in the MCU\_FSS0\_HPB0\_MC\_MBAR\_y registers to determine the external chip select that is accessed.

##### 12.3.3.5.1.1 HyperFlash Access

**Table 12-310. HyperFlash Access Sequence**

Step	Description
1.	Ensure that the FIFO RAM auto-initialization is complete by reading the MCU_FSS0_HPB0_SS_RAM_STAT_REG[0] INIT_DONE bit .
2.	<p>Lock MDLL:</p> <ul style="list-style-type: none"> <li>- To ensure that the MDLL in the HyperBus is locked, the reset to the HyperBus module should be de-asserted only after all the clock inputs are stable at the desired operating frequency (see <i>MCU_FSS0_HPB0 Clocks and Resets</i>). The MDLL can lose the lock if the frequencies of the clock inputs to the HyperBus module are changed during operation.</li> <li>- To ensure that the MDLL is stabilized, the following sequence is required. Boot code should attempt to read 64 bytes of Flash data, for 16 iterations, and if the data is the same in 4 successive iterations, the DLL can be considered to be stabilized and the software proceed with normal Flash access .</li> <li>- The MCU_FSS0_HPB0_SS_DLL_STAT_REG[0] MDLL_LOCK and MCU_FSS0_HPB0_SS_DLL_STAT_REG[1] SDL_LOCK bits can be used to determine if the master delay line and slave delay line have locked, respectively .</li> </ul>
3.	<p>Initialize Memory Configuration register (MCU_FSS0_HPB0_MC_MCR_y):</p> <ul style="list-style-type: none"> <li>- MCU_FSS0_HPB0_MC_MCR_y[31] MAXEN bit and MCU_FSS0_HPB0_MC_MCR_y[26-18] MAXLEN bit field based on burst transaction length to memory.</li> <li>- MCU_FSS0_HPB0_MC_MCR_y[17] TCMO = 1h, to enable HBMC to merge multiple accesses with sequential addresses into a single memory access. This will improve memory throughput.</li> <li>- MCU_FSS0_HPB0_MC_MCR_y[16] ACS = 0h, to set 'No asymmetry cache system support'.</li> <li>- MCU_FSS0_HPB0_MC_MCR_y[5] CRT = 0h, to set 'Memory space' instead 'CR space'.</li> <li>- MCU_FSS0_HPB0_MC_MCR_y[4] DEVTYPE = 0h, to set 'HyperFlash' instead 'HyperRAM'.</li> <li>- MCU_FSS0_HPB0_MC_MCR_y[1-0] WRAPSIZE = 00h, since the HBMC does not support wrap bursts.</li> </ul>
4.	Initialize Memory Timing register (MCU_FSS0_HPB0_MC_MTR_y) based on timing of the memory device being used.
5.	<p>Initialize Memory Base Address register (MCU_FSS0_HPB0_MC_MBAR_y):</p> <ul style="list-style-type: none"> <li>- MCU_FSS0_HPB0_MC_MBAR_y[31-24] A_MSB = 8 MSB bits of memory address space. This will define the start of the 16 MB address region in the system memory where the HyperFlash can be accessed. The HBMC will initiate HyperFlash access to any memory mapped access in this range.</li> </ul>
6.	Check the MCU_FSS0_HPB0_MC_CSR[10] RRSTOERR bit in Controller Status register to ensure that the HyperFlash is out of reset.
7.	Normal HyperFlash access can be performed after this.
8.	The HyperFlash device registers and CFI (Common Flash Interface) region can be accessed by read/write transactions to the offset from the base address as specified in the device command summary table.
9.	The HyperFlash memory data array can be read as memory mapped access.
10.	The HyperFlash write sequence needs to be followed to update contents of the memory array.

### 12.3.3.5.1.2 HyperRAM Access

**Table 12-311. HyperRAM Access Sequence**

Step	Description
1.	Wait 150 $\mu$ s for RAM to power up after reset. The HyperRAM does not provide any feedback on reset/ready state to the HBMC .
2.	Ensure that the FIFO RAM auto-initialization is complete by reading the MCU_FSS0_HPB0_SS_RAM_STAT_REG[0] INIT_DONE bit .
3.	<p>Lock MDLL:</p> <ul style="list-style-type: none"> <li>- To ensure that the MDLL in the HyperBus is locked, the reset to the HyperBus module should be de-asserted only after all the clock inputs are stable at the desired operating frequency (see <i>MCU_FSS0_HPB0 Clocks and Resets</i>). The MDLL can lose the lock if the frequencies of the clock inputs to the HyperBus module are changed during operation.</li> <li>- To ensure that the MDLL is stabilized, the following sequence is required. Boot code should attempt to read 64 bytes of RAM data, for 16 iterations, and if the data is the same in 4 successive iterations, the DLL can be considered to be stabilized and the software proceed with normal RAM access .</li> <li>- The MCU_FSS0_HPB0_SS_DLL_STAT_REG[0] MDLL_LOCK and MCU_FSS0_HPB0_SS_DLL_STAT_REG[1] SDL_LOCK bits can be used to determine if the master delay line and slave delay line have locked, respectively .</li> </ul>
4.	<p>Initialize Memory Configuration register (MCU_FSS0_HPB0_MC_MCR_y):</p> <ul style="list-style-type: none"> <li>- MCU_FSS0_HPB0_MC_MCR_y[31] MAXEN bit and MCU_FSS0_HPB0_MC_MCR_y[26-18] MAXLEN bit field based on burst transaction length to memory.</li> <li>- MCU_FSS0_HPB0_MC_MCR_y[17] TCMO = 0h, since HBMC does not support command merging for HyperRAM accesses.</li> <li>- MCU_FSS0_HPB0_MC_MCR_y[16] ACS = 0h, to set 'No asymmetry cache system support'.</li> <li>- MCU_FSS0_HPB0_MC_MCR_y[5] CRT = 0h or 1h, depending on what needs to be accessed (memory or register space).</li> <li>- MCU_FSS0_HPB0_MC_MCR_y[4] DEVTYPE = 1h, to set 'HyperRAM' instead 'HyperFlash'.</li> <li>- MCU_FSS0_HPB0_MC_MCR_y[1-0] WRAPSIZE = 00h, since the HBMC does not support wrap bursts.</li> </ul>
5.	Initialize MCU_FSS0_HPB0_MC_MTR_y register based on timing of the memory device being used.
6.	<p>Initialize MCU_FSS0_HPB0_MC_MCR_y register:</p> <ul style="list-style-type: none"> <li>- MCU_FSS0_HPB0_MC_MCR_y[31-24] A_MSB = 8 MSB bits of memory address space. This will define the start of the 16 MB address region in the system memory where the HyperRAM can be accessed. The controller will initiate HyperRAM access to any memory mapped access in this range.</li> </ul>
7.	Check the MCU_FSS0_HPB0_SS_DLL_STAT_REG[0] MDLL_LOCK bit to ensure the Master DLL is locked.
8.	Normal HyperRAM access can be performed after this.
9.	The HyperRAM device registers can be accessed by read/write transactions to the offset from the base address as specified in the device command summary table with MCU_FSS0_HPB0_MC_MCR_y[5] CRT bit set to register space.
10.	The HyperRAM memory data array can be read/written to as memory mapped access with MCU_FSS0_HPB0_MC_MCR_y[5] CRT bit set to memory space.

#### Note

HyperRAM devices have a 150  $\mu$ s startup time. Software needs to wait this duration after reset to initiate transactions to a HyperRAM device.

### 12.3.3.5.2 HyperBus Real-time Operating Requirements

The HBMC may assert the memory interrupt based on the status of the memory transaction. Software needs to handle this interrupt using the MCU\_FSS0\_HPB0\_MC\_ISR register. The MCU\_FSS0\_HPB0\_MC\_ISR register can provide the status information to determine the cause of the interrupt.

#### **12.3.3.5.3 HyperBus Power Up/Down Sequence**

Software needs to ensure that there are no pending transactions before stopping clock and/or power to the HBMC.

### 12.3.4 General-Purpose Memory Controller (GPMC)

This section describes the General-Purpose Memory Controller (GPMC) for the device.

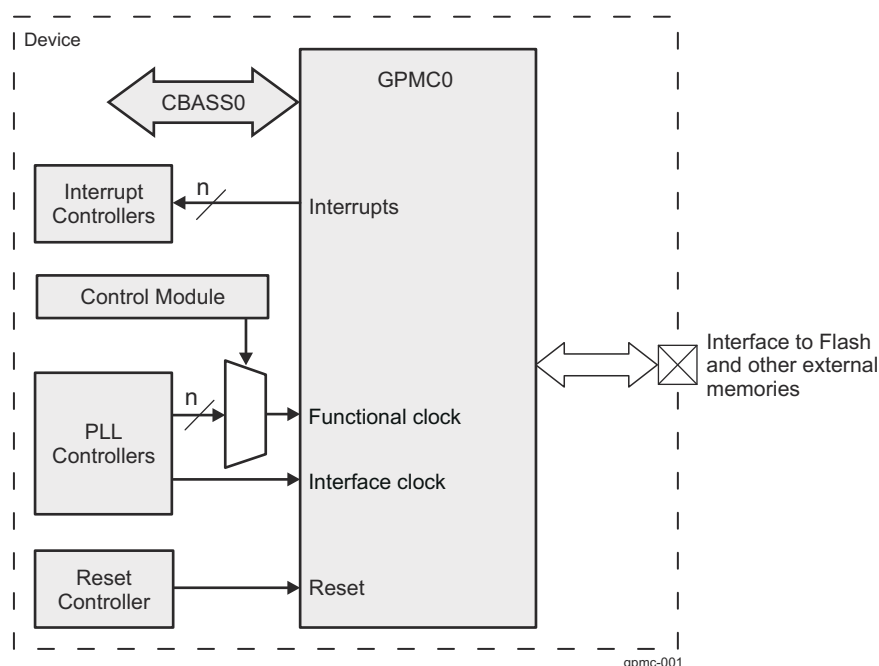
#### 12.3.4.1 GPMC Overview

The General-Purpose Memory Controller is a unified memory controller dedicated for interfacing with external memory devices like:

- Asynchronous SRAM-like memories and application-specific integrated circuit (ASIC) devices
- Asynchronous, synchronous, and page mode (available only in non-multiplexed mode) burst NOR flash devices
- NAND flash
- Pseudo-SRAM devices

shows the GPMC module allocation within device domains.

Figure 12-231 shows the GPMC0 module overview.



**Figure 12-231. GPMC0 Overview**

#### 12.3.4.1.1 GPMC Features

The main features of the GPMC are:

- 8- or 16-bit-wide data path to external memory device
- Supports up to 4 chip select regions of programmable size and programmable base addresses in a total address space of 1GB
- Supports on-the-fly error code detection using the Bose-Chaudhuri-Hocquenghem (BCH) ( $t = 4, 8, \text{ or } 16$ ) or Hamming code to improve the reliability of NAND with a minimum effect on software (NAND flash with 512-byte page size or greater)
- Fully pipelined operation for optimal memory bandwidth usage
- The clock to the external memory is provided from GPMC\_FCLK divided by 1, 2, 3, or 4
- Supports programmable autoclock gating when no access is detected
- Independent and programmable control signal timing parameters for setup and hold time on a per-chip basis. Parameters are set according to the memory device timing parameters with a timing granularity of one GPMC\_FCLK clock cycle.

- Flexible internal access time control (wait state) and flexible handshake mode using external WAIT pin monitoring
- Support bus keeping
- Support bus turnaround
- Prefetch and write-posting engine associated with DMA controller at system level to achieve full performance from the NAND device with minimum effect on NOR/SRAM concurrent access
- 32-bit interconnect slave interface which supports non-wrapping and wrapping burst of up to 16x32 bits.

The GPMC supports the following various access types:

- Asynchronous read/write access
- Asynchronous read page access (4, 8, and 16 Word16)
- Synchronous read/write access
- Synchronous read/write burst access without wrap capability (4, 8 and 16 Word16)
- Synchronous read/write burst access with wrap capability (4, 8 and 16 Word16)
- Address-data-multiplexed (AD) access
- Address-address-data (AAD) multiplexed access
- Little-endian access only

The GPMC can communicate with a wide range of external devices:

- External asynchronous or synchronous 8-bit wide memory or device (non burst device)
- External asynchronous or synchronous 16-bit wide memory or device
- External 16-bit non-multiplexed NOR flash device
- External 16-bit address and data multiplexed NOR Flash device
- External 8-bit and 16-bit NAND flash device
- External 16-bit pseudo-SRAM (pSRAM) device

---

#### **Note**

Page mode is available only in non-multiplexed mode.

---

#### **12.3.4.1.2 GPMC Not Supported Features**

The following features are not supported on this family of devices:

- DMA mode is not supported.
- Asynchronous page write mode is not supported.
- Multiple write access in asynchronous mode is not supported.
- Multiple read access in asynchronous mode is not supported in address/data-multiplexed and AAD-multiplexed modes.

### 12.3.4.2 GPMC Environment

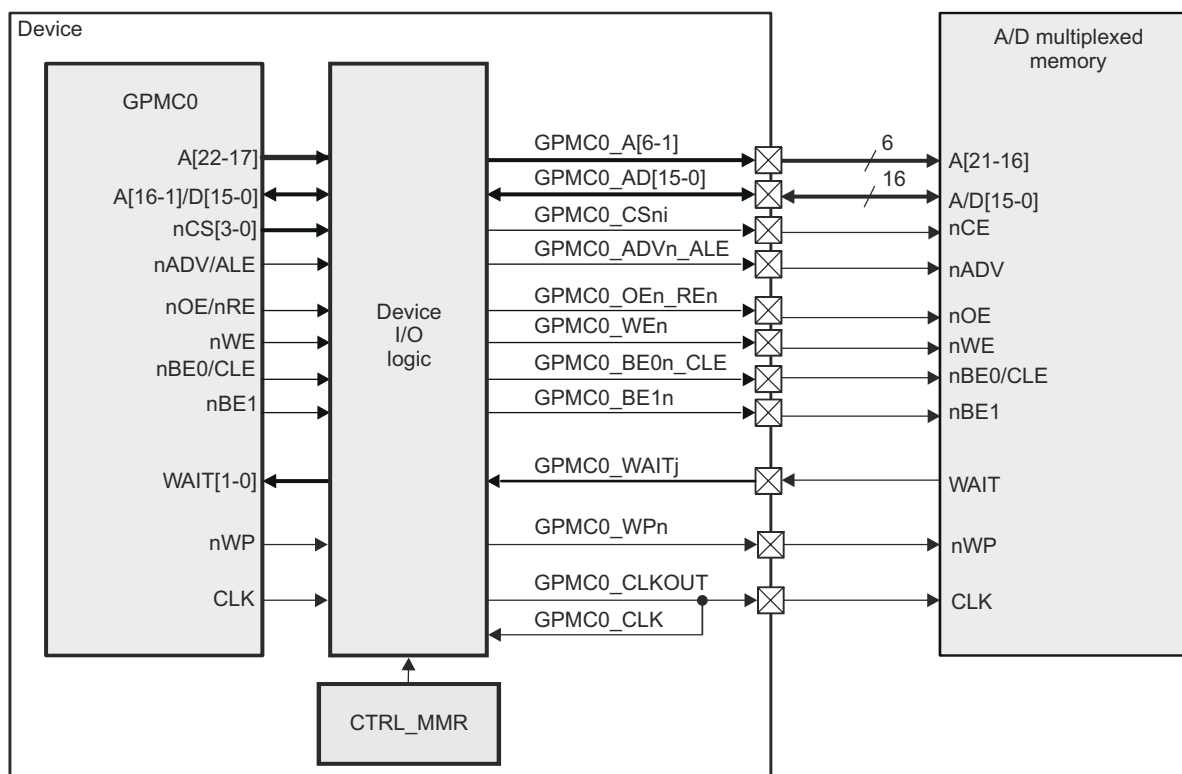
The GPMC0 module is hereinafter referred to as GPMC.

This section describes the GPMC0 external connections (environment).

#### 12.3.4.2.1 GPMC Modes

This section shows four GPMC0 external connection options:

- [Figure 12-232](#) shows a connection between the GPMC0 and a 16-bit synchronous address/data-multiplexed (or AAD-multiplexed but this protocol uses fewer address pins) external memory device.
- [Figure 12-233](#) shows a connection between the GPMC0 and a 16-bit synchronous non-multiplexed external memory device.
- [Figure 12-234](#) shows a connection between the GPMC0 and an 8-bit synchronous non-multiplexed external memory device.
- [Figure 12-235](#) shows a connection between the GPMC0 and an 8-bit NAND device.

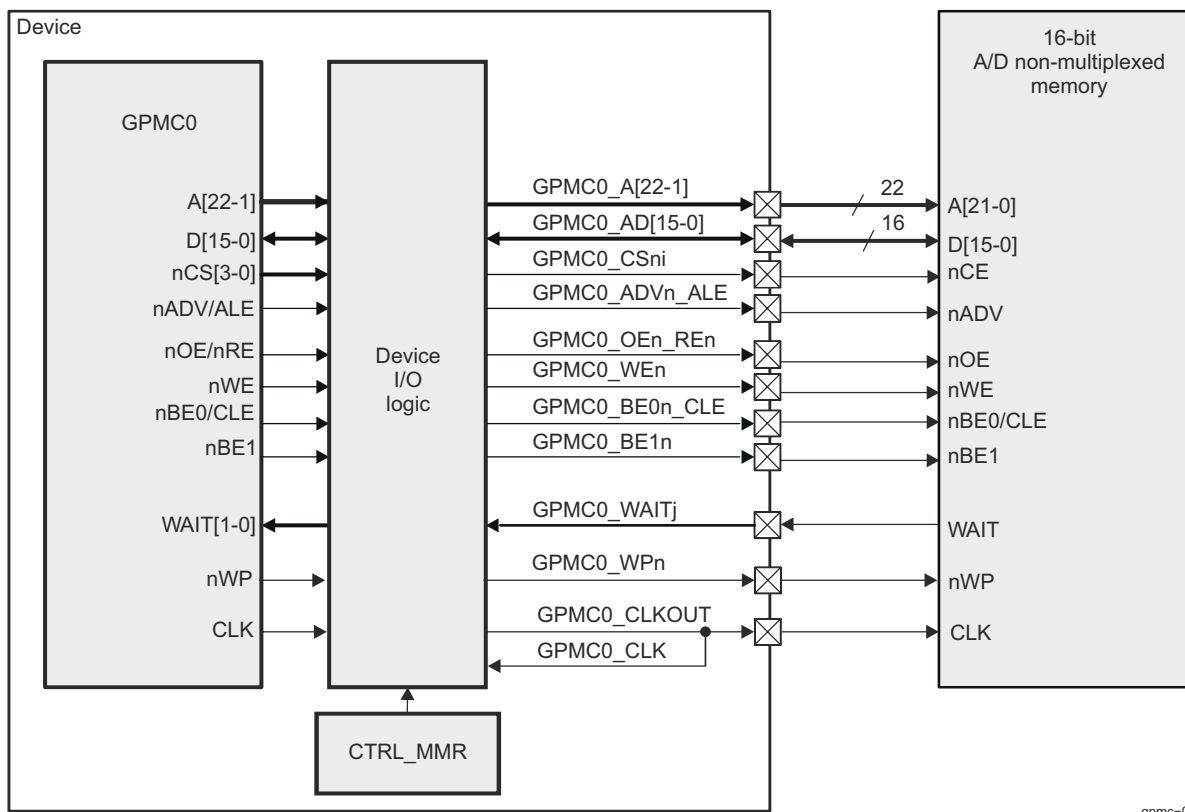


gpmc-002

*i* = 0 to 3

*j* = 0 to 1

**Figure 12-232. GPMC to 16-Bit Address/Data-Multiplexed Memory**

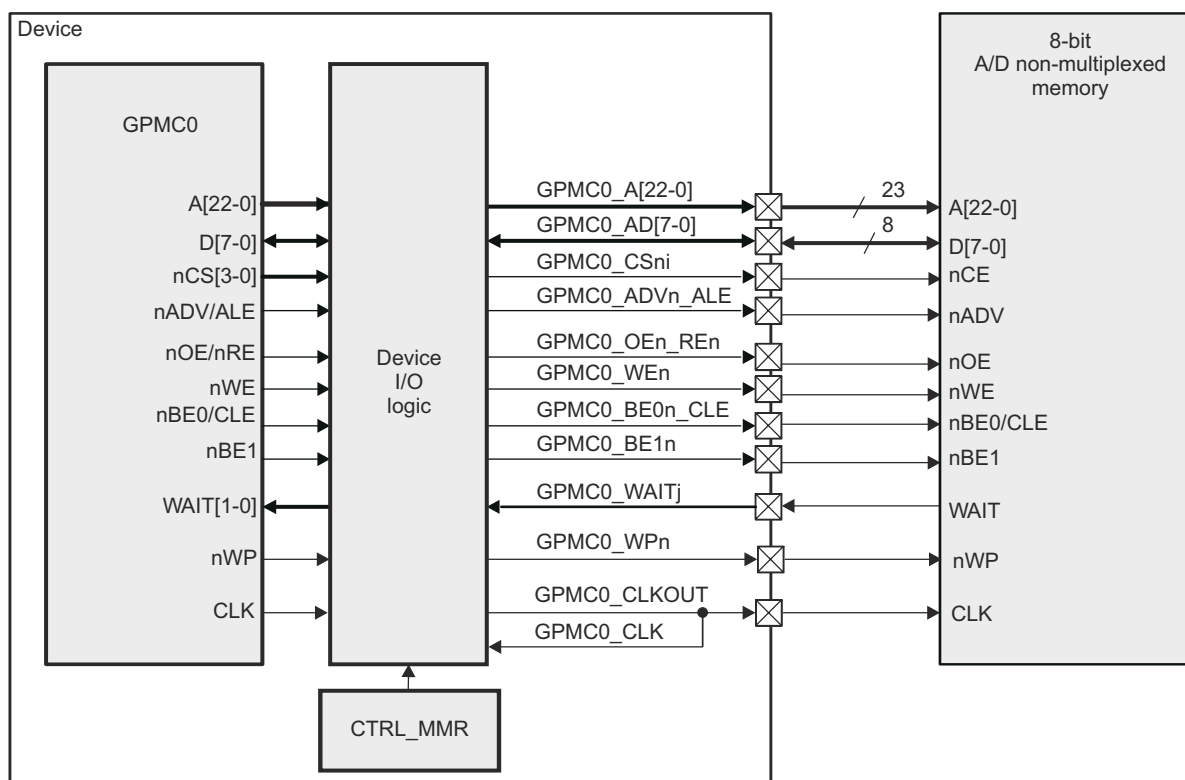


gpmc-045

i = 0 to 3

j = 0 to 1

**Figure 12-233. GPMC to 16-Bit Non-Multiplexed Memory**

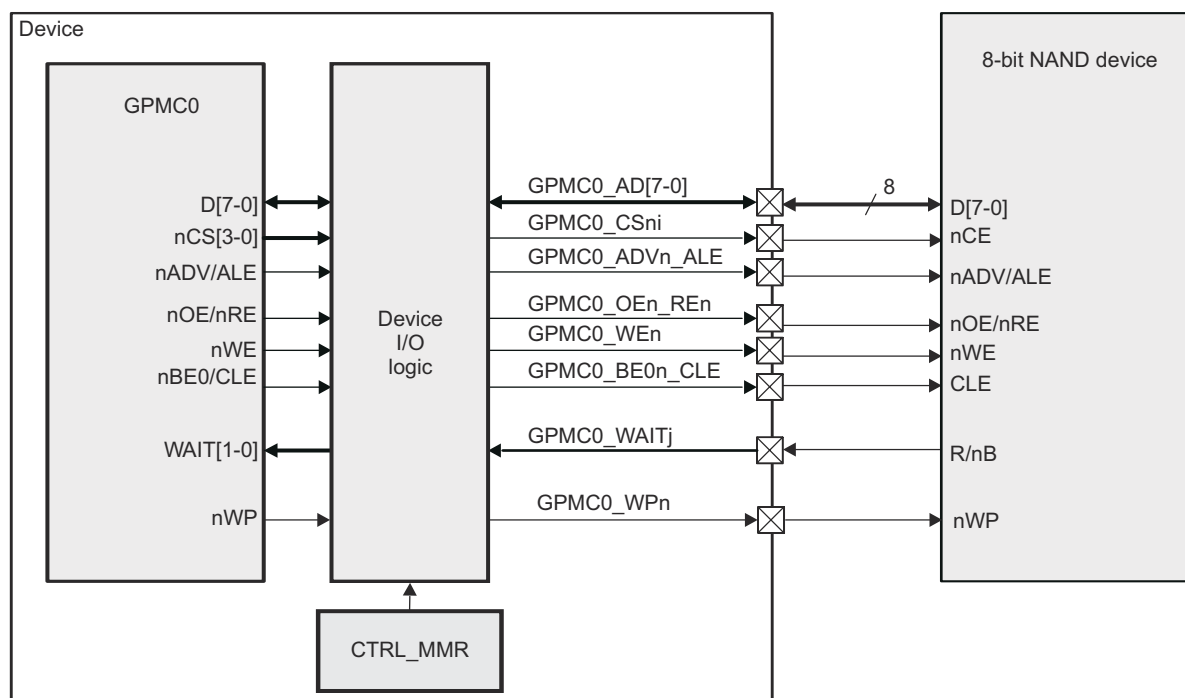


gpmc-045a

i = 0 to 3

j = 0 to 1

**Figure 12-234. GPMC to 8-Bit Non-Multiplexed Memory**



gpmc-003

i = 0 to 3



j = 0 to 1

**Figure 12-235. GPMC to 8-Bit NAND Device**

#### 12.3.4.2.2 GPMC I/O Signals

Table 12-312 lists the GPMC subsystem input/output (I/O) pins.

**Table 12-312. GPMC I/O Signals (Master Mode)**

Module Pin	Device Level Signal	I/O <sup>(1)</sup>	Description	Module Pin Reset Value <sup>(2)</sup>
<b>GPMC0</b>				
A[22-0]	GPMC0_A[22-0]	O	23-bit output address bus	-
A[16-1]/D[15-0]	GPMC0_AD[15-0]	I/O	Multiplexed address/data	-
nCS[3-0]	GPMC0_CS[n][3-0]	O	Chip-selects (active low)	-
CLK	GPMC0_CLKOUT	O	Clock generated for the external memory or device. For more information, see <a href="#">Figure 12-236</a> .	-
RET_CLK				
N/A	GPMC0_FCLK_MUX	O	Free running clock. GPMC functional clock (GPMC0_FCLK) propagated on a device pad. For more information on the GPMC0_FCLK_MUX integration, see <a href="#">Figure 12-236</a> .	-
nADV/ALE	GPMC0_ADVn_ALE	O	Address valid (active low). Also used as address latch enable (active high) for NAND protocol memories.	-
nOE/nRE	GPMC0_OEn_REn	O	Output enable (active low). Also used as read enable (active low) for NAND protocol memories.	-
nWE	GPMC0_WEn	O	Write enable (active low)	-
nBE0/CLE	GPMC0_BE0n_CLE	O	Lower-byte enable (active low). Also used as command latch enable for NAND protocol memories.	-
nBE1	GPMC0_BE1n	O	Upper-byte enable (active low)	-
WAIT[1-0]	GPMC0_WAIT[1-0]	I	External wait signal for NOR and NAND protocol memories. Can be mapped on any of the chip-selects.	-
nWP	GPMC0_WPn	O	Write protect (active low)	-
DIR	GPMC0_DIR	O	This signal can be used to control an external buffer direction. Also controls the signal direction of D[15-0]. Low during transmit (for write access: data OUT from GPMC0 to memory). High during receive (for read access: data IN from memory to GPMC0).	-

(1) I = Input; O = Output; I/O - Bidirectional

(2) HiZ = High Impedance

#### Note

For GPMC output clock signal (CLK) to work properly, the RXACTIVE bit of the appropriate CTRLMMR\_PADCONFIGy registers should be set to 0x1 because of retiming purposes.

#### Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

Table 12-313 shows the use of address and data GPMC pins based on the type of external device.

**Table 12-313. GPMC Pin Multiplexing Options**

<b>GPMC Pin</b>	<b>Multiplexed Address Data 16-Bit Device</b>	<b>non-multiplexed Address Data 16-Bit Device (incomplete 28-bit address range)</b>	<b>non-multiplexed Address Data 8-Bit Device (incomplete 28-bit address range)</b>	<b>16-Bit NAND Device</b>	<b>8-Bit NAND Device</b>
GPMC0_A[22]	Not used	A22	A22	Not used	Not used
GPMC0_A[21]	Not used	A21	A21	Not used	Not used
GPMC0_A[20]	Not used	A20	A20	Not used	Not used
GPMC0_A[19]	Not used	A19	A19	Not used	Not used
GPMC0_A[18]	Not used	A18	A18	Not used	Not used
GPMC0_A[17]	Not used	A17	A17	Not used	Not used
GPMC0_A[16]	Not used	A16	A16	Not used	Not used
GPMC0_A[15]	Not used	A15	A15	Not used	Not used
GPMC0_A[14]	Not used	A14	A14	Not used	Not used
GPMC0_A[13]	Not used	A13	A13	Not used	Not used
GPMC0_A[12]	Not used	A12	A12	Not used	Not used
GPMC0_A[11]	Not used	A11	A11	Not used	Not used
GPMC0_A[10]	A26	A10	A10	Not used	Not used
GPMC0_A[9]	A25	A9	A9	Not used	Not used
GPMC0_A[8]	A24	A8	A8	Not used	Not used
GPMC0_A[7]	A23	A7	A7	Not used	Not used
GPMC0_A[6]	A22	A6	A6	Not used	Not used
GPMC0_A[5]	A21	A5	A5	Not used	Not used
GPMC0_A[4]	A20	A4	A4	Not used	Not used
GPMC0_A[3]	A19	A3	A3	Not used	Not used
GPMC0_A[2]	A18	A2	A2	Not used	Not used
GPMC0_A[1]	A17	A1	A1	Not used	Not used
GPMC0_A[0] <sup>(1)</sup>	A0 - Not used	Not used	A0	Not used	Not used
GPMC0_AD[15]	A16/D15	D15	Not used	D15	Not used
GPMC0_AD[14]	A15/D14	D14	Not used	D14	Not used
GPMC0_AD[13]	A14/D13	D13	Not used	D13	Not used
GPMC0_AD[12]	A13/D12	D12	Not used	D12	Not used
GPMC0_AD[11]	A12/D11	D11	Not used	D11	Not used
GPMC0_AD[10]	A11/D10	D10	Not used	D10	Not used
GPMC0_AD[9]	A10/D9	D9	Not used	D9	Not used
GPMC0_AD[8]	A9/D8	D8	Not used	D8	Not used
GPMC0_AD[7]	A8/D7	D7	D7	D7	D7
GPMC0_AD[6]	A7/D6	D6	D6	D6	D6
GPMC0_AD[5]	A6/D5	D5	D5	D5	D5
GPMC0_AD[4]	A5/D4	D4	D4	D4	D4
GPMC0_AD[3]	A4/D3	D3	D3	D3	D3
GPMC0_AD[2]	A3/D2	D2	D2	D2	D2
GPMC0_AD[1]	A2/D1	D1	D1	D1	D1
GPMC0_AD[0]	A1/D0	D0	D0	D0	D0

(1) Used to effectively address 8-bit (only) non-multiplexed memories

With all device types, the GPMC does not drive unnecessary address lines. They stay at their reset value of 0x0.

Address mapping supports address/data-multiplexed 16-bit-wide devices:

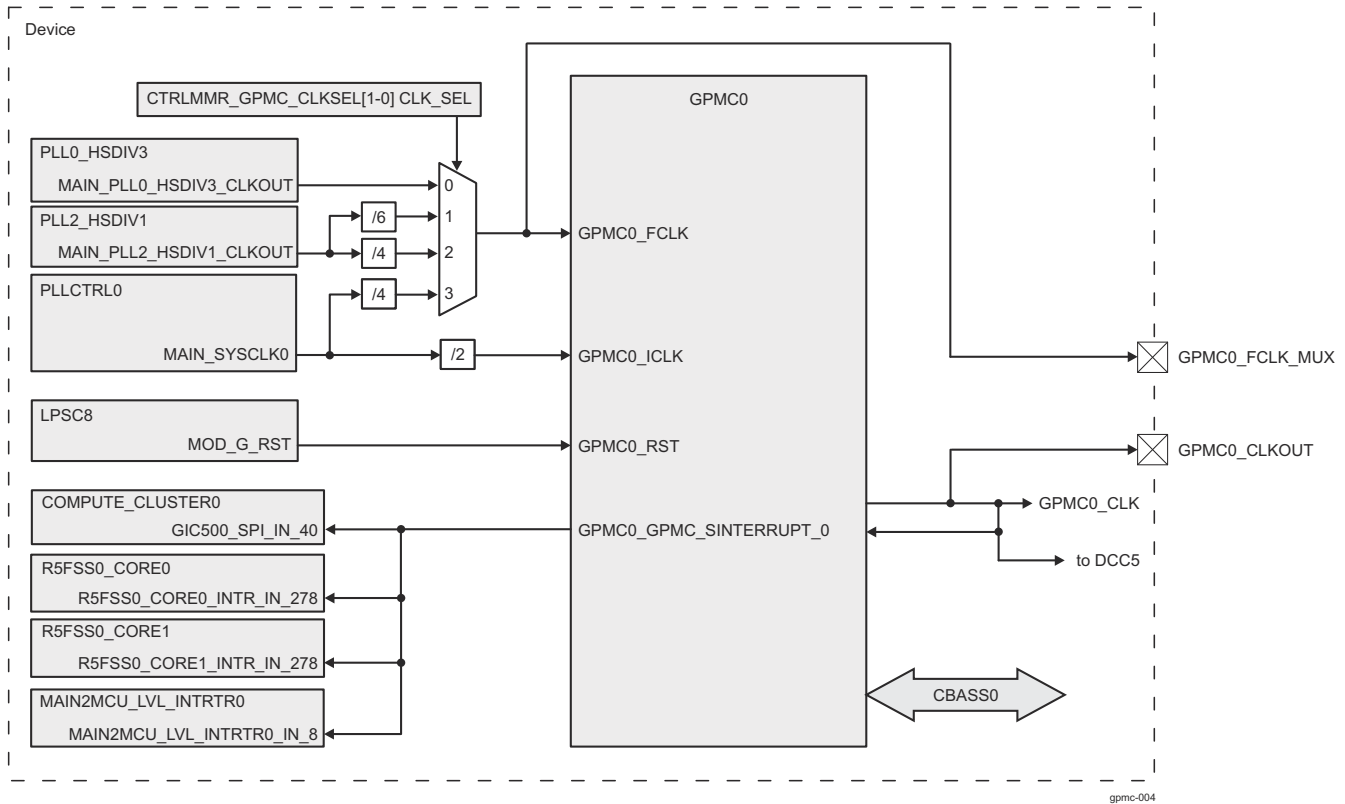
- The NOR flash memory controller still supports non-multiplexed address and data memory devices.
- Multiplexing mode can be selected through the GPMC\_CONFIG1\_i[9-8] MUXADDDATA bit field (where i = 0 to 3).

#### 12.3.4.3 GPMC Integration

This section describes module integration in the device, including information about clocks, resets, and hardware requests.

### 12.3.4.3.1 GPMC Integration in MAIN Domain

A single GPMC module is integrated in the device MAIN domain - GPMC0. Figure 12-236 shows the GPMC0 integration.



**Figure 12-236. GPMC0 Integration**

Table 12-314 through Table 12-316 summarize the integration of GPMC0 in device MAIN domain.

**Table 12-314. GPMC0 Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
GPMC0	PSC0	PD0	LPSC8	CBASS0

**Table 12-315. GPMC0 Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
GPMC0	GPMC0_FCLK	MAIN_PLL0_HSDIV3_CLKO UT	PLL0_HSDIV3	GPMC0 Functional Clock. For more information about clock multiplexing, see CTRLMMR_GPMC_CLKSEL[1-0] CLK_SEL in <i>Control Module (CTRL_MMR)</i> .
		MAIN_PLL2_HSDIV1_CLKO UT/6	PLL2_HSDIV1	
		MAIN_PLL2_HSDIV1_CLKO UT/4		
		MAIN_SYSCLK0/4	PLLCTRL0	
	GPMC0_ICLK	MAIN_SYSCLK0/2	PLLCTRL0	GPMC0 Interface Clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
GPMC0	GPMC0_RST	MOD_G_RST	LPSC8	GPMC0 Asynchronous Reset

**Table 12-316. GPMC0 Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
GPMC0	GPMC0_GPMC_SINTER RUPT_0	GIC500_SPI_IN_40	COMPUTE_CLUSTER0	GPMC0 Interrupt Request	Level
		R5FSS0_CORE0_INTR_IN_278	R5FSS0_CORE0	GPMC0 Interrupt Request	
		R5FSS0_CORE1_INTR_IN_278	R5FSS0_CORE1	GPMC0 Interrupt Request	
		MAIN2MCU_LVL_INTRTR0_IN_8	MAIN2MCU_LVL_INTR TR0	GPMC0 Interrupt Request	

### Note

GPMC0 interrupts are further described in [Section 12.3.4.4.4](#), *GPMC Interrupt Requests*.

#### 12.3.4.4 GPMC Functional Description

The GPMC basic programming model offers maximum flexibility to support various access protocols for each of the four configurable chip-selects. Use optimal chip-select settings, based on the characteristics of the external device:

- Different protocols can be selected to support generic asynchronous or synchronous random-access devices (NOR flash, SRAM) or to support specific NAND devices.
- The address and data bus can be multiplexed on the same external bus.
- Read and write access can be independently defined as asynchronous or synchronous.
- System requests (byte, 16-bit word, burst) are performed through single or multiple accesses. External access profiles (single, multiple with optimized burst length, native- or emulated-wrap) are based on external device characteristics (supported protocol, bus width, data buffer size, native-wrap support).
- System burst read or write requests are synchronous-burst (multiple-read or multiple-write). When neither burst nor page mode is supported by external memory or ASIC devices, system burst read or write requests are translated to successive single synchronous or asynchronous accesses (single reads or single writes). 8-bit wide devices are supported only in single synchronous or single asynchronous read or write mode.
- To simulate a programmable internal-wait-state, an external WAIT pin can be monitored to dynamically control external access at the beginning (initial access time) of and during a burst access.

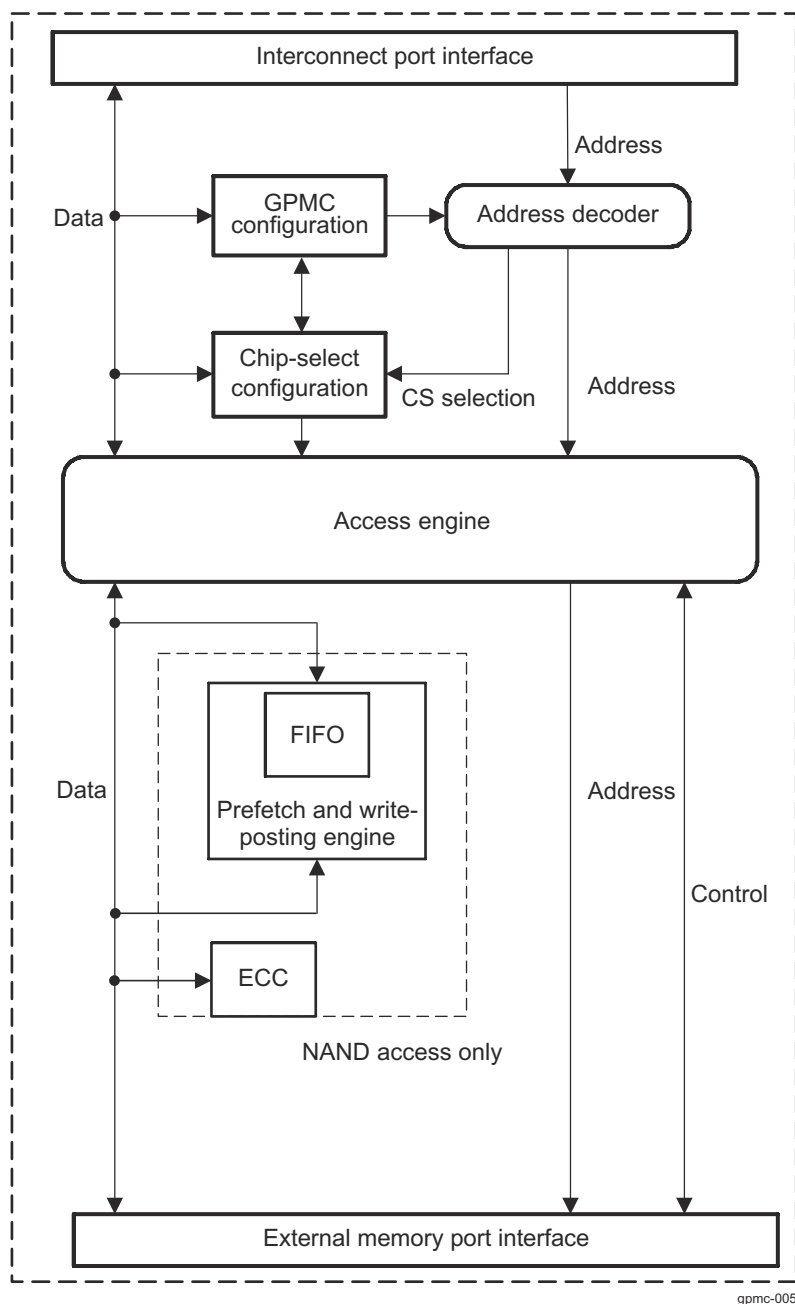
Each control signal is controlled independently for each chip-select. The internal functional clock of the GPMC (GPMC\_FCLK) is used as a time reference to specify the following:

- Read- and write-access duration
- Most GPMC external interface control-signal assertion and deassertion times
- Data-capture time during read access
- External wait-pin monitoring time
- Duration of idle time between accesses, when required

##### 12.3.4.4.1 GPMC Block Diagram

[Figure 12-237](#) shows the GPMC functional block diagram. The GPMC consists of six blocks:

- Interconnect port interface
- Address decoder, GPMC configuration, and chip-select configuration register file
- Access engine
- Prefetch and write-posting engine
- Error correction code engine (ECC)
- External device/memory port interface



**Figure 12-237. GPMC Block Diagram**

The GPMC can access various external devices. The flexible programming model allows a wide range of attached device types and access schemes.

Based on the programmed configuration bit fields stored in the GPMC registers, the GPMC can generate the timing of all control signals depending on the attached device and access type.

Given the chip-select decoding and its associated configuration registers, the GPMC selects the appropriate control signal timing for the device type.

#### 12.3.4.4.2 GPMC Clock Configuration

Table 12-317 describes the GPMC clocks.



**Table 12-317. GPMC Clocks**

Signal	I/O <sup>(1)</sup>	Description
GPMC_FCLK	I	Functional clock
GPMC_ICLK	I	Interface clock
CLK (GPMC_CLKOUT pin)	O	External clock provided to synchronous external memory devices and to DCC5 in the device.

(1) I = Input; O = Output; I/O - Bidirectional

The GPMC output clock (CLK) is generated by the GPMC from the internal GPMC\_FCLK clock. The source of the GPMC\_FCLK is described in *GPMC0 Clocks*. The GPMC output clock is configured using the GPMC\_CONFIG1\_i[1-0] GPMCFCLKDIVIDER bit field (where i = 0 to 3), as shown in [Table 12-318](#).

**Table 12-318. GPMC Output Clock Configuration**

Source Clock	GPMC_CONFIG1_i[1-0] GPMCFCLKDIVIDER	GPMC Output Clock Provided to External Memory Device
GPMC_FCLK	00	GPMC_FCLK
	01	GPMC_FCLK/2
	10	GPMC_FCLK/3
	11	GPMC_FCLK/4

When using synchronous interface protocols, the GPMC output clock (CLK), toggles only during the read or write access cycle. In some applications, it may be desirable to have a continuous clock running at the GPMC interface clock frequency for clocking attached devices. This option is enabled by an optional clock path from GPMC functional clock input (GPMC\_FCLK) to GPMC\_FCLK\_MUX. And output of this mux can be selected by CLKOUT\_SEL bit field for GPMC\_CONTROL Register present in MSS\_CTRL. For more details, see MSS\_CTRL chapter in GPMC Global Configuration.

Note that when using such synchronous interface protocols with the continuous clock option, user should ensure that the GPMC outputs are timed to the same frequency (GPMC\_CONFIG1\_i[1-0] GPMCFCLKDIVIDER = 0).

#### 12.3.4.4.3 GPMC Power Management

[Table 12-319](#) describes the local power-management features available for the GPMC module.

**Table 12-319. GPMC Local Power-Management Features**

Feature	Registers	Description
Clock autogating	GPMC_SYSCONFIG[0] AUTOIDLE	This bit allows a local power optimization inside the module, by gating the GPMC_ICLK clock upon the internal activity.
Target idle modes	GPMC_SYSCONFIG[4-3] IDLEMODE	Force-idle, no-idle and smart-idle modes are available.

#### 12.3.4.4.4 GPMC Interrupt Requests

The GPMC generates one interrupt request (see *GPMC0 Hardware Requests*).

[Table 12-320](#) lists the event flags, and their mask, that can cause module interrupts.

**Table 12-320. GPMC Interrupt Events**

Event Flag	Event Mask	Sensitivity	Description
GPMC_IRQSTATUS[9] WAIT1EDGE DETECTIONSTATUS	GPMC_IRQENABLE[9] WAIT1EDGE DETECTIONENABLE	Edge	Wait1 edge detection interrupt: Triggered if a rising or falling edge is detected on the GPMC_WAIT1 signal. The rising or falling edge detection of Wait1 is selected through the GPMC_CONFIG[9] WAIT1PINPOLARITY bit.

**Table 12-320. GPMC Interrupt Events (continued)**

Event Flag	Event Mask	Sensitivity	Description
GPMC_IRQSTATUS[8] WAIT0EDGE DETECTIONSTATUS	GPMC_IRQENABLE[8] WAIT0EDGE DETECTIONENABLE	Edge	Wait0 edge detection interrupt: Triggered if a rising or falling edge is detected on the GPMC_WAIT0 signal. The rising or falling edge detection of Wait0 is selected through the GPMC_CONFIG[8] WAIT0PINPOLARITY bit.
GPMC_IRQSTATUS[1] TERMINAL COUNTSTATUS	GPMC_IRQENABLE[1] TERMINAL COUNTENABLE	Level	Terminal count event: Triggered on prefetch process completion; that is, when the number of currently remaining data to be requested reaches 0.
GPMC_IRQSTATUS[0] FIFOEVENTSTATUS	GPMC_IRQENABLE[0] FIFOEVENTENABLE	Level	FIFO event interrupt: Indicates available FIFO levels for write-posting mode and prefetch mode. GPMC_PREFETCH_CONFIG1[2] DMAMODE must be set to 0.

#### 12.3.4.4.5 GPMC Interconnect Port Interface

##### Note

Some of the GPMC features described in this section may not be supported on this family of devices. For more information, see *GPMC Not Supported Features*.

The GPMC interconnect interface is a pipelined interface including a 16 × 32-bit word write buffer.

Any system host can issue external access requests through the GPMC.

The device system can issue the following requests through this interface:

- One 8-, 16-, or 32-bit interconnect access (read/write)
- Two incrementing 32-bit interconnect accesses (read/write)
- Two wrapped 32-bit interconnect accesses (read/write)
- Four incrementing 32-bit interconnect accesses (read/write)
- Four wrapped 32-bit interconnect accesses (read/write)
- Eight incrementing 32-bit interconnect accesses (read/write)
- Eight wrapped 32-bit interconnect accesses (read/write)

Only linear burst transactions are supported; interleaved burst transactions are not supported. Only power-of-two-length precise bursts 2 × 32, 4 × 32, 8 × 32, and 16 × 32, with the burst base address aligned on the total burst size, are supported (this limitation applies to incrementing bursts only).

This interface also provides one interrupt and one DMA request line for specific event control.

It is recommended to program the GPMC\_CONFIG1\_i[24-23] ATTACHEDDEVICEPAGELENGTH bit field according to the page length of the effective attached device and to enable the GPMC\_CONFIG1\_i[31] WRAPBURST bit if the attached device supports wrapping burst.

It is possible, however, to emulate wrapping burst on a nonwrapping memory by providing relevant addresses within the page or by splitting transactions. Bursts larger than the memory page length are chopped into multiple burst transactions. Because of the alignment requirements, a page boundary is never crossed.

#### 12.3.4.4.6 GPMC Address and Data Bus

The current application supports GPMC connection to NAND devices and to address/data-multiplexed memories or devices. Connection to address/data-non-multiplexed memories or devices is supported with an address range of 4MB.

Depending on the GPMC configuration of each chip-select, address and data bus lines that are not required for a particular access protocol are not updated (changed from current value) and are not sampled when input (input data bus).

- For address/data-multiplexed and AAD-multiplexed NOR devices, the address is multiplexed on the data bus.

- 8-bit-wide NOR devices do not use GPMC I/O: GPMC\_AD[15-8] for data (they are used for address if needed).
- 16-bit-wide NAND devices do not use GPMC I/O: GPMC\_A[22-0].
- 8-bit-wide NAND devices do not use GPMC I/O: GPMC\_A[22-0] and GPMC I/O: GPMC\_AD[15-8].

#### 12.3.4.4.6.1 GPMC I/O Configuration Setting

##### Note

In this section and the following sections, the *i* in GPMC\_CONFIGx\_i stands for the GPMC chip-select *i*, where *i* = 0 to 3.

To select a NAND device, program the following register fields:

- GPMC\_CONFIG1\_i[11-10] DEVICETYPE = 0b10
- GPMC\_CONFIG1\_i[9-8] MUXADDDATA = 0b00

To select an address/data-multiplexed device, program the following register fields:

- GPMC\_CONFIG1\_i[11-10] DEVICETYPE = 0b00
- GPMC\_CONFIG1\_i[9-8] MUXADDDATA = 0b10

To select an address/address/data-multiplexed device, program the following register fields:

- GPMC\_CONFIG1\_i[11-10] DEVICETYPE = 0b00
- GPMC\_CONFIG1\_i[9-8] MUXADDDATA = 0b01

To select an address/data-non-multiplexed device, program the following register fields:

- GPMC\_CONFIG1\_i[11-10] DEVICETYPE = 0b00
- GPMC\_CONFIG1\_i[9-8] MUXADDDATA = 0b00

#### 12.3.4.4.7 GPMC Address Decoder and Chip-Select Configuration

Addresses are decoded accordingly with the address request of the chip-select and the content of the chip-select base address register file, which includes a set of global GPMC configuration registers and four sets of chip-select configuration registers.

The GPMC configuration register file is memory-mapped and can be read or written with byte, 16-bit word, or 32-bit word accesses. The register file must be configured as a noncacheable, nonbufferable region to prevent any desynchronization between host execution (write request) and the completion of register configuration (write completed with register updated).

After the chip-select is configured, the access engine accesses the external device, drives the external interface control signals, and applies the interface protocol based on user-defined timing parameters and settings.

##### 12.3.4.4.7.1 Chip-Select Base Address and Region Size

Any external memory or ASIC device attached to the GPMC external interface can be accessed by any device system host within the GPMC 128MB address space. For more information, see *Memory Map*.

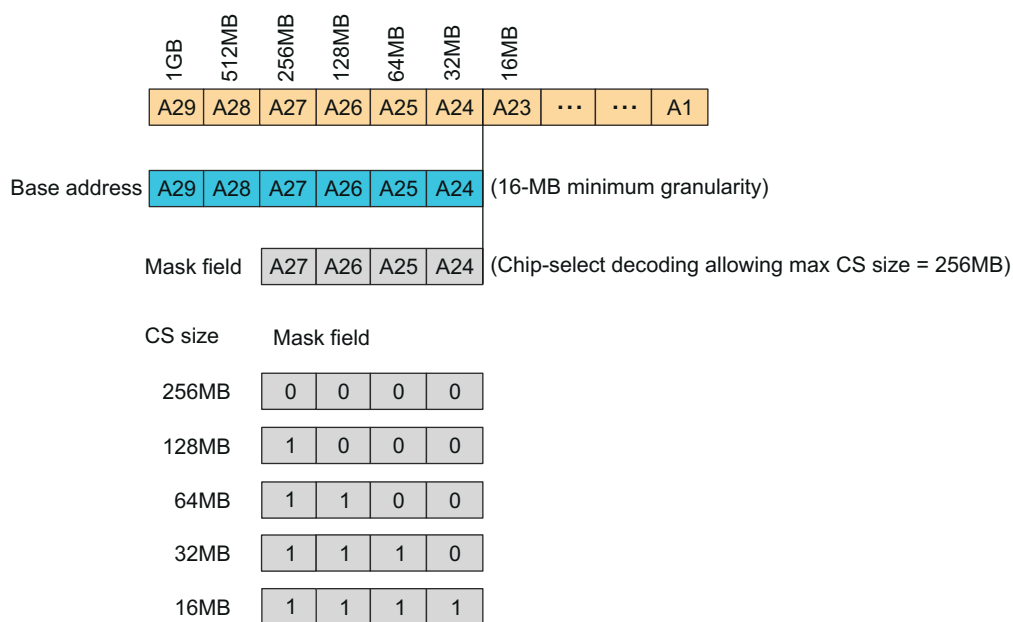
##### Note

Even though GPMC supports total address space of 1GB, only 128MB are physically available in this device.

The GPMC 128MB address space can be divided into a maximum of four chip-select regions with programmable base address and programmable chip-select size. The chip-select size is programmable from 16MB to 256MB (must be a power-of-two) and is defined by the mask field. Attached memory smaller than the programmed chip-select region size is accessed through the entire chip-select region (aliasing).

Each chip-select has a 6-bit base address encoding and 4-bit decoding mask, which must be programmed according to the following rules:

- The programmed chip-select region base address must be aligned on the chip-select region size address boundary and is limited to a power-of-two address value. During access decoding, the value of the register base address is used to compare the address with the address bit line mapping, as shown in [Figure 12-238](#) (with A0 as the device system byte-address line). The base address is programmed through the GPMC\_CONFIG7\_i[5-0] BASEADDRESS bit field.
- The register mask is used to exclude some address lines from the decoding. A register mask bit field set to 0 suppresses the associated address line from the address comparison (incoming address bit line is don't care). The value of the register mask must be limited to the subsequent value, based on the desired chip-select region size. Any other value has an undefined result. When multiple chip-select regions with overlapping addresses are enabled concurrently, access to these chip-select regions is cancelled and a GPMC access error is posted. The mask field is programmed through the GPMC\_CONFIG7\_i[11-8] MASKADDRESS bit field.



gpmc-006

**Figure 12-238. Chip-Select Address Mapping and Decoding Mask**

For example, to map the 128MB address space (from 2000 0000h to 27FF FFFFh), the GPMC\_CONFIG7\_i[5-0] BASEADDRESS bit field should be set to 28h.

Eg : 68000000h = 0110\_1000\_0000\_0000\_0000\_0000\_0000b

Base Address[29:24] = 10\_1000 = 28h

Chip-select configuration (base and mask address or any protocol and timing settings) must be performed while the associated chip-select is disabled through the GPMC\_CONFIG7\_i[6] CSVALID bit (where i stands for the GPMC chip-select i, where i = 0 to 3). In addition, a chip-select configuration can be disabled only if there is no ongoing access to that chip-select. This requires monitoring the activity of the prefetch or write-posting engine if the engine is active on the chip-select. Also, the write buffer state must be monitored to wait for any posted write completion to the chip-select.

Any access attempted to a nonvalid GPMC address region (CSVALID disabled or address decoding outside a valid chip-select region) is not propagated to the external interface and a GPMC access error is posted. In case of overlapping chip-selects, an error is generated and no access occurs on either chip-select.

CS0 is the only chip-select region enabled after a power up or GPMC reset.

### CAUTION

Although the GPMC interface can drive up to four chip-selects, the frequency specified for this interface is for a specific load. If this load is exceeded, the maximum frequency cannot be reached. One solution is to implement a board with buffers to allow the slowest device to maintain the total load on the lines at the value specified in the device-specific Datasheet.

#### 12.3.4.4.7.2 Access Protocol

##### 12.3.4.4.7.2.1 Supported Devices

The access protocol of each chip-select can be independently specified through the GPMC\_CONFIG1\_i[11-10] DEVICETYPE parameter (where i = 0 to 3) for:

- Random-access synchronous or asynchronous memory, such as NOR flash and SRAM
- NAND flash asynchronous devices

### Note

For more information about the NAND flash GPMC basic programming model and NAND support, see [Section 12.3.4.4.11, GPMC NAND Device Basic Programming Model](#), and [Section 12.3.4.4.11.1, NAND Memory Device in Byte or Word16 Stream Mode](#).

#### 12.3.4.4.7.2.2 Access Size Adaptation and Device Width

Each chip-select can be independently configured through the GPMC\_CONFIG1\_i[13-12] DEVICESIZE bit field (where i = 0 to 3) to interface with a 16- or 8-bit-wide device. System requests with data width greater than the external device data bus width are split into successive accesses according to the external device data-bus width and little-endian data organization.

##### 12.3.4.4.7.2.3 Address/Data-Multiplexing Interface

For random synchronous or asynchronous memory interfacing (DEVICETYPE = 0b00), an address- and data-multiplexing protocol can be selected through the GPMC\_CONFIG1\_i[9-8] MUXADDDATA bit field (where i = 0 to 3). The nADV signal must be used as the external device address latch control signal. For the associated chip-select configuration, nADV assertion and deassertion time and nOE assertion time must be set to the appropriate value to meet the address latch setup/hold time requirements of the external device. See *GPMC Integration*.

### Note

This address/data-multiplexing interface is not applicable to NAND device interfacing. NAND devices require a specific address, command, and data-multiplexing protocol. See [Section 12.3.4.4.11, NAND Device Basic Programming Model](#).

#### 12.3.4.4.7.3 External Signals

##### 12.3.4.4.7.3.1 WAIT Pin Monitoring Control

GPMC access time can be dynamically controlled using an external GPMC\_WAIT pin when the external device access time is not deterministic and cannot be defined and controlled using only the GPMC internal RDACCESSTIME, WRACCESSTIME, and PAGEBURSTACCESSTIME wait-state generator.

The GPMC features two input WAIT pins: GPMC\_WAIT1, and GPMC\_WAIT0. These pins allow control of external devices with different WAIT pin polarity. They also allow the overlap of WAIT pin assertion from different devices without affecting access to devices for which the WAIT pin is not asserted.

- The GPMC\_CONFIG1\_i[17-16] WAITPINSELECT bit field (where i = 0 to 3) selects which input GPMC\_WAIT pin is used for the device attached to the corresponding chip-select.

- The polarity of the WAIT pin is defined through the WAITxPINPOLARITY bit of the GPMC\_CONFIG register. A WAIT pin configured to be active low means that low level on the WAIT signal indicates that the data is not ready and that the data bus is invalid. When a WAIT pin is inactive, data is valid.

The GPMC access engine can be configured per chip-select to monitor or not the WAIT pin of the external memory device, based on the access type: read or write.

- The GPMC\_CONFIG1\_i[22] WAITREADMONITORING bit defines whether or not the WAIT pin must be monitored during read accesses.
- The GPMC\_CONFIG1\_i[21] WAITWRITEMONITORING bit defines whether or not the WAIT pin must be monitored during write accesses.

The GPMC access engine can be configured to monitor the WAIT pin of the external memory device asynchronously or synchronously with the GPMC output clock, depending on the access type: synchronous or asynchronous (the GPMC\_CONFIG1\_i[29] READTYPE and GPMC\_CONFIG1\_i[27] WRITETYPE bits).

#### 12.3.4.4.7.3.1.1 Wait Monitoring During Asynchronous Read Access

When WAIT pin monitoring is enabled for read accesses (WAITREADMONITORING), the effective access time is a logical AND combination of the RDACCESSTIME timing completion and the wait-deasserted state.

During asynchronous read accesses with WAIT pin monitoring enabled, the WAIT pin must be at a valid level (asserted or deasserted) for at least two GPMC clock cycles before RDACCESSTIME completes, to ensure correct dynamic access-time control through WAIT pin monitoring. The advance pipelining of the two GPMC clock cycles is the result of the internal synchronization requirements for the WAIT signal.

In this context, RDACCESSTIME is used as a wait invalid timing window and is set to such a value that the WAIT pin is at a valid state two GPMC clock cycles before RDACCESSTIME completes.

Similarly, during a multiple-access cycle (for example, asynchronous read page mode), the effective access time is a logical AND combination of PAGEBURSTACCESSTIME timing completion and the wait-deasserted state. Wait monitoring pipelining is also applicable to multiple accesses (access within a page).

- Wait monitored as active freezes the CYCLETIME counter. For an access within a page, when the CYCLETIME counter is by definition in a lock state, wait monitored as asserted extends the current access time in the page. Control signals are kept in their current state. The data bus is considered invalid, and no data are captured during this clock cycle.
- Wait monitored as inactive unfreezes the CYCLETIME counter. For an access within a page, when the CYCLETIME counter is by definition in a lock state, wait monitored as inactive completes the current access time and starts the next access phase in the page. The data bus is considered valid, and data are captured during this clock cycle. In case of a single access or if this was the last access in a multiple-access cycle, all signals are controlled according to their related control timing value and according to the CYCLETIME counter status.

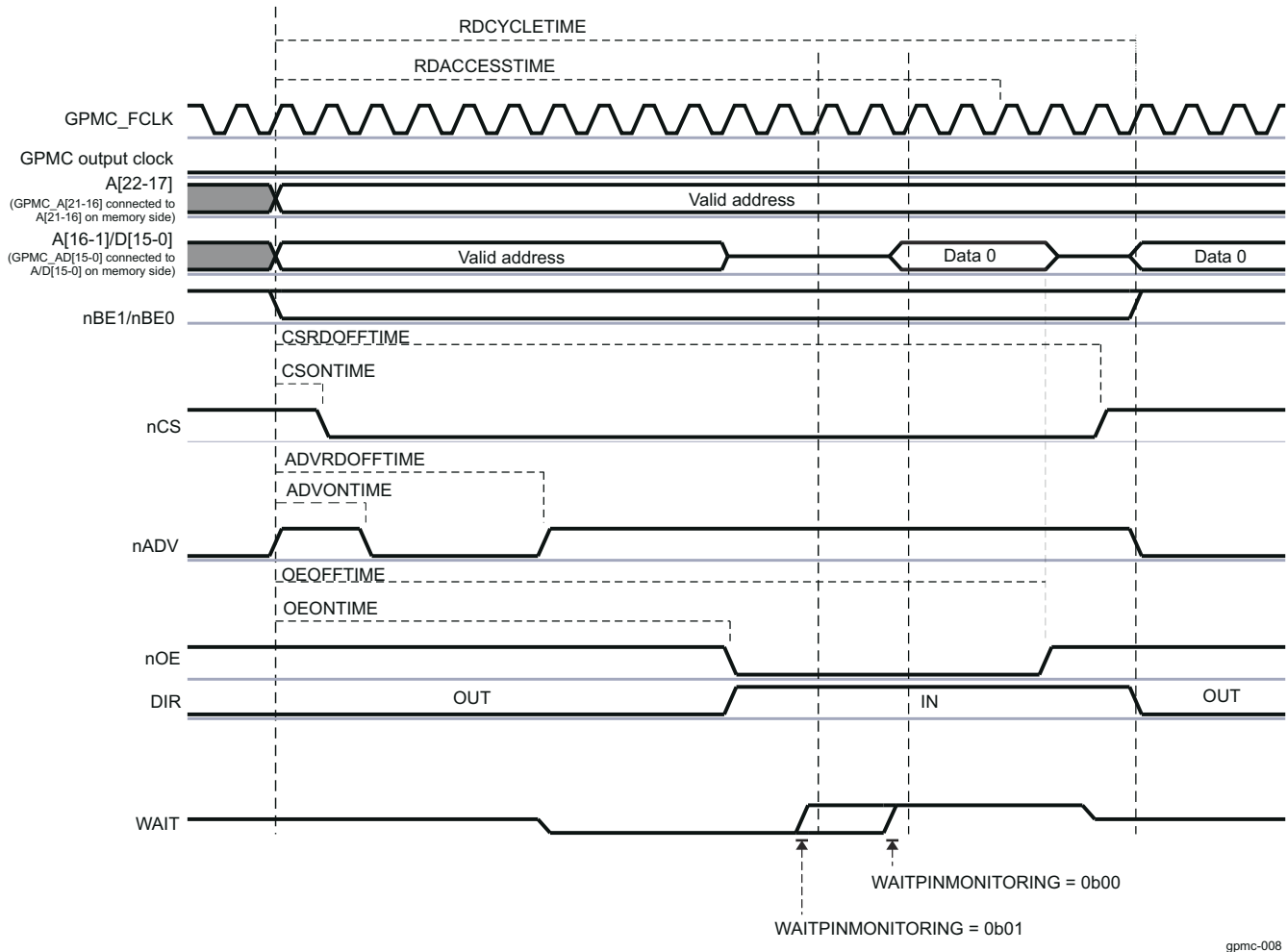
When a delay larger than two GPMC clocks must be observed between wait-pin deactivation time and data valid time (including the required GPMC and the device data setup time), an extra delay can be added between wait-pin deassertion time detection and effective data-capture time and the effective unlock of the CYCLETIME counter. This extra delay can be programmed in the GPMC\_CONFIG1\_i[19-18] WAITMONITORINGTIME bit field (where i = 0 to 3).

#### Note

- The WAITMONITORINGTIME parameter does not delay the WAIT pin active or inactive detection, nor does it modify the two GPMC clocks pipelined detection delay.
- This extra delay is expressed as a number of GPMC output clock cycles, even though the access is defined as asynchronous, and no GPMC output clock is provided to the external device. Still, because GPMC\_CONFIG1\_i[1-0] GPMCFCLKDIVIDER is used as a divider for the GPMC clock, it must be programmed to define the correct WAITMONITORINGTIME delay.

Figure 12-239 shows wait behavior during an asynchronous single read access.





**Figure 12-239. Wait Behavior During an Asynchronous Single Read Access (GPMCFCLKDivider = 1)**

#### Note

The WAIT signal is active low. GPMC\_CONFIG1\_i[19-18] WAITMONITORINGTIME = 0b00, or 0b01.

#### 12.3.4.7.3.1.2 Wait Monitoring During Asynchronous Write Access

When WAIT pin monitoring is enabled for write accesses (GPMC\_CONFIG1\_i[21] WAITWRITEMONITORING bit = 0x1), the wait invalid timing window is defined by the WRACCESSTIME field. WRACCESSTIME must be set so that the WAIT pin is at a valid state two GPMC clock cycles before WRACCESSTIME completes. The advance pipelining of the two GPMC clock cycles is the result of the internal synchronization requirements for the WAIT signal.

- Wait monitored as active freezes the CYCLETIME counter. This informs the GPMC that the data bus is not captured by the external device. The control signals are kept in their current state. The data bus still drives the data.
- Wait monitored as inactive unfreezes the CYCLETIME counter. This informs that the data bus is correctly captured by the external device. All signals, including the data bus, are controlled according to their related control timing value and to the CYCLETIME counter status.

When a delay larger than two GPMC clock cycles must be observed between wait-pin deassertion time and the effective data write into the external device (including the required GPMC data setup time and the device data setup time), an extra delay can be added between wait-pin deassertion time detection and effective data write

time into the external device and the effective unfreezing of the CYCLETIME counter. This extra delay can be programmed in the GPMC\_CONFIG1\_i[19-18] WAITMONITORINGTIME bit field (where i = 0 to 3).

#### Note

- The WAITMONITORINGTIME parameter does not delay the WAIT pin assertion or deassertion detection, nor does it modify the two GPMC clock cycles pipelined detection delay.
- This extra delay is expressed as a number of GPMC output clock cycles, even though the access is defined as asynchronous, and even though no clock is provided to the external device. Still, because the GPMC\_CONFIG1\_i[1-0] GPMCFCLKDIVIDER bit field is used as a divider for the GPMC clock, it must be programmed to define the correct WAITMONITORINGTIME delay.

#### 12.3.4.4.7.3.1.3 Wait Monitoring During Synchronous Read Access

During synchronous accesses with WAIT pin monitoring enabled, the WAIT pin is captured synchronously with GPMC output clock, using the rising edge of this clock.

The WAIT signal can be programmed to apply to the same clock cycle in which it is captured. Alternatively, it can be sampled one or two GPMC output clock cycles ahead of the clock cycle to which it applies. This pipelining is applicable to the entire burst access and to all data phases in the burst access. This wait pipelining depth is programmed in the GPMC\_CONFIG1\_i[19-18] WAITMONITORINGTIME bit field (where i = 0 to 3), and is expressed as a number of GPMC output clock cycles.

In synchronous mode, when WAIT pin monitoring is enabled (the GPMC\_CONFIG1\_i[22] WAITREADMONITORING bit), the effective access time is a logical AND combination of the RDACCESSTIME timing completion and the wait-deasserted state detection.

Depending on the programmed value of WAITMONITORINGTIME, the WAIT pin must be at a valid level, either asserted or deasserted:

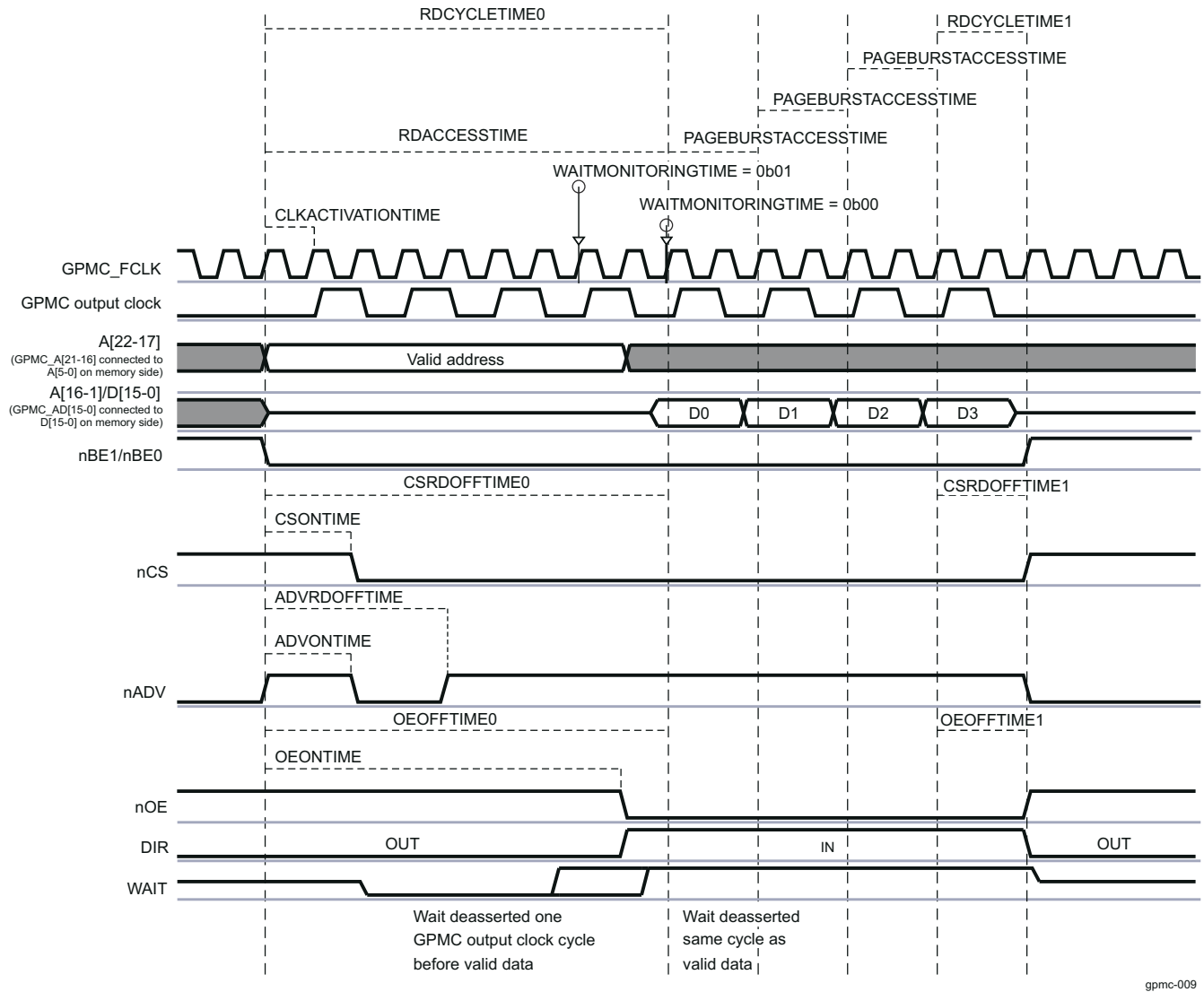
- In the same clock cycle the data is valid if WAITMONITORINGTIME = 0 (at RDACCESSTIME completion)
- In the WAITMONITORINGTIME x (GPMCFCLKDIVIDER + 1) GPMC\_FCLK clock cycles before RDACCESSTIME completion if WAITMONITORINGTIME is not equal to 0

Similarly, during a multiple-access cycle (burst mode), the effective access time is a logical AND combination of PAGEBURSTACCESSTIME timing completion and the WAIT-INACTIVE state. The wait pipelining-depth programming applies to the whole burst access.

- Wait monitored as active freezes the CYCLETIME counter. For an access within a burst (when the CYCLETIME counter is by definition in a lock state), wait monitored as active extends the current access time in the burst. Control signals are kept in their current state. The data bus is considered invalid, and no data are captured during this clock cycle.
- Wait monitored as inactive unfreezes the CYCLETIME counter. For an access within a burst (when the CYCLETIME counter is by definition in lock state), wait monitored as inactive completes the current access time and starts the next access phase in the burst. The data bus is considered valid, and data are captured during this clock cycle. In a single access or if this was the last access in a multiple-access cycle, all signals are controlled according to their relative control timing value and the CYCLETIME counter status.

Figure 12-240 shows wait behavior during a synchronous read burst access.





**Figure 12-240. Wait Behavior During a Synchronous Read Burst Access**

**Note**

The WAIT signal is active low. WAITMONITORINGTIME = 0b00, 0b01.

**12.3.4.4.7.3.1.4 Wait Monitoring During Synchronous Write Access**

During synchronous accesses with WAIT pin monitoring enabled (the GPMC\_CONFIG1\_i[21] WAITWRITEMONITORING bit), the WAIT pin is captured synchronously with GPMC output clock, using the rising edge of this clock.

If enabled, external WAIT pin monitoring can be used in combination with WRACCESSTIME to delay the GPMC output clock capture edge of the effective memory device.

WAIT-monitoring pipelining depth is similar to synchronous read access:

- At WRACCESSTIME completion if WAITMONITORINGTIME = 0
- In the WAITMONITORINGTIME x (GPMCFCLKDIVIDER + 1) GPMC\_FCLK cycles before WRACCESSTIME completion if WAITMONITORINGTIME is not equal to 0

Wait-monitoring pipelining definition applies to whole burst accesses:

- Wait monitored as active freezes the CYCLETIME counter. For accesses within a burst, when the CYCLETIME counter is by definition in a lock state, wait monitored as active indicates that the data bus is not being captured by the external device. Control signals are kept in their current state. The data bus is kept in its current state.
- Wait monitored as inactive unfreezes the CYCLETIME counter. For accesses within a burst, when the CYCLETIME counter is by definition in a lock state, wait monitored as inactive indicates the effective data capture of the bus by the external device and starts the next access of the burst. In case of a single access or if this was the last access in a multiple access cycle, all signals, including the data bus, are controlled according to their related control timing value and the CYCLETIME counter status.

#### Note

WAIT monitoring is supported for all configurations except GPMC\_CONFIG1\_i[19-18]  
WAITMONITORINGTIME = 0x0 (where i = 0 to 3) for write bursts with a clock divider of 1 or 2  
(the GPMC\_CONFIG1\_i[1-0] GPMCFCLKDIVIDER bit field is equal to 0x0 or 0x1, respectively).

#### 12.3.4.4.7.3.1.5 Wait With NAND Device

For information about the use of the WAIT pin for communication with a NAND flash external device, see [Section 12.3.4.4.11.2, NAND Device-Ready Pin](#).

#### 12.3.4.4.7.3.1.6 Idle Cycle Control Between Successive Accesses

##### 12.3.4.4.7.3.1.6.1 Bus Turnaround (BUSTURNAROUND)

To prevent data-bus contention, an access that follows a read access to a slow memory/device must be delayed (in other words, control the nCS/nOE deassertion to data bus in high-impedance delay).

The bus turnaround is a time-out counter starting after nCS or nOE deassertion time, whichever occurs first, and delays the next access start-cycle time. The counter is programmed through the GPMC\_CONFIG6\_i[11-8] BUSTURNAROUND bit field (where i = 0 to 3).

After a read access to a chip-select with a nonzero BUSTURNAROUND, the next access is delayed until the BUSTURNAROUND delay completes, if the next access is one of the following:

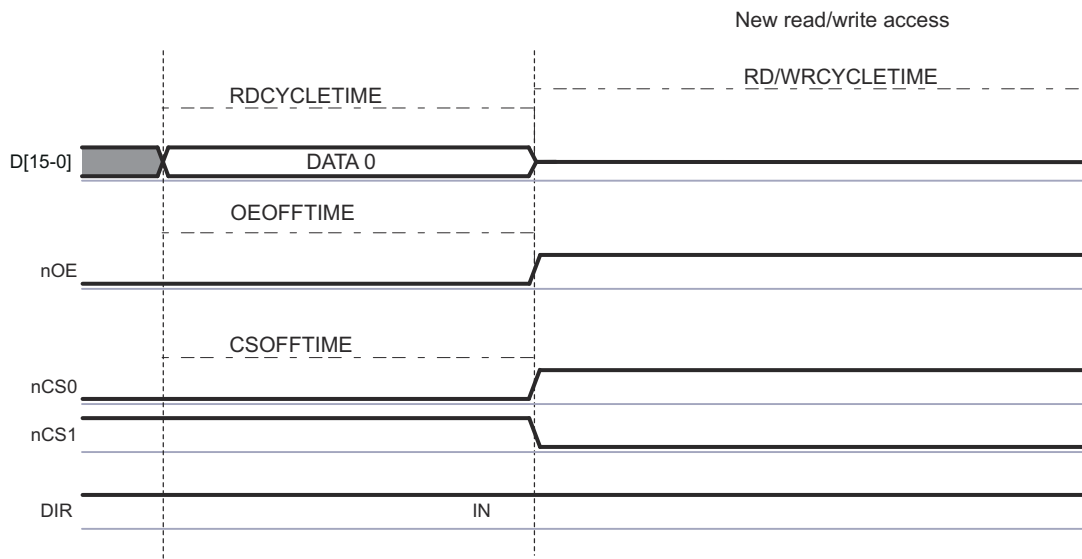
- A write access to any chip-select (the same or different chip-select from which the data was read)
- A read access to a different chip-select than the chip-select from which the data was read access
- A read or write access to a chip-select associated with an address/data-multiplexed device

#### Note

Bus keeping starts after bus turnaround completion so that DIR changes from IN to OUT after bus turnaround. The bus does not have enough time to go into high-impedance even though it can be driven with the same value before bus turnaround timing.

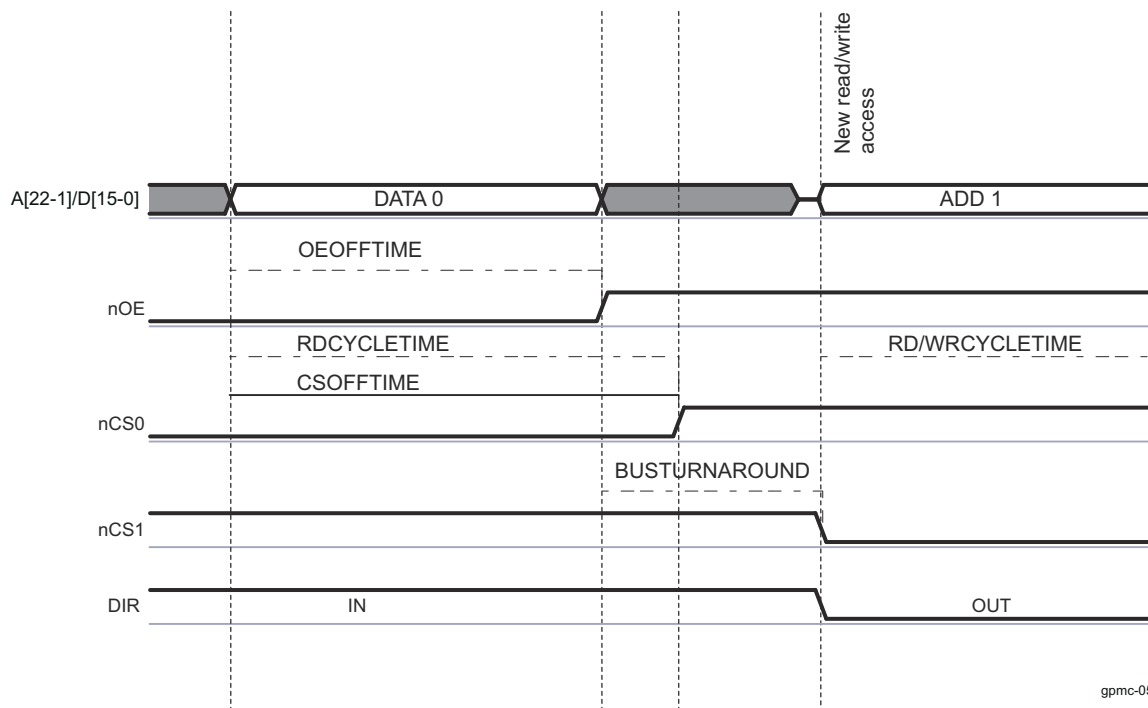
BUSTURNAROUND delay runs in parallel with GPMC\_CONFIG6\_i[11:8] CYCLE2CYCLEDELAY bit field delays. BUSTURNAROUND is a timing parameter for the ending chip-select access, while CYCLE2CYCLEDELAY is a timing parameter for the following chip-select access. The effective minimum delay between successive accesses is driven by these delay timing parameters and by the access type of the following access (see [Figure 12-241](#) through [Figure 12-243](#)).

Another way to prevent bus contention is to define an earlier nCS or nOE deassertion time for slow devices or to extend the value of RDCYCLETIME. Doing this prevents bus contention, but it also affects all accesses of this specific chip-select.



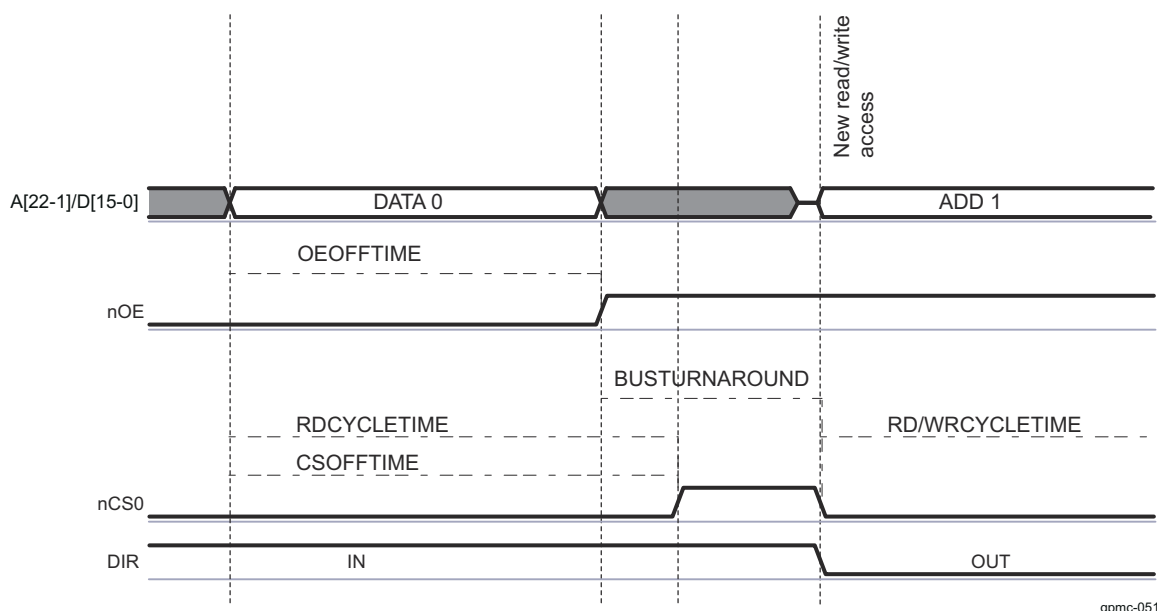
gpmc-049

**Figure 12-241. Read-to-Read for an Address-Data Multiplexed Device, on Different Chip-Select, Without Bus Turnaround (nCS Attached to a Fast Device)**



gpmc-050

**Figure 12-242. Read- to-Read/Write for an Address-Data Multiplexed Device, on Different Chip-Select, With Bus Turnaround**



**Figure 12-243. Read-to-Read/Write for a Address-Data or AAD-Multiplexed Device, on Same Chip-Select, With Bus Turnaround**

#### 12.3.4.4.7.3.1.6.2 Idle Cycles Between Accesses to Same Chip-Select (CYCLE2CYCLESAMECSSEN, CYCLE2CYCLEDELAY)

Some devices require a minimum chip-select signal inactive time between accesses. The GPMC\_CONFIG6\_i[7] CYCLE2CYCLESAMECSSEN bit (where i = 0 to 3) enables insertion of a minimum number of GPMC\_FCLK cycles, defined by the GPMC\_CONFIG6\_i[11-8] CYCLE2CYCLEDELAY bit field, between successive accesses of any type (read or write) to the same chip-select.

If CYCLE2CYCLESAMECSSEN is enabled, any subsequent access to the same chip-select is delayed until its CYCLE2CYCLEDELAY completes. The CYCLE2CYCLEDELAY counter starts when CSRDFFTIME/CSWROFFTIME completes.

The same applies to successive accesses occurring during 32-bit word or burst accesses split into successive single accesses when the single-access mode is used (GPMC\_CONFIG1\_i[30] READMULTIPLE = 0 or GPMC\_CONFIG1\_i[28] WRITEMULTIPLE = 0).

All control signals (CS, ADV#/ALE, BE0#/CLE, WE#, and GPMC output clock (CLK)) are kept inactive (ADV#/ALE, BE0#/CLE, and GPMC output clock at low level; and CS, OE#/RE, and WE at high level) during the idle GPMC\_FCLK cycles. This prevents back-to-back accesses to the same chip-select without idle cycles between accesses.

#### 12.3.4.4.7.3.1.6.3 Idle Cycles Between Accesses to Different Chip-Select (CYCLE2CYCLEDIFFCSSEN, CYCLE2CYCLEDELAY)

Because of the pipelined behavior of the system, successive accesses to different chip-selects can occur back-to-back with no idle cycles between accesses. Depending on the control signals (nCS, nADV/ALE, nBE0/CLE, nOE/RE, nWE) assertion and deassertion timing parameters and on the device timing parameters, the assertion times of some control signals may overlap between the successive accesses to a different chip-select. Similarly, some control signals (WE, OE/RE) may not respect required transition times.

To work around overlapping and to observe the required control-signal transitions, a minimum of CYCLE2CYCLEDELAY inactive cycles is inserted between the access being initiated to this chip-select and the previous access ending for a different chip-select. This applies to any type of access (read or write).

If the GPMC\_CONFIG6\_i[6] CYCLE2CYCLEDIFFCSSEN bit is enabled, the chip-select access is delayed until CYCLE2CYCLEDELAY cycles have expired since the end of a previous access to a different chip-select.

CYCLE2CYCLEDELAY count starts at CSRDOFFTIME/CSWROFFTIME completion. All control signals are kept inactive during the idle GPMC\_FCLK cycles.

### Note

CYCLE2CYCLESAMECSEN and CYCLE2CYCLEDIFFCSEN must be set in the GPMC\_CONFIG6\_i registers to get idle cycles inserted between accesses on this chip-select and after accesses to a different chip-select, respectively.

The CYCLE2CYCLEDELAY delay runs in parallel with the BUSTURNAROUND delay. The BUSTURNAROUND is a timing parameter defined for the ending chip-select access, whereas CYCLE2CYCLEDELAY is a timing parameter defined for the starting chip-select access. The effective minimum delay between successive accesses is based on the larger delay timing parameter and on access type combination, because bus turnaround does not apply to all access types. For more information about bus turnaround, see [Section 3.4.4.7.3.1.6.1, Bus Turnaround \(BUSTURNAROUND\)](#).

[Table 12-321](#) describes the configuration required for idle cycle insertion.

**Table 12-321. Idle Cycle Insertion Configuration**

1st Access Type	BUSTURN AROUND Timing Parameter	Second Access Type	Chip-Select	Add/Data Multiplexed	CYCLE2 CYCLE SAMECSEN Parameter	CYCLE2 CYCLE DIFFCSEN Parameter	Idle Cycle Insertion Between the Two Accesses
R/W	= 0	R/W	Any	Any	0	x	No idle cycles are inserted if the two accesses are well pipelined.
R	> 0	R	Same	Nonmuxed	x	0	No idle cycles are inserted if the two accesses are well pipelined.
R	> 0	R	Different	Nonmuxed	0	0	BUSTURNAROUND cycles are inserted.
R	> 0	R/W	Any	Muxed	0	0	BUSTURNAROUND cycles are inserted.
R	> 0	W	Any	Any	0	0	BUSTURNAROUND cycles are inserted.
W	> 0	R/W	Any	Any	0	0	No idle cycles are inserted if the two accesses are well pipelined.
R/W	= 0	R/W	Same	Any	1	x	CYCLE2CYCLEDELAY cycles are inserted.
R/W	= 0	R/W	Different	Any	x	1	CYCLE2CYCLEDELAY cycles are inserted.
R/W	> 0	R/W	Same	Any	1	x	CYCLE2CYCLEDELAY cycles are inserted. If BTA idle cycles already apply on these two back-to-back accesses, the effective delay is max (BUSTURNAROUND, CYCLE2CYCLEDELAY).
R/W	> 0	R/W	Different	Any	x	1	CYCLE2CYCLEDELAY cycles are inserted. If BTA idle cycles already apply on these two back-to-back accesses, the effective delay is maximum (BUSTURNAROUND, CYCLE2CYCLEDELAY).

#### 12.3.4.4.7.3.1.7 Slow Device Support (TIMEPARAGRANULARITY Parameter)

All access-timing parameters can be multiplied by 2 by setting the GPMC\_CONFIG1\_i[4] TIMEPARAGRANULARITY bit (where i stands for the GPMC chip-select i, where i = 0 to 3). Increasing all access timing parameters allows support of slow devices.

#### 12.3.4.4.7.3.2 DIR Pin

The DIR pin is used to control I/O direction on the GPMC data bus GPMC\_D[15-0]. Depending on pad multiplexing, this signal can be output and used externally to the device, if required. The DIR pin is low during transmit (OUT) and high during receive (IN).

For write accesses, the DIR pin stays OUT from start-cycle time to end-cycle time.

For read accesses, the DIR pin goes from OUT to IN at nOE assertion time and stays IN until:

- BUSTURNAROUND is enabled
  - The DIR pin goes from IN to OUT at end-cycle time plus programmable bus turnaround time.
- BUSTURNAROUND is disabled
  - After an asynchronous read access, the DIR pin goes from IN to OUT at RDACCESSTIME + 1 GPMC\_FCLK cycle or when RDCYCLETIME completes, whichever occurs last.
  - After a synchronous read access, the DIR pin goes from IN to OUT at RDACCESSTIME + 2 GPMC\_FCLK cycles or when RDCYCLETIME completes, whichever occurs last.

Because of the bus-keeping feature of the GPMC, after a read or write access and with no other accesses pending, the default value of the DIR pin is OUT (see [Section 12.3.4.4.8.10, Bus Keeping Support](#)). In non-multiplexed devices, the DIR pin stays IN between two successive read accesses to prevent unnecessary toggling.

#### 12.3.4.4.7.3.3 Reset

No reset signal is sent to the external memory device by the GPMC.

GPMC\_RST is the reset signal for the GPMC module. It is connected and controlled by LPSC8 in VD\_CORE. That reset signal initializes the internal state-machine and the internal configuration registers.

#### 12.3.4.4.7.3.4 Write Protect Signal (nWP)

When connected to the attached memory device, the write protect signal can enable or disable the lockdown function of the attached memory. The nWP output pin value is controlled through the GPMC\_CONFIG[4] WRITEPROTECT bit which is common for all chip selects.

#### 12.3.4.4.7.3.5 Byte Enable (nBE1/nBE0)

Byte enable signals (nBE1/nBE0) are:

- Valid (asserted or nonasserted according to the incoming system request) from access start to access completion for asynchronous and synchronous single accesses
- Asserted low from access start to access completion for asynchronous and synchronous multiple read accesses
- Valid (asserted or nonasserted, according to the incoming system request) synchronously to each written data for synchronous multiple write accesses.

#### 12.3.4.4.7.4 Error Handling

When an error occurs in the GPMC, the error information is stored in the GPMC\_ERR\_TYPE register and the address of the illegal access is stored in the GPMC\_ERR\_ADDRESS register. The GPMC keeps only the first error abort information until the GPMC\_ERR\_TYPE register is reset. Subsequent accesses that cause errors are not logged until the error is cleared by hardware with the GPMC\_ERR\_TYPE[0] ERRORVALID bit.

- ERRORNOTSUPPADD occurs when an incoming system request address decoding does not match any valid chip-select region, or if two chip-select regions are defined as overlapped, or if a register file access is tried outside the valid address range of 1KB.

- **ERRORNOTSUPPMCMD** occurs when an unsupported command request is decoded at the interconnect interface.
- **ERRORTIMEOUT**: A time-out mechanism prevents the system from hanging. The start value of the 9-bit time-out counter is defined in the `GPMC_TIMEOUT_CONTROL` register and enabled with the `GPMC_TIMEOUT_CONTROL[0] TIMEOUTENABLE` bit. When enabled, the counter starts at start-cycle time until it reaches 0 and data is not responded to from memory, and then a time-out error occurs. When data are sent from memory, this counter is reset to its start value. With multiple accesses (asynchronous page mode or synchronous burst mode), the counter is reset to its start value for each data access within the burst.

The GPMC does not generate interrupts on these errors. An interrupt generation is handled at interconnect level.

#### 12.3.4.4.8 GPMC Timing Setting

The GPMC offers maximum flexibility to support various access protocols. Most of the timing parameters of the protocol access used by the GPMC to communicate with attached memories or devices are programmable on a chip-select basis. Assertion and deassertion times of control signals are defined to match the attached memory or device timing specifications and to get maximum performance during accesses. For more information about `GPMC_CLKOUT` and `GPMC_FCLK`, see [Section 12.3.4.4.8.6, GPMC\\_CLKOUT](#).

#### Note

In the following sections, the start access time refers to the time at which the access begins.

##### 12.3.4.4.8.1 Read Cycle Time and Write Cycle Time (*RDCYCLETIME* / *WRCYCLETIME*)

The `GPMC_CONFIG5_i[4-0] RDCYCLETIME` and `GPMC_CONFIG5_i[12-8] WRCYCLETIME` bit fields (where  $i = 0$  to 3) define the address bus and byte-enable valid time for read and write accesses. To ensure a correct duty cycle of GPMC output clock between accesses, `RDCYCLETIME` and `WRCYCLETIME` are expressed in `GPMC_FCLK` cycles and must be multiples of the GPMC output clock cycle. The `RDCYCLETIME` and `WRCYCLETIME` bit fields can be set with a granularity of 1 or 2 through the `GPMC_CONFIG5_i[4] TIMEPARAGRANULARITY` bit.

When `RDCYCLETIME` or `WRCYCLETIME` completes, if they are not already deasserted, all control signals (`nCS`, `nADV/ALE`, `nOE/RE`, `nWE`, and `BE0/CLE`) are deasserted to their reset values, regardless of their deassertion time parameters.

An exception to this forced deassertion occurs when a pipelined request to the same chip-select or to a different chip-select is pending. In such a case, it is not necessary to deassert a control signal with deassertion time parameters equal to the cycle-time parameter. This exception to forced deassertion prevents any unnecessary glitches. This requirement also applies to BE signals, thus avoiding an unnecessary BE glitch transition when pipelining requests.

#### Note

All control signals (`CS`, `ADV#/ALE`, `BE0#/CLE`, `WE#`, and GPMC output clock) are kept inactive (`ADV#/ALE`, `BE0#/CLE`, and GPMC output clock at low level; and `CS`, `OE#/RE`, and `WE` at high level) during the idle `GPMC_FCLK` cycles.

If no inactive cycles are required between successive accesses to the same chip-select or a different chip-select (`GPMC_CONFIG6_i[7] CYCLE2CYCLESAMECSSEN = 0` or `GPMC_CONFIG6_i[6] CYCLE2CYCLEDIFFCSSEN = 0`, where  $i = 0$  to 3), and if assertion-time parameters associated with the pipelined access are equal to 0, asserted control signals (`nCS`, `nADV/ALE`, `nBE0/CLE`, `nWE`, and `nOE/RE`) are kept asserted. This applies to any read/write to read/write access combination.

If inactive cycles are inserted between successive accesses (that is, `CYCLE2CYCLESAMECSSEN = 1` or `CYCLE2CYCLEDIFFCSSEN = 1`), the control signals are forced to their respective default reset values for the number of `GPMC_FCLK` cycles defined in `CYCLE2CYCLEDELAY`.



#### **12.3.4.4.8.2 nCS: Chip-Select Signal Control Assertion/Deassertion Time (CSONTIME / CSRDOFFTIME / CSWROFFTIME / CSEXTRADELAY)**

The GPMC\_CONFIG2\_i[3-0] CSONTIME bit field (where i = 0 to 3) defines the nCS signal-assertion time relative to the start access time. It is common for read and write accesses.

The GPMC\_CONFIG2\_i[12-8] CSRDOFFTIME (read access) and GPMC\_CONFIG2\_i[20-16] CSWROFFTIME (write access) bit fields define the nCS signal deassertion time relative to start access time.

The CSONTIME, CSRDOFFTIME, and CSWROFFTIME parameters apply to synchronous and asynchronous modes. CSONTIME can be used to control an address and byte-enable setup time before chip-select assertion. CSRDOFFTIME and CSWROFFTIME can be used to control an address and byte-enable hold time after chip-select deassertion.

nCS signal transitions, as controlled through CSONTIME, CSRDOFFTIME, and CSWROFFTIME, can be delayed by a half-GPMC\_FCLK period by enabling the GPMC\_CONFIG2\_i[7] CSEXTRADELAY bit. This half-GPMC\_FCLK period provides more granularity on the nCS assertion and deassertion time to ensure proper setup and hold time relative to GPMC output clock. CSEXTRADELAY is especially useful in configurations where GPMC output clock and GPMC\_FCLK have the same frequency, but it can also be used for all GPMC configurations. If enabled, CSEXTRADELAY applies to all parameters that control nCS transitions.

The CSEXTRADELAY bit must be used carefully to avoid control signal overlap between successive accesses to different chip-selects. This implies the need to program the RDCYCLETIME and WRCYCLETIME bit fields to be greater than the nCS signal-deassertion time, including the extra half-GPMC\_FCLK-period delay.

#### **12.3.4.4.8.3 nADV/ALE: Address Valid/Address Latch Enable Signal Control Assertion/Deassertion Time (ADVONTIME / ADVRDOFFTIME / ADVWROFFTIME / ADVEXTRADELAY/ADVAADMUXONTIME/ADVAADMUXRDOFFTIME/ADVAADMUXWROFFTIME)**

The GPMC\_CONFIG3\_i[3-0] ADVONTIME field (where i = 0 to 3) defines the nADV/ALE signal-assertion time relative to start access time. It is common to read and write accesses.

The GPMC\_CONFIG3\_i[12-8] ADVRDOFFTIME (read access) and GPMC\_CONFIG3\_i[20-16] ADVWROFFTIME (write access) bit fields define the nADV/ALE signal-deassertion time relative to start access time.

ADVONTIME can be used to control an address and byte-enable valid setup time control before nADV/ALE assertion. ADVRDOFFTIME and ADVWROFFTIME can be used to control an address and byte-enable valid hold time control after nADV/ALE deassertion. ADVRDOFFTIME and ADVWROFFTIME apply to synchronous and asynchronous modes.

The nADV/ALE signal transitions as controlled through ADVONTIME, ADVRDOFFTIME, and ADVWROFFTIME can be delayed by a half-GPMC\_FCLK period by enabling the GPMC\_CONFIG3\_i[7] ADVEXTRADELAY bit. This half-GPMC\_FCLK period provides more granularity on nADV/ALE assertion and deassertion time to ensure proper setup and hold time relative to GPMC output clock. The ADVEXTRADELAY configuration parameter is especially useful in configurations where GPMC output clock and GPMC\_FCLK have the same frequency, but can be used for all GPMC configurations. If enabled, ADVEXTRADELAY applies to all parameters controlling nADV/ALE transitions.

ADVEXTRADELAY must be used carefully to avoid control-signal overlap between successive accesses to different chip-selects. This implies the need to program the RDCYCLETIME and WRCYCLETIME bit fields to be greater than nADV/ALE signal-deassertion time, including the extra half-GPMC\_FCLK-period delay.

GPMC\_CONFIG3\_i[6-4] ADVAADMUXONTIME, GPMC\_CONFIG3\_i[26-24] ADVAADMUXRDOFFTIME, and GPMC\_CONFIG3\_i[30-28] ADVAADMUXWROFFTIME parameters have the same functions as ADVONTIME, ADVRDOFFTIME, and ADVWROFFTIME, but apply to the first address phase in the AAD-multiplexed protocol. The user must ensure that ADVAADMUXxxOFFTIME is programmed to a value less than or equal to ADVxxOFFTIME. Functionality in AAD-multiplexed mode is undefined if the settings do not comply with this requirement. ADVAADMUXxxOFFTIME can be programmed to the same value as ADVONTIME if no high nADV



pulse is needed between the two AAD-multiplexed address phases, which is the typical case in synchronous mode. In this configuration, nADV is kept low until it reaches the correct ADVxxOFFTIME.

For more information about the use of ADVONTIME, ADVRDOFFTIME, ADVWROFFTIME, and ADVAADMUXRDOFFTIME and ADVAADMUXWROFFTIME for command latch enable (CLE) and address latch enable (ALE) use for a NAND flash interface, see [Section 12.3.4.4.11](#), *GPMC NAND Access Description*.

#### **12.3.4.4.8.4 nOE/nRE: Output Enable/Read Enable Signal Control Assertion/Deassertion Time (OEONTIME / OEOFFTIME / OEEXTRADELAY / OEAADMUXONTIME / OEAADMUXOFFTIME)**

The GPMC\_CONFIG4\_i[3-0] OEONTIME bit field (where i = 0 to 3) defines the nOE/nRE signal assertion time relative to start access time. It applies only to read accesses.

The GPMC\_CONFIG4\_i[12-8] OEOFFTIME bit field defines the nOE/nRE signal deassertion time relative to start access time. It applies only to read accesses. nOE/nRE is not asserted during a write cycle.

The OEONTIME, OEOFFTIME, OEAADMUXONTIME, and OEAADMUXOFFTIME parameters apply to synchronous and asynchronous modes. OEONTIME can be used to control an address and byte enable valid setup time control before nOE/nRE assertion. OEOFFTIME can be used to control an address and byte-enable valid hold time control after nOE/nRE assertion.

The OEAADMUXONTIME and OEAADMUXOFFTIME parameters have the same functions as OEONTIME and OEOFFTIME, but apply to the first OE assertion in the AAD-multiplexed protocol for a read phase, or to the only OE assertion for a write phase. The user must ensure that OEAADMUXOFFTIME is programmed to a value less than OEONTIME. Functionality in AAD-multiplexed mode is undefined if the settings do not comply with this requirement. OEAADMUXOFFTIME must never be equal to OEONTIME because the AAD-multiplexed protocol requires a second address phase with the nOE signal deasserted before nOE can be asserted again to define a read command.

The nOE/RE signal transitions as controlled through OEONTIME, OEOFFTIME, OEAADMUXONTIME, and OEAADMUXOFFTIME can be delayed by a half-GPMC\_FCLK period by enabling the GPMC\_CONFIG4\_i[7] OEEXTRADELAY bit. This half-GPMC\_FCLK period provides more granularity on the nOE/RE assertion and deassertion time to ensure proper setup and hold time relative to GPMC output clock. If enabled, OEEXTRADELAY applies to all parameters controlling nOE/nRE transitions.

OEEXTRADELAY must be used carefully, to avoid control-signal overlap between successive accesses to different chip-selects. This implies the need to program RDCYCLETIME and WRCYCLETIME to be greater than the nOE/RE signal-deassertion time, including the extra half-GPMC\_FCLK-period delay.

#### **Note**

When the GPMC generates a read access to an address-/data-multiplexed device, it drives the address bus until nOE assertion time.

#### **12.3.4.4.8.5 nWE: Write Enable Signal Control Assertion/Deassertion Time (WEONTIME / WEOFFTIME / WEEXTRADELAY)**

The GPMC\_CONFIG4\_i[19-16] WEONTIME bit field (where i = 0 to 3) defines the nWE signal-assertion time relative to start access time. The GPMC\_CONFIG4\_i[28-24] WEOFFTIME bit field defines the nWE signal-deassertion time relative to start access time. These bit fields apply only to write accesses. nWE is not asserted during a read cycle.

WEONTIME can be used to control an address and byte-enable valid setup time control before nWE assertion. WEOFFTIME can be used to control an address and byte-enable valid hold time control after nWE assertion.

nWE signal transitions as controlled through WEONTIME, and WEOFFTIME can be delayed by a half-GPMC\_FCLK period by enabling the GPMC\_CONFIG4\_i[23] WEEXTRADELAY bit. This half-GPMC\_FCLK period provides more granularity on nWE assertion and deassertion time to ensure proper setup and hold time relative to GPMC output clock. If enabled, WEEXTRADELAY applies to all parameters controlling nWE transitions.

The WEEXTRADELAY bit must be used carefully to avoid control-signal overlap between successive accesses to different chip-selects. This implies the need to program the WRCYCLETIME bit field to be greater than the nWE signal-deassertion time, including the extra half-GPMC\_FCLK-period delay.

#### 12.3.4.4.8.6 GPMC\_CLKOUT

The GPMC output clock generated for external synchronous memory or device is GPMC\_CLKOUT.

- The GPMC\_CLKOUT clock frequency is the GPMC\_FCLK functional clock frequency divided by 1, 2, 3, or 4, depending on the GPMC\_CONFIG1\_i[1-0] GPMCFCLKDIVIDER bit field (where i = 0 to 3), with a guaranteed 50-percent duty cycle. For information about the duty cycle error, see the device-specific Datasheet.
- The GPMC\_CLKOUT clock is activated only when the access in progress is defined as synchronous (read or write access).
- The GPMC\_CONFIG1\_i[26-25] CLKACTIVATIONTIME bit field (where i = 0 to 3) defines the number of GPMC\_FCLK cycles from start access time to GPMC\_CLKOUT activation.
- The GPMC\_CLKOUT clock is stopped when cycle time completes and is asserted low between accesses.
- The GPMC\_CLKOUT clock is kept low when access is defined as asynchronous.

#### CAUTION

When the cycle time completes, the GPMC\_CLKOUT may be high because of the GPMC\_CONFIG1\_i[1-0] GPMCFCLKDIVIDER bit field. To ensure correct stoppage of the GPMC\_CLKOUT clock within the required 50-percent duty cycle, the user must extend the RDCYCLETIME or WRCYCLETIME value.

#### Note

To ensure a correct external clock cycle, the following rules must be applied:

- $(RDCYCLETIME - CLKACTIVATIONTIME)$  must be a multiple of  $(GPMCFCLKDIVIDER + 1)$ .
- The PAGEBURSTACCESSTIME value must be a multiple of  $(GPMCFCLKDIVIDER + 1)$ .

#### 12.3.4.4.8.7 GPMC Output Clock and Control Signals Setup and Hold

Control-signal transition (assertion and deassertion) setup and hold values with respect to the GPMC output clock edge can be controlled in the following ways:

- For the GPMC output clock signal, the GPMC\_CONFIG1\_i[26-25] CLKACTIVATIONTIME bit field (where i = 0 to 3) allows setup and hold control of control-signal assertion time.
- The use of a divided GPMC output clock allows setup and hold control of the control-signal assertion and deassertion times.
- When the GPMC output clock runs at the GPMC\_FCLK frequency so that GPMC output clock edge and control-signal transitions refer to the same GPMC\_FCLK edge, the control-signal transitions can be delayed by a half-GPMC\_FCLK period to provide minimum setup and hold times. This half-GPMC\_FCLK delay is enabled with the CSEXTRADELAY, ADVEXTRADELAY, OEEXTRADELAY, or WEEXTRADELAY parameter. This delay must be used carefully to prevent control-signal overlap between successive accesses to different chip-selects. This implies that the RDCYCLETIME and WRCYCLETIME are greater than the last control-signal deassertion time, including the extra half-GPMC\_FCLK cycle.

#### 12.3.4.4.8.8 Access Time (RDACCESSTIME / WRACCESSTIME)

The read/write access time durations can be programmed independently through the GPMC\_CONFIG5\_i[20-16] RDACCESSTIME and GPMC\_CONFIG6\_i[28-24] WRACCESSTIME bit fields (where i = 0 to 3). This allows nOE and GPMC data-capture timing parameters to be independent of nWE and memory device data capture timing parameters. The RDACCESSTIME and WRACCESSTIME bit fields can be set with a granularity of 1 or 2 through the GPMC\_CONFIG1\_i[4] TIMEPARAGRANULARITY bit.

#### 12.3.4.4.8.1 Access Time on Read Access

In asynchronous read mode, for single and paged accesses, the GPMC\_CONFIG5\_i[20-16] RDACCESSTIME bit field (where i = 0 to 3) defines the number of GPMC\_FCLK cycles from start access time to the GPMC\_FCLK rising edge used for the first data capture. RDACCESSTIME must be programmed to the rounded greater value (in GPMC\_FCLK cycles) of the read access time of the attached memory device.

In synchronous read mode, for single or burst accesses, RDACCESSTIME defines the number of GPMC\_FCLK cycles from the start access time to the GPMC\_FCLK rising edge corresponding to the GPMC output clock rising edge used for the first data capture.

GPMC output clock, which is sent to the memory device for synchronization with the GPMC controller, is internally retimed to correctly latch the returned data. The GPMC\_CONFIG5\_i[4-0] RDCYCLETIME bit field must be greater than RDACCESSTIME to let the GPMC latch the last return data using the internally retimed GPMC output clock.

The external WAIT signal can be used in conjunction with RDACCESSTIME to control the effective GPMC data-capture GPMC\_FCLK edge on read access in asynchronous and synchronous modes. For more information about wait monitoring, see [Section 12.3.4.4.7.3.1, WAIT Pin Monitoring Control](#).

#### 12.3.4.4.8.2 Access Time on Write Access

In asynchronous write mode, the GPMC\_CONFIG6\_i[28-24] WRACCESSTIME timing parameter is not used to define the effective write access time. Instead, it is used as a wait invalid timing window and must be set to a correct value so that the GPMC\_WAIT pin is at a valid state two GPMC output clock cycles before WRACCESSTIME completes. For more information about wait monitoring, see [Section 12.3.4.4.7.3.1, WAIT Pin Monitoring Control](#).

In synchronous write mode, for single or burst accesses, WRACCESSTIME defines the number of GPMC\_FCLK cycles from the start access time to the GPMC output clock rising edge used by the memory device for the first data capture.

The external WAIT signal can be used in conjunction with WRACCESSTIME to control the effective memory device data-capture GPMC output clock edge for a synchronous write access. For more information about wait monitoring, see [Section 12.3.4.4.7.3.1, WAIT Pin Monitoring Control](#).

#### 12.3.4.4.8.9 Page Burst Access Time (PAGEBURSTACCESSTIME)

The GPMC\_CONFIG5\_i[27-24] PAGEBURSTACCESSTIME bit field (where i = 0 to 3) can be set with a granularity of 1 or 2 through the GPMC\_CONFIG1\_i[4] TIMEPARAGRANULARITY bit.

##### 12.3.4.4.8.9.1 Page Burst Access Time on Read Access

In asynchronous page read mode, the delay between successive word captures in a page is controlled through the PAGEBURSTACCESSTIME bit field. The PAGEBURSTACCESSTIME parameter must be programmed to the rounded greater value (in GPMC\_FCLK cycles) of the read access time of the attached device.

In synchronous burst read mode, the delay between successive word captures in a burst is controlled through the PAGEBURSTACCESSTIME bit field.

The external WAIT signal can be used in conjunction with PAGEBURSTACCESSTIME to control the effective GPMC data-capture GPMC\_FCLK edge on read access. For more information about wait monitoring, see [Section 12.3.4.4.7.3.1, WAIT Pin Monitoring Control](#).

##### 12.3.4.4.8.9.2 Page Burst Access Time on Write Access

Asynchronous page write mode is not supported. PAGEBURSTACCESSTIME is irrelevant in this case.

In synchronous burst write mode, PAGEBURSTACCESSTIME controls the delay between successive memory device word captures in a burst.

The external WAIT signal can be used in conjunction with PAGEBURSTACCESSTIME to control the effective memory device data capture GPMC output clock edge in synchronous write mode. For more information about wait monitoring, see [Section 12.3.4.4.7.3.1](#), *WAIT Pin Monitoring Control*.

#### **12.3.4.4.8.10 Bus Keeping Support**

At the end cycle time of a read access, if no other access is pending, the GPMC drives the bus with the last data read after RDCYCLETIME completes to prevent bus floating and reduce power consumption.

After a write access, if no other access is pending, the GPMC keeps driving the data bus after WRCYCLETIME completes with the same data to prevent bus floating and power consumption.

#### **12.3.4.4.9 GPMC NOR Access Description**

For each chip-select configuration, the read access can be specified as asynchronous or synchronous access through the GPMC\_CONFIG1\_i[29] READTYPE bit (where i = 0 to 3). For each chip-select configuration, the write access can be specified as synchronous or asynchronous access through the GPMC\_CONFIG1\_i[27] WRITETYPE bit where (i = 0 to 3).

Asynchronous and synchronous read and write access time and related control signals are controlled through timing parameters that refer to GPMC\_FCLK. The primary difference of synchronous mode is the availability of a configurable clock interface to control the external device. Synchronous mode also affects data-capture and wait-pin monitoring schemes in read access.

For more information about asynchronous and synchronous access, see the descriptions of GPMC output clock (CLK), RdAccessTime, WrAccessTime, and WAIT pin monitoring.

For more information about timing-parameter settings, see the sample timing diagrams in this chapter.

---

#### **Note**

The address bus and nBE[1-0] are fixed for the duration of a synchronous burst read access, but they are updated for each byte of an asynchronous page-read access.

---

#### **12.3.4.4.9.1 Asynchronous Access Description**

This section describes:

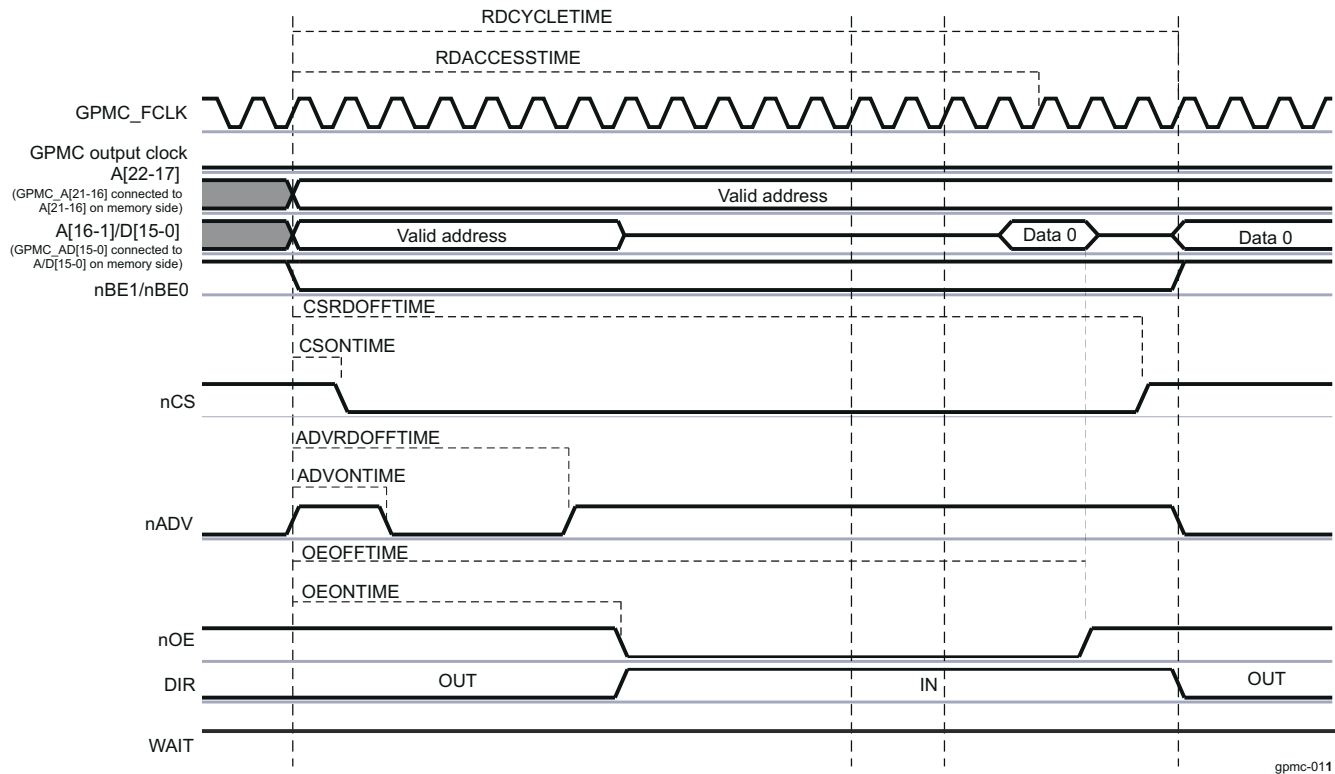
- Asynchronous single-read operation on an address/data multiplexed device
- Asynchronous single write operation on an address/data-multiplexed device
- Asynchronous single read operation on an AAD-multiplexed device
- Asynchronous single write operation on an AAD-multiplexed device
- Asynchronous multiple (page) read operation on a non-multiplexed device

In asynchronous operations GPMC output clock is not provided outside the GPMC and is kept low.

##### **12.3.4.4.9.1.1 Access on Address/Data Multiplexed Devices**

##### **12.3.4.4.9.1.1.1 Asynchronous Single-Read Operation on an Address/Data Multiplexed Device**

[Figure 12-244](#) shows an asynchronous single read operation on an address/data-multiplexed device.



**Figure 12-244. Asynchronous Single Read on an Address/Data-Multiplexed Device**

For formulas to calculate timing parameters, see [Section 12.3.4.5.6.1, GPMC Timing Parameters Formulas](#).

[Table 12-361](#) lists the timing bit fields to set up to configure the GPMC in asynchronous single-read mode.

When the GPMC generates a read access to an address/data-multiplexed device, it drives the address bus until nOE assertion time. For more information, see [Section 12.3.4.4.7.2.3, Address/Data-Multiplexing Interface](#).

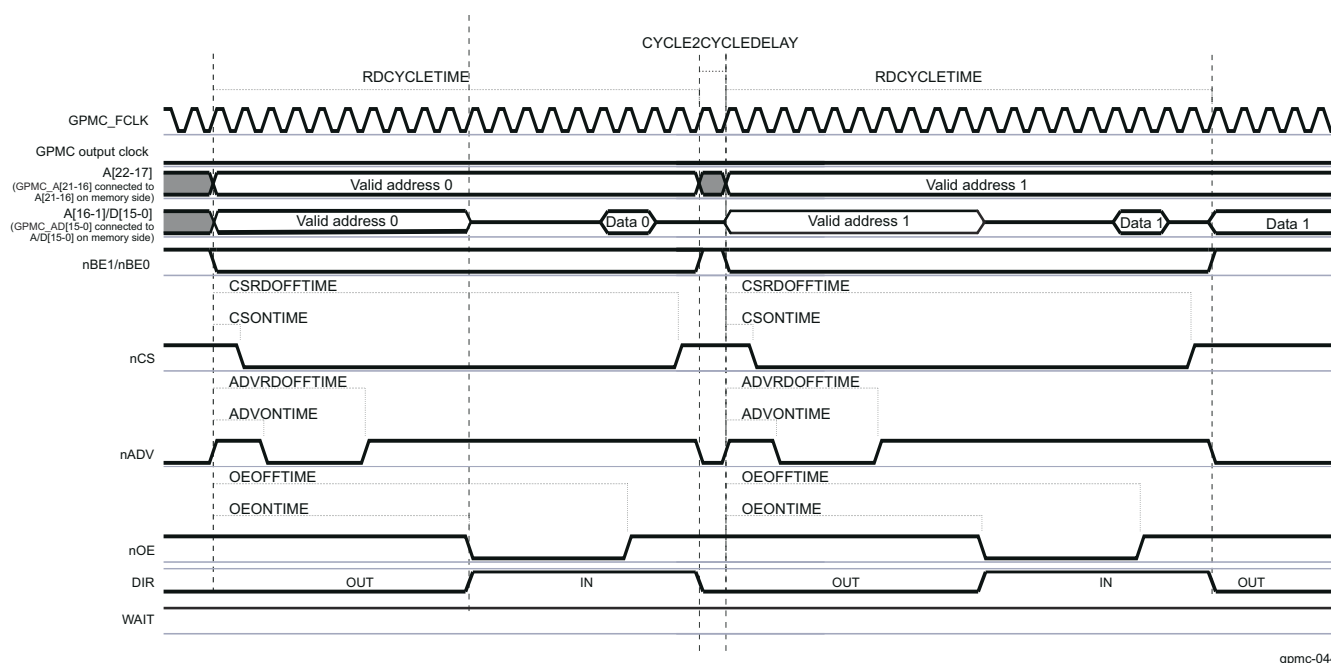
Address bits (A[16-1] from a GPMC perspective, A[15-0] from an external device perspective) are placed on the address/data bus, and the remaining address bits GPMC\_A[22-16] are placed on the address bus. The address phase ends at nOE assertion, when the DIR signal goes from OUT to IN.

- Chip-select signal nCS:
  - nCS assertion time is controlled by the GPMC\_CONFIG2\_i[3-0] CSONTIME bit field. It controls the address setup time to nCS assertion.
  - nCS deassertion time is controlled by the GPMC\_CONFIG2\_i[12-8] CSRDOFFTIME bit field. It controls the address hold time from nCS deassertion.
- Address valid signal nADV:
  - nADV assertion time is controlled by the GPMC\_CONFIG3\_i[3-0] ADVONTIME bit field.
  - nADV deassertion time is controlled by the GPMC\_CONFIG3\_i[12-8] ADVRDOFFTIME bit field.
- Output enable signal nOE:
  - nOE assertion indicates a read cycle.
  - nOE assertion time is controlled by the GPMC\_CONFIG4\_i[3-0] OEONTIME bit field.
  - nOE deassertion time is controlled by the GPMC\_CONFIG4\_i[12-8] OEOFFTIME bit field.
- Read data is latched when RDACCESSTIME completes. Access time is defined in the GPMC\_CONFIG5\_i[20-16] RDACCESSTIME bit field.
- Direction signal DIR: DIR goes from OUT to IN at the same time that nOE is asserted.
- The end of the access is defined by the GPMC\_CONFIG5\_i[4-0] RDCYCLETIME parameter.

In the GPMC, when a 16-bit wide device is attached to the controller, a 32-bit word write access is split into two 16-bit word write accesses. For more information about GPMC access size and type adaptation, see [Section 12.3.4.4.9.5, System Burst Versus External Device Burst Support](#).

Between two successive accesses, if an nCS pulse is needed:

- The GPMC\_CONFIG6\_i[11-8] CYCLE2CYCLEDELAY bit field can be programmed with the GPMC\_CONFIG6\_i[7] CYCLE2CYCLESAMECSN bit enabled.
- The CSWROFFTIME and CSONTIME parameters also allow a chip-select pulse, but this affects all other types of access.

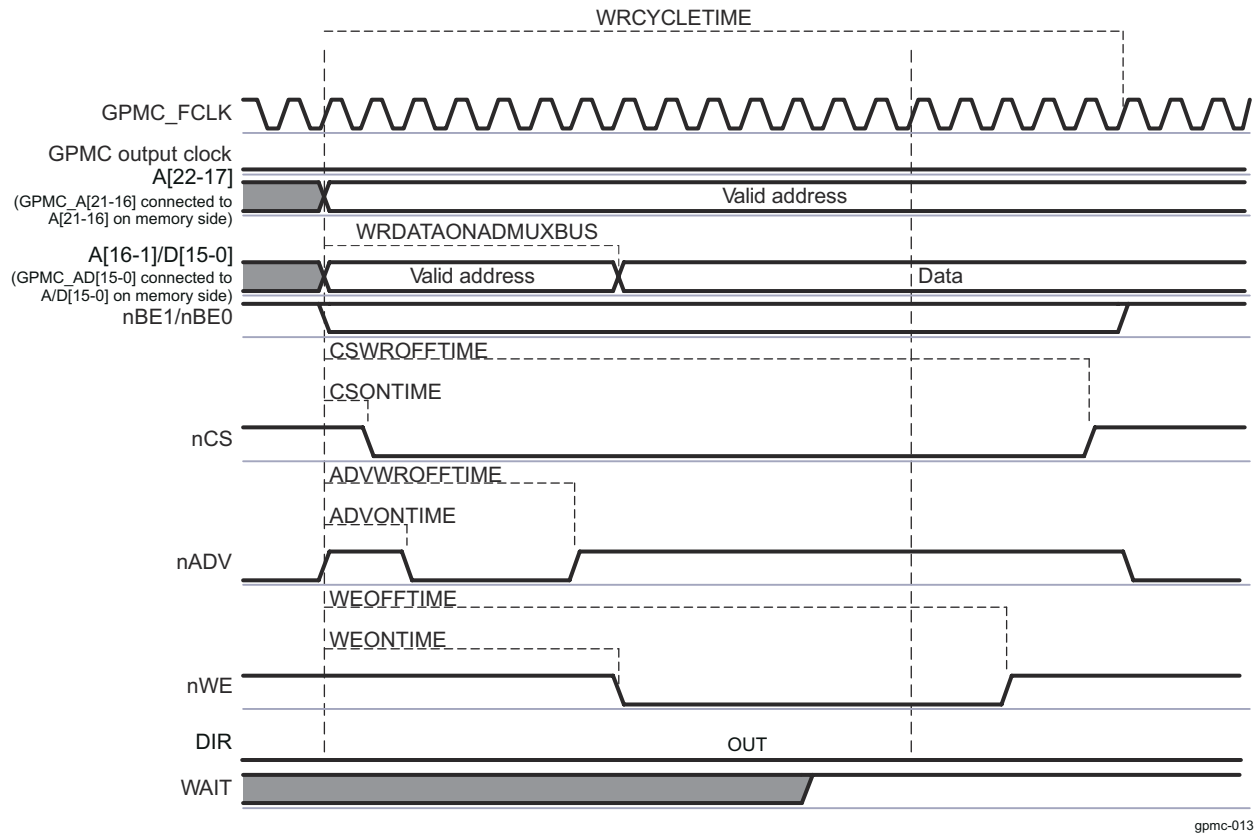


**Figure 12-245. Two Asynchronous Single-Read Accesses on an Address/Data-Multiplexed Device (32-Bit Read Split Into 2 x 16-Bit Read)**

#### 12.3.4.4.9.1.2 Asynchronous Single-Write Operation on an Address/Data-Multiplexed Device

[Figure 12-246](#) shows an asynchronous single-write operation on an address/data-multiplexed device.





**Figure 12-246. Asynchronous Single-Write on an Address/Data-Multiplexed Device**

For formulas to calculate timing parameters, see [Section 12.3.4.5.6.1, GPMC Timing Parameters Formulas](#).

[Table 12-361](#) lists the timing bit fields to set up to configure the GPMC in asynchronous single-write mode.

When the GPMC generates a write access to an address/data-multiplexed device, it drives the address bus until nWE assertion time. For more information, see [Section 12.3.4.4.7.2.3, Address/Data-Multiplexing Interface](#).

The nCS and nADV signals are controlled in the same way as for a asynchronous single-read operation on an address/data-multiplexed device.

- Write enable signal nWE:
  - nWE assertion indicates a write cycle.
  - nWE assertion time is controlled by the GPMC\_CONFIG4\_i[19-16] WEONTIME bit field.
  - nWE deassertion time is controlled by the GPMC\_CONFIG4\_i[28-24] WEOFFTIME bit field.
- Direction signal DIR: DIR signal is OUT during the entire access.
- The end of the access is defined by the GPMC\_CONFIG5\_i[12-8] WRCYCLETIME parameter.

Address bits A[16-1] (GPMC point of view) are placed on the address/data bus at the start of cycle time, and the remaining address bits A[22-17] are placed on the address bus.

Data is driven on the address/data bus at a GPMC\_CONFIG6\_i[19-16] WRDATAONADMUXBUS time.

#### Note

Multiple write access in asynchronous mode is not supported. If WRITEMULTIPLE is enabled with WRITETYPE as asynchronous, the GPMC processes single asynchronous accesses.

After a write operation, if no other access (read or write) is pending, the data bus keeps its previous value. See [Section 12.3.4.4.8.10, Bus Keeping Support](#).

### 12.3.4.4.9.1.1.3 Asynchronous Multiple (Page) Write Operation on an Address/Data-Multiplexed Device

Write multiple (page) access in asynchronous mode is not supported for address/data-multiplexed devices.

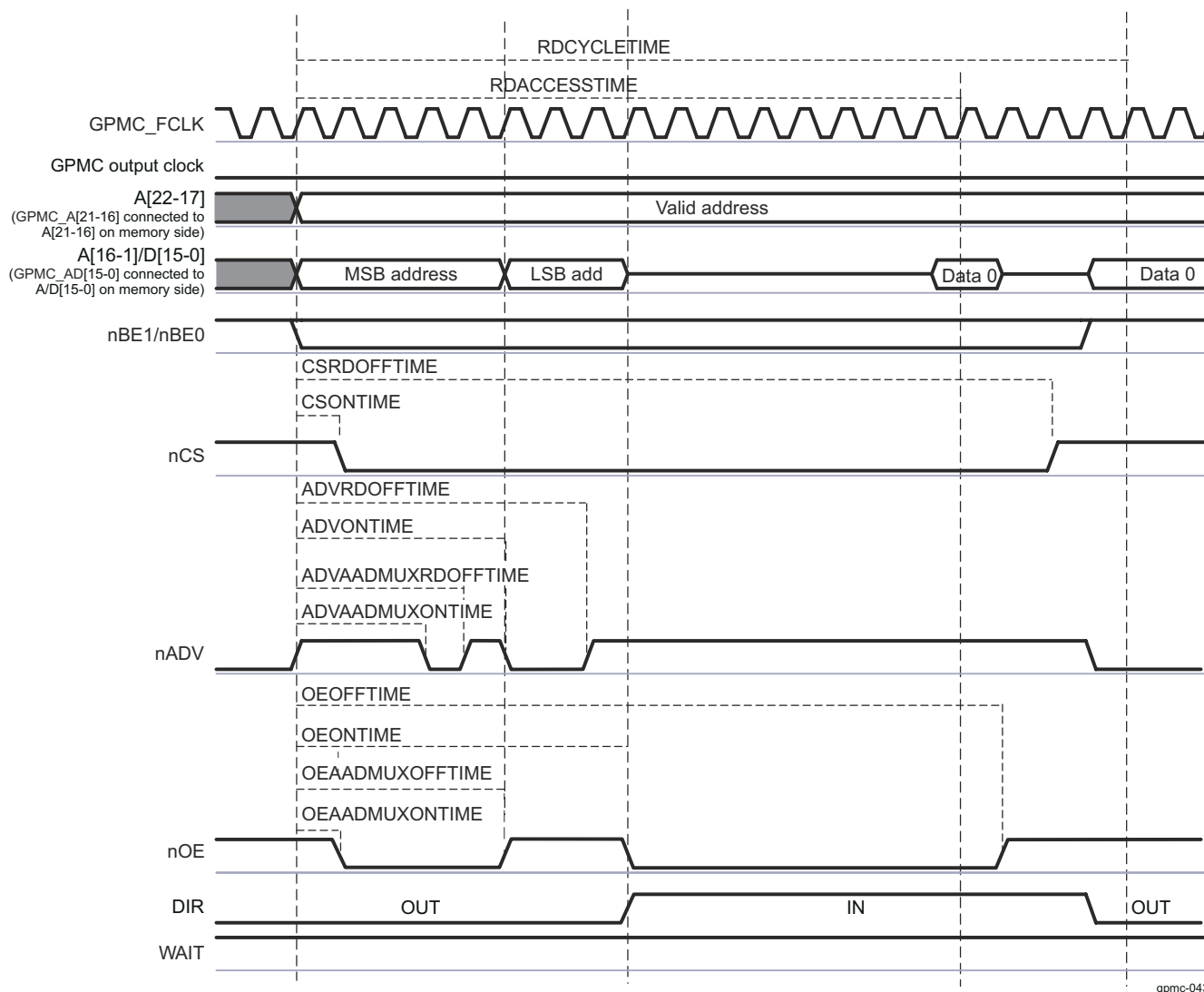
If the GPMC\_CONFIG1\_i[28] WRITEMULTIPLE bit is enabled (0x1) with the GPMC\_CONFIG1\_i[27] WRITETYPE bit as asynchronous (0x0), the GPMC processes single asynchronous accesses.

For accesses on non-multiplexed devices, see [Section 12.3.4.4.9.3, Asynchronous and Synchronous Accesses in non-multiplexed Mode](#).

### 12.3.4.4.9.1.2 Access on Address/Address/Data-Multiplexed Devices

#### 12.3.4.4.9.1.2.1 Asynchronous Single Read Operation on an AAD-Multiplexed Device

Figure 12-247 shows an asynchronous single-read operation on an AAD-multiplexed device.



**Figure 12-247. Asynchronous Single Read on an AAD-Multiplexed Device**

For formulas to calculate timing parameters, see [Section 12.3.4.5.6.1, GPMC Timing Parameters Formulas](#).

[Table 12-361](#) lists the timing bit fields to set up to configure the GPMC in asynchronous single write mode.

When the GPMC generates a read access to an AAD-multiplexed device, all address bits are driven onto the address/data bus in two separate phases. The first phase is used for the MSB address and is qualified with nOE



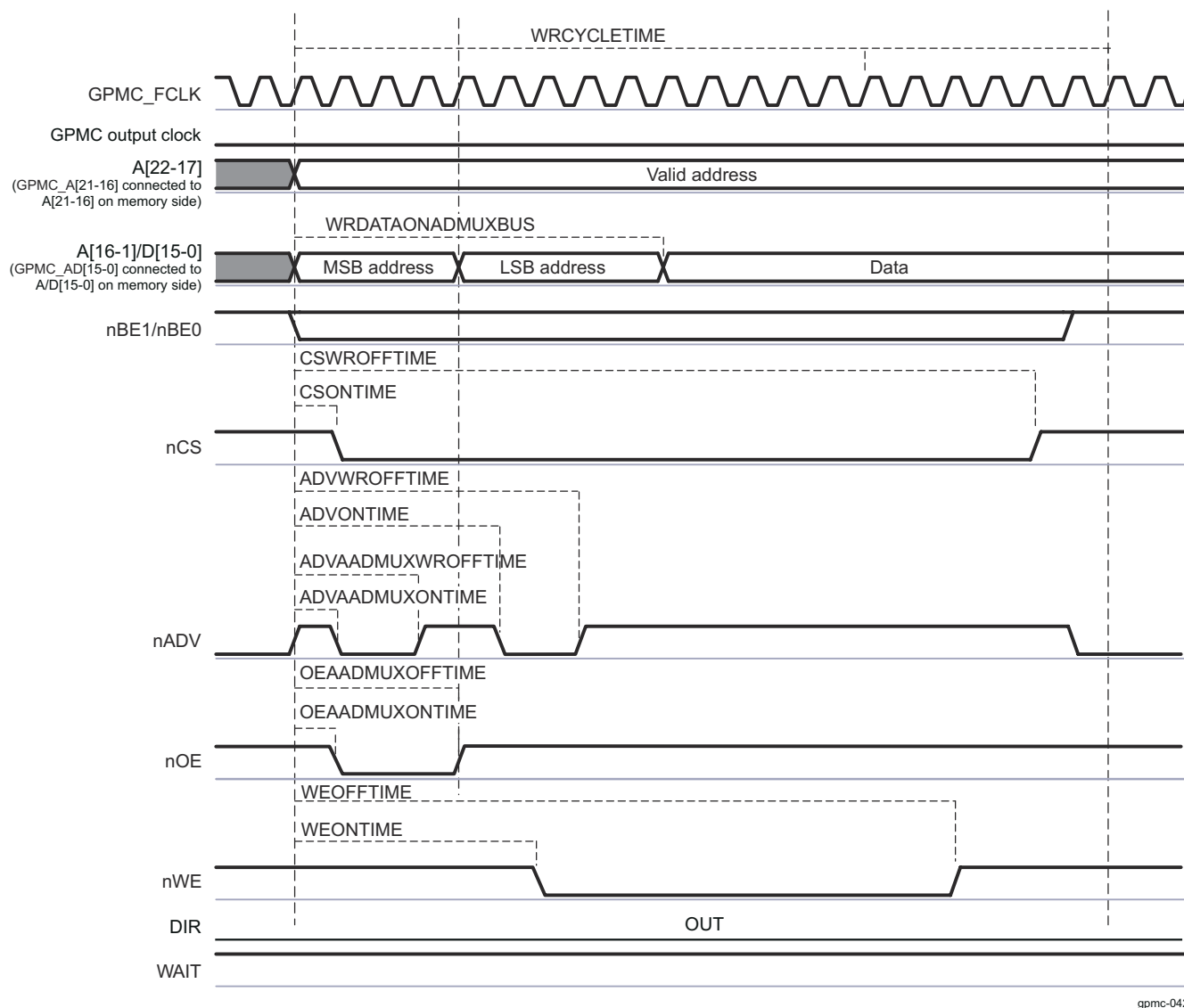
driven low. The first address phase ends at the first nOE deassertion time. The second phase for LSB address is qualified with nOE driven high. The second address phase ends at the second nOE assertion time, when the DIR signal goes from OUT to IN.

The nCS and DIR signals are controlled in the same way as for an asynchronous single-read operation on an address/data-multiplexed device.

- Address valid signal nADV. nADV is asserted and deasserted twice during a read transaction:
  - nADV first assertion time is controlled by the GPMC\_CONFIG3\_i[6-4] ADVAADMUXONTIME bit field.
  - nADV first deassertion time is controlled by the GPMC\_CONFIG3\_i[26-24] ADVAADMUXRDOFFTIME bit field.
  - nADV second assertion time is controlled by the GPMC\_CONFIG3\_i[3-0] ADVONTIME bit field.
  - nADV second deassertion time is controlled by the GPMC\_CONFIG3\_i[12-8] ADVRDOFFTIME bit field.
- Output Enable signal nOE. nOE is asserted and deasserted twice during a read transaction (nOE second assertion indicates a read cycle):
  - nOE first assertion time is controlled by the GPMC\_CONFIG4\_i[6-4] OEAADMUXONTIME bit field.
  - nOE first deassertion time is controlled by the GPMC\_CONFIG3\_i[15-13] OEAADMUXOFFTIME bit field.
  - nOE second assertion time is controlled by the GPMC\_CONFIG4\_i[3-0] OEONTIME bit field.
  - nOE second deassertion time is controlled by the GPMC\_CONFIG4\_i[12-8] OEOFFTIME bit field.

#### **12.3.4.4.9.1.2.2 Asynchronous Single-Write Operation on an AAD-Multiplexed Device**

Figure 12-248 shows an asynchronous single-write operation on an AAD-multiplexed device.



gpmc-042

**Figure 12-248. Asynchronous Single Write on an AAD-Multiplexed Device**

For formulas to calculate timing parameters, see [Section 12.3.4.5.6.1, GPMC Timing Parameters Formulas](#).

[Table 12-361](#) lists the timing bit fields to set up to configure the GPMC in asynchronous single-write mode.

When the GPMC generates a write access to an AAD-multiplexed device, all address bits are driven onto the address/data bus in two separate phases. The first phase is used for the MSB address and is qualified with nOE driven low. The second phase for LSB address is qualified with nOE driven high. The address phase ends at nWE assertion time.

The nCS, nWE, and DIR signals are controlled in the same way as for an asynchronous single-write operation on an address/data-multiplexed device. See [Table 12-352, NAND Memory Type](#).

- Address valid signal nADV is asserted and deasserted twice during a write transaction:
  - nADV first assertion time is controlled by the GPMC\_CONFIG3\_i[6-4] ADVAADMUXONTIME bit field.
  - nADV first deassertion time is controlled by the GPMC\_CONFIG3\_i[30-28] ADVAADMUXWROFFTIME bit field.
  - nADV second assertion time is controlled by the GPMC\_CONFIG3\_i[3-0] ADVONTIME bit field.
  - nADV second deassertion time is controlled by the GPMC\_CONFIG3\_i[20-16] ADVWROFFTIME bit field.
- Output enable signal nOE is asserted during the address phase of a write transaction:

- nOE assertion time is controlled by the GPMC\_CONFIG4\_i[6-4] OEAADMUXONTIME bit field.
- nOE deassertion time is controlled by the GPMC\_CONFIG3\_i[15-13] OEAADMUXOFFTIME bit field.

The address bits for the first address phase are driven onto the data bus until nOE deassertion. Data is driven onto the address/data bus at the clock edge defined by the GPMC\_CONFIG6\_i[19-16] WRDATAONADMUXBUS parameter.

#### **12.3.4.4.9.1.2.3 Asynchronous Multiple (Page) Read Operation on an AAD-Multiplexed Device**

Write multiple (page) access in asynchronous mode is not supported for AAD-multiplexed devices.

If the GPMC\_CONFIG1\_i[28] WRITEMULTIPLE bit is enabled (0x1) with the GPMC\_CONFIG1\_i[27] WRITETYPE bit as asynchronous (0x0), the GPMC processes single asynchronous accesses.

For accesses on non-multiplexed devices, see [Section 12.3.4.4.9.3, Asynchronous and Synchronous Accessed in non-multiplexed Mode](#).

#### **12.3.4.4.9.2 Synchronous Access Description**

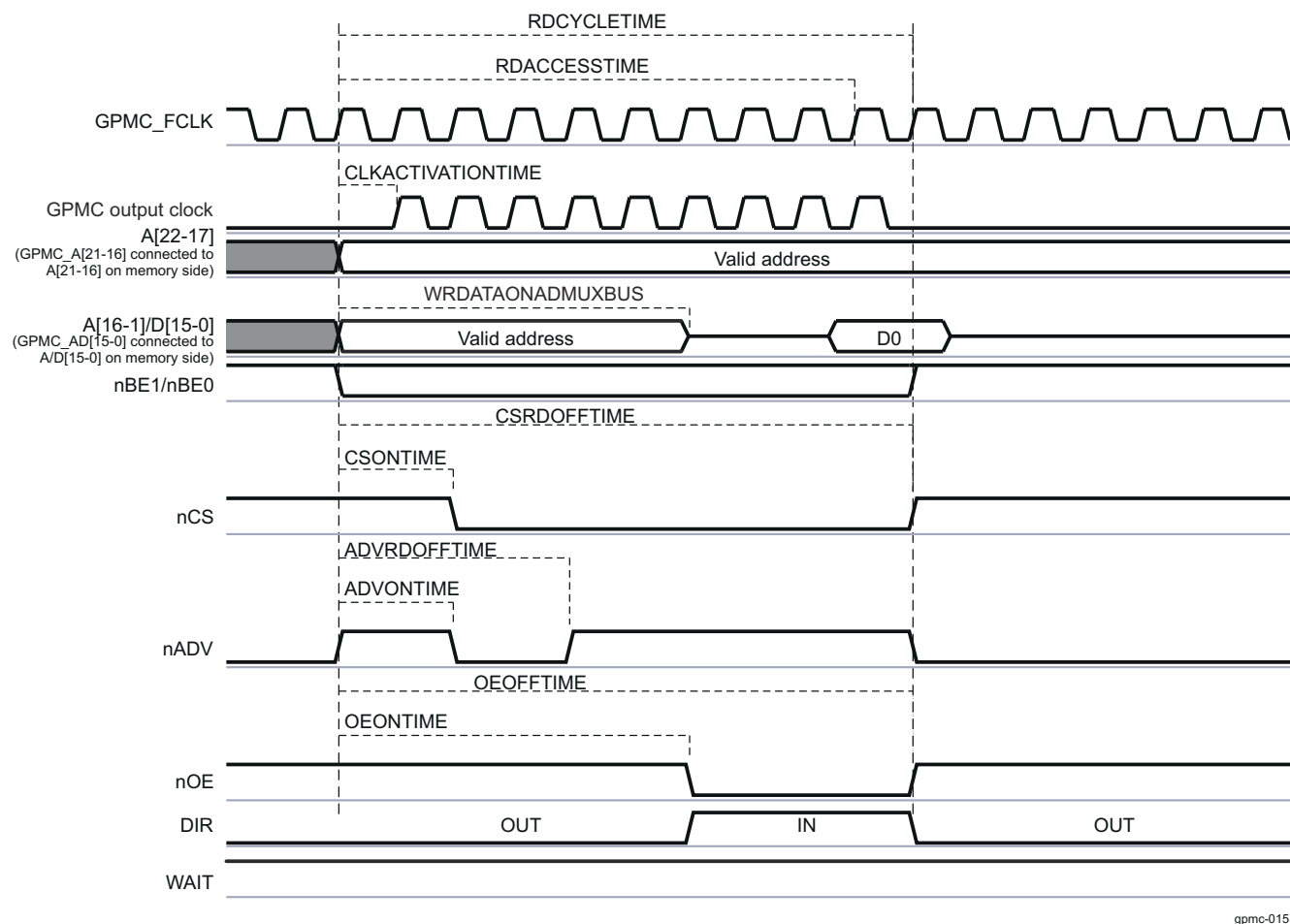
This section describes read and write synchronous accesses on address/data-multiplexed devices. All information in this section can be applied to any type of memory (non-multiplexed, address and data-multiplexed, or AAD-multiplexed) with the difference limited to the address phase. For accesses on non-multiplexed devices, see [Section 12.3.4.4.9.3, Asynchronous and Synchronous Accessed in non-multiplexed Mode](#).

In synchronous operations:

- The GPMC\_CLKOUT clock is provided outside the GPMC when accessing the memory device.
- The GPMC\_CLKOUT clock is derived from the GPMC\_FCLK clock using the GPMC\_CONFIG1\_i[1-0] GPMCFCLKDIVIDER bit field. In the following section i stands for the chip-select number, i = 0 to 3.
- The GPMC\_CONFIG1\_i[26-25] CLKACTIVATIONTIME bit field specifies that the GPMC\_CLKOUT is provided outside the GPMC for 0 to 2 GPMC\_FCLK cycles after start access time until RDCYCLETIME or WRCYCLETIME completes.

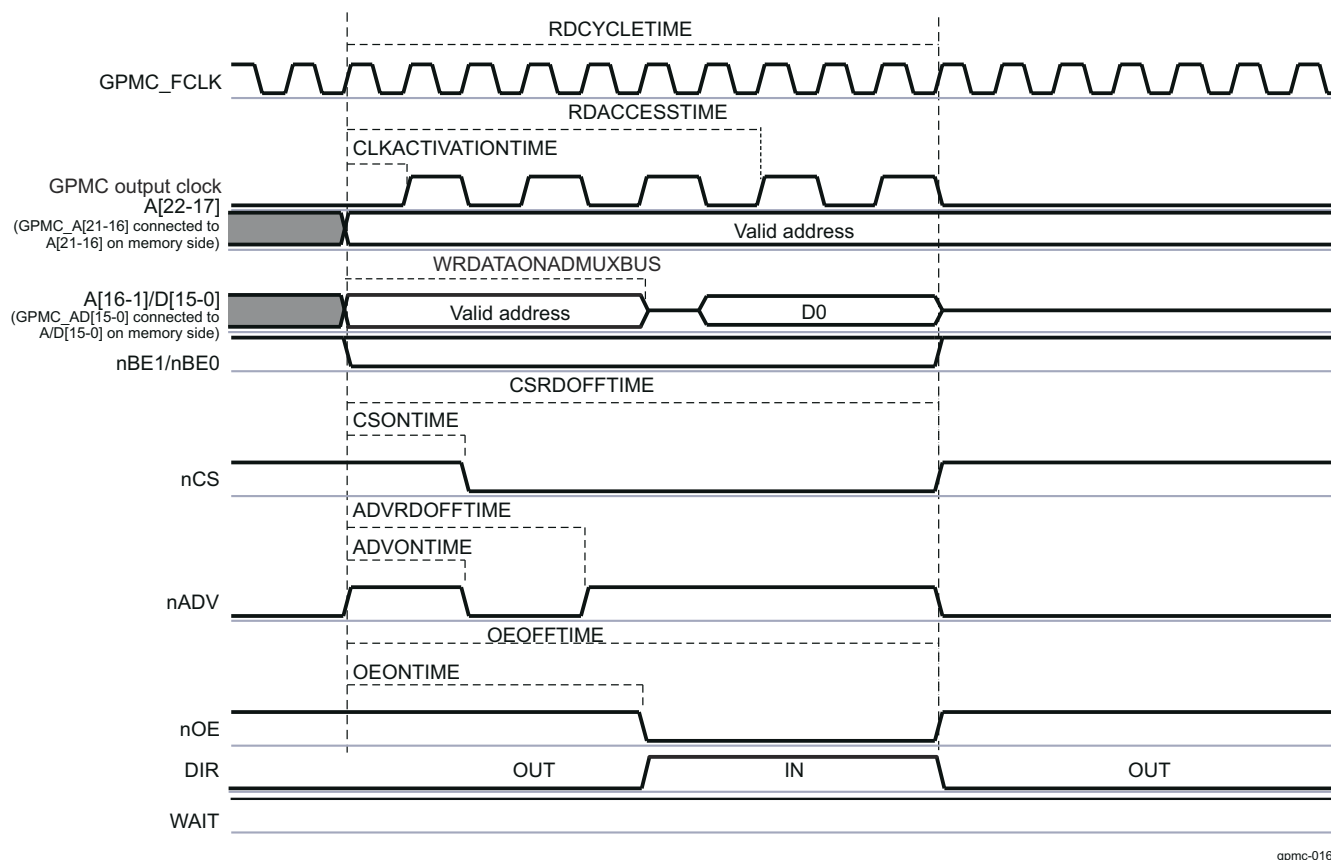
#### **12.3.4.4.9.2.1 Synchronous Single Read**

[Figure 12-249](#) and [Figure 12-250](#) show a synchronous single-read operation with GPMCFCLKDIVIDER equal to 0 and 1, respectively.



gpmc-015

**Figure 12-249. Synchronous Single Read (GPMCCLKDIVIDER = 0)**



gpmc-016

**Figure 12-250. Synchronous Single Read (GPMCCLKDIVIDER = 1)**

For formulas to calculate timing parameters, see [Section 12.3.4.5.6.1, GPMC Timing Parameters Formulas](#).

[Table 12-361](#) lists the timing bit fields to set up to configure the GPMC in asynchronous single-read mode.

When the GPMC generates a read access to an address/data-multiplexed device, it drives the address bus until nOE assertion time. For more information, see [Section 12.3.4.4.7.2.3, Address/Data-Multiplexing Interface](#).

- Chip-select signal nCS:
  - nCS assertion time is controlled by the GPMC\_CONFIG2\_i[3-0] CSONTIME bit field and ensures address setup time to nCS assertion.
  - nCS deassertion time is controlled by the GPMC\_CONFIG2\_i[12-8] CSRDOFFTIME bit field and ensures address hold time to nCS deassertion.
- Address valid signal nADV:
  - nADV assertion time is controlled by the GPMC\_CONFIG3\_i[3-0] ADVONTIME bit field.
  - nADV deassertion time is controlled by the GPMC\_CONFIG3\_i[12-8] ADVRDOFFTIME bit field.
- Output enable signal nOE:
  - nOE assertion indicates a read cycle.
  - nOE assertion time is controlled by the GPMC\_CONFIG4\_i[3-0] OEONTIME bit field.
  - nOE deassertion time is controlled by the GPMC\_CONFIG4\_i[12-8] OEOFFTIME bit field.
- Initial latency for the first read data is controlled by GPMC\_CONFIG5\_i[20-16] RDACCESSTIME bit field or by monitoring the WAIT signal.
- Total access time (the GPMC\_CONFIG5\_i[4-0] RDCYCLETIME bit field) corresponds to RDACCESSTIME plus the address hold time from nCS deassertion, plus time from RDACCESSTIME to CSRDOFFTIME.
- Direction signal DIR: DIR goes from OUT to IN at the same time as nOE assertion.

When the GPMC generates a write access to an AAD-multiplexed device, all address bits are driven onto the address/data bus in two separate phases. The first phase is used for the MSB address and is qualified with nOE driven low. The second phase for LSB address is qualified with nOE driven high. The address phase ends at nWE assertion time.

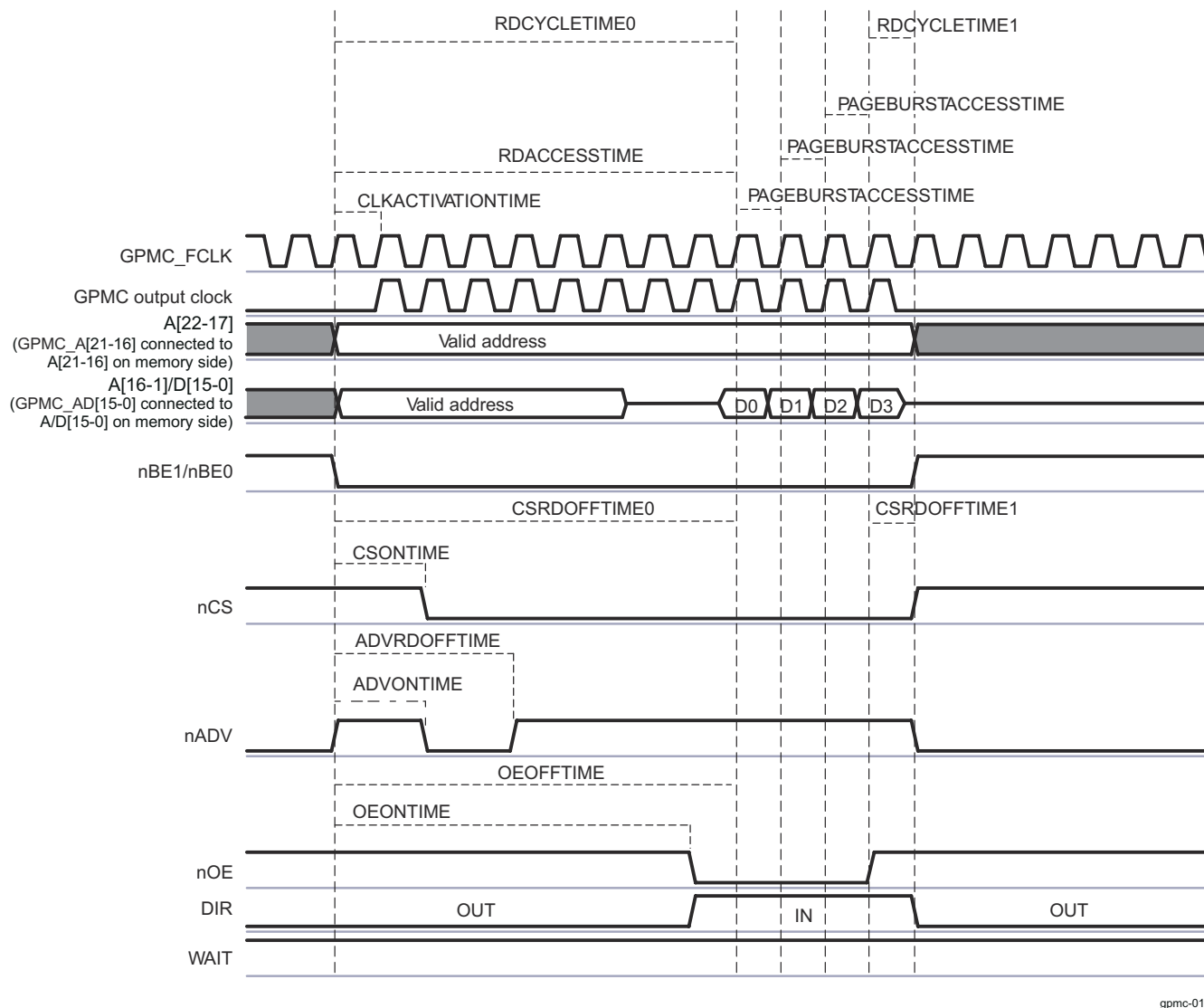
The nCS and DIR signals are controlled in the same way as for a synchronous single-read operation on an address/data-multiplexed device.

- Address valid signal nADV is asserted and deasserted twice during a read transaction:
  - nADV first assertion time is controlled by the GPMC\_CONFIG3\_i[6-4] ADVAADMUXONTIME bit field.
  - nADV first deassertion time is controlled by the GPMC\_CONFIG3\_i[26-24] ADVAADMUXRDOFFTIME bit field.
  - nADV second assertion time is controlled by the GPMC\_CONFIG3\_i[3-0] ADVONTIME bit field.
  - nADV second deassertion time is controlled by the GPMC\_CONFIG3\_i[12-8] ADVRDOFFTIME bit field.
- Output Enable signal nOE is asserted and deasserted twice during a read transaction (nOE second assertion indicates a read cycle):
  - nOE first assertion time is controlled by the GPMC\_CONFIG4\_i[6-4] OEAADMUXONTIME bit field.
  - nOE first deassertion time is controlled by the GPMC\_CONFIG3\_i[15-13] OEAADMUXOFFTIME bit field.
  - nOE second assertion time is controlled by the GPMC\_CONFIG4\_i[3-0] OEONTIME bit field.
  - nOE second deassertion time is controlled by the GPMC\_CONFIG4\_i[12-8] OEOFFTIME bit field.

After a read operation, if no other access (read or write) is pending, the data bus is driven with the previous read value. See [Section 12.3.4.4.8.10, Bus Keeping Support](#).

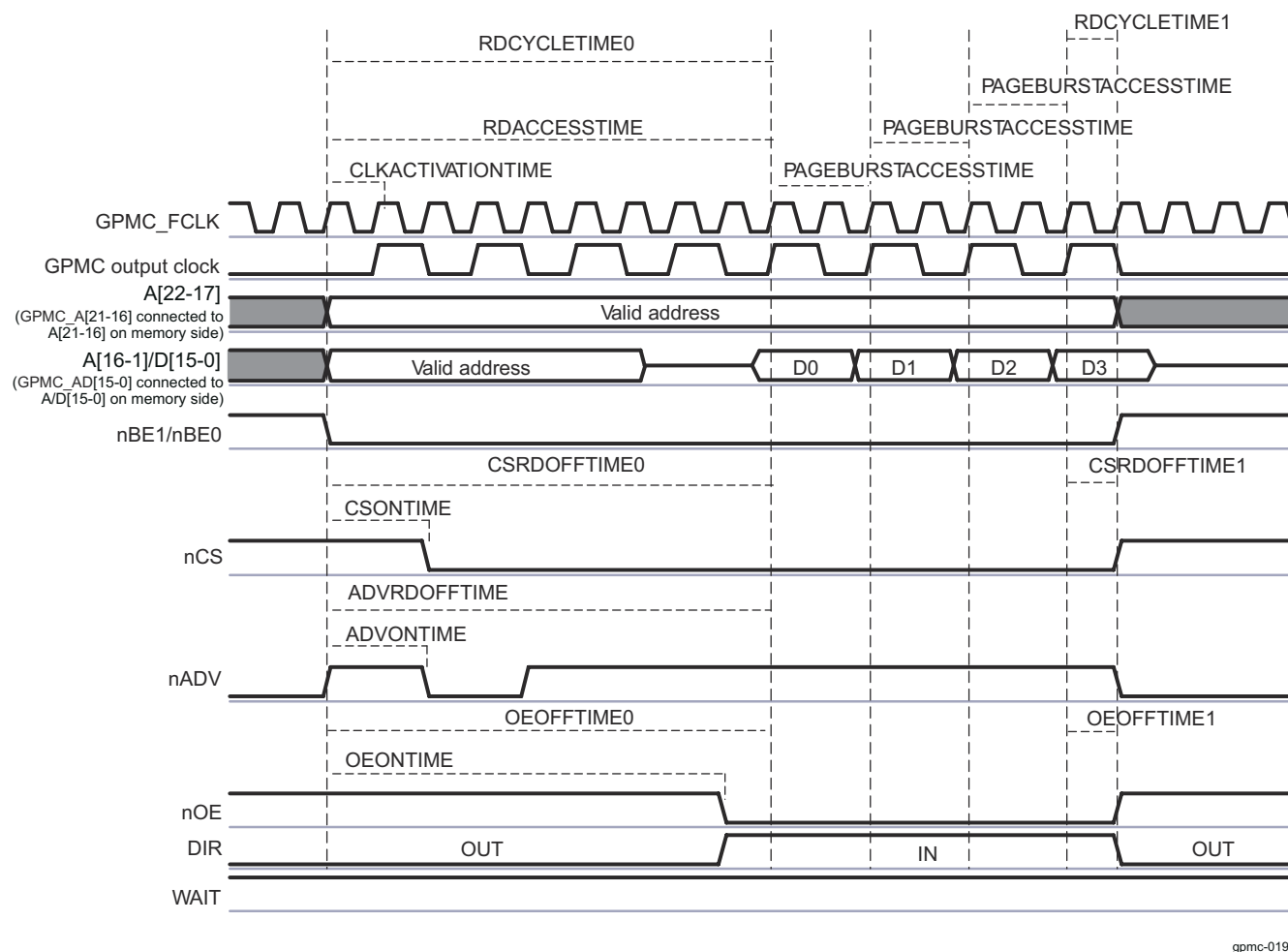
#### **12.3.4.4.9.2.2 Synchronous Multiple (Burst) Read (4-, 8-, 16-Word16 Burst With Wraparound Capability)**

[Figure 12-251](#) and [Figure 12-252](#) show a synchronous multiple-read operation with GPMCFCLKDivider equal to 0 and 1, respectively.



gpmc-018

**Figure 12-251. Synchronous Multiple (Burst) Read (GPMCFCLKDIVIDER = 0)**



gpmc-019

**Figure 12-252. Synchronous Multiple (Burst) Read (GPMCFCLKDIVIDER = 1)**

When the GPMC\_CONFIG5\_i[20-16] RDACCESSTIME bit field completes, control-signal timings are frozen during the multiple data transactions, corresponding to the GPMC\_CONFIG5\_i[27-24] PAGEBURSTACCESSTIME bit field multiplied by the number of remaining data transactions.

The nCS, nADV, nOE, and DIR signals are controlled in the same way as for a synchronous single-read operation. See [Table 12-347, NOR Memory Type](#).

Initial latency for the first read data is controlled by RDACCESSTIME or by monitoring the WAIT signal. Successive read data are provided by the memory device every one or two GPMC\_CLKOUT cycles. The PAGEBURSTACCESSTIME parameter must be set accordingly with the GPMC\_CONFIG1\_i[1-0] GPMCFCLKDIVIDER bit field and the memory-device internal configuration. Depending on the device page length, the GPMC checks the device page crossing during a new burst request and purposely inserts initial latency (of RDACCESSTIME) when required.

Total access time GPMC\_CONFIG5\_i[4-0] RDCYCLETIME corresponds to RDACCESSTIME plus the address hold time from nCS deassertion. In [Figure 12-252](#), the programmed value of RDCYCLETIME equals RDCYCLETIME0 + RDCYCLETIME1.

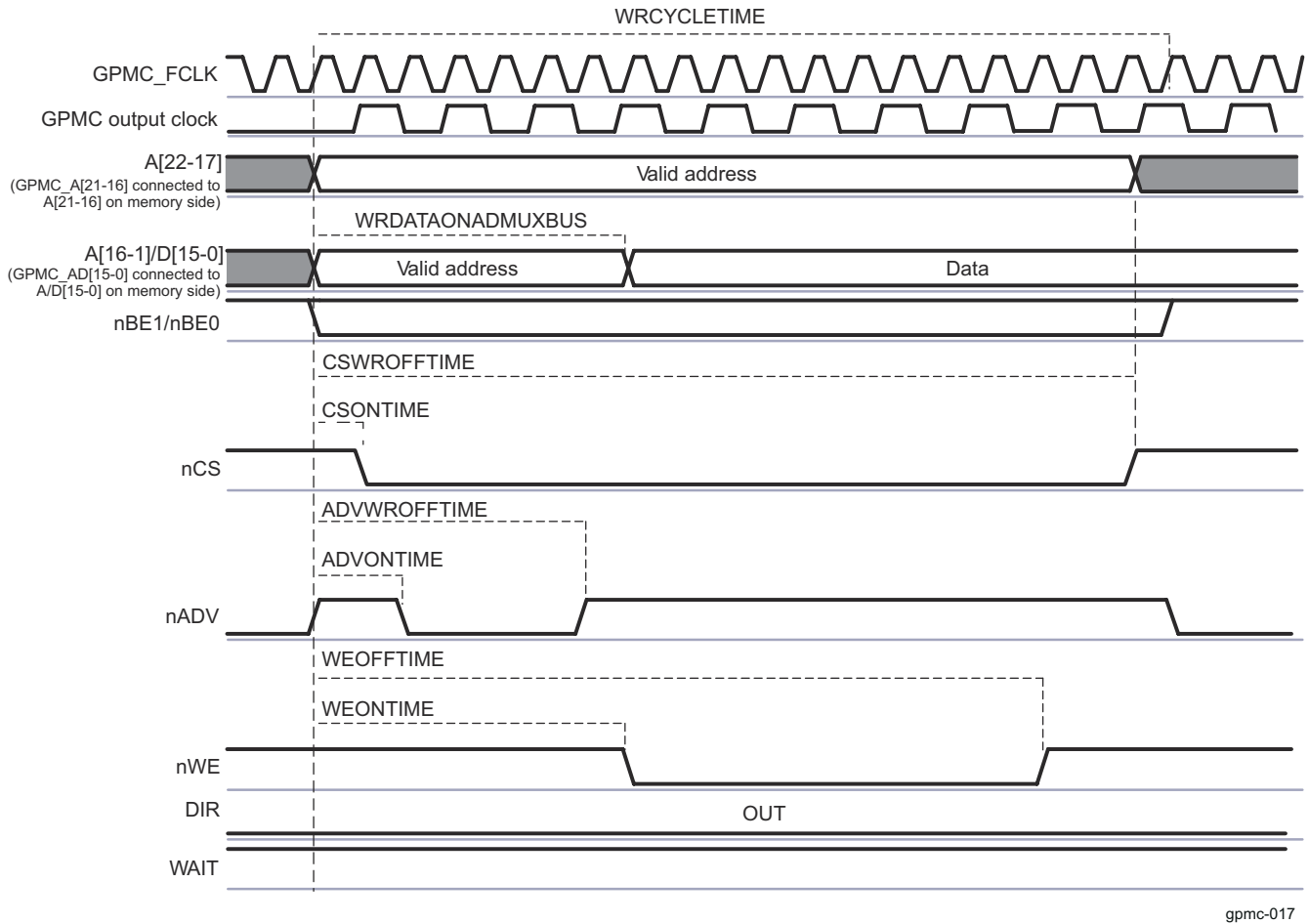
After a read operation, if no other access (read or write) is pending, the data bus is driven with the previous read value. See [Section 12.3.4.4.8.10, Bus Keeping Support](#).

Burst wraparound is enabled through the GPMC\_CONFIG1\_i[31] WRAPBURST bit and allows a 4-, 8-, or 16-Word linear burst access to wrap within its burst-length boundary through the GPMC\_CONFIG1\_i[24-23] ATTACHEDDEVICEPAGELENGTH bit field.



#### 12.3.4.4.9.2.3 Synchronous Single Write

Burst write mode is used for synchronous single or burst accesses.



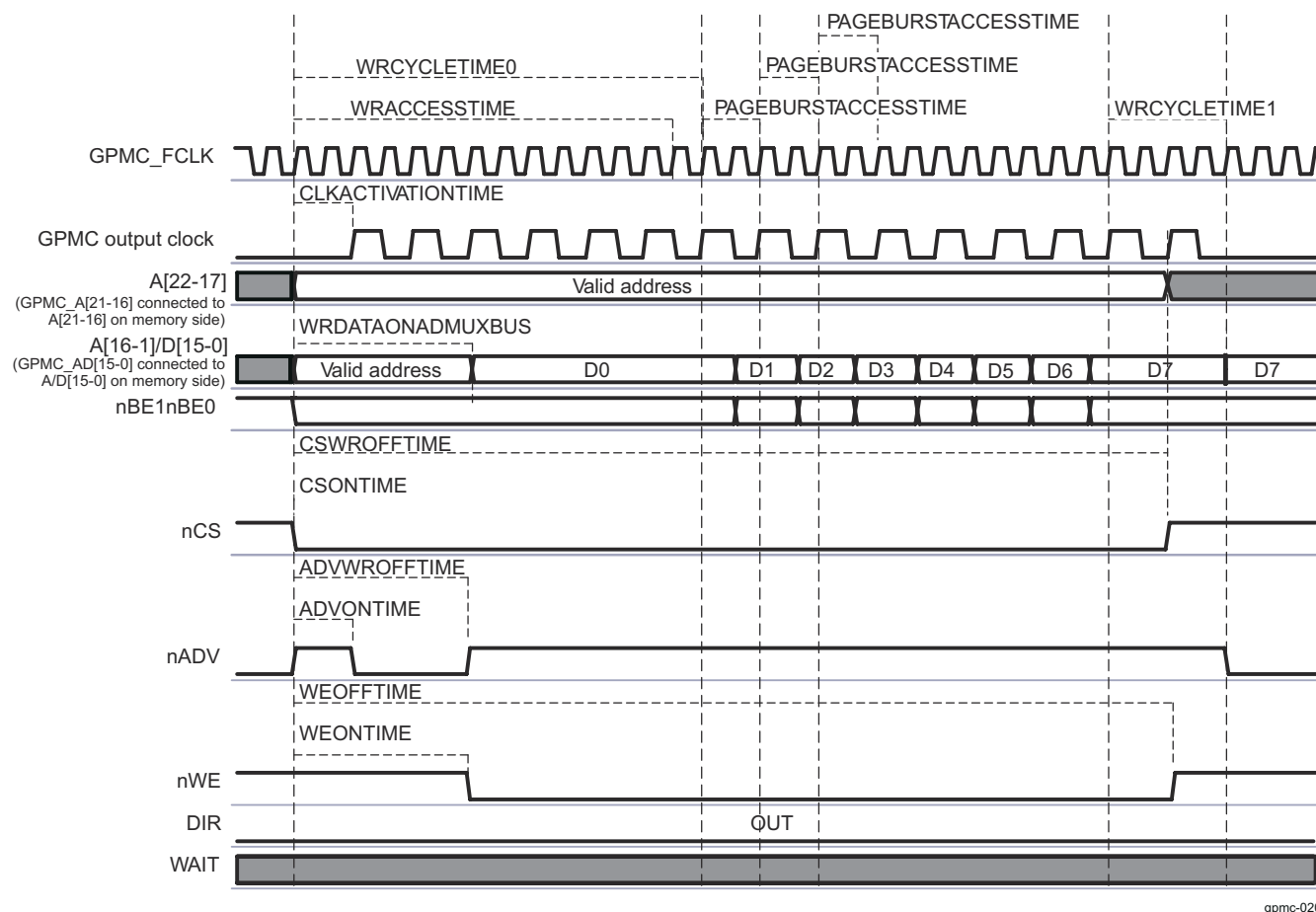
**Figure 12-253. Synchronous Single Write on an Address/Data-Multiplexed Device**

When the GPMC generates a write access to an address/data-multiplexed device, it drives the data bus (with address bits A[16-1]) until the GPMC\_CONFIG6\_i[19-16] WRDATAONADMUXBUS bit field time. The first data of the burst is driven on the address/data bus at WRDATAONADMUXBUS time.

#### 12.3.4.4.9.2.4 Synchronous Multiple (Burst) Write

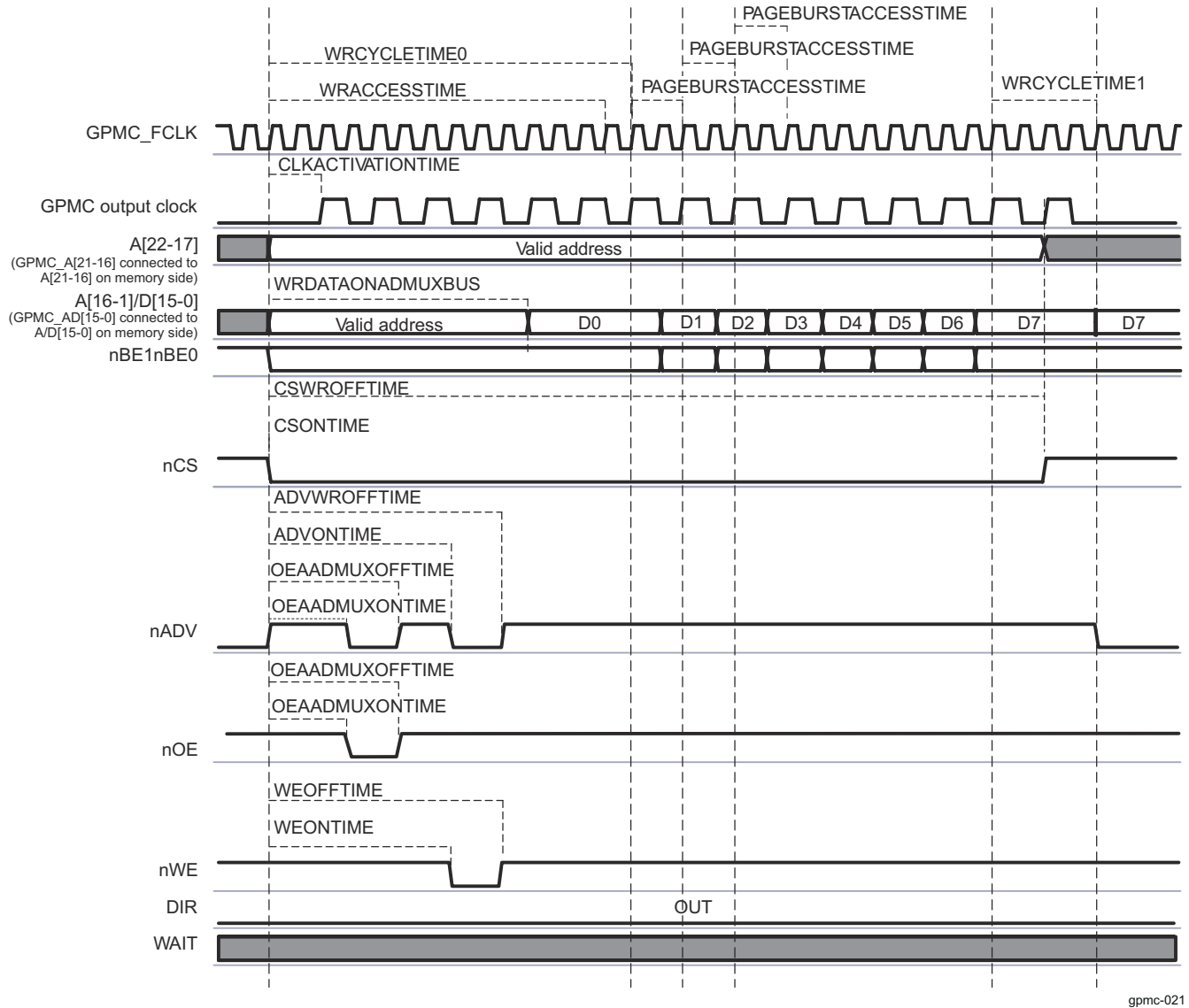
Synchronous burst write mode provides synchronous single or consecutive accesses.

Figure 12-254 shows a synchronous burst write access when the chip-select is configured in address/data-multiplexed mode.



**Figure 12-254. Synchronous Multiple Write (Burst Write) in Address/Data-Multiplexed Mode**

Figure 12-255 shows the same synchronous burst write access when the chip-select is configured in address/address/data-multiplexed (AAD-multiplexed) mode.



**Figure 12-255. Synchronous Multiple Write (Burst Write) in Address/Address/Data-Multiplexed Mode**

The first data of the burst is driven on the A/D bus at the GPMC\_CONFIG6\_i[19-16] WRDATAONADMUXBUS bit field.

When WRACCESTIME completes, control-signal timings are frozen during the multiple data transactions, corresponding to the GPMC\_CONFIG5\_i[27-24] PAGEBURSTACCESTIME bit field multiplied by the number of remaining data transactions.

When the GPMC generates a read access to an address/data-multiplexed device, it drives the address bus until nOE assertion time. For more information, see [Section 12.3.4.4.7.2.3, Address/Data-Multiplexing Interface](#).

- Chip-select signal nCS:
  - nCS assertion time is controlled by the GPMC\_CONFIG2\_i[3-0] CSONTIME bit field (where i = 0 to 3) and ensures address setup time to nCS assertion.
  - nCS deassertion time controlled by the GPMC\_CONFIG2\_i[20-16] CSWROFFTIME bit field and ensures address hold time to nCS deassertion.
- Address valid signal nADV:
  - nADV assertion time is controlled by the GPMC\_CONFIG3\_i[3-0] ADVONTIME bit field.
  - nADV deassertion time is controlled by the GPMC\_CONFIG3\_i[20-16] ADVWROFFTIME bit field.

- Write enable signal nWE:
  - nWE assertion indicates a read cycle.
  - nWE assertion time is controlled by the GPMC\_CONFIG4\_i[19-16] WEONTIME bit field.
  - nWE deassertion time is controlled by the GPMC\_CONFIG4\_i[28-24] WEOFFTIME bit field.

#### Note

The nWE falling edge must not be used to control the time when the burst first data is driven in the address/data bus, because some new devices require the nWE signal to be low during the address phase.

- Direction signal DIR is OUT during the entire access.

When the GPMC generates a write access to an AAD-multiplexed device, all address bits are driven onto the address/data bus in two separate phases. The first phase is used for the MSB address and is qualified with nOE driven low. The second phase for LSB address is qualified with nOE driven high. The address phase ends at nWE assertion time.

The nCS, and DIR signals are controlled as previously described.

- Address valid signal nADV is asserted and deasserted twice during a read transaction:
  - nADV first assertion time is controlled by the GPMC\_CONFIG3\_i[6-4] ADVAADMUXONTIME bit field.
  - nADV first deassertion time is controlled by the GPMC\_CONFIG3\_i[26-24] ADVAADMUXRDOFFTIME bit field.
  - nADV second assertion time is controlled by the GPMC\_CONFIG3\_i[3-0] ADVONTIME bit field.
  - nADV second deassertion time is controlled by the GPMC\_CONFIG3\_i[12-8] ADVRDOFFTIME bit field.
- Output Enable signal nOE is asserted and deasserted twice during a read transaction (nOE second assertion indicates a read cycle):
  - nOE first assertion time is controlled by the GPMC\_CONFIG4\_i[6-4] OEAADMUXONTIME bit field.
  - nOE first deassertion time is controlled by the GPMC\_CONFIG4\_i[15-13] OEAADMUXOFFTIME bit field.
  - nOE second assertion time is controlled by the GPMC\_CONFIG4\_i[3-0] OEONTIME bit field.
  - nOE second deassertion time is controlled by the GPMC\_CONFIG4\_i[12-8] OEOFFTIME bit field.

First write data is driven by the GPMC at GPMC\_CONFIG6\_i[19-16] WRDATAONADMUXBUS, when in address/data-multiplexed configuration. The next write data of the burst is driven on the bus at WRACCESSTIME + 1 during GPMC\_CONFIG5\_i[27-24] PAGEBURSTACCESSTIME GPMC\_FCLK cycles. The last data of the synchronous burst write is driven until GPMC\_CONFIG5\_i[12-8] WRCYCLETIME completes.

- WRACCESSTIME is defined in the GPMC\_CONFIG6\_i[28-24] bit field.
- The PAGEBURSTACCESSTIME parameter must be set accordingly with GPMCFCLKDIVIDER and the memory-device internal configuration.

Total access time GPMC\_CONFIG5\_i[12-8] WRCYCLETIME corresponds to WRACCESSTIME plus the address hold time from nCS deassertion. In [Figure 12-254](#), the programmed value of WRCYCLETIME equals WRCYCLETIME0 + WRCYCLETIME1. WRCYCLETIME0 and WRCYCLETIME1 delays are not actual parameters and are only a graphical representation of the full WRCYCLETIME value.

After a write operation, if no other access (read or write) is pending, the data bus keeps the previous value. See [Section 12.3.4.4.8.10, Bus Keeping Support](#).

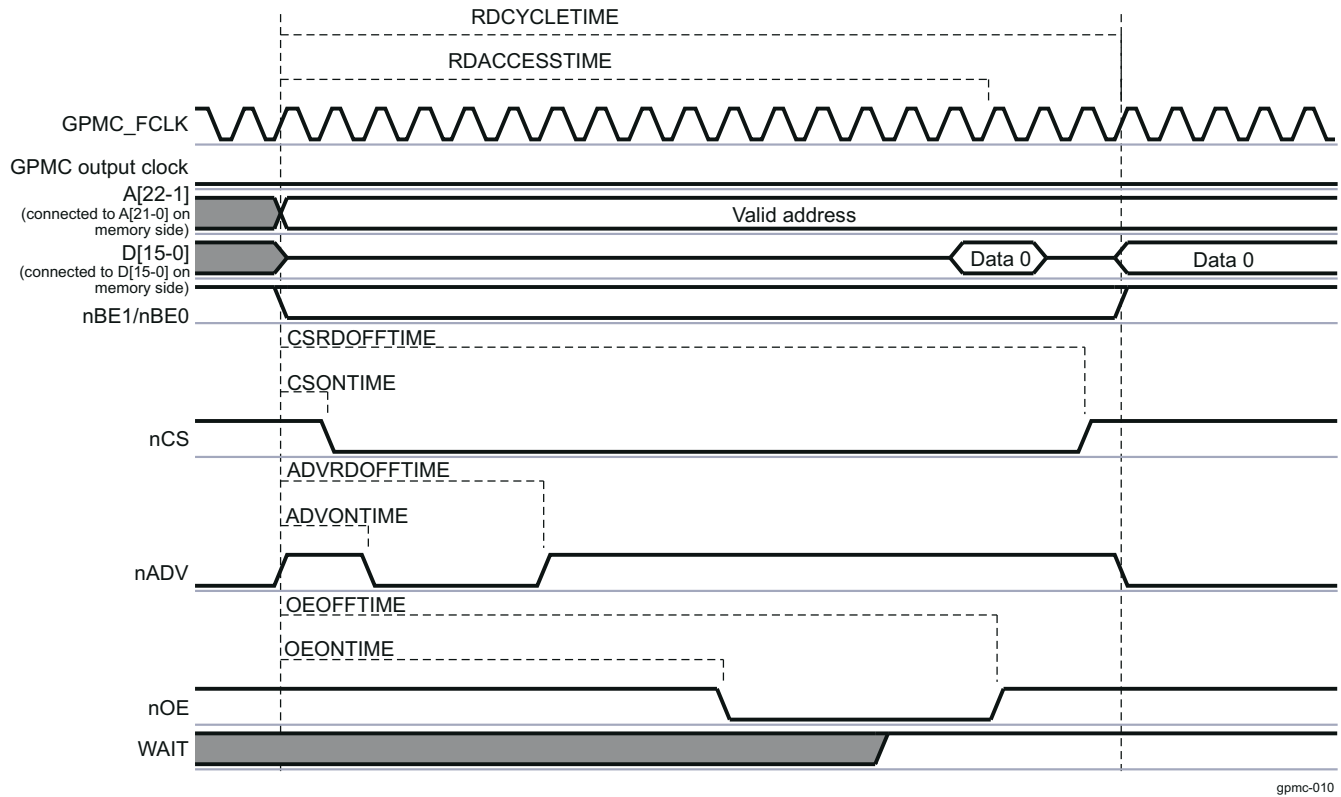
#### 12.3.4.4.9.3 Asynchronous and Synchronous Accesses in non-multiplexed Mode

Page mode is available only in non-multiplexed mode.

- Asynchronous single-read operation on a non-multiplexed device
- Asynchronous single-write operation on a non-multiplexed device
- Asynchronous multiple- (page mode) read operation on a non-multiplexed device
- Synchronous operations on a non-multiplexed device

##### 12.3.4.4.9.3.1 Asynchronous Single-Read Operation on non-multiplexed Device

[Figure 12-256](#) shows an asynchronous single-read operation on a non-multiplexed device.



**Figure 12-256. Asynchronous Single Read on an Address/Data-non-multiplexed Device**

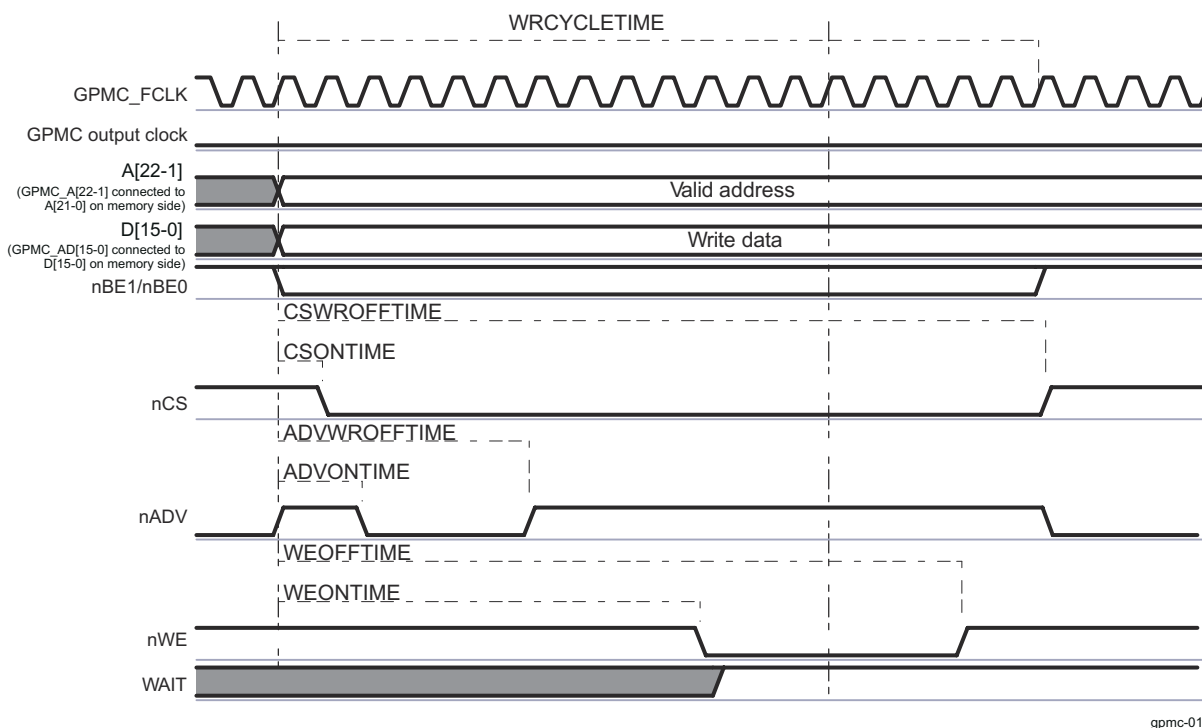
The 22-bit address (For a 16-bit data memory device, hence GPMC A[0] is not necessary to be output) is driven onto the address bus A[22-1] and the 16-bit data is driven onto the data bus D[15-0].

Read data is latched at GPMC\_CONFIG1\_5[20-16] RDACCESSTIME completion time. The end of the access is defined by the GPMC\_CONFIG1\_5[4-0] RDCYCLETIME parameter.

The nCS, nADV, nOE, and DIR signals are controlled in the same way as address/data-multiplexed accesses (see [Table 12-352, NAND Memory Type](#)).

#### 12.3.4.4.9.3.2 Asynchronous Single-Write Operation on non-multiplexed Device

[Figure 12-257](#) shows an asynchronous single-write operation on a non-multiplexed device.

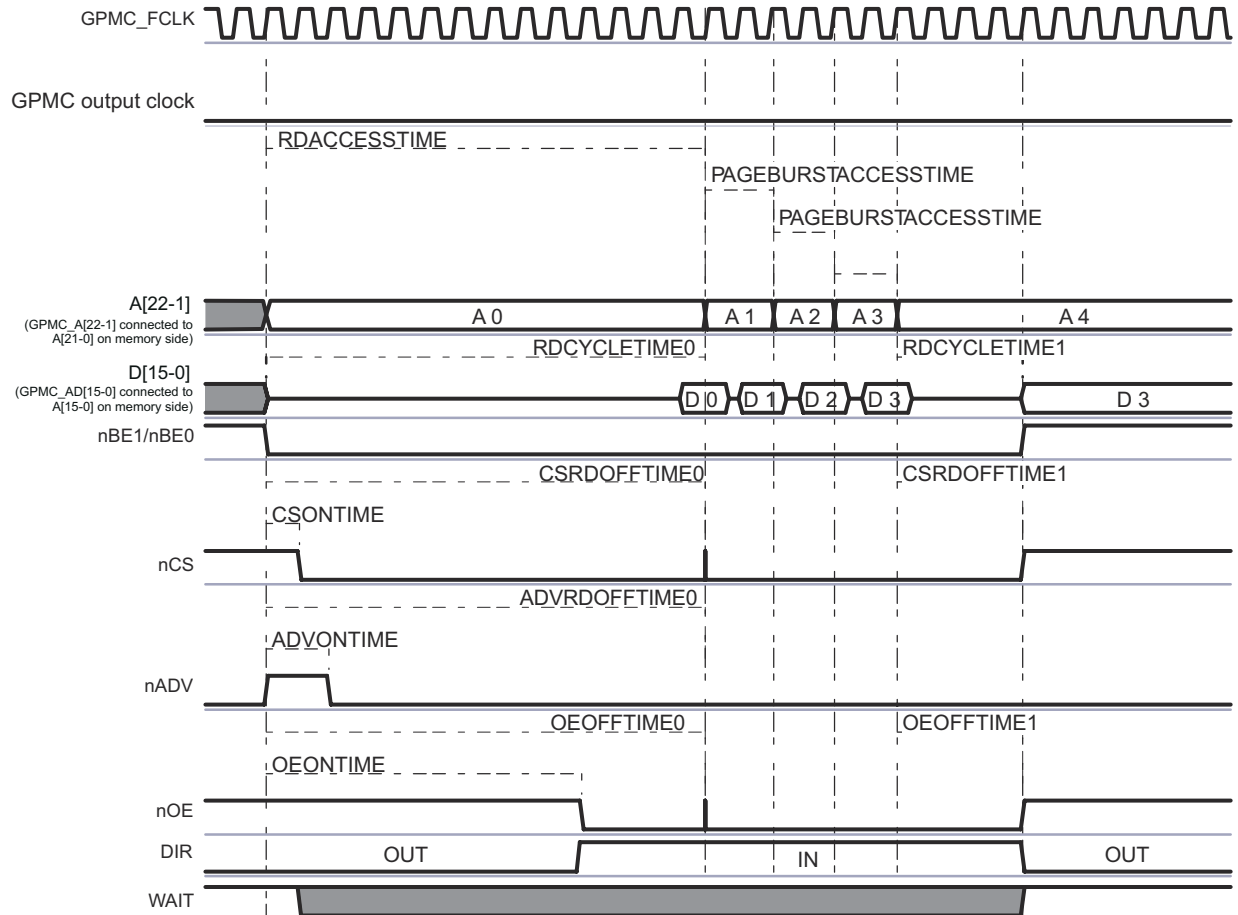


**Figure 12-257. Asynchronous Single Write on an Address/Data-non-multiplexed Device**

The nCS, nADV, nWE, and DIR signals are controlled in the same way as address/data-multiplexed accesses (see [Table 12-352](#)).

#### 12.3.4.4.9.3.3 Asynchronous Multiple (Page Mode) Read Operation on non-multiplexed Device

[Figure 12-258](#) shows an asynchronous multiple-read operation on a non-multiplexed device in which two word32 host read accesses to the GPMC are split into one multiple- (page mode of 4 word16) read access to the attached device.



gpmc-014

**Figure 12-258. Asynchronous Multiple (Page Mode) Read**

### Note

The WAIT signal is active low.

The nCS, nADV, nOE, and DIR signals are controlled in the same way as address/data-multiplexed accesses (see [Table 12-352](#)).

When RDACCESSTIME completes, control signal timings are frozen during the multiple data transactions, corresponding to PAGEBURSTACCESSTIME multiplied by the number of remaining data transactions.

Read data is latched at GPMC\_CONFIG5\_i[20-16] RDACCESSTIME completion time (where i = 0 to 3). The end of the access is defined by the GPMC\_CONFIG5\_i[4-0] RDCYCLETIME parameter.

During consecutive accesses, the GPMC increments the address after each data read completes.

Delay between successive read data in the page is controlled by the GPMC\_CONFIG5\_i[27-24] PAGEBURSTACCESSTIME parameter. Depending on the device page length, the GPMC can control device page crossing during a burst request and insert initial RDACCESSTIME latency. Page crossing is possible only with a new burst access, meaning a new initial access phase is initiated.

Total access time RDCYCLETIME corresponds to RDACCESSTIME, plus the address hold time, starting from the nCS deassertion.

- The read cycle time is defined in the GPMC\_CONFIG5\_i[4-0] RDCYCLETIME bit field.

- In [Figure 12-258](#), the programmed value of RDCYCLETIME equals RDCYCLETIME0 (before paged accesses) + RDCYCLETIME1 (after paged accesses).

#### 12.3.4.4.9.3.4 Synchronous Operations on a non-multiplexed Device

All information for this section is equivalent to similar operations for address/data-multiplexed or AAD-multiplexed accesses. The only difference resides in the address phase. See [Section 12.3.4.5.3](#), *GPMC Configuration in NOR Mode*.

#### 12.3.4.4.9.4 Page and Burst Support

Each chip-select can be configured to process system single or burst requests into successive single accesses or asynchronous page/synchronous burst accesses, with appropriate access size adaptation.

Depending on the external device page or burst capability, read and write accesses can be independently configured through the GPMC. The GPMC\_CONFIG1\_i[30] READMULTIPLE and GPMC\_CONFIG1\_i[28] WRITEMULTIPLE bits (where i = 0 to 3) are associated with the READTYPE and WRITETYPE parameters.

#### Note

- Asynchronous write page mode is not supported.
- 8-bit-wide device support is limited to nonburstable devices (READMULTIPLE and WRITEMULTIPLE are ignored).
- Not applicable to NAND device interfacing.

#### 12.3.4.4.9.5 System Burst vs External Device Burst Support

The device system can issue the following requests to the GPMC:

- Byte, 16-bit word, 32-bit word requests (byte-enable-controlled). This is always a single request from the interconnect point of view.
- Incrementing fixed-length bursts of two, four, and eight words
- Wrapped (critical word access first) fixed-length burst of two, four, or eight words

To process a system request with the optimal protocol, the READMULTIPLE (and READTYPE) and WRITEMULTIPLE (and WRITETYPE) parameters must be set according to the burstable capability (synchronous or asynchronous) of the attached device.

The GPMC access engine issues only fixed-length bursts. The maximum length that can be issued is defined per chip-select by the GPMC\_CONFIG1\_i[24-23] ATTACHEDDEVICEPAGELENGTH bit field (where i = 0 to 3). When the value of ATTACHEDDEVICEPAGELENGTH is less than the length of the system burst request (including the appropriate access size adaptation according to the device width), the GPMC splits the system burst request into multiple bursts. Within the specified 4-, 8-, or 16-word value, the value of the ATTACHEDDEVICEPAGELENGTH bit field must correspond to the maximum length burst supported by the memory device configured in fixed-length burst mode (as opposed to continuous burst mode).

To get optimal performance from memory devices that natively support 16 Word16-length-wrapping burst capability (critical word access first), the ATTACHEDDEVICEPAGELENGTH parameter must be set to 16 words and the GPMC\_CONFIG1\_i[31] WRAPBURST bit (where i = 0 to 3) must be set to 1. Similarly DEVICEPAGELENGTH is set to 4 and 8 for memories supporting 4 and 8 Word16-length-wrapping burst, respectively.

When the memory device does not offer (or is not configured to offer) native 16 Word16-length-wrapping burst, the WRAPBURST parameter must be cleared, and the GPMC access engine emulates the wrapping burst by issuing the appropriate burst sequences according to the value of ATTACHEDDEVICEPAGELENGTH.

When the memory device does not support native-wrapping burst, there is usually no difference in behavior between a fixed-burst length mode and a continuous-burst mode configuration (except for a potential power increase from a memory-speculative data prefetch in a continuous burst read). However, even though continuous burst mode is compatible with GPMC behavior, because the GPMC access engine issues only



fixed-length burst and does not benefit from continuous burst mode, it is best to configure the memory device in fixed-length burst mode.

The memory device maximum-length burst (configured in fixed-length burst wrap or nonwrap mode) usually corresponds to the memory device data buffer size. Memory devices with a minimum of 16 half-word buffers are the most appropriate (especially with wrap support), but memory devices with smaller buffer size (4 or 8) are also supported, assuming that the GPMC\_CONFIG1\_i[24-23] ATTACHEDDEVICEPAGELENGTH bit field is set accordingly to 4 or 8 words.

The device system issues only requests with addresses or starting addresses for nonwrapping burst requests; that is, the request size boundary is aligned. In case of an eight-word-wrapping burst, the wrapping address always occurs on the eight-word boundary. As a consequence, all words requested must be available from the memory data buffer when the buffer size is equal to or greater than the value of ATTACHEDDEVICEPAGELENGTH. This usually means that data can be read from or written to the buffer at a constant rate (number of cycles between data) without wait-states between data accesses. If the memory does not behave this way (nonzero wait-state burstable memory), WAIT pin monitoring must be enabled to dynamically control data access completion within the burst.

#### Note

When the system burst request length is less than the value of ATTACHEDDEVICEPAGELENGTH, the GPMC proceeds with the required accesses.

#### 12.3.4.4.10 GPMC pSRAM Access Specificities

pSRAM devices are SRAM-pin-compatible low-power memories that contain a self-refreshed DRAM memory array. The GPMC\_CONFIG1\_i[11-10] DEVICETYPE bit field (where i = 0 to 3) must be set to 0b00.

The pSRAM device uses the NOR protocol. It supports the following operations:

- Asynchronous single read
- Asynchronous page read
- Asynchronous single write
- Synchronous single read and write
- Synchronous burst read
- Synchronous burst write (not supported by NOR flash memory)

pSRAM devices must be powered up and initialized in a predefined manner according to the specifications of the attached device.

pSRAM devices can be programmed to use either mode: fixed or variable latency. pSRAM devices can automatically schedule autorefresh operations, which force the GPMC to use its WAIT signal capability when read or write operations occur during an internal self-refresh operation, or they can automatically include the autorefresh operation in the access time. These devices do not require additional WAIT signal capability or a minimum nCS high pulse width between consecutive accesses to ensure that the correct internal refresh operation is scheduled.

#### 12.3.4.4.11 GPMC NAND Access Description

NAND (8-bit and 16-bit) memory devices using a standard NAND asynchronous address/data-multiplexing scheme can be supported on any chip-select with the appropriate asynchronous configuration settings.

As for any other type of memory compatible with the GPMC interface, accesses to a chip-select allocated to a NAND device can be interleaved with accesses to chip-selects allocated to other external devices. This interleaved capability limits the system to *chip enable don't care* NAND devices, because the chip-select allocated to the NAND device must be deasserted if accesses to other chip-selects are requested.

### 12.3.4.4.11.1 NAND Memory Device in Byte or 16-bit Word Stream Mode

NAND devices require correct command and address programming before data array read or write accesses. The GPMC does not include specific hardware to translate a random address system request into a NAND-specific multiphase access. In that sense, GPMC NAND support, as opposed to random memory-map device support, is data stream-oriented (byte or 16-bit word).

The GPMC NAND programming model depends on a software driver for address and command formatting with the correct data address pointer value according to the block and page structure. Because of NAND structure and protocol interface diversity, the GPMC does not support automatic command and address phase programming, and software drivers must access the NAND device ID to ensure that correct command and address formatting are used for the identified device.

NAND device data read and write accesses are achieved through an asynchronous read or write access. The associated chip-select signal timing control must be programmed according to the NAND device timing specification.

Any chip-select region can be qualified as a NAND region to constrain the nADV/ALE signal as ALE (ALE active high, default state value at low) during address program access, and the nBE0/CLE signal as CLE (CLE active high, default state value at low) during command program access. GPMC address lines are not used (the previous value is not changed) during NAND access.

#### 12.3.4.4.11.1.1 Chip-Select Configuration for NAND Interfacing in Byte or Word Stream Mode

The GPMC\_CONFIG7\_i register (where i = 0 to 3) associated with a NAND device region interfaced in byte or word stream mode can be initialized with a minimum size of 16MB, because any address location in the chip-select memory region can be used to access a NAND data array. The NAND flash protocol specifies an address sequence where address bits are passed through the data bus in a series of write accesses with the ALE pin asserted. After this address phase, all operations are streamed and the system requests address is irrelevant.

#### CAUTION

To allow correct command, address, and data-access controls, the GPMC\_CONFIG1\_i register associated with a NAND device region must be initialized in asynchronous read and write modes with the parameters listed in [Table 12-322](#). Failure to comply with these settings corrupts the NAND interface protocol.

**Table 12-322. Chip-Select Configuration for NAND Interfacing**

Bit Field	Register	Value	Comments
WRAPBURST	GPMC_CONFIG1_i[31] <sup>(1)</sup>	0	No wrap
READMULTIPLE	GPMC_CONFIG1_i[30]	0	Single access
READTYPE	GPMC_CONFIG1_i[29]	0	Asynchronous mode
WRITEMULTIPLE	GPMC_CONFIG1_i[28]	0	Single access
WRITETYPE	GPMC_CONFIG1_i[27]	0	Asynchronous mode
CLKACTIVATIONTIME	GPMC_CONFIG1_i[26-25]	0b00	
ATTACHEDDEVICEPAGELENGTH	GPMC_CONFIG1_i[24-23]	Don't care	Single-access mode
WAITREADMONITORING	GPMC_CONFIG1_i[22]	0	Wait not monitored by GPMC access engine
WAITWRITEMONITORING	GPMC_CONFIG1_i[21]	0	Wait not monitored by GPMC access engine
WAITMONITORINGTIME	GPMC_CONFIG1_i[19-18]	Don't care	Wait not monitored by GPMC access engine
WAITPINSELECT	GPMC_CONFIG1_i[17-16]		Select which wait is monitored by edge detectors
DEVICESIZE	GPMC_CONFIG1_i[13-12]	0b00 or 0b01	8- or 16-bit interface

**Table 12-322. Chip-Select Configuration for NAND Interfacing (continued)**

Bit Field	Register	Value	Comments
DEVICETYPE	GPMC_CONFIG1_i[11-10]	0b10	NAND device in stream mode
MUXADDDATA	GPMC_CONFIG1_i[9-8]	0b00	non-multiplexed mode
TIMEPARAGRANULARITY	GPMC_CONFIG1_i[4]	0	Timing achieved with best GPMC clock granularity
GPMCFCLKDIVIDER	GPMC_CONFIG1_i[1-0]	Don't care	Asynchronous mode

(1)  $i = 0$  to 3

The GPMC\_CONFIG1\_i to GPMC\_CONFIG4\_i registers (where  $i = 0$  to 3) associated with a NAND device region must be initialized with the correct control-signal timing value according to the NAND device timing parameters.

#### 12.3.4.4.11.1.2 NAND Device Command and Address Phase Control

NAND devices require multiple address programming phases. The software driver must issue the correct number of command and address program accesses, according to the device command set and the device address-mapping scheme.

NAND device-command and address-phase programming is achieved through write requests to the GPMC\_NAND\_COMMAND\_i and GPMC\_NAND\_ADDRESS\_i register locations (where  $i = 0$  to 3) with the correct command and address values. These locations are mapped in the associated chip-select register region. The associated chip-select signal timing control must be programmed according to the NAND device timing specification.

Command and address values are not latched during the access and cannot be read back at the register location.

- Only write accesses must be issued to these locations, but the GPMC does not discard any read access. Accessing a NAND device with nOE and CLE or ALE asserted (read access) can produce undefined results.
- Write accesses to the GPMC\_NAND\_COMMAND\_i and GPMC\_NAND\_ADDRESS\_i register locations must be posted for faster operations (where  $i = 0$  to 3). The GPMC\_CONFIG[0] NANDFORCEPOSTEDWRITE bit enables write accesses to these locations as posted, even if they are defined as nonposted.

A write buffer is used to store write transaction information before the external device is accessed:

- Up to eight consecutive posted write accesses can be accepted and stored in the write buffer.
- For nonposted write, the pipeline is one deep.
- An GPMC\_STATUS[0] EMPTYWRITEBUFFERSTATUS bit stores the empty status of the write buffer.

The GPMC\_NAND\_COMMAND\_i and GPMC\_NAND\_ADDRESS\_i registers (where  $i = 0$  to 3) are 32-bit word locations, which means any 32- or 16-bit word access is split into 4- or 2-byte accesses if an 8-bit-wide NAND device is attached. For multiple-command phase or multiple-address phase, the software driver can use 32- or 16-bit word access to these registers, but it must consider the splitting and little-endian ordering scheme. When only one byte command or address phase is required, only byte write access to the GPMC\_NAND\_COMMAND\_i and GPMC\_NAND\_ADDRESS\_i registers can be used, and any of the four byte locations of the registers is valid.

The same applies to a GPMC\_NAND\_COMMAND\_i and a GPMC\_NAND\_ADDRESS\_i (where  $i = 0$  to 3) 32-bit word write access to a 16-bit-wide NAND device (split into two 16-bit word accesses). In the case of a 16-bit word write access, the MSByte of the 16-bit word value must be set according to the NAND device requirement (usually 0). Either 16-bit word location or any one of the four byte locations of the registers is valid.

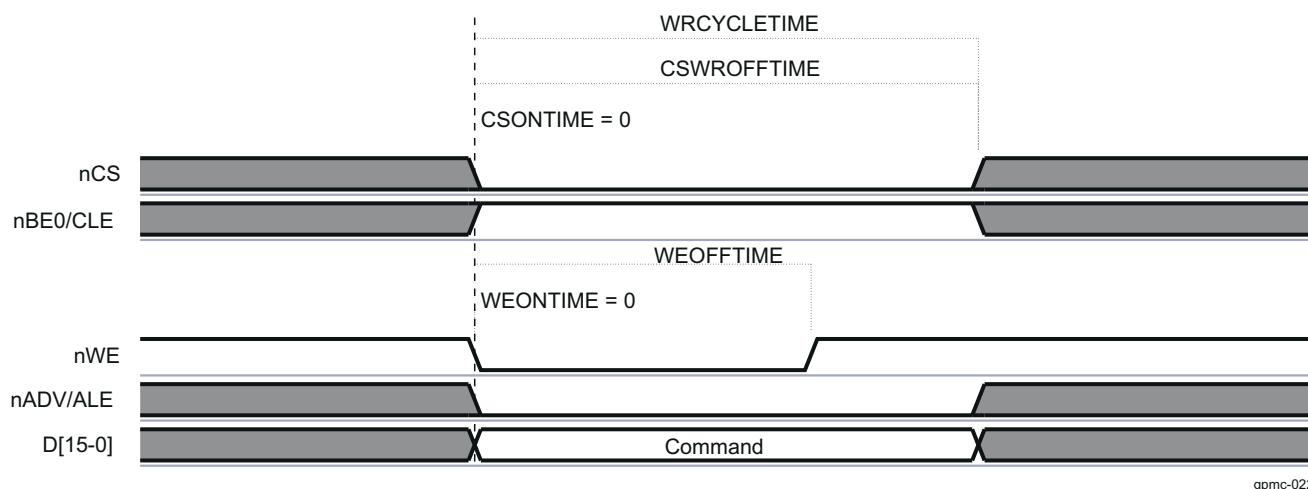
#### 12.3.4.4.11.1.3 Command Latch Cycle

Writing data at the GPMC\_NAND\_COMMAND\_i location (where  $i = 0$  to 3) places the data as the NAND command value on the bus, using a regular asynchronous write access.

- nCE is controlled by the CSONTIME and CSWROFFTIME timing parameters.
- CLE is controlled by the ADVONTIME and ADVWROFFTIME timing parameters.

- nWE is controlled by the WEONTIME and WEOFFTIME timing parameters.
- ALE and nRE (nOE) are maintained inactive.

Figure 12-259 shows the NAND command latch cycle.



**Figure 12-259. NAND Command Latch Cycle**

#### Note

CLE is shared with the nBE0 output signal and has an inverted polarity from BE0. The NAND qualifier deals with this. During the asynchronous NAND data access cycle, nBE0 (also nBE1) must not toggle, because it is shared with CLE.

NAND flash memories do not use byte-enable signals.

#### 12.3.4.4.11.1.4 Address Latch Cycle

Writing data at the GPMC\_NAND\_ADDRESS\_i location (where i = 0 to 3) places the data as the NAND partial address value on the bus, using a regular asynchronous write access.

- nCS is controlled by the CSONTIME and CSWROFFTIME timing parameters.
- ALE is controlled by the ADVONTIME and ADVWROFFTIME timing parameters.
- nWE is controlled by the WEONTIME and WEOFFTIME timing parameters.
- CLE and nRE (nOE) are maintained inactive.

Figure 12-260 shows the NAND address latch cycle.

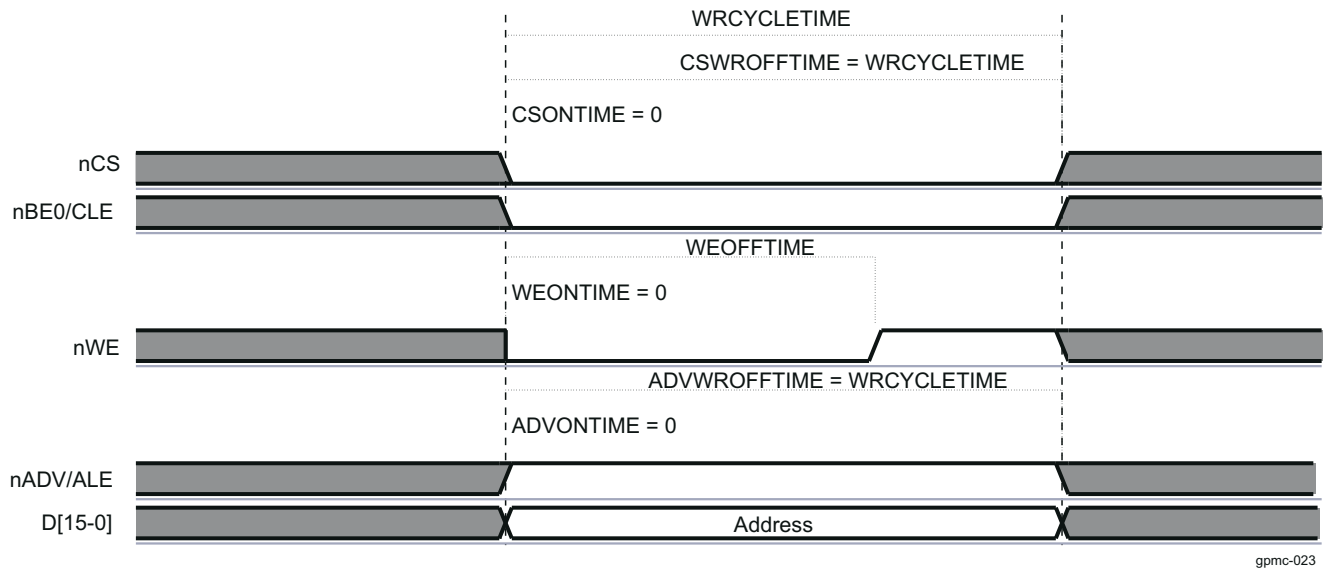


Figure 12-260. NAND Address Latch Cycle

**Note**

ALE is shared with the nADV output signal and has an inverted polarity from ADV. The NAND qualifier deals with this. During the asynchronous NAND data access cycle, ALE is kept stable.

**12.3.4.4.11.1.5 NAND Device Data Read and Write Phase Control in Stream Mode**

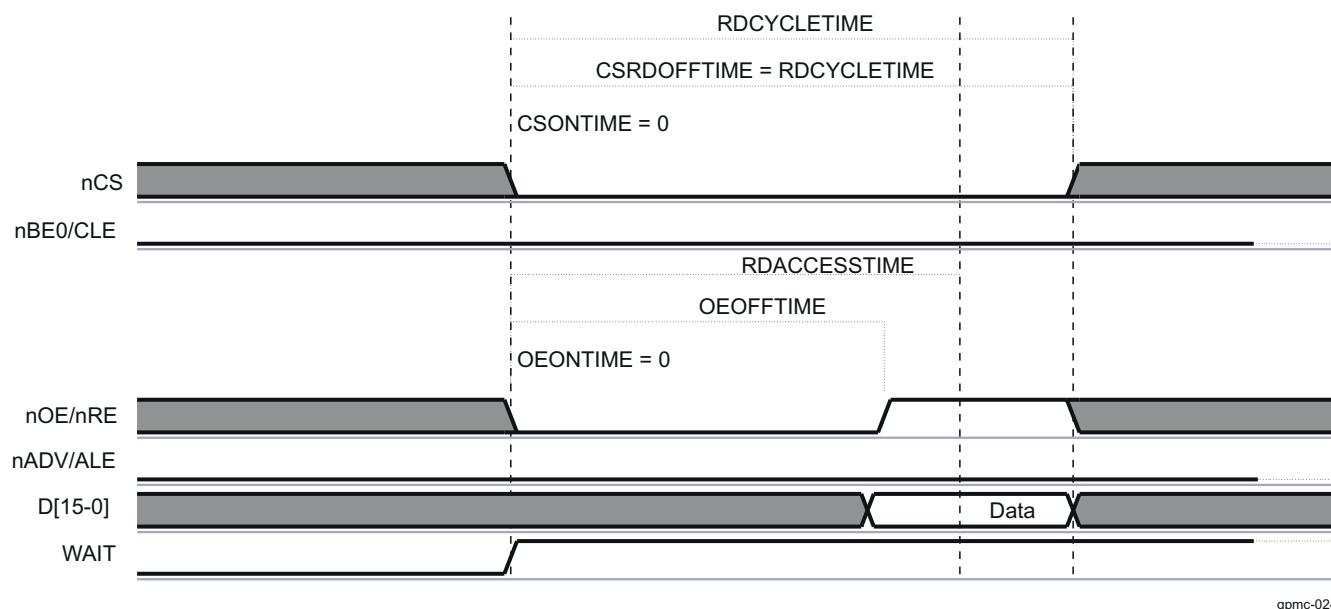
NAND device data read and write accesses are achieved through a read or write request to the chip-select-associated memory region at any address location in the region or through a read or write request to the GPMC\_NAND\_DATA\_i location (where i = 0 to 3) mapped in the chip-select-associated control register region. GPMC\_NAND\_DATA\_i is not a true register, but an address location to enable nRE or nWE signal control. The associated chip-select signal timing control must be programmed according to the NAND device timing specification.

Reading data from the GPMC\_NAND\_DATA\_i location or from any location in the associated chip-select memory region activates an asynchronous read access.

- nCS is controlled by the CSONTIME and CSRDOFFTIME timing parameters.
- nRE is controlled by the OEONTIME and OEOFFTIME timing parameters.
- To take advantage of nRE high-to-data invalid minimum timing value, RDACCESSTIME can be set so that data are effectively captured after nRE deassertion. This allows optimization of NAND read access cycle time completion. For optimal timing parameter settings, see the NAND device and the device timing parameters.

ALE, CLE, and nWE are maintained inactive.

Figure 12-261 shows the NAND data read cycle.

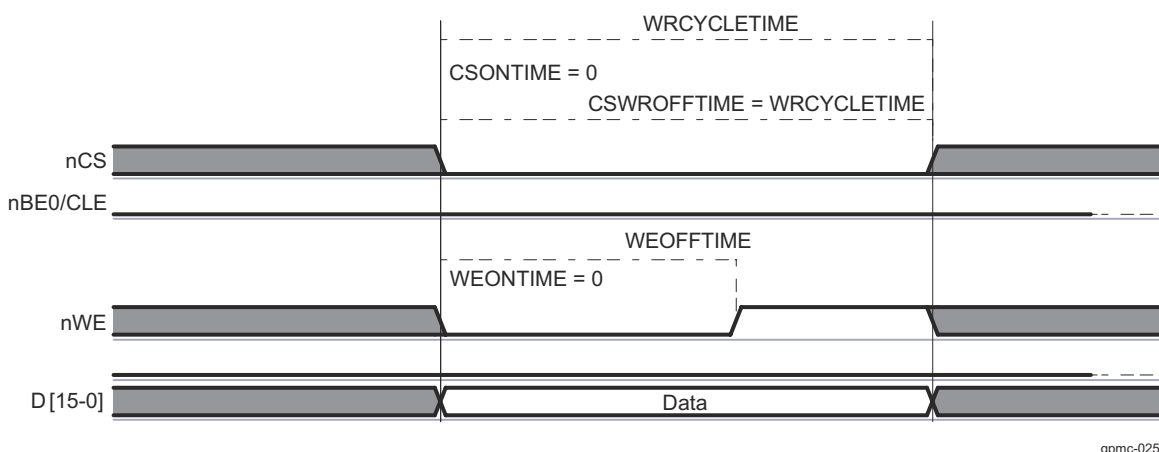


**Figure 12-261. NAND Data Read Cycle**

Writing data to the GPMC\_NAND\_DATA\_i location or to any location in the associated chip-select memory region activates an asynchronous write access.

- nCS is controlled by the CSONTIME and CSWROFFTIME timing parameters.
- nWE is controlled by the WEONTIME and WEOFFTIME timing parameters.
- ALE, CLE, and nRE (nOE) are maintained inactive.

Figure 12-262 shows the NAND data write cycle.



**Figure 12-262. NAND Data Write Cycle**

#### 12.3.4.4.11.6 NAND Device General Chip-Select Timing Control Requirement

For most NAND devices, read data access time is dominated by nCS-to-data-valid timing and has faster nRE-to-data-valid timing. Successive accesses with nCS deassertions between accesses are affected by this timing constraint. Because accesses to a NAND device can be interleaved with other chip-select accesses, there is no certainty that nCS always stays low between two accesses to the same chip-select. Moreover, an nCS deassertion time between the same chip-select NAND accesses is likely to be required as follows: the nCS deassertion requires programming CYCLETIME and RDACCESSTIME according to the nCS-to-data-valid critical timing.

To get full performance from NAND read and write accesses, the prefetch engine can dynamically reduce the following on back-to-back NAND accesses (to the same memory) and suppress the minimum nCS high pulse width between accesses:

- RDCYCLETIME
- WRCYCLETIME
- RDACCESSTIME
- WRACCESSTIME
- CSRDOFFTIME
- CSWROFFTIME
- ADVRDOFFTIME
- ADVWROFFTIME
- OEOFFTIME
- WEOFFTIME

For more information about optimal prefetch engine access, see [Section 12.3.4.4.11.4, Prefetch and Write-Posting Engine](#).

Some NAND devices require minimum write-to-read idle time, especially for device-status read accesses following status-read command programming (write access). If such write-to-read transactions are used, a minimum nCS high pulse width must be set. For this, CYCLE2CYCLESAMECSN and CYCLE2CYCLEDELAY must be set according to the appropriate timing requirement to prevent any timing violation.

NAND devices usually have an important nRE high-to-data bus in three-state mode. This requires a bus turnaround setting (BUSTURNAROUND = 1) so that the next access to a different chip-select is delayed until the BUSTURNAROUND delay completes. Back-to-back NAND read accesses to the same NAND flash are not affected by the programmed bus turnaround delay.

#### **12.3.4.4.11.1.7 Read and Write Access Size Adaptation**

##### **12.3.4.4.11.1.7.1 8-Bit-Wide NAND Device**

Host 16- and 32-bit word read and write access requests to a chip-select associated with an 8-bit-wide NAND device are split into successive read and write byte accesses to the NAND memory device. Byte access is ordered according to little-endian organization. A NAND 8-bit-wide device must be interfaced on the D0D7 interface bus lane. GPMC data accesses are justified on this bus lane when the cs is associated with an 8-bit-wide NAND device.

##### **12.3.4.4.11.1.7.2 16-Bit-Wide NAND Device**

Host 32-bit word read and write access requests to a chip-select associated with a 16-bit-wide NAND device are split into successive read and write 16-bit word accesses to the NAND memory device. 16-bit word access is ordered according to little-endian organization.

Host byte read and write access requests to a 16-bit-wide NAND device are completed as 16-bit accesses on the device itself, because there is no byte-addressing capability on 16-bit-wide NAND devices. This means that the NAND device address pointer is incremented on a 16-bit word basis and not on a byte basis. For a read access, only the requested byte is given back to the host, but the remaining byte is not stored or saved by the GPMC, and the next byte or 16-bit word read access gets the next 16-bit word NAND location. For a write access, the invalid byte part of the 16-bit word is driven to FF, and the next byte or 16-bit word write access programs the next 16-bit word NAND location.

Generally, byte access to a 16-bit-wide NAND device must be avoided, especially when ECC calculation is enabled. 8- or 16-bit ECC-based computations are corrupted by a byte read to a 16-bit-wide NAND device, because the nonrequested byte is considered invalid on a read access (not captured on the external data bus; FF is fed to the ECC engine) and is set to FF on a write access.

Host requests (read/write) issued in the chip-select memory region are translated in successive single or split accesses (read/write) to the attached device. Therefore, incrementing 32-bit burst requests are translated in multiple 32-bit sequential accesses following the access adaptation of the 32-bit to 8- or 16-bit device.



#### 12.3.4.4.11.2 NAND Device-Ready Pin

The NAND memory device provides a ready pin to indicate data availability after a block/page opening and to indicate that data programming is complete. The ready pin can be connected to one of the wait GPMC input pins; data read accesses must not be tried when the ready pin is sampled inactive (device is not ready) even if the associated chip-select WAITREADMONITORING bit field is set. The duration of the NAND device busy state after the block/page opening is so long (up to 50 micro second) that accesses occurring when the ready pin is sampled inactive can stall GPMC access and eventually cause a system time-out.

#### Note

If a read access to a NAND flash is done using WAIT monitoring mode, the device is blocked during a page opening, and so is the GPMC. If the correct settings are used, other chip-selects can be used while the memory processes the page opening command.

To avoid a time-out caused by a block/page opening delay in NAND flash, disable the WAIT pin monitoring for read and write accesses (that is, set the GPMC\_CONFIG1\_i[21] WAITWRITEMONITORING and GPMC\_CONFIG1\_i[22] WAITREADMONITORING bits to 0, where i = 0 to 3), and use one of the following methods instead:

- Use software to poll the WAITxSTATUS bit (where x = 0 to 1) of the GPMC\_STATUS.
- Configure an interrupt that is generated on the WAIT signal change (through the GPMC\_IRQENABLE register bits[11-8]).

Even if the READWAITMONITORING bit is not set, the external memory nR/B pin status is captured in the programmed wait bit in the GPMC\_STATUS register.

The READWAITMONITORING bit method must be used for other memories than NAND flash, if they require the use of a WAIT signal.

##### 12.3.4.4.11.2.1 Ready Pin Monitored by Software Polling

The ready signal state can be monitored through the GPMC\_STATUS[9-8] WAITxSTATUS bit (where x = 0 to 1). Software must monitor the ready pin only when the signal is declared valid. Refer to the NAND device timing parameters to set the correct software temporization to monitor ready only after the invalid window is complete from the last read command written to the NAND device.

##### 12.3.4.4.11.2.2 Ready Pin Monitored by Hardware Interrupt

Each GPMC\_WAIT input pin can generate an interrupt when a wait-to-no-wait transition is detected. Depending on whether the GPMC\_CONFIG[9-8] WAITxPINPOLARITY bits (where x = 0 to 1) is active low or active high, the wait-to-no-wait transition is a low-to-high external WAIT signal transition or a high-to-low external WAIT signal transition, respectively.

The wait transition pin detector must be cleared before any transition detection. This is done by writing 1 to the WAITxEDGEDETECTIONSTATUS bit (where x = 0 to 1) of the GPMC\_IRQSTATUS register according to the GPMC\_WAIT pin used for the NAND device-ready signal monitoring. To detect a wait-to-no-wait transition, the transition detector requires a wait active time detection of a minimum of two GPMC\_FCLK cycles. Software must incorporate precautions to clear the wait transition pin detector before wait (busy) time completes.

A wait-to-no-wait transition detection can issue a GPMC interrupt if the WAITxEDGEDETECTIONENABLE bit in the GPMC\_IRQENABLE register is set and if the WAITxEDGEDETECTIONSTATUS bit field in the GPMC\_IRQSTATUS register is set.

The WAITMONITORINGTIME bit field does not affect wait-to-no-wait transition time detection.

It is also possible to poll the WAITxEDGEDETECTIONSTATUS bit field in the GPMC\_IRQSTATUS register according to the GPMC\_WAIT pin used for NAND device ready signal monitoring.



#### 12.3.4.4.11.3 ECC Calculator

The GPMC includes an error code correction (ECC) calculator circuitry that enables ECC calculation on the fly during data read or data program (that is, write) operations. The page size supported by the ECC calculator in one calculation/context is 512 bytes.

The user can choose from two different algorithms with different error correction capabilities through the GPMC\_ECC\_CONFIG[16] ECCALGORITHM bit:

1. Hamming code for 1-bit error code correction on 8- or 16-bit NAND flash organized with page size greater than 512 bytes
2. Bose-Chaudhuri-Hocquenghem (BCH) code for 4- to 16-bit error correction

The GPMC does not handle the error code correction directly. During writes, the GPMC computes parity bits. During reads, the GPMC provides enough information for the processor to correct errors without reading the data buffer all over again.

The Hamming code ECC is based on a 2-dimensional (2D) (row and column) bit parity accumulation. This parity accumulation is accomplished on the programmed number of bytes or 16-bit words read from the memory device, or is written to the memory device in stream mode.

Because the ECC engine includes only one accumulation context, it can be allocated to only one chip-select at a time through the GPMC\_ECC\_CONFIG[3-1] ECCCS bit field. Even if two chip-selects use different ECC algorithms, one the Hamming code and the other a BCH code, they must define separate ECC contexts because some of the ECC registers are common to all types of algorithms.

##### 12.3.4.4.11.3.1 Hamming Code

All references to ECC in this subsection refer to the 1-bit error correction Hamming code.

The ECC is based on a 2D (row and column) bit parity accumulation known as the Hamming code. The parity accumulation is done for a programmed number of bytes or 16-bit word read from the memory device or written to the memory device in stream mode.

There is no automatic error detection or correction, and the software NAND driver must read the multiple ECC calculation results, compare them to the expected code value, and take the appropriate corrective actions according to the error handling strategy (ECC storage in spare byte, error correction on read, block invalidation).

The ECC engine includes a single accumulation context. It can be allocated to a single designated chip-select at a time, and parallel computations on different chip-selects are not possible. Because it is allocated to a single chip-select, the ECC computation is not affected by interleaved GPMC accesses to other chip-selects and devices. The ECC accumulation is sequentially processed in the order of data read from or written to the memory on the designated chip-select. The ECC engine does not differentiate read accesses from write accesses and does not differentiate data from command or status information. Software must ensure that only relevant data are passed to the NAND flash memory while the ECC computation engine is active.

The starting NAND page location must be programmed first, followed by an ECC accumulation context reset with an ECC enabling, if required. The NAND device accesses discussed in the following sections must be limited to data read or write until the specified number of ECC calculations is complete.

##### 12.3.4.4.11.3.1.1 ECC Result Register and ECC Computation Accumulation Size

The GPMC includes up to nine ECC result registers (GPMC\_ECCj\_RESULT, where j = 1 to 9) to store ECC computation results when the specified number of bytes or 16-bit words has been computed.

The ECC result registers are used sequentially: one ECC result is stored in one ECC result register on the list, the next ECC result is stored in the next ECC result register on the list, and so forth, until the last ECC computation. The value of the GPMC\_ECCj\_RESULT register is valid only when the programmed number of bytes or 16-bit words has been accumulated, which means that the same number of bytes or 16-bit words has been read from or written to the NAND device in sequence.

The GPMC\_ECC\_CONTROL[3-0] ECCPOINTER bit field must be set to the correct value to select the ECC result register to be used first in the list for the incoming ECC computation process. The ECCPointer can be read to determine which ECC register is used in the next ECC result storage for the ongoing ECC computation. The value of the GPMC\_ECCj\_RESULT register (where j = 1 to 9) can be considered valid when ECCPOINTER equals j + 1. When the GPMC\_ECCj\_RESULT register (where j = 9) is updated, ECCPOINTER is frozen at 10, and ECC computing is stopped (ECCENABLE = 0).

The ECC accumulator must be reset before any ECC computation accumulation process. The GPMC\_ECC\_CONTROL[8] ECCCLEAR bit must be set to 1 (nonpersistent bit) to clear the accumulator and all ECC result registers.

For each ECC result (each GPMC\_ECCj\_RESULT register, where j = 1 to 9), the number of bytes or 16-bit words used for ECC computing accumulation can be selected from between two programmable values.

The ECCjRESULTSIZe bits (where j = 1 to 9) in the GPMC\_ECC\_SIZE\_CONFIG register select which programmable size value (ECCSIZE0 or ECCSIZE1) must be used for this ECC result (stored in the GPMC\_ECCj\_RESULT register).

The ECCSIZE0 and ECCSIZE1 bit fields allow selection of the number of bytes or 16-bit words used for ECC computation accumulation. Any even values from 2 to 512 are allowed.

Flexibility in the number of ECCs computed and the number of bytes or 16-bit words used in the successive ECC computations enables different NAND page error-correction strategies. Usually based on 256 or 512 bytes and on 128 or 256 16-bit word, the number of ECC results required is a function of the NAND device page size. Specific ECC accumulation size can be used when computing the ECC on the NAND spare byte.

For example, with a 2-KB data page, 8-bit-wide NAND device, eight ECCs accumulated on 256 bytes can be computed and added to one extra ECC computed on the 24 spare bytes area where the eight ECC results used for comparison and correction with the computed data page ECC are stored. The GPMC then provides nine GPMC\_ECCj\_RESULT registers (j = 1 to 9) to store the results. In this case, ECCSIZE0 is set to 256, and ECCSIZE1 is set to 24; the ECC[1-8]RESULTSIZe bits are set to 0, and the ECC9RESULTSIZe bit is set to 1.

#### 12.3.4.4.11.3.1.2 ECC Enabling

The GPMC\_ECC\_CONFIG[3-1] ECCCS bit field selects the allocated chip-select. The GPMC\_ECC\_CONFIG[0] ECCENABLE bit enables ECC computation on the next detected read or write access to the selected chip-select.

The following fields must not be changed or cleared while an ECC computation is in progress:

- ECCPOINTER
- ECCCLEAR
- ECCSIZE
- ECCjRESULTSIZe (where j = 1 to 9)
- ECC16B
- ECCCS

The ECC accumulator and ECC result register must not be changed or cleared while an ECC computation is in progress.

[Table 12-323](#) describes the ECC enable settings.

**Table 12-323. ECC Enable Settings**

Bit Field	Register	Value	Comments
ECCCS	GPMC_ECC_CONFIG	0–3	Selects the chip-select where ECC is computed
ECC16B	GPMC_ECC_CONFIG	0/1	Selects column number for ECC calculation
ECCCLEAR	GPMC_ECC_CONTROL	0–7	Clears all ECC result registers
ECCPOINTER	GPMC_ECC_CONTROL	0–7	A write to this bit field selects the ECC result register where the first ECC computation is stored. Set to 1 by default.

**Table 12-323. ECC Enable Settings (continued)**

Bit Field	Register	Value	Comments
ECSSIZE1	GPMC_ECC_SIZE_CONFIG	0x00–0xFF	Defines ECSSIZE1
ECSSIZE0	GPMC_ECC_SIZE_CONFIG	0x00–0xFF	Defines ECSSIZE0
ECCjRESULTSIZE (j from 1 to 9)	GPMC_ECC_SIZE_CONFIG	0/1	Selects the size of ECCn result register
ECCENABLE	GPMC_ECC_CONFIG	1	Enables the ECC computation

#### 12.3.4.4.11.3.1.3 ECC Computation

The ECC algorithm is a multiple parity bit accumulation computed on the odd and even bit streams extracted from the byte or Word16 streams. The parity accumulation is split into row and column accumulations, as shown in Figure 12-263 and Figure 12-264. The intermediate row and column parities are used to compute the upper level row and column parities. Only the final computation of each parity bit is used for ECC comparison and correction.

P1o = bit7 XOR bit5 XOR bit3 XOR bit1 on each byte of the data stream

P1e = bit6 XOR bit4 XOR bit2 XOR bit0 on each byte of the data stream

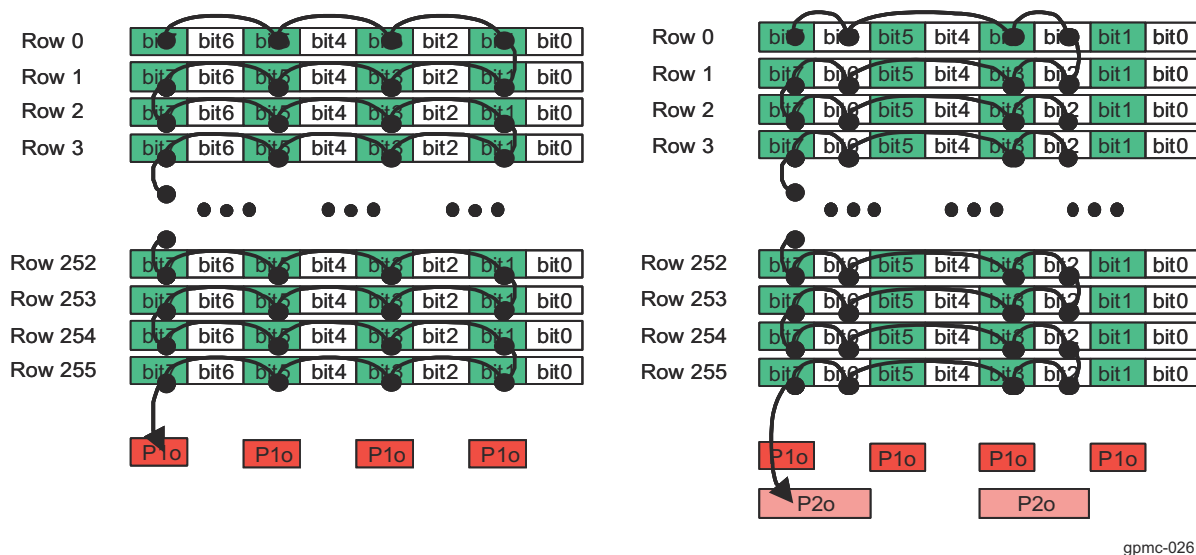
P2o = bit7 XOR bit6 XOR bit3 XOR bit2 on each byte of the data stream

P2e = bit5 XOR bit4 XOR bit1 XOR bit0 on each byte of the data stream

P4o = bit7 XOR bit6 XOR bit5 XOR bit4 on each byte of the data stream

P4e = bit3 XOR bit2 XOR bit1 XOR bit0 on each byte of the data stream

Each column parity bit is XORed with the previous accumulated value.


**Figure 12-263. Hamming Code Accumulation Algorithm (1/2)**

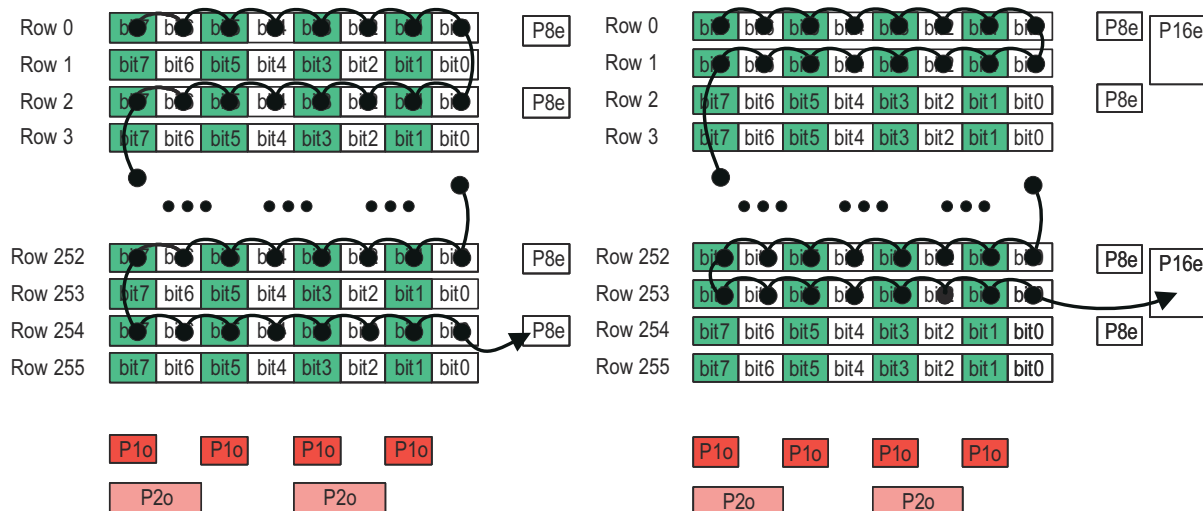
For line parities, the bits of each new data are XORed together, and line parity bits are computed as described below:

P8e = row0 XOR row2 XOR row4 XOR ... XOR row254

P8o = row1 XOR row3 XOR row5 XOR ... XOR row255

P16e = row0 XOR row1 XOR row4 XOR row5 XOR ... XOR row252 XOR row 253

P16o = row2 XOR row3 XOR row6 XOR row7 XOR ... XOR row254 XOR row 255

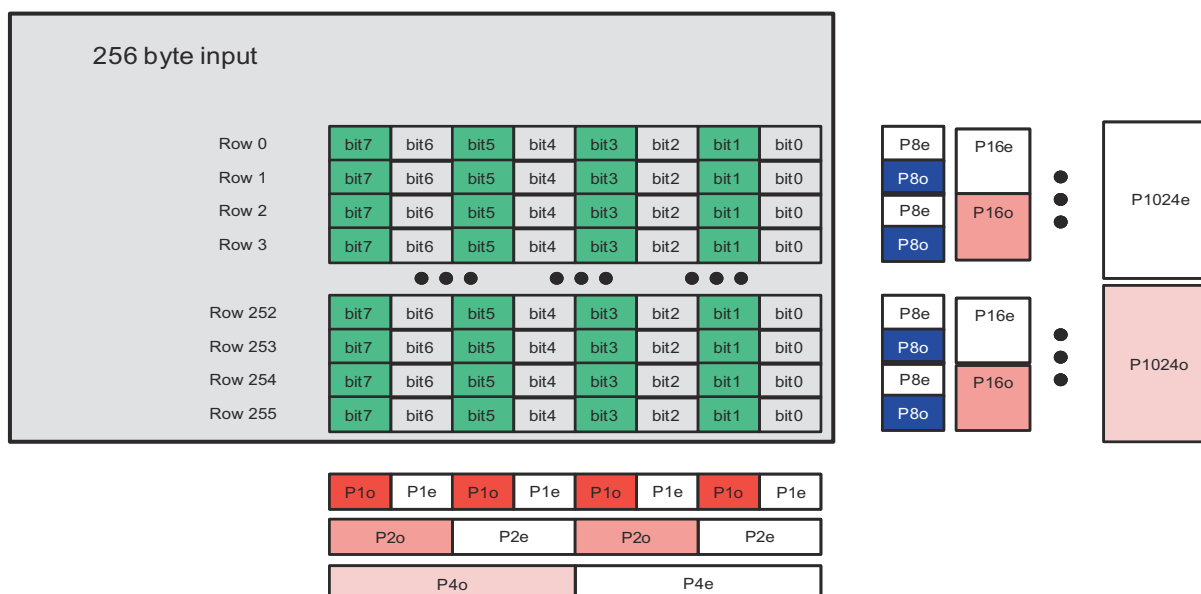


gpmc-027

**Figure 12-264. Hamming Code Accumulation Algorithm (2/2)**

Unused parity bits in the result registers are set to 0.

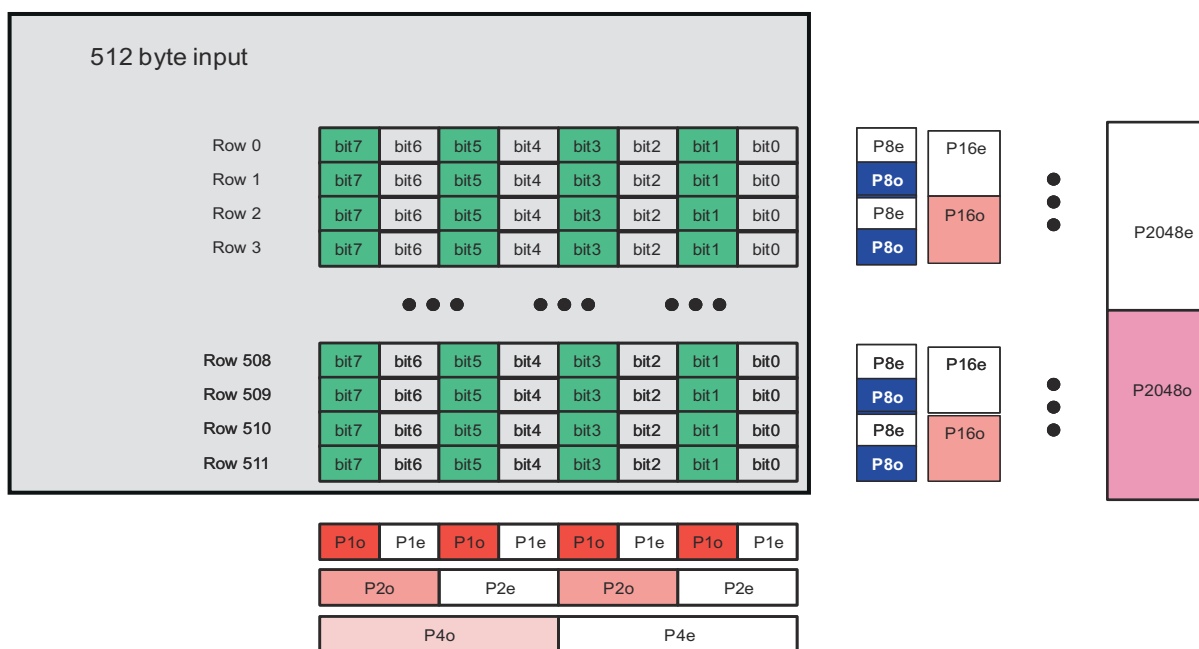
Figure 12-265 shows ECC computation for a 256-byte data stream (read or write). The result includes six column parity bits (P1o-P2o-P4o for odd parities, and P1e-P2e-P4e for even parities) and sixteen row parity bits (P8o-P16o-P32o--P1024o for odd parities, and P8e-P16e-P32e--P1024e for even parities).



gpmc-028

**Figure 12-265. ECC Computation for a 256-Byte Data Stream (Read or Write)**

Figure 12-266 shows ECC computation for a 512-byte data stream (read or write). The result includes six column parity bits (P1o-P2o-P4o for odd parities, and P1e-P2e-P4e for even parities) and eighteen row parity bits (P8o-P16o-P32o--P1024o- - P2048o for odd parities, and P8e-P16e-P32e--P1024e- P2048e for even parities).



gpmc-029

**Figure 12-266. ECC Computation for a 512-Byte Data Stream (Read or Write)**

For a 2-KB page, four 512 bytes ECC calculations plus 1 for the spare area are required. Results are stored in the GPMC\_ECCj\_RESULT registers (where j = 1 to 9).

#### 12.3.4.4.11.3.1.4 ECC Comparison and Correction

To detect an error, the computed ECC result must be XORed with the parity value stored in the spare area of the accessed page.

- If the result of this logical XOR is all 0s, no error is detected and the read data is correct.
- If every second bit in the parity result is a 1, 1 bit is corrupted and is located at bit address (P2048o, P1024o, P512o, P256o, P128o, P64o, P32o, P16o, P8o, P4o, P2o, P1o). Software must correct the corresponding bit.
- If only 1 bit in the parity result is 1, it is an ECC error and the read data is correct.

#### 12.3.4.4.11.3.1.5 ECC Calculation Based on 8-Bit Word

The 8-bit-based ECC computation is used for 8-bit-wide NAND device interfacing.

The 8-bit-based ECC computation can be used for 16-bit-wide NAND device interfacing to get backward compatibility on the error-handling strategy used with 8-bit-wide NAND devices. In this case, the 16-bit-wide data read from or written to the NAND device is fragmented into 2 bytes. According to little-endian access, the LSB of the 16-bit-wide data is ordered first in the byte stream used for 8-bit-based ECC computation.

#### 12.3.4.4.11.3.1.6 ECC Calculation Based on 16-Bit Word

ECC computation based on an 16-bit word is used for 16-bit-wide NAND device interfacing. This ECC computation is not supported when interfacing an 8-bit-wide NAND device, and the GPMC\_ECC\_CONFIG[7] ECC16B bit must be set to 0 when interfacing an 8-bit-wide NAND device.

The parity computation based on 16-bit words affects the row and column parity mapping. The main difference is that the odd and even parity bits P8o and P8e are computed on rows for an 8-bit-based ECC and on columns for a 16-bit based ECC. [Figure 12-267](#) and [Figure 12-268](#) show a 128 Word16 ECC computation scheme and a 256 16-bit word ECC computation scheme.

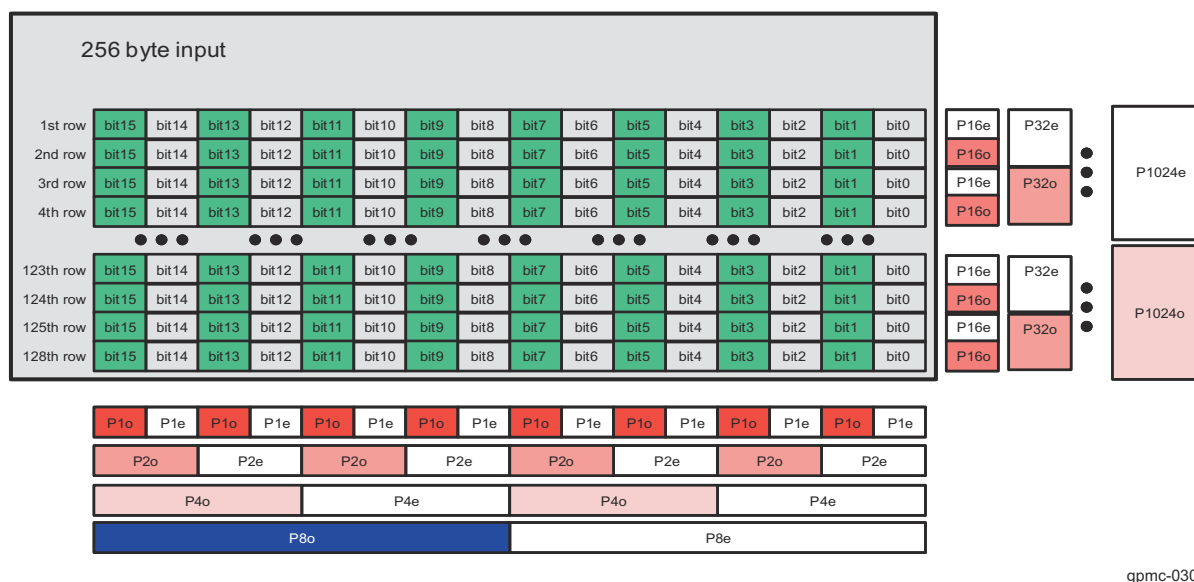


Figure 12-267. 128 Word16 ECC Computation

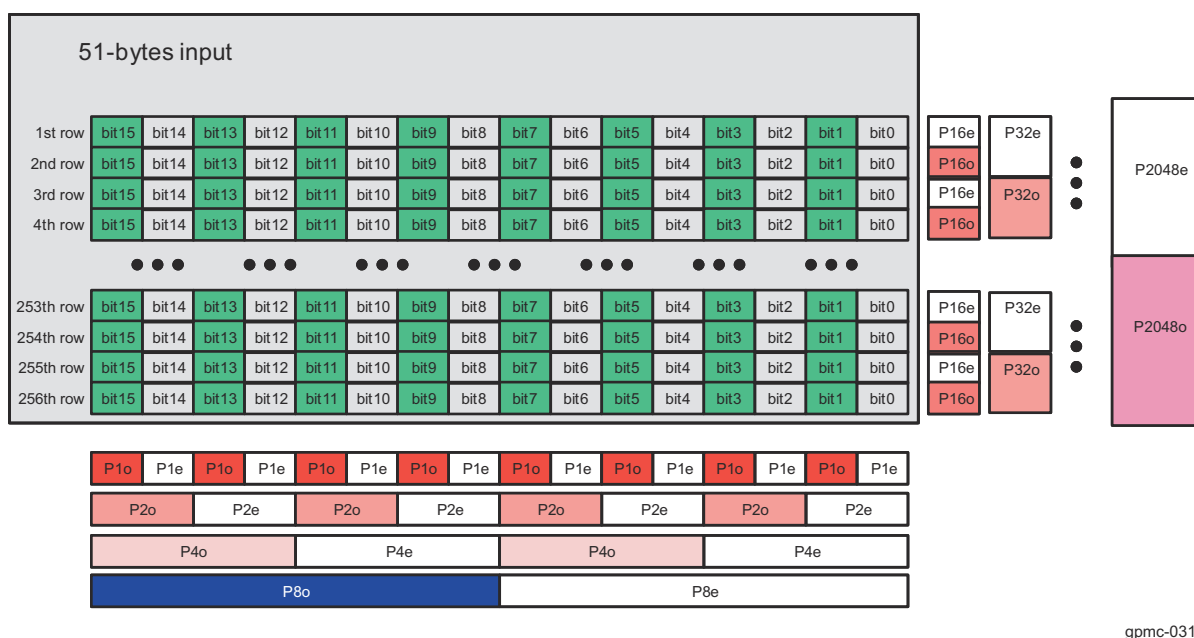


Figure 12-268. 256 Word16 ECC Computation

#### 12.3.4.4.11.3.2 BCH Code

All references to ECC in this subsection refer to the 4- to 16-bit error correction BCH code.

##### 12.3.4.4.11.3.2.1 Requirements

- Read and write accesses to a NAND flash take place by whole pages, in a predetermined sequence: first the data byte page itself, and then some spare bytes, including the BCH ECC (and other information). The NAND device can cache a full page, including spares, for read and write accesses.
  - Typical page write sequence:
    - Sequential write to NAND cache of main data plus spare data, for a page. ECC is calculated on the fly. Calculated ECC can be inserted on the fly in the spares or replaced by dummy accesses.



- When the calculated ECC is replaced by dummy accesses, it must be written to the cache in a second, separate phase. The ECC module is disabled during that time.
- NAND writes its cache line (page) to the array.
- Typical page read sequence:
  - Sequential read of a page. ECC is calculated on the fly.
  - The status of the ECC module buffers determines the presence of errors.
- 2. Accesses to several memories can be interleaved by the GPMC, but only one of those memories at a time can be a NAND using the BCH engine; in other words, only one BCH calculation (for example, for a single page) can be ongoing at any time. The sequential nature of NAND accesses ensures that the data is always written or read out in the same order. BCH-relevant accesses are selected by the chip-selects of the GPMC.
- 3. Each page can hold up to 4KB of data, spare bytes not included. This means up to  $8 \times 512$ -byte BCH messages. Because all the data is written or read out first, followed by the BCH ECC, the BCH engine must be able to hold eight 104-bit remainders or syndromes (or smaller, 52-bit ones) at the same time.

The BCH module can store all remainders internally. After the page start, an internal counter is used to detect the 512-byte sector boundaries. On those boundaries, the current remainder is stored and the divider reset for the next calculation. At the end of the page, the BCH module contains all remainders.

- 4. NAND access cycles hold 8 or 16 bits of data each (1 or 2 bytes); Each NAND cycle takes at least four cycles of the GPMC internal clock. This means the NAND flash timing parameters must define a RDCYCLETIME and a WRCYCLETIME of at least four clock cycles after optimization when using the BCH calculator.
- 5. The spare area is assumed to be large enough to hold the BCH ECC; that is, to have a message of at least 13 bytes available per 512-byte sector of data. The zone of unused spare area by the ECC may or may not be protected by the same ECC scheme, by extending the BCH message beyond 512 bytes (the maximum codeword is 1023 bytes long, ECC included, which leaves much space to cover spare bytes).

#### 12.3.4.4.11.3.2.2 Memory Mapping of BCH Codeword

BCH encoding considers a block of data to protect as a polynomial message  $M(x)$ . In a standard case, 512 bytes of data (that is,  $2^{12}$  bits = 4096 bits) are seen as a polynomial of degree  $2^{12} - 1 = 4095$ , with parameters ranging from  $M_0$  to  $M_{4095}$ . For 512 bytes of data, 52 bits are required for 4-bit error correction, 104 bits are required for 8-bit error correction, and 207 bits are required for 16-bit error correction. The ECC is a remainder polynomial  $R(x)$  of degree 103 (or 51, depending on the selected mode). The complete codeword  $C(x)$  is the concatenation of  $M(x)$  and  $R(x)$ , as described in [Table 12-324](#).

**Table 12-324. Flattened BCH Codeword Mapping (512 Bytes + 104 Bits)**

	Message $M(x)$			ECC $R(x)$		
Bit Number	M4095	...	M0	R103	...	R0

If the message is extended by the addition of spare bytes to be protected by the same ECC, the principle is still valid. For example, a 3-byte extension of the message gives a polynomial message  $M(x)$  of degree  $((512 + 3) \times 8) - 1 = 4119$ , for a total of  $3 + 13 = 16$  spare bytes of spare, all protected as part of the same codeword.

The message and the ECC bits are manipulated and mapped in the GPMC byte-oriented system. The ECC bits are stored in the following registers (where  $i = 0$  to 3):

- GPMC\_BCH\_RESULT0\_i
- GPMC\_BCH\_RESULT1\_i
- GPMC\_BCH\_RESULT2\_i
- GPMC\_BCH\_RESULT3\_i

#### 12.3.4.4.11.3.2.2.1 Memory Mapping of Data Message

The data message mapping must follow the following rules:

- Bit endianness within a byte is little-endian; that is, the bytes LSB is also the lowest-degree polynomial parameter: a byte  $b_7$ - $b_0$  (with  $b_0$  the LSB) represents a segment of polynomial  $b_7 * x^{(7+i)} + b_6 * x^{(6+i)} + \dots + b_0 * x^i$

- The message is mapped in the NAND starting with the highest-order parameters, that is, in the lowest addresses of a NAND page.
- Byte endianness within the 16-bit words in the NAND is big-endian (that is, the same message mapped in 8- and 16-bit memories has the same content at the same byte address).

### Note

The BCH module has no visibility over actual addresses. The most important point is the sequence of data words the BCH sees. However, the NAND page is always scanned incrementally in read and write accesses, which produces the mapping patterns described in the following.

Table 12-325 and Table 12-326 describe the mapping of the same 512-byte vector (typically, a BCH message) in the NAND memory space. The byte address is only an offset module 512 (0x200), because the same page may contain several contiguous 512-byte sectors (BCH blocks). The LSB and MSB are, respectively, the bits M0 and M(2<sup>12</sup>–1) of the codeword mapping discussed previously. In both cases the data vectors are aligned; that is, their boundaries coincide with the RAM data word boundaries.

**Table 12-325. Aligned Message Byte Mapping in 8-Bit NAND**

Byte Offset	8-Bit Word
0x000	(MSB) Byte 511 (0x1FF)
0x001	Byte 510 (0x1FE)
...	...
0x1FF	Byte 0 (0x0) (LSB)

**Table 12-326. Aligned Message Byte Mapping in 16-Bit NAND**

Byte Offset	16-Bit Word MSB	16-Bit Word LSB
0x000	Byte 510 (0x1FE)	(MSB) Byte 511 (0x1FF)
0x002	Byte 508 (0x1FC)	Byte 509 (0x1FD)
...	...	...
0x1FE	Byte 0 (0x0)	(LSB) Byte 1 (0x1)

Table 12-327 through Table 12-332 list the mapping in memory of arbitrarily-sized messages, starting on access (byte or 16-bit word) boundaries for more clarity. Note that message may actually start and stop on arbitrary nibbles. A nibble is a 4-bit entity. The unused nibbles are not discarded, and they can still be used by the BCH module, but as part of the next message section (for example, on the ECC of another sector).

**Table 12-327. Aligned Nibble Mapping of Message in 8-Bit NAND**

Byte Offset	8-Bit Word	
	4-Bit Most Significant Nibble	4-Bit Least Significant Nibble
1	(MSB) Nibble S-1	Nibble S-2
2	Nibble S-3	Nibble S-4
...	...	...
S/2 – 2	Nibble 3	Nibble 2
S/2 – 1	Nibble 1	Nibble 0 (LSB)

**Table 12-328. Misaligned Nibble Mapping of Message in 8-Bit NAND**

Byte Offset	8-Bit Word	
	4-Bit Most Significant Nibble	4-Bit Least Significant Nibble
1	(MSB) Nibble S-1	Nibble S-2
2	Nibble S-3	Nibble S-4
...	...	...
(S + 1) / 2 – 2	Nibble 2	Nibble 1



**Table 12-328. Misaligned Nibble Mapping of Message in 8-Bit NAND (continued)**

Byte Offset	8-Bit Word
$(S + 1) / 2 - 1$	Nibble 0 (LSB)

**Table 12-329. Aligned Nibble Mapping of Message in 16-Bit NAND**

Byte Offset	16-Bit Word			
	4-Bit Most Significant Nibble		4-Bit Least Significant Nibble	
0	Nibble S-3	Nibble S-4	(MSB) Nibble S-1	Nibble S-2
2	Nibble S-7	Nibble S-8	Nibble S-5	Nibble S-6
...	...	...	...	...
$S/2 - 4$	Nibble 5	Nibble 4	Nibble 7	Nibble 6
$S/2 - 2$	Nibble 1	Nibble 0 (LSB)	Nibble 3	Nibble 2

**Table 12-330. Misaligned Nibble Mapping of Message in 16-Bit NAND (1 Unused Nibble)**

Byte Offset	16-Bit Word			
	4-Bit Most Significant Nibble		4-Bit Least Significant Nibble	
0	Nibble S-3	Nibble S-4	(MSB) Nibble S-1	Nibble S-2
2	Nibble S-7	Nibble S-8	Nibble S-5	Nibble S-6
...	...	...	...	...
$(S + 1) / 2 - 4$	Nibble 4	Nibble 3	Nibble 6	Nibble 5
$(S + 1) / 2 - 2$	Nibble 0 (LSB)		Nibble 2	Nibble 1

**Table 12-331. Misaligned Nibble Mapping of Message in 16-Bit NAND (2 Unused Nibbles)**

Byte Offset	16-Bit Word			
	4-Bit Most Significant Nibble		4-Bit Least Significant Nibble	
0	Nibble S-3	Nibble S-4	(MSB) Nibble S-1	Nibble S-2
2	Nibble S-7	Nibble S-8	Nibble S-5	Nibble S-6
...	...	...	...	...
$(S + 2) / 2 - 4$	Nibble 3	Nibble 2	Nibble 5	Nibble 4
$(S + 2) / 2 - 2$			Nibble 1	Nibble 0 (LSB)

**Table 12-332. Misaligned Nibble Mapping of Message in 16-Bit NAND (3 Unused Nibbles)**

Byte Offset	16-Bit Word			
	4-Bit Most Significant Nibble		4-Bit Least Significant Nibble	
0	Nibble S-3	Nibble S-4	(MSB) Nibble S-1	Nibble S-2
2	Nibble S-7	Nibble S-8	Nibble S-5	Nibble S-6
...	...	...	...	...
$(S + 3) / 2 - 4$	Nibble 2	Nibble 1	Nibble 4	Nibble 3
$(S + 3) / 2 - 2$			Nibble 0 (LSB)	

Many other cases exist than those given in the previous tables; for example, where the message does not start on a word boundary.

#### 12.3.4.4.11.3.2.2 Memory-Mapping of the ECC

The ECC (or remainder) is presented by the BCH module as a single 104-bit (or 52-bit), little-endian vector. Software must fetch those 13 bytes (or 6 bytes) from the module interface and then store them to the spare area (page write) in the NAND or to an intermediate buffer for comparison with the stored ECC (page read). There are no constraints on the ECC mapping inside the spare area: it is software-controlled.

It is advised, however, to maintain a coherence in the respective formats of the message or the ECC remainder once they have been read out of the NAND. The error correction algorithm works from the complete codeword

(concatenated message and remainder) once an error is detected. The creation of this codeword must be made as straightforward as possible.

There are cases in which the same NAND access contains both data and the ECC protecting that data. This is the case when the data/ECC boundary (which can be on any nibble) does not coincide with an access boundary. The ECC is calculated on the fly following the write. In that case, the write must also contain part of the ECC because it is impossible to insert the ECC on the fly. Instead:

- During the initial page write (BCH encoding), the ECC is replaced by dummy bits. The BCH encoder is by definition turned OFF during the ECC section, so the BCH result is unmodified.
- During a second phase, the ECC is written to the correct location, next to the actual data.
- The completed line buffer is then written to the NAND array.

#### 12.3.4.4.11.3.2.2.3 Wrapping Modes

For a given wrapping mode, the module automatically goes through a specific number of sections as data is being fed into the module. For each section, the BCH core can be enabled (in which case the data is fed to the BCH divider) or not (in which case the BCH simply counts to the end of the section). When enabled, the data is added to the ongoing calculation for a given sector number (for example, number 0).

Wrapping modes are described as follows. To better understand and see the real-life read and write sequences implemented with each mode, see [Section 12.3.4.4.11.3.2.3, Supported NAND Page Mappings and ECC Schemes](#).

For each mode:

- A sequence describes the mode in pseudo language, with, for each section, the size and the buffer used for ECC processing (if ON). The programmable lengths are size, size0, and size1.
- A checksum condition is given. If the checksum condition is not respected for a given mode, the module behavior is unpredictable. S is the number of sectors in the page; size0 and size1 are the section sizes programmed for the mode, in nibbles.

Wrapping modes 8 through 11 insert a 1-nibble padding where the BCH processing is OFF. This is intended for  $t = 4$  ECC, where ECC is 6 bytes long and the ECC area is expected to include (at least) 1 unused nibble to remain byte-aligned.

##### 12.3.4.4.11.3.2.2.3.1 Manual Mode (0x0)

This mode is intended for short sequences, added manually to a given buffer through the software data port input. A complete page may be built out of several such sequences.

To process an arbitrary sequence of 4-bit nibbles, accesses to the software data port, containing the appropriate data, must be made. If the sequence end does not coincide with an access boundary (for example, to process 5 nibbles = 20 bits in 16-bit access mode) and those nibbles must be skipped, a number of unused nibbles must be programmed in GPMC\_ECC\_SIZE\_CONFIG[31-22] ECCSIZE1. In the same example, 5 nibbles to process + 3 to discard = 8 nibbles =  $2 \times 16$ -bit accesses. Software must set:

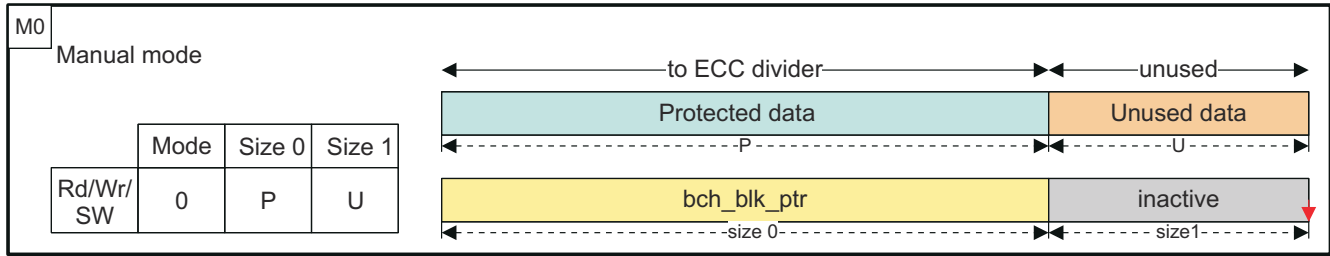
- GPMC\_ECC\_SIZE\_CONFIG[21-12] ECCSIZE0 = 0x5
- GPMC\_ECC\_SIZE\_CONFIG[31-22] ECCSIZE1 = 0x3

---

#### Note

In the following figures, size and size0 are the same parameter.

---



gpmc-032

**Figure 12-269. Manual Mode Sequence and Mapping**

Section processing sequence:

- One time with buffer
  - size0 nibbles of data, processing ON
  - size1 nibbles of unused data, processing OFF

Checksum: size0 + size1 nibbles must fit in a whole number of accesses.

In the following sections, S is the number of sectors in the page.

#### 12.3.4.4.11.3.2.2.3.2 Mode 0x1

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- Repeat with buffer 0 to S-1
  - size0 nibbles spare, processing ON
  - size1 nibbles spare, processing OFF

Checksum: Spare area size (nibbles) = S – (size0 + size1)

#### 12.3.4.4.11.3.2.2.3.3 Mode 0xA (10)

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- Repeat with buffer 0 to S-1
  - size0 nibbles spare, processing ON
  - 1 nibble pad spare, processing OFF
  - size1 nibbles spare, processing OFF

Checksum: Spare area size (nibbles) = S – (size0 + 1 + size1)

#### 12.3.4.4.11.3.2.2.3.4 Mode 0x2

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- Repeat with buffer 0 to S-1
  - size0 nibbles spare, processing OFF
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = S – (size0 + size1)

#### 12.3.4.4.11.3.2.2.3.5 Mode 0x3

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- One time with buffer 0
  - size0 nibbles spare, processing ON
- Repeat with buffer 0 to S-1
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = size0 + (S – size1)

#### 12.3.4.4.11.3.2.2.3.6 Mode 0x7

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- One time with buffer 0
  - size0 nibbles spare, processing ON
- Repeat S times (no buffer used)
  - size1 nibbles spare, processing OFF

Checksum: Spare area size (nibbles) = size0 + (S – size1)

#### 12.3.4.4.11.3.2.2.3.7 Mode 0x8

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- One time with buffer 0
  - size0 nibbles spare, processing ON
- Repeat with buffer 0 to S-1
  - 1 nibble padding spare, processing OFF
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = size0 + (S – (1 + size1))

#### 12.3.4.4.11.3.2.2.3.8 Mode 0x4

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- One time (no buffer used)
  - size0 nibbles spare, processing OFF
- Repeat with buffer 0 to S-1
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = size0 + (S – size1)

#### 12.3.4.4.11.3.2.2.3.9 Mode 0x9

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- One time (no buffer used)
  - size0 nibbles spare, processing OFF
- Repeat with buffer 0 to S-1
  - 1 nibble padding spare, processing OFF
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = size0 + (S – (1 + size1))

#### 12.3.4.4.11.3.2.2.3.10 Mode 0x5

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- Repeat with buffer 0 to S-1
  - size0 nibbles spare, processing ON
- Repeat with buffer 0 to S-1
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = S – (size0 + size1)

#### 12.3.4.4.11.3.2.2.3.11 Mode 0xB (11)

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- Repeat with buffer 0 to S-1
  - size0 nibbles spare, processing ON
- Repeat with buffer 0 to S-1
  - 1 nibble padding spare, processing OFF
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = S – (size0 + 1 + size1)

#### 12.3.4.4.11.3.2.2.3.12 Mode 0x6

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- Repeat with buffer 0 to S-1
  - size0 nibbles spare, processing ON
- Repeat S times (no buffer used)
  - size1 nibbles spare, processing OFF

Checksum: Spare area size (nibbles) = S – (size0 + size1)

#### 12.3.4.4.11.3.2.3 Supported NAND Page Mappings and ECC Schemes

The following rules apply to the entire mapping description:

- Main data area (sectors) size is hardcoded to 512 bytes.
- Spare area size is programmable.
- All page sections (of main area data bytes, protected spare bytes, unprotected spare bytes, and ECC) are defined as explained in [Section 3.4.4.11.3.2.2.1, Memory Mapping of Data Message](#).

Each of the following sections gives a NAND page mapping example (per-sector spare mappings, pooled spare mapping, per-sector spare mapping, with ECC separated at the end of the page).

In the mapping diagrams, sections that belong to the same BCH codeword have the same color (blue or green); unprotected sections are not covered (orange) by the BCH scheme.

Below each mapping diagram, a write (encoding) and read (decoding: syndrome generation) sequence is given, with the number of the active buffers at each point in time (yellow). In the inactive zones (grey), no computing is taking place but the data counter is still active.

In Figure 12-270 through Figure 12-272, the tables on the left summarize the mode, size0, and size1 parameters to program for, respectively, write and read processing of a page, with the given mapping, where:

- P is the size of spare byte section Protected by the ECC (in nibbles)
- U is the size of spare byte section Unprotected by the ECC (in nibbles)
- E is the size of the ECC (in nibbles)
- S is the number of Sectors per page (two in the current diagrams)

Each time the processing of a BCH block is complete (ECC calculation for write/encoding, syndrome generation for read/decoding, indicated by red arrows), the update pointer is pulsed. The processing for block 0 can be the first or the last to complete, depending on the NAND page mapping and operation (read or write). All examples show a page size of 1 KB + spares; that is, S = 2 sectors of 512 bytes. The same principles can be extended to larger pages by adding more sectors.

The actual BCH codeword size is used during the error location work to restrict the search range: by definition, errors can happen only in the codeword that was actually written to the NAND, not in the mathematical codeword of  $n = 2^{13} - 1 = 8191$  bits; that codeword (higher-order bits) is all-zero and implicit during computations.

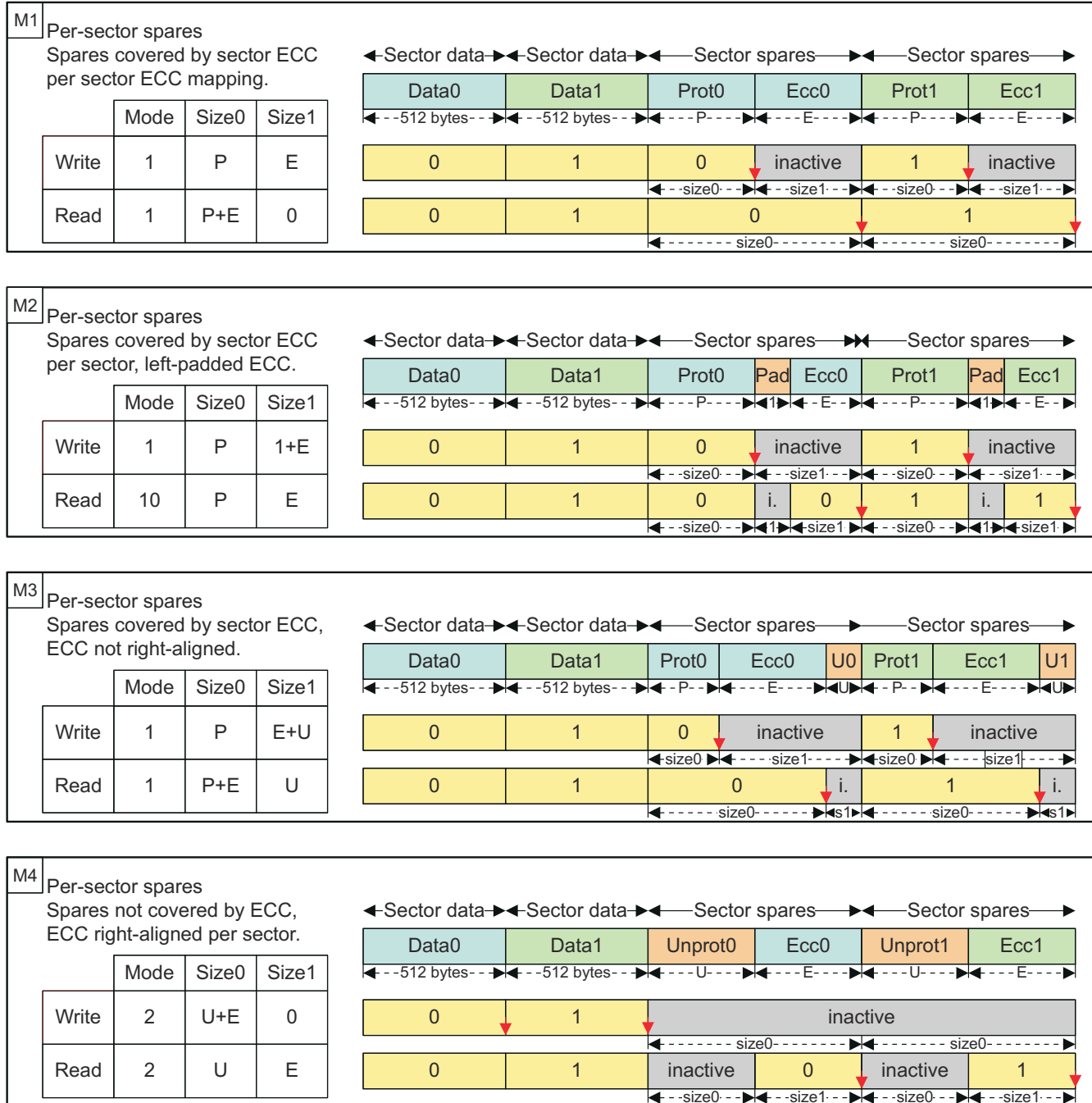
The actual BCH codeword size depends on the mode, the programmed sizes, and the sector number (all sizes in nibbles):

- Spares mapped and protected per sector (below: see M1-M2-M3-M9-M10):
  - All sectors:  $(512) + P + E$
- Spares pooled and protected by sector 0 (below: see M5-M6):
  - Sector 0 codeword:  $(512) + P + E$
  - Other sectors:  $(512) + E$
- Unprotected spares (below: see M4-M7-M8-M11-M12):
  - All codewords  $(512) + E$

#### 12.3.4.4.11.3.2.3.1 Per-Sector Spare Mappings

In these schemes, each 512-byte sector of the main area has its own dedicated section of the spare area. The spare area of each sector is composed of:

- ECC, which must be located after the data it protects
- Other data, which may or may not be protected by the ECC for its sector.



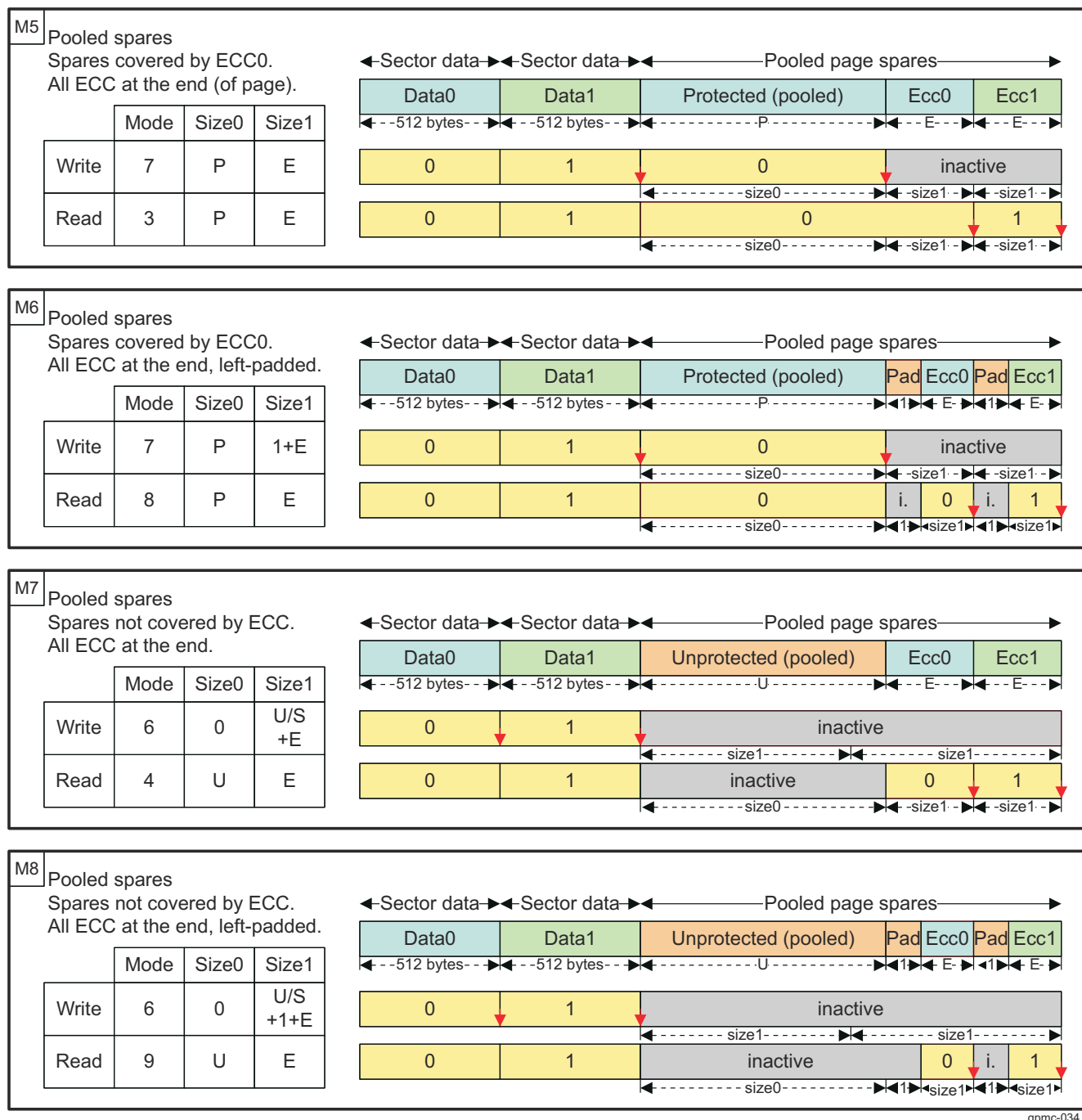
gpmc-033

**Figure 12-270. NAND Page Mapping and ECC: Per-Sector Schemes**

#### 12.3.4.4.11.3.2.3.2 Pooled Spare Mapping

In the following schemes, the spare area is pooled for the page.

- The ECC of each sector is aligned at the end of the spare area.
- The non-ECC spare data may or may not be covered by the ECC of sector 0.



gpmc-034

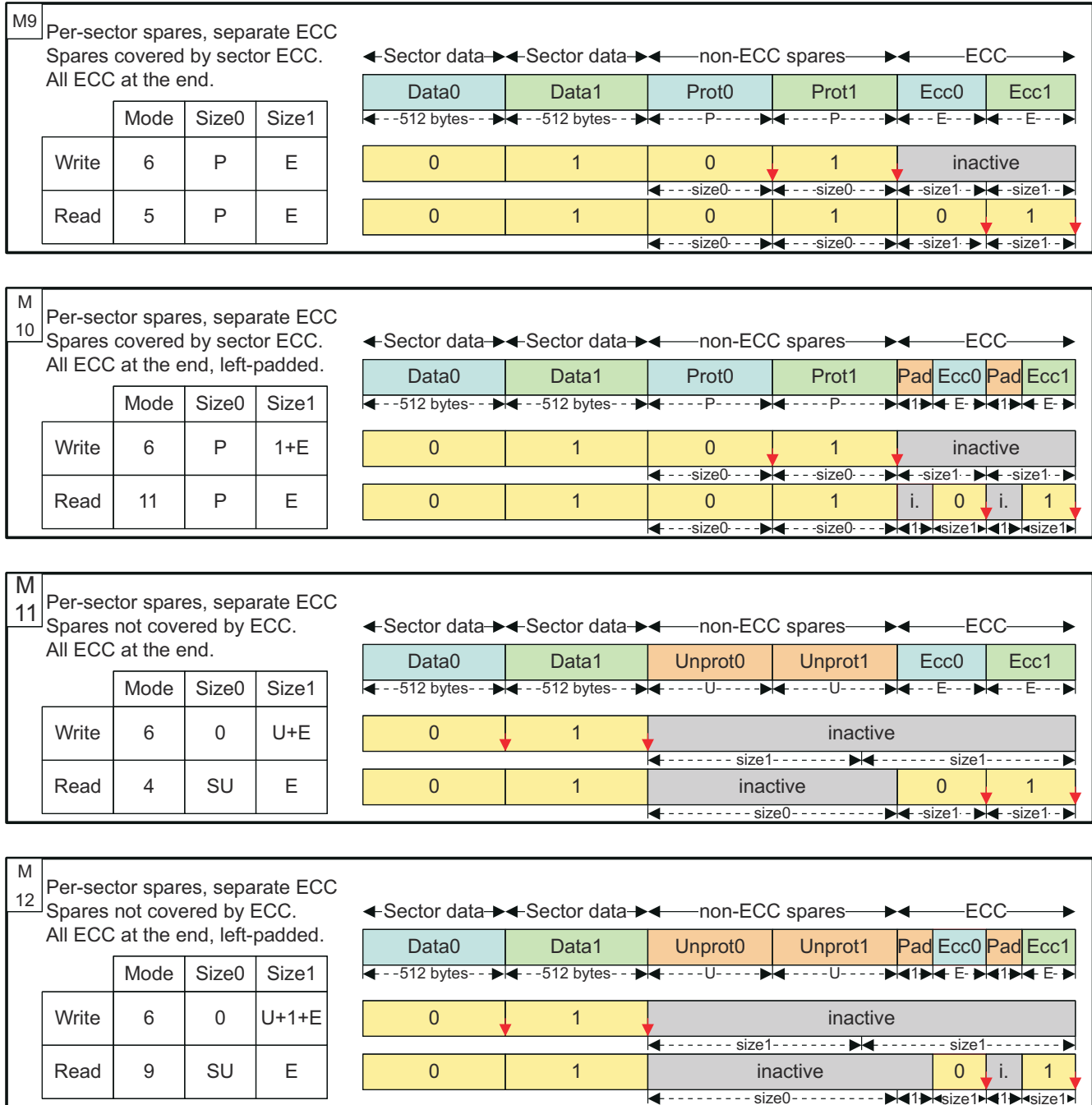
**Figure 12-271. NAND Page Mapping and ECC: Pooled Spare Schemes**

#### 12.3.4.4.11.3.2.3.3 Per-Sector Spare Mapping, with ECC Separated at the End of the Page

In these schemes, each 512-byte sector of the main area is associated with two sections of the spare area.

- ECC section, all aligned at the end of the page
- Other data section, aligned before the ECCs, each of which may or may not be protected by the ECC for its sector.





gpmc\_035

**Figure 12-272. NAND Page Mapping and ECC: Per-Sector Schemes, With Separate ECC**

#### 12.3.4.4.11.4 Prefetch and Write-Posting Engine

##### Note

Some of the GPMC features described in this section may not be supported on this family of devices. For more information, see *GPMC Not Supported Features*.

NAND device data access cycles are usually much slower than the CPU system frequency; such NAND read or write accesses issued by the processor affect the overall system performance, especially considering long

read or write sequences required for NAND page loading or programming. To minimize this effect on system performance, the GPMC includes a prefetch and write-posting engine, which can be used to read from or write to any chip-select location in a buffered manner.

The prefetch and write-posting engine is a simplified embedded-access requester that presents requests to the access engine on a user-defined chip-select target. The access engine interleaves these requests with any request coming from the interconnect interface; as a default, the prefetch and write-posting engine has the lowest priority.

The prefetch and write-posting engine is dedicated to data-stream access (as opposed to random data access); thus, it is primarily dedicated to NAND support. The engine does not include an address generator; the request is limited to chip-select target identification. It includes a 64-byte FIFO associated with a DMA request synchronization line, for optimal DMA-based use.

The prefetch and write-posting engine uses an embedded 64-byte (32 16-bit word) FIFO to prefetch data from the NAND device in read mode (prefetch mode) or to store host data to be programmed into the NAND device in write mode (write-posting mode). The FIFO draining and filling (read and write) can be controlled by a device host processor through interrupt synchronization (an interrupt is triggered whenever a programmable threshold is reached) or by a device DMA module through DMA request synchronization, with a programmable request byte size in prefetch or posting mode.

The prefetch and write-posting engine includes a single memory pool. Therefore, only one mode, read or write, can be used at any given time. In other words, the prefetch and write-posting engine is a single-context engine that can be allocated to only one chip-select at a time for a read prefetch or a write-posting process.

The engine does not support atomic command and address phase programming and is limited to linear memory read or write access. As a consequence, it is limited to NAND data-stream access. The engine depends on the NAND software driver to control block and page opening with the correct data address pointer initialization, before the engine can read from or write to the NAND memory device.

Once started, engine data read and write sequencing is based solely on FIFO location availability and until the total programmed number of bytes is read or written.

Any host-concurrent accesses to a different chip-select are correctly interleaved with ongoing engine accesses. The engine has the lowest priority access so that host accesses to a different chip-select do not suffer a large latency.

A round-robin arbitration scheme can be enabled to ensure minimum bandwidth to the prefetch and write-posting engine in the case of back-to-back direct memory requests to a different chip-select. If the GPMC\_PREFETCH\_CONFIG1[23] PFPWENROUNDROBIN bit is enabled, the arbitration grants the prefetch and write-posting engine access to the GPMC bus for a number of requests programmed in the GPMC\_PREFETCH\_CONFIG1[19-16] PFPWWWEIGHTEDPRIO bit field.

The prefetch/write-posting engine read or write request is routed to the access engine with the chip-select destination ID. After the required arbitration phase, the access engine processes the request as a single access with the data access size equal to the device size specified in the corresponding chip-select configuration.

---

#### Note

The destination chip-select configuration must be set to the NAND protocol-compatible configuration for which address lines are not used (the address bus is not changed from its current value). Selecting a different chip-select configuration can produce undefined behavior.

---

#### 12.3.4.4.11.4.1 General Facts About the Engine Configuration

The engine can be configured only if the GPMC\_PREFETCH\_CONTROL[0] STARTENGINE bit is deasserted.

The engine must be correctly configured in prefetch or write-posting mode and must be linked to a NAND chip-select before it can be started. The chip-select is linked using the GPMC\_PREFETCH\_CONFIG1[26-24] ENGINECSSELECTOR bit field.

In prefetch and write-posting modes, the engine uses byte or 16-bit word access requests, respectively, for an 8- or 16-bit-wide NAND device attached to the linked chip-select. The FIFOTHRESHOLD and TRANSFERCOUNT bit fields must be programmed accordingly as a number of bytes.

When the GPMC\_PREFETCH\_CONFIG1[7] ENABLEENGINE bit is set, the FIFO entry on the interconnect port side is accessible at any address in the associated chip-select memory region. When the ENABLEENGINE bit is set, any host access to this chip-select is rerouted to the FIFO input. Directly accessing the NAND device linked to this chip-select from the host is still possible through the following registers (where i = 0 to 3):

- GPMC\_NAND\_COMMAND\_i
- GPMC\_NAND\_ADDRESS\_i
- GPMC\_NAND\_DATA\_i

The FIFO entry on the interconnect port can be accessed with byte, 16-bit word, or 32-bit word access size, according to little-endian format, even though the FIFO input is 32 bits wide.

The FIFO control is made easier through the use of interrupts or DMA requests associated with the FIFOTHRESHOLD bit field. The GPMC\_PREFETCH\_STATUS[30-24] FIFOPOINTER bit field monitors the number of available bytes to be read in prefetch mode or the number of free empty slots that can be written in write-posting mode. The GPMC\_PREFETCH\_STATUS[13-0] COUNTVALUE bit field monitors the number of remaining bytes to be read or written by the engine according to the value of the TRANSFERCOUNT bit field. The FIFOPOINTER and COUNTVALUE bit fields are always expressed as a number of bytes even if a 16-bit-wide NAND device is attached to the linked chip-select.

In prefetch mode, when the FIFOPOINTER equals 0 (that is, the FIFO is empty), a host read access receives the byte last read from the FIFO as its response. In case of 32-bit word or 16-bit word read accesses, the last byte read from the FIFO is copied the required number of times to fit the requested word size. In write-posting mode, when the FIFOPOINTER equals 0 (that is, the FIFO is full), a host write overwrites the last FIFO byte location. There is no underflow or overflow error reporting in the GPMC.

#### 12.3.4.4.11.4.2 Prefetch Mode

The prefetch mode is selected when the GPMC\_PREFETCH\_CONFIG1[0] ACCESSMODE bit is cleared.

The NAND software driver must issue the block and page opening (READ) command with the correct data address pointer initialization before the engine can be started to read from the NAND memory device. The engine is started by asserting the GPMC\_PREFETCH\_CONTROL[0] STARTENGINE bit. The STARTENGINE bit automatically clears when the prefetch process completes.

If required, the ECC calculator engine must be initialized (that is, reset, configured, and enabled) before the prefetch engine is started so that the ECC is computed correctly on all data read by the prefetch engine.

When the GPMC\_PREFETCH\_CONFIG1[3] SYNCHROMODE bit is cleared, the prefetch engine starts requesting data as soon as the STARTENGINE bit is set. If using this configuration, the host must monitor the NAND device-ready pin so that it sets the STARTENGINE bit only when the NAND device is in a ready state (that is, data is valid for prefetching).

When the SYNCHROMODE bit is set, the prefetch engine starts requesting data when an active-to-inactive WAIT signal transition is detected. The transition detector must be cleared before any transition detection (see [Section 12.3.4.4.11.2.2, Ready Pin Monitored by Hardware Interrupt](#)). The GPMC\_PREFETCH\_CONFIG1[5-4] WAITPINSELECTOR bit field selects which GPMC\_WAIT pin edge detector triggers the prefetch engine in this synchronized mode.

If the STARTENGINE bit is set after the NAND address phase (page opening command), the engine is effectively started only after the actual NAND address phase completion. To prevent GPMC stall during this NAND address phase, set the STARTENGINE bit before NAND address phase completion when in synchronized mode. The prefetch engine starts when an active-to-inactive WAIT signal transition is detected. The STARTENGINE bit is automatically cleared on prefetch process completion.

The prefetch engine issues a read request to fill the FIFO with the amount of data specified by the GPMC\_PREFETCH\_CONFIG2[13-0] TRANSFERCOUNT bit field.

Table 12-333 describes the prefetch mode configuration.

**Table 12-333. Prefetch Mode Configuration**

Bit Field	Register	Value	Comments
STARTENGINE	GPMC_PREFETCH_CONTROL[0]	0	Prefetch engine can be configured only if STARTENGINE is set to 0.
ENGINECSSELECTOR	GPMC_PREFETCH_CONFIG1[26-24]	0 to 3	Selects the chip-select associated with a NAND device where the prefetch engine is active.
ACCESSMODE	GPMC_PREFETCH_CONFIG1[0]	0	Selects prefetch mode
FIFOTHRESHOLD	GPMC_PREFETCH_CONFIG1[14-8]		Selects the maximum number of bytes read or written by the host on DMA or interrupt request
TRANSFERCOUNT	GPMC_PREFETCH_CONFIG2[13-0]		Selects the number of bytes to be read or written by the engine to the selected chip-select
SYNCHROMODE	GPMC_PREFETCH_CONFIG1[3]	0/1	Selects when the engine starts the access to the chip-select
WAITPINSELECT	GPMC_PREFETCH_CONFIG1[17-16]	0 to 1	Selects WAIT pin edge detector (if GPMC_PREFETCH_CONFIG1[3] SYNCHROMODE = 0x1)
ENABLEOPTIMIZEDACCESS	GPMC_PREFETCH_CONFIG1[27]	0/1	See <a href="#">Section 12.3.4.4.11.4.6, Optimizing NAND Access Using the Prefetch and Write-Posting Engine</a> .
CYCLOOPTIMIZATION	GPMC_PREFETCH_CONFIG1[30-28]		Number of clock cycle removed to timing parameters
ENABLEENGINE	GPMC_PREFETCH_CONFIG1[7]	1	Engine enabled
STARTENGINE	GPMC_PREFETCH_CONTROL[0]	1	Starts the prefetch engine

#### 12.3.4.4.11.4.3 FIFO Control in Prefetch Mode

##### Note

Some of the GPMC features described in this section may not be supported on this family of devices. For more information, see *GPMC Not Supported Features*.

The FIFO can be drained directly by a device host processor or a DMA module channel.

In draining mode, the FIFO status can be monitored through the GPMC\_PREFETCH\_STATUS[30-24] FIFOPOINTER bit field or through the GPMC\_PREFETCH\_STATUS[16] FIFOTHRESHOLDSTATUS bit. The FIFOPOINTER indicates the current number of available data to be read; FIFOTHRESHOLDSTATUS set to 1 indicates that at least FIFOTHRESHOLD bytes are available from the FIFO.

An interrupt can be triggered by the GPMC if the GPMC\_IRQENABLE[0] FIFOEVENTENABLE bit is set. The FIFO interrupt event is logged, and the GPMC\_IRQSTATUS[0] FIFOEVENTSTATUS bit is set. To clear the interrupt, all the available bytes must be read, or at least enough bytes to get below the programmed FIFO threshold, and the FIFOEVENTSTATUS bit must be cleared to enable further interrupt events. The FIFOEVENTSTATUS bit must always be reset before asserting the FIFOEVENTENABLE bit to clear any out-of-date logged interrupt event. This interrupt generation must be enabled after enabling the STARTENGINE bit.

Prefetch completion can be monitored through the GPMC\_PREFETCH\_STATUS[13-0] COUNTVALUE bit field. COUNTVALUE indicates the number of currently remaining data to be requested according to the TRANSFERCOUNT value. An interrupt can be triggered by the GPMC when the prefetch process is complete (that is, COUNTVALUE equals 0) if the GPMC\_IRQENABLE[1] TERMINALCOUNTEVENTENABLE bit is set. At prefetch completion, the TERMINALCOUNT interrupt event is also logged, and the GPMC\_IRQSTATUS[1] TERMINALCOUNTSTATUS bit is set. To clear the interrupt, the TERMINALCOUNTSTATUS bit must

be cleared. The TERMINALCOUNTSTATUS bit must always be cleared prior to asserting the TERMINALCOUNTEVENTENABLE bit to clear any out-of-date logged interrupt event.

#### Note

The COUNTVALUE value is valid only when the prefetch engine is active (started), and an interrupt is only triggered when COUNTVALUE reaches 0, that is, when the prefetch engine automatically goes from an active to an inactive state.

The number of bytes to be prefetched (programmed in TRANSFERCOUNT) must be a multiple of the programmed FIFOTHRESHOLD to trigger the correct number of interrupts allowing a deterministic and transparent FIFO control. If this guideline is respected, the number of ISR accesses is always required and the FIFO is always empty after the last interrupt is triggered. In other cases, the TERMINALCOUNT interrupt must be used to read the remaining bytes in the FIFO (the number of remaining bytes being lower than the FIFOTHRESHOLD value).

In DMA draining mode, the GPMC\_PREFETCH\_CONFIG1[2] DMAMODE bit must be set so that the GPMC issues a DMA hardware request when at least FIFOTHRESHOLD bytes are ready to be read from the FIFO. The DMA channel that owns this DMA request must be programmed so that the number of bytes programmed in FIFOTHRESHOLD is read from the FIFO during the DMA request process. The DMA request is kept active until this number of bytes has effectively been read from the FIFO, and no other DMA request can be issued until the ongoing active request is complete.

In prefetch mode, the TERMINALCOUNT event is also a source of DMA requests if the number of bytes to be prefetched is not a multiple of FIFOTHRESHOLD, the remaining bytes in the FIFO can be read by the DMA channel using the last DMA request. This assumes that the number of remaining bytes to be read is known and controlled through the DMA channel programming model.

Any potentially active DMA request is cleared when the prefetch engine goes from inactive-to-active prefetch (the STARTENGINE bit is set to 1). The associated DMA channel must always be enabled after setting the STARTENGINE bit so that the out-of-date active DMA request does not trigger spurious DMA transfers.

#### 12.3.4.4.11.4.4 Write-Posting Mode

The write-posting mode is selected when the GPMC\_PREFETCH\_CONFIG1[0] ACCESSMODE bit is set.

The NAND software driver must issue the correct address pointer initialization command (page program) before the engine can start writing data into the NAND memory device. The engine starts when the GPMC\_PREFETCH\_CONTROL[0] STARTENGINE bit is set to 1. The STARTENGINE bit clears automatically when posting completes. When all data have been written to the NAND memory device, the NAND software driver must issue the second cycle program command and monitor the status for programming process completion (adding ECC handling, if required).

If used, the ECC calculator engine must be started (configured, reset, and enabled) before the posting engine is started so that the ECC parities are calculated properly on all data written by the prefetch engine to the associated chip-select.

In write-posting mode, the GPMC\_PREFETCH\_CONFIG1[3] SYNCHROMODE bit must be cleared so that posting starts as soon as the STARTENGINE bit is set and the FIFO is not empty.

If the STARTENGINE bit is set after the NAND address phase (page program command), the STARTENGINE setting is effective only after the actual NAND command completion. To prevent GPMC stall during this NAND command phase, set the STARTENGINE bit field before the NAND address completion and ensure that the associated DMA channel is enabled after the NAND address phase.

The posting engine issues a write request when valid data are available from the FIFO and until the programmed GPMC\_PREFETCH\_CONFIG2[13-0] TRANSFERCOUNT accesses are complete.

The STARTENGINE bit clears automatically when posting completes. When all data have been written to the NAND memory device, the NAND software driver must issue the second cycle program command and monitor



the status for programming process completion. The closing program command phase must be issued only when the full NAND page has been written into the NAND flash write buffer, including the spare area data and the ECC parities, if used.

Table 12-334 describes the write-posting configuration.

**Table 12-334. Write-Posting Mode Configuration**

Bit Field	Register	Value	Comments
STARTENGINE	GPMC_PREFETCH_CONTROL[0]	0	Write-posting engine can be configured only if STARTENGINE is set to 0.
ENGINECSSELECTOR	GPMC_PREFETCH_CONFIG1[26-24]	0 to 3	Selects the chip-select associated with a NAND device where the prefetch engine is active
ACCESSMODE	GPMC_PREFETCH_CONFIG1[0]	1	Selects write-posting mode
FIFOTHRESHOLD	GPMC_PREFETCH_CONFIG1[14-8]		Selects the maximum number of bytes read or written by the host on DMA or interrupt request
TRANSFERCOUNT	GPMC_PREFETCH_CONFIG2[13-0]		Selects the number of bytes to be read or written by the engine from/to the selected chip-select
SYNCHROMODE	GPMC_PREFETCH_CONFIG1[3]	0	Engine starts the access to chip-select as soon as STARTENGINE is set.
ENABLEOPTIMIZEDACCESS	GPMC_PREFETCH_CONFIG1[27]	0/1	See <a href="#">Section 12.3.4.4.11.4.6, Optimizing NAND Access Using the Prefetch and Write-Posting Engine</a> .
CYCLOPTIMIZATION	GPMC_PREFETCH_CONFIG1[30-28]		
ENABLEENGINE	GPMC_PREFETCH_CONFIG1[7]	1	Engine enabled
STARTENGINE	GPMC_PREFETCH_CONTROL[0]	1	Starts the prefetch engine

#### 12.3.4.4.11.4.5 FIFO Control in Write-Posting Mode

##### Note

Some of the GPMC features described in this section may not be supported on this family of devices. For more information, see *GPMC Not Supported Features*.

The FIFO can be filled directly by a device host processor or a DMA module channel.

In filling mode, the FIFO status can be monitored through the FIFOPOINTER or through the GPMC\_PREFETCH\_STATUS[16] FIFOTHRESHOLDSTATUS bit. FIFOPOINTER indicates the current number of available free byte places in the FIFO, and the FIFOTHRESHOLDSTATUS bit, when set, indicates that at least FIFOTHRESHOLD free byte places are available in the FIFO.

An interrupt can be issued by the GPMC if the GPMC\_IRQENABLE[0] FIFOEVENTENABLE bit is set. When the interrupt is fired, the GPMC\_IRQSTATUS[0] FIFOEVENTSTATUS bit is set. To clear the interrupt, enough bytes must be written to fill the FIFO or to get below the programmed threshold, and the FIFOEVENTSTATUS bit must be cleared to get further interrupt events. The FIFOEVENTSTATUS bit must always be cleared before asserting the FIFOEVENTENABLE bit to clear any out-of-date logged interrupt event. This interrupt must be enabled after enabling the STARTENGINE bit.

The posting completion can be monitored through the GPMC\_PREFETCH\_STATUS[13-0] COUNTVALUE bit field. COUNTVALUE indicates the current number of remaining data to be written based on the value of the TRANSFERCOUNT bit field. An interrupt is issued by the GPMC when the write-posting process completes (that is, COUNTVALUE equal to 0) if the GPMC\_IRQENABLE[1] TERMINALCOUNTEVENTENABLE bit is set. When the interrupt is fired, the GPMC\_IRQSTATUS[1] TERMINALCOUNTSTATUS bit is set. To clear the interrupt, the TERMINALCOUNTSTATUS bit must be cleared. The TERMINALCOUNTSTATUS bit must always be cleared before asserting the TERMINALCOUNTEVENTENABLE bit to clear any out-of-date logged interrupt event.

### Note

The value of the COUNTVALUE bit field is valid only if the write-posting engine is active and started, and an interrupt is issued only when COUNTVALUE reaches 0; that is, when the posting engine automatically goes from active to inactive.

In DMA filling mode, the DMAMode bit field in the GPMC\_PREFETCH\_CONFIG1[2] DMAMODE bit must be set so that the GPMC issues a DMA hardware request when at least FIFOTHRESHOLD bytes-free places are available in the FIFO. The DMA channel that owns this DMA request must be programmed so that a number of bytes equal to the value programmed in the FIFOTHRESHOLD bit field are written into the FIFO during the DMA access. The DMA request remains active until the associated number of bytes has effectively been written into the FIFO, and no other DMA request can be issued until the ongoing active request completes.

Any potentially active DMA request is cleared when the prefetch engine goes from inactive-to-active prefetch (STARTENGINE set to 1). The associated DMA channel must always be enabled after setting the STARTENGINE bit so that an out-of-date active DMA request does not trigger spurious DMA transfers.

In write-posting mode, DMA or CPU fills the FIFO with no consideration for the associated byte enables. Any byte stored in the FIFO is written into the memory device.

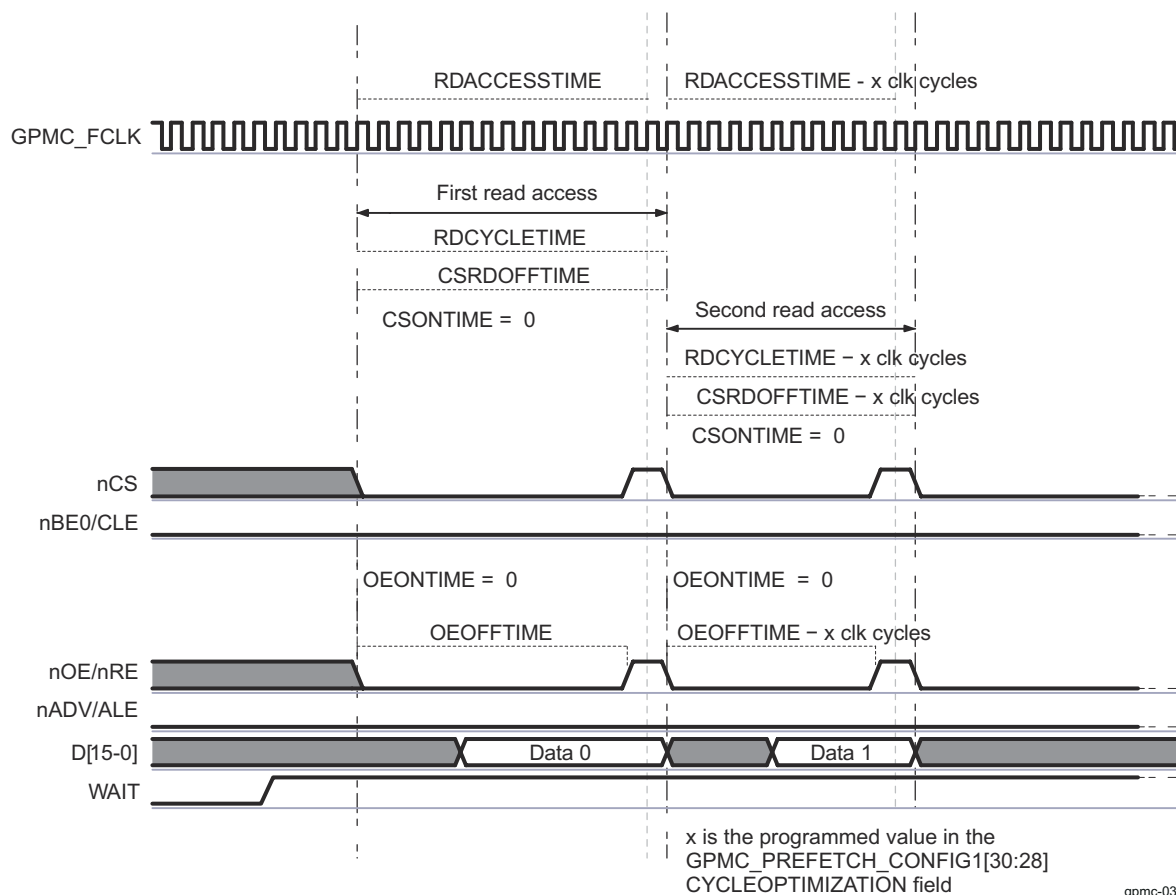
#### 12.3.4.4.11.4.6 Optimizing NAND Access Using the Prefetch and Write-Posting Engine

Access time to a NAND memory device can be optimized for back-to-back accesses if the associated nCS signal is not deasserted between accesses. The GPMC access engine can track prefetch engine accesses to optimize the access timing parameter programmed for the allocated chip-select, if no accesses to other chip-selects (that is, interleaved accesses) occur. Similarly, the access engine also eliminates CYCLE2CYCLEDELAY even if CYCLE2CYCLESAMECSN is set. This capability is limited to the prefetch and write-posting engine accesses, and accesses to a NAND memory device (through the defined chip-select memory region or through the GPMC\_NAND\_DATA\_i location, where i = 0 to 3) are never optimized.

The GPMC\_PREFETCH\_CONFIG1[27] ENABLEOPTIMIZEDACCESS bit must be set to enable optimized accesses. To optimize access time, the GPMC\_PREFETCH\_CONFIG1[30-28] CYCLEOPTIMIZATION bit field defines the number of GPMC\_FCLK cycles to be suppressed from the following timing parameters:

- RDCYCLETIME
- WRCYCLETIME
- RDACCESSTIME
- WRACCESSTIME
- CSOFFTIME
- ADVOFFTIME
- OEOFFTIME
- WEOFFTIME

Figure 12-273 shows that in the case of back-to-back accesses to the NAND flash through the prefetch engine, CYCLE2CYCLESAMECSN is forced to 0 when using optimized accesses. The first access uses the regular timing settings for this chip-select. All accesses after this one use settings reduced by x clock cycles, x being defined by the GPMC\_PREFETCH\_CONFIG1[30-28] CYCLEOPTIMIZATION bit field.



gpmc-036

**Figure 12-273. NAND Read Cycle Optimization Timing Description**

#### 12.3.4.4.11.4.7 Interleaved Accesses Between Prefetch and Write-Posting Engine and Other Chip-Selects

Any on-going read or write access from the prefetch and write-posting engine is completed before an access to any other chip-select can be initiated. As a default, the arbiter uses a fixed-priority algorithm, and the prefetch and write-posting engine has the lowest priority. The maximum latency added to access starting time in this case equals the RDCYCLETIME or WRCYCLETIME (optimized or not) plus the requested BUSTURNAROUND delay for bus turnaround completion programmed for the chip-select to which the NAND device is connected.

Alternatively, a round-robin arbitration can be used to prioritize accesses to the external bus. This arbitration scheme is enabled by setting the GPMC\_PREFETCH\_CONFIG1[23] PFPWENROUNDROBIN bit. When a request to another chip-select is received while the prefetch and write-posting engine is active, priority is given to the new request. The request processed thereafter is the prefetch and write-posting engine request, even if another interconnect request is passed in the mean time. The engine keeps control of the bus for an additional number of requests programmed in the GPMC\_PREFETCH\_CONFIG1[19-16] PFPWWEIGHTEDPRIO bit field. Control is then passed to the direct interconnect request.

As an example, the round-robin arbitration scheme is selected with PFPWWEIGHTEDPRIO set to 0x2. Considering that the prefetch and write-posting engine and the interconnect interface are always requesting access to the external interface, the GPMC grants priority to the direct interconnect access for one request. The GPMC then grants priority to the engine for three requests, and finally back to the direct interconnect access, until the arbiter is reset when one of the two initiators stops initiating requests.



#### 12.3.4.4.12 GPMC Use Cases and Tips

##### 12.3.4.4.12.1 How to Set GPMC Timing Parameters for Typical Accesses

###### 12.3.4.4.12.1.1 External Memory Attached to the GPMC Module

As discussed in the introduction to this chapter, the GPMC module supports the following external memory types:

- Asynchronous or synchronous, 8- or 16-bit-wide memory or device
- 16-bit address/data-multiplexed or non-multiplexed NOR flash device
- 8- or 16-bit NAND flash device

The following examples describe how to calculate GPMC timing parameters by showing a typical parameter setup for the access to be performed.

The example is based on a 512-Mb multiplexed NOR flash memory with the following characteristics:

- Type: NOR flash (address/data-multiplexed mode)
- Size: 512M bits
- Data Bus: 16 bits wide
- Speed: 104-MHz clock frequency
- Read access time: 80 ns

###### 12.3.4.4.12.1.2 Typical GPMC Setup

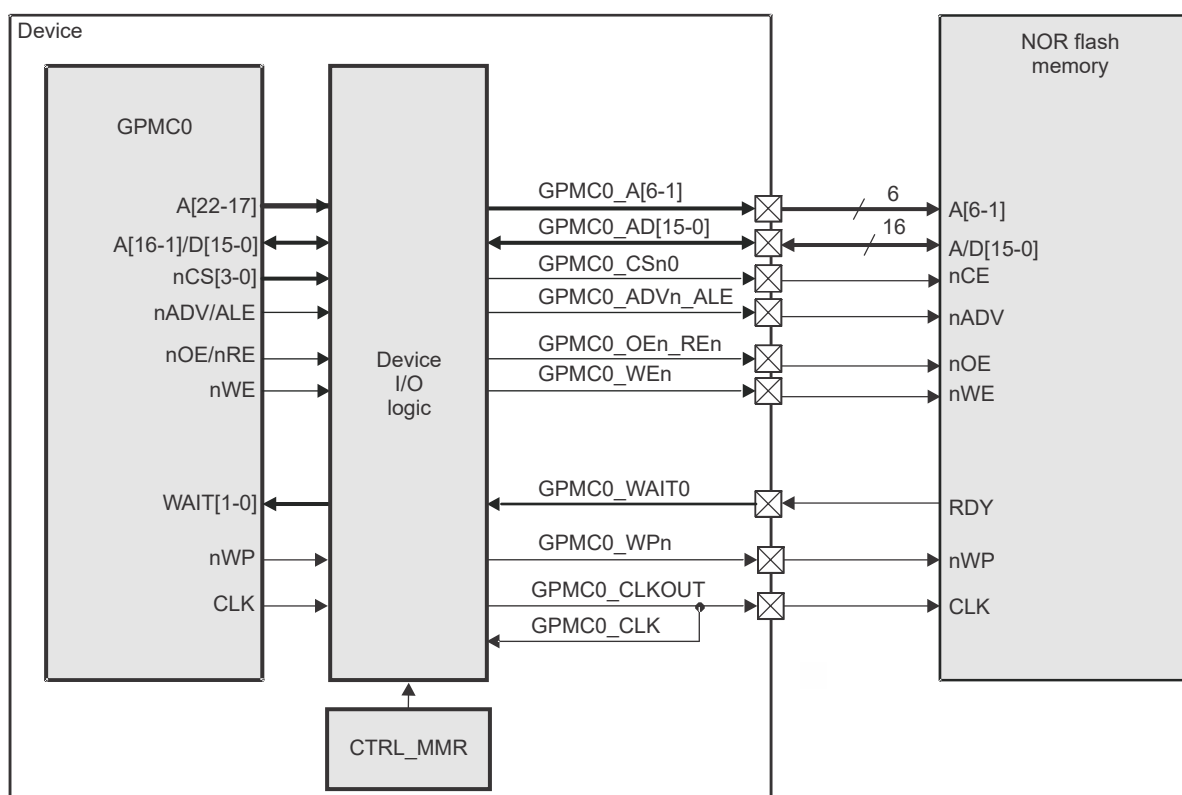
[Table 12-335](#) lists some of the I/Os of the GPMC module.

**Table 12-335. Typical GPMC Setup Signals**

Module Pin	I/O <sup>(1)</sup>	Description
GPMC0_FCLK	Internal	Functional clock. Acts as the time reference.
GPMC0_ICLK	Internal	Interface clock. Acts as the time reference.
GPMC0_CLKOUT	O	External clock provided to the external device for synchronous operations
GPMC0_A[21-16]	O	Address
GPMC0_AD[15-0]	I/O	Data-multiplexed with addresses A[16-1] on memory side
GPMC0_CSn[3-0]	O	Chip-selects
GPMC0_ADVn_ALE	O	Address valid enable
GPMC0_OEn_REn	O	Output enable (read access only)
GPMC0_WEn	O	Write enable (write access only)
GPMC0_WAIT[1-0]	I	Ready signal from memory device. Indicates when valid burst data is ready to be read

(1) I = Input; O = Output

[Figure 12-274](#) shows the typical connection between the GPMC module and an attached NOR Flash memory.



**Figure 12-274. GPMC Connection to an External NOR Flash Memory**

The following sections demonstrate how to calculate GPMC parameters for three access types:

- Synchronous burst read
- Asynchronous read
- Asynchronous single write

#### 12.3.4.4.12.1.2.1 GPMC Configuration for Synchronous Burst Read Access

##### Note

The examples in [Section 12.3.4.4.12.1.2.1](#) through [Section 12.3.4.4.12.1.2.3](#) are based on a clock rate of 104 MHz. See the device-specific Datasheet for the maximum frequency appropriate for this device and the memory datasheet for the maximum frequency for the particular memory device.

The clock runs at 104 MHz ( $f = 104 \text{ MHz}$ ;  $T = 9.615 \text{ ns}$ ).

[Table 12-336](#) shows the timing parameters (on the memory side) that determine the parameters on the GPMC side.

[Table 12-337](#) shows how to calculate timings for the GPMC using the memory parameters.

[Figure 12-275](#) shows the synchronous burst read access.

**Table 12-336. Useful Timing Parameters on the Memory Side**

AC Read Characteristics on the Memory Side	Description	Duration (ns)
tCES	nCS setup time to clock	0
tACS	Address setup time to clock	3

**Table 12-336. Useful Timing Parameters on the Memory Side (continued)**

AC Read Characteristics on the Memory Side	Description	Duration (ns)
tIACC	Synchronous access time	80
tBACC	Burst access time valid clock to output delay	5.2
tCEZ	Chip-select to High-Z	7
tOEZ	Output enable to High-Z	7
tAVC	nADV setup time	6
tAVD	nADV pulse	6
tACH	Address hold time from clock	3

The following terms, which describe the timing interface between the controller and its attached device, are used to calculate the timing parameters on the GPMC side:

- Read access time (GPMC side): Time required to activate the clock + read access time requested on the memory side + data setup time required for optimal capture of a burst of data
- Data setup time (GPMC side): Ensures a good capture of a burst of data (as opposed to taking a burst of data out). One word of data is processed in one clock cycle ( $T = 9.615$  ns). The read access time between two bursts of data is  $tBACC = 5.2$  ns. Therefore, data setup time is a clock period –  $tBACC = 4.415$  ns of data setup.
- Access completion (GPMC side): (Different from page burst access time) Time required between the last burst access and access completion: nCS/nOE hold time (nCS and nOE must be released at the end of an access. These signals are held to allow the access to complete).
- Read cycle time (GPMC side): Read access time + access completion
- Write cycle time for burst access: Not supported for NOR flash memory

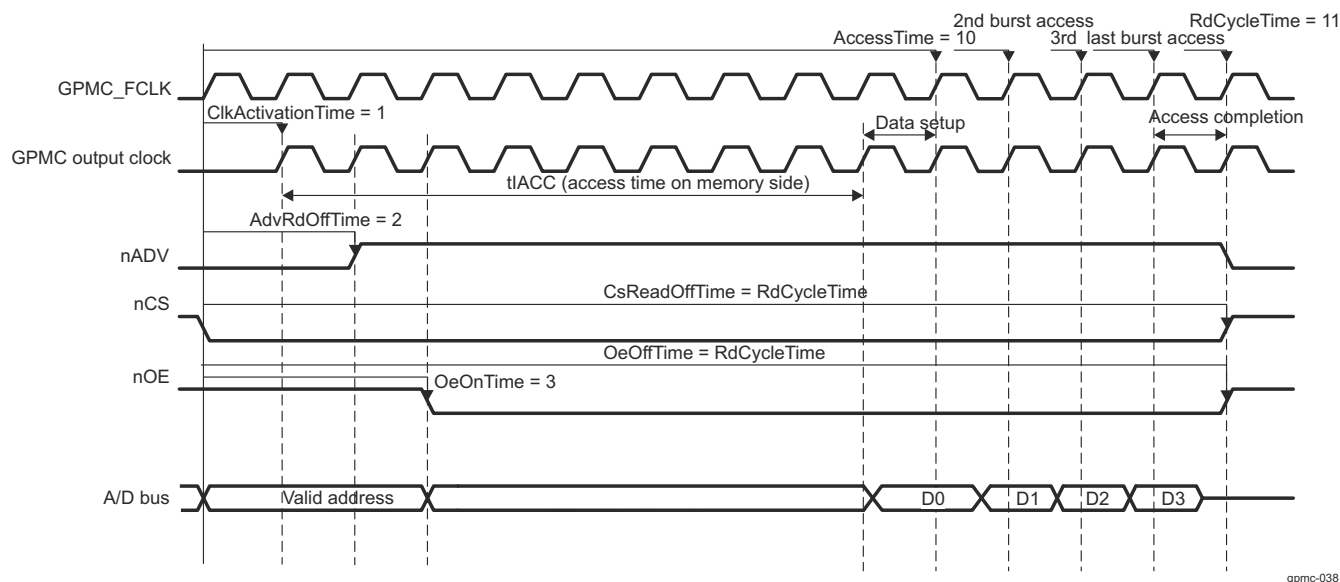
**Table 12-337. Calculating GPMC Timing Parameters**

Parameter Name on GPMC Side	Formula	Duration (ns)	Number of Clock Cycles (F = 104 MHz)	GPMC Register Configurations
GPMC FCLK Divider	–	–	–	GPMCFCLKDIVIDER = 0x0
ClkActivation Time	$\min(tCES, tACS)$	3	1	CLKACTIVATIONTIME = 0x1
RdAccessTime	$\text{roundmax}(\text{ClkActivationTime} + tIACC + \text{DataSetupTime})$	94.03: (9.615 + 80 + 4.415)	10: $\text{roundmax}(94.03 / 9.615)$	RDACCESSTIME = 0xA
PageBurst RdAccessTime	$\text{roundmax}(tBACC)$	$\text{roundmax}(5.2)$	1	PAGEBURSTACCESSTIME = 0x1
RdCycleTime	$\text{RdAccess time} + \max(tCEZ, tOEZ)$	101.03: (94.03 + 7)	11	RDCYCLETIME = 0xB
CsOnTime	tCES	0	0	CSONTIME = 0x0
CsReadOffTime	RdCycleTime	–	11	CSRDOFFTIME = 0xB
AdvOnTime	tAVC <sup>(1)</sup>	0	0	ADVONTIME = 0x0
AdvRdOffTime	tAVD + tAVC <sup>(2)</sup>	12	2	ADVRDOFFTIME = 0x2
OeOnTime <sup>(3)</sup>	$(\text{ClkActivationTime} + tACH) < \text{OeOnTime}(\text{ClkActivationTime} + tIACC)$	–	3, for instance	OEONTIME = 0x3
OeOffTime	RdCycleTime	–	11	OEOFFTIME = 0xB

(1) The external clock provided to the NOR flash is not yet available.

(2)  $\text{AdvRdOffTime} - \text{AdvOnTime} = tAVD$ ; thus,  $\text{AdvRdOffTime} = tAVD + \text{AdvOnTime} = tAVD + tAVC$ .

(3) OeOnTime must ensure that addresses are available. It must not exceed the availability of the first burst of data.



**Figure 12-275. Synchronous Burst Read Access (Timing Parameters in Clock Cycles)**

#### 12.3.4.4.12.1.2.2 GPMC Configuration for Asynchronous Read Access

The clock runs at 104 MHz ( $f = 104 \text{ MHz}$ ;  $T = 9.615 \text{ ns}$ ).

Table 12-338 shows the timing parameters (on the memory side) that determine the parameters on the GPMC side.

Table 12-339 shows how to calculate timings for the GPMC using the memory parameters.

Figure 12-276 shows the asynchronous read access.

**Table 12-338. AC Characteristics for Asynchronous Read Access**

AC Read Characteristics on the Memory Side	Description	Duration (ns)
tCE	Read Access time from nCS low	80
tAAVDS	Address setup time to rising edge of nADV	3
tAVDP	nADV low time	6
tCAS	nCS setup time to nADV	0
tOE	Output enable to output valid	6
tOEZ	Output enable to High-Z	7

Use the following formula to calculate the RdCycleTime parameter for this typical access:

$$\text{RdCycleTime} = \text{RdAccessTime} + \text{AccessCompletion} = \text{RdAccessTime} + 1 \text{ clock cycle} + \text{tOEZ}$$

1. On the memory side, the external memory makes the data available to the output bus. This is the memory-side read access time defined in Table 12-338: the number of clock cycles between the address capture (nADV rising edge) and the data valid on the output bus.

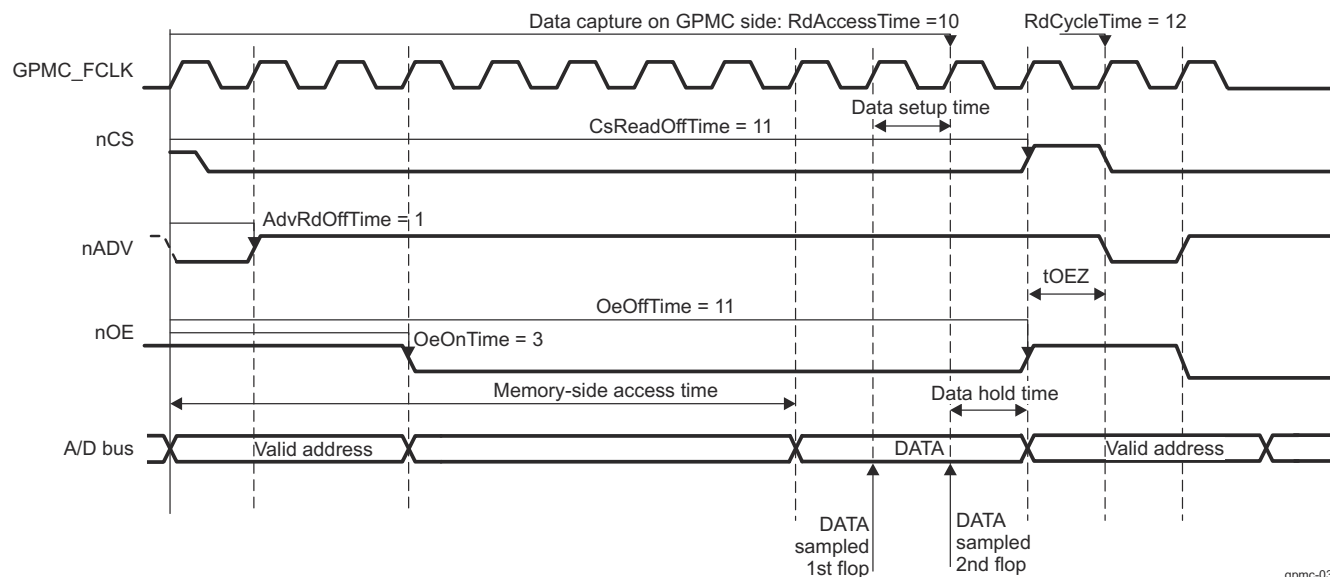
The GPMC requires some hold time to allow the data to be captured correctly and the access to be finished.

2. To read the data correctly, the GPMC must be configured to meet the data setup time requirement of the memory; the GPMC module captures the data on the next rising edge. This is access time on the GPMC side.
3. There must also be a data hold time for correctly reading the data (checking that there is no nOE/nCS deassertion while reading the data). This data hold time is one clock cycle (that is, RdAccessTime + 1).
4. To complete the access, nOE/nCS signals are driven to High-Z. RdAccessTime + 1 + tOEZ is the read cycle time.

5. Addresses can now be relatched and a new read cycle begun.

**Table 12-339. GPMC Timing Parameters for Asynchronous Read Access**

Parameter Name on GPMC Side	Formula	Duration (ns)	Number of Clock Cycles (F = 104 MHz)	GPMC Register Configurations
ClkActivationTime		n/a (asynchronous mode)		
RdAccessTime	round max (tCE)	80	10	RDACCESSTIME = 0xA
PageBurstAccessTime	N/A (single access)			
RdCycleTime	RdAccessTime + 1 cycle + tOEZ	96.615	12	RDCYCLETIME = 0xC
CsOnTime	tCAS	0	0	CSONTIME = 0x0
CsReadOffTime	RdAccessTime + 1 cycle	89.615	11	CSRDOFFTIME = 0xB
AdvOnTime	tAAVDS	3	1	ADVONTIME = 0x1
AdvRdOffTime	tAAVDS + tAVDP	9	1	ADVRDOFFTIME = 0x1
OeOnTime	OeOnTime >= AdvRdOffTime (multiplexed mode)	-	3, for instance	OEONTIME = 0x3
OeOffTime	RdAccessTime + 1 cycle	89.615	11	OEOFFTIME = 0xB



gpmc-039

**Figure 12-276. Asynchronous Single Read Access (Timing Parameters in Clock Cycles)**

#### 12.3.4.4.12.1.2.3 GPMC Configuration for Asynchronous Single Write Access

The clock runs at 104 MHz: (f = 104 MHz; T = 9.615 ns).

Table 12-341 shows how to calculate timings for the GPMC using the memory parameters.

Table 12-340 shows the timing parameters (on the memory side) that determine the parameters on the GPMC side.

Figure 12-277 shows the asynchronous single write access.

**Table 12-340. AC Characteristics for Asynchronous Single Write (Memory Side)**

AC Characteristics on the Memory Side	Description	Duration (ns)
tWC	Write cycle time	60
tAVDP	nADV low time	6
tWP	Write pulse width	25
tWPH	Write pulse width high	20

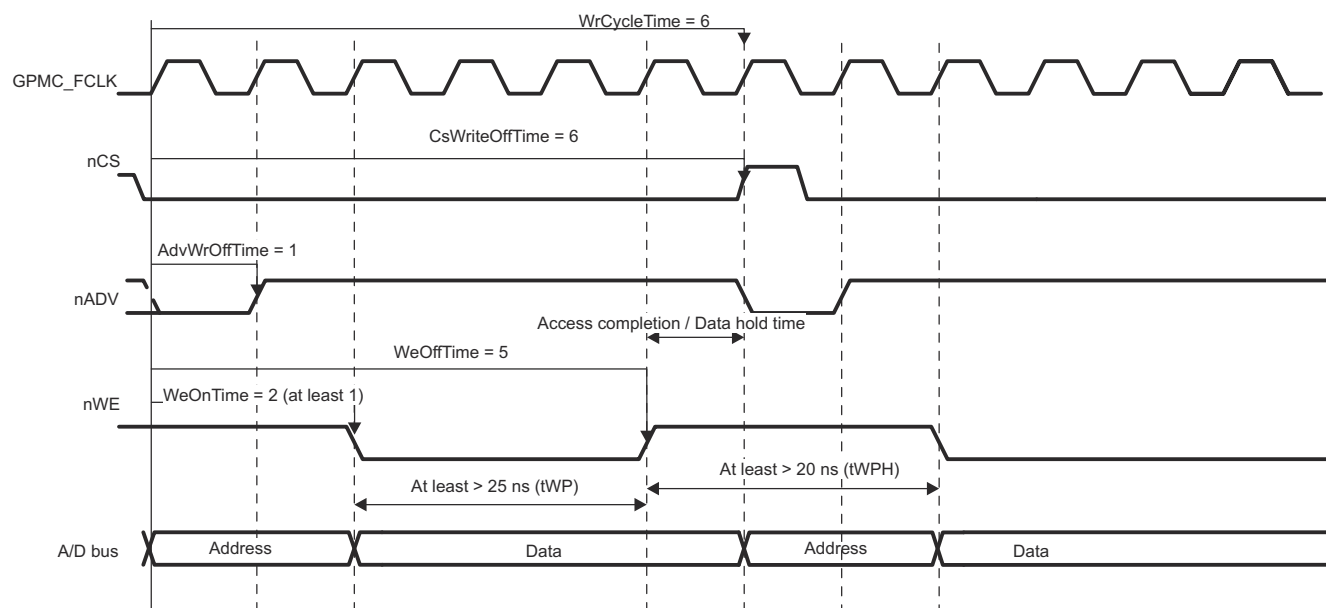
**Table 12-340. AC Characteristics for Asynchronous Single Write (Memory Side) (continued)**

AC Characteristics on the Memory Side	Description	Duration (ns)
tCS	nCS setup time to nWE	3
tCAS	nCS setup time to nADV	0
tAVSC	nADV setup time	3

For asynchronous single write access, write cycle time is  $WrCycleTime = WeOffTime + AccessCompletion = WeOffTime + 1$ . For the AccessCompletion, the GPMC requires one cycle of data hold time (nCS deassertion). For more information, see the device-specific Datasheet.

**Table 12-341. GPMC Timing Parameters for Asynchronous Single Write**

Parameter Name on GPMC Side	Formula	Duration (ns)	Number of Clock Cycles (F = 104 MHz)	GPMC Registers Configuration
ClkActivationTime		N/A (asynchronous mode)		
WdAccessTime	Applicable only to WAITMONITORING (the value is the same as for read access)			
PageBurstAccessTime		N/A (single access)		
WrCycleTime	WeOffTime + AccessCompletion	57.615	6	WRCYCLETIME = 0x6
CsOnTime	tCAS	0	0	CSONTIME = 0x0
CsWrOffTime	WeOffTime + 1	57.615	6	CSWROFFTIME = 0x6
AdvOnTime	tAVSC	3	1	ADVONTIME = 0x1
AdvWrOffTime	tAVSC + tAVDP	9	1	ADVWROFFTIME = 0x1
WeOnTime	tCS	3	1	WEONTIME = 0x1
WeOffTime	tCS + tWP + tWPH	48	5	WEOFFTIME = 0x5



gpmc-040

**Figure 12-277. Asynchronous Single Write Access (Timing Parameters in Clock Cycles)**

#### 12.3.4.4.12.2 How to Choose a Suitable Memory to Use With the GPMC

This section is intended to help the user select a suitable memory device to interface with the GPMC controller.

#### 12.3.4.4.12.2.1 Supported Memories or Devices

NAND flash and NOR flash architectures are the two flash technologies. The GPMC supports various types of external memory or devices, basically any one that supports NAND or NOR protocols:

- 8- and 16-bit-wide asynchronous or synchronous memory or device (only 8-bit: nonburst device)
- 16-bit address and data-multiplexed NOR flash devices (pSRAM, and so on)
- 8- and 16-bit NAND flash devices

##### 12.3.4.4.12.2.1.1 Memory Pin Multiplexing

This section describes the interfacing differences of the GPMC supported memories.

**Table 12-342. Supported Memory Interfaces**

Function	16-Bit Address/ Data-Multiplexed pSRAM or NOR Flash <sup>(1)</sup>	16-Bit NAND	8-Bit NAND
GPMC_A[22]			
GPMC_A[21]			
GPMC_A[20]			
GPMC_A[19]			
GPMC_A[18]			
GPMC_A[17]			
GPMC_A[16]			
GPMC_A[15]			
GPMC_A[14]			
GPMC_A[13]			
GPMC_A[12]			
GPMC_A[11]			
GPMC_A[10]	A26		
GPMC_A[9]	A25		
GPMC_A[8]	A24		
GPMC_A[7]	A23		
GPMC_A[6]	A22		
GPMC_A[5]	A21		
GPMC_A[4]	A20		
GPMC_A[3]	A19		
GPMC_A[2]	A18		
GPMC_A[1]	A17		
GPMC_AD[15]	D15 or A16	IO15	
GPMC_AD[14]	D14 or A15	IO14	
GPMC_AD[13]	D13 or A14	IO13	
GPMC_AD[12]	D12 or A13	IO12	
GPMC_AD[11]	D11 or A12	IO11	
GPMC_AD[10]	D10 or A11	IO10	
GPMC_AD[9]	D9 or A10	IO9	
GPMC_AD[8]	D8 or A9	IO8	
GPMC_AD[7]	D7 or A8		IO7
GPMC_AD[6]	D6 or A7		IO6
GPMC_AD[5]	D5 or A6		IO5
GPMC_AD[4]	D4 or A5		IO4
GPMC_AD[3]	D3 or A4		IO3

**Table 12-342. Supported Memory Interfaces (continued)**

Function	16-Bit Address/ Data-Multiplexed pSRAM or NOR Flash <sup>(1)</sup>	16-Bit NAND	8-Bit NAND
GPMC_AD[2]	D2 or A3		IO2
GPMC_AD[1]	D1 or A2		IO1
GPMC_AD[0]	D0 or A1		IO0
GPMC_CLKOUT	CLK		
GPMC_CSn0	nCS0 (chip-select)		nCE0 (chip-enable)
GPMC_CSn1	nCS1		nCE1
GPMC_CSn2	nCS2		nCE2
GPMC_CSn3	nCS3		nCE3
GPMC_ADVn_ALE	nADV (address valid)		ALE (address latch enable)
GPMC_OEn_REn	nOE (output enable)		nRE (read enable)
GPMC_WEn	nWE (Write enable)		nWE (write enable)
GPMC_BE0n_CLE	nBE0 (byte enable)		CLE (command latch enable)
GPMC_BE1n	nBE1		
GPMC_WAIT0	WAIT0		R/nB0 (ready/busy)
GPMC_WAIT1	WAIT1		R/nB1
GPMC_WPn	nWP (Write Protect)		nWP (Write Protect)

(1) Addresses seen from the device side. When interfacing to the external device, A1 is connected to the memory A0, A2 to the memory A1, and so on.

#### 12.3.4.4.12.2.1.2 NAND Interface Protocol

NAND flash architecture, introduced in 1989, is a flash technology. NAND is a page-oriented memory device; that is, read and write accesses are done by pages. NAND achieves great density by sharing common areas of the storage transistor, which creates strings of serially connected transistors (in NOR devices, each transistor stands alone). Because of its high density NAND is best suited to devices that require high capacity data storage, such as pictures, music, and data files. NAND nonvolatility makes of it a good storage solution for many applications where mobility, low power, and speed are key factors. Low pin count and simple interface are other advantages of NAND.

Table 12-343 summarizes the NAND interface signals level applied to external device or memories.

**Table 12-343. NAND Interface Bus Operations Summary**

Bus Operation	CLE	ALE	nCE	nWE <sup>(1)</sup>	nRE <sup>(1)</sup>	nWP
Read (cmd input)	H	L	L	RE	H	x
Read (add input)	L	H	L	RE	H	x
Write (cmd input)	H	L	L	RE	H	H
Write (add input)	L	H	L	RE	H	H
Data input	L	L	L	RE	H	H
Data output	L	L	L	H	FE	x
Busy (during read)	x	x	H <sup>(2)</sup>	H <sup>(2)</sup>	H <sup>(2)</sup>	x
Busy (during program)	x	x	x	x	x	H
Busy (during erase)	x	x	x	x	x	H
Write protect	x	x	x	x	x	L
Standby	x	x	H	x	x	H/L

(1) RE stands for rising edge; FE stands for falling edge

(2) Can be nCE high, or WE and nRE high.



#### 12.3.4.4.12.2.1.3 NOR Interface Protocol

NOR flash architecture, introduced in 1988, is a flash technology. Unlike NAND, which is a sequential access device, NOR is directly addressable; that is, it is designed to be a random access device. NOR is best suited to devices used to store and run code or firmware, usually in small capacities. While NOR has fast read capabilities, it also has slow write and erase functions when compared to the NAND architecture.

[Table 12-344](#) summarizes the level of the NOR interface signals applied to external devices or memories.

**Table 12-344. NOR Interface Bus Operations Summary**

Bus Operation	CLK	nADV	nCS	nOE	nWE	WAIT	DQ[15-0]
Read (asynchronous)	x	L	L	L	H	Asserted	Output
Read (synchronous)	Running	L	L	L	H	Driven	Output
Read (burst suspend)	Halted	x	L	H	H	Active	Output
Write	x	L	L	H	L	Asserted	Input
Output disable	x	x	L	H	H	Asserted	High-Z
Standby	x	x	H	x	x	High-Z	High-Z

#### 12.3.4.4.12.2.1.4 Other Technologies

Other supported device types interact with the GPMC through the NOR interface protocol.

FPGA (Field-Programmable Gate Array) is a low-power integrated circuit based on an array of programmable logic blocks.

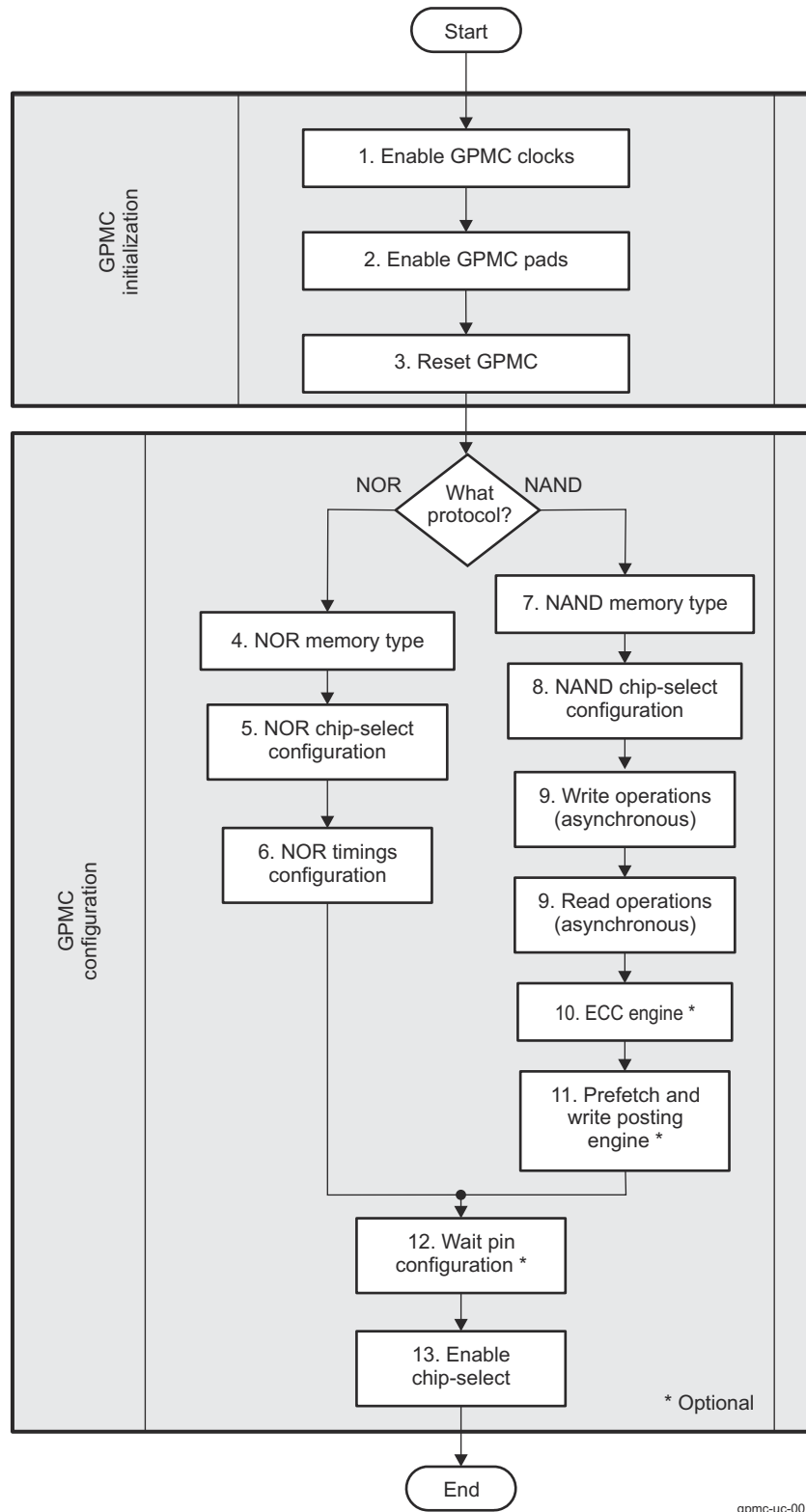
pSRAM (pseudo-static random access memory) is a low-power memory device. pSRAM is based on the DRAM cell with internal refresh and address control features, and interfaces as a synchronous NOR flash. It has synchronous write capability.

### 12.3.4.5 GPMC Basic Programming Model

#### 12.3.4.5.1 GPMC High-Level Programming Model Overview

The goal of the basic high-level programming model is to introduce a top-down approach to users that need to configure the GPMC module.

[Figure 12-278](#) and [Table 12-345](#) through [Table 12-346](#) show a programming model top-level diagram for the GPMC, and a description of each step. Each block of the diagram is described in one of the following sections through a set of registers to configure.



**Figure 12-278. Programming Model Top-Level Diagram**

**Table 12-345. GPMC Configuration in NOR Mode**

Step	Description
NOR Memory Type	See <a href="#">Table 12-347</a> .

**Table 12-345. GPMC Configuration in NOR Mode (continued)**

Step	Description
NOR Chip-Select Configuration	See <a href="#">Table 12-348</a> .
NOR Timings Configuration	See <a href="#">Table 12-349</a> .
WAIT Pin Configuration	See <a href="#">Table 12-357</a> .
Enable Chip-Select	See <a href="#">Table 12-358</a> .

**Table 12-346. GPMC Configuration in NAND Mode**

Step	Description
NAND Memory Type	See <a href="#">Table 12-352</a> .
NAND Chip-Select Configuration	See <a href="#">Table 12-353</a> .
Write Operations (Asynchronous)	See <a href="#">Table 12-354</a> .
Read Operations (Asynchronous)	See <a href="#">Table 12-354</a> .
ECC Engine	See <a href="#">Table 12-355</a> .
Prefetch and Write-Posting Engine	See <a href="#">Table 12-356</a> .
WAIT Pin Configuration	See <a href="#">Table 12-357</a> .
Enable Chip-Select	See <a href="#">Table 12-358</a> .

### 12.3.4.5.2 GPMC Initialization

GPMC can be reset via software bit in LPSC. For more information, see *Reset*.

### 12.3.4.5.3 GPMC Configuration in NOR Mode

This section gives a generic configuration for parameters related to the NOR memory connected to the GPMC.

**Table 12-347. NOR Memory Type**

Subprocess Name	Register / Bit Field	Value
Set the NOR protocol.	GPMC_CONFIG1_i[11-10] DEVICETYPE	0x0
Set a device size.	GPMC_CONFIG1_i[13-12] DEVICESIZE	x
Select an address and data multiplexing protocol.	GPMC_CONFIG1_i[9-8] MUXADDDATA	x
Set the attached device page length.	GPMC_CONFIG1_i[24-23] ATTACHEDDEVICEPAGELENGTH	x
Set the wrapping burst capabilities.	GPMC_CONFIG1_i[31] WRAPBURST	x
Select a timing signals latencies factor.	GPMC_CONFIG1_i[4] TIMEPARAGRANULARITY	x
Select an output clock frequency <sup>(1)</sup> .	GPMC_CONFIG1_i[1-0] GPMCFCLKDIVIDER	x
Choose an output clock activation time <sup>(1)</sup> .	GPMC_CONFIG1_i[26-25] CLKACTIVATIONTIME	x
Set a single or multiple access for read operations <sup>(1)</sup> .	GPMC_CONFIG1_i[30] READMULTIPLE	x
Set a synchronous or asynchronous mode for read operations.	GPMC_CONFIG1_i[29] READTYPE	x
Set a single or multiple access for write operations.	GPMC_CONFIG1_i[28] WRITEMULTIPLE	x
Set a synchronous or asynchronous mode for write operations.	GPMC_CONFIG1_i[27] WRITETYPE	x

(1) Applies only to synchronous configurations (or non-multiplexed asynchronous for multiple access one)

**Table 12-348. NOR Chip-Select Configuration**

Subprocess Name	Register/Bit Field	Value
Select the chip-select base address.	GPMC_CONFIG7_i[5-0] BASEADDRESS	x
Select the chip-select mask address.	GPMC_CONFIG7_i[11-8] MASKADDRESS	x

**Table 12-349. NOR Timings Configuration**

Subprocess Name	Register/Bit Field	Value
Configure adequate timing parameters in various memory modes.	See <a href="#">Section 12.3.4.5.6, GPMC Timing Parameters</a>	

**Table 12-350. WAIT Pin Configuration**

Subprocess Name	Register/Bit Field	Value
Enable or disable WAIT pin monitoring for read operations.	GPMC_CONFIG1_i[22] WAITREADMONITORING	x
Enable or disable WAIT pin monitoring for write operations.	GPMC_CONFIG1_i[21] WAITWRITEMONITORING	x
Select a WAIT pin monitoring time.	GPMC_CONFIG1_i[19-18] WAITMONITORINGTIME	x
Choose the input WAIT pin for the chip-select.	GPMC_CONFIG1_i[17-16] WAITPINSELECT	x

**Table 12-351. Enable Chip-Select**

Subprocess Name	Register/Bit Field	Value
When all parameters are configured, enable the chip-select.	GPMC_CONFIG7_i[6] CSVALID	x

#### 12.3.4.5.4 GPMC Configuration in NAND Mode

This section gives a generic configuration for parameters related to the NAND memory connected to the GPMC.

#### Note

Some of the GPMC features described in this section may not be supported on this family of devices. For more information, see *GPMC Not Supported Features*.

**Table 12-352. NAND Memory Type**

Subprocess Name	Register/Bit Field	Value
Set the NAND protocol.	GPMC_CONFIG1_i[11-10] DEVICETYPE	0x2
Set a device size.	GPMC_CONFIG1_i[13-12] DEVICESIZE	x
Set the address and data multiplexing protocol to non-multiplexed attached device.	GPMC_CONFIG1_i[9-8] MUXADDDATA	0x0
Select a timing signals latencies factor.	GPMC_CONFIG1_i[4] TIMEPARAGRANULARITY	x
Set a synchronous or asynchronous mode and a single or multiple access for read and write operations.	See <a href="#">Section 12.3.4.5.5, Set Memory Access</a> .	x

**Table 12-353. NAND Chip-Select Configuration**

Subprocess Name	Register/Bit Field	Value
Select the chip-select base address.	GPMC_CONFIG7_i[5-0] BASEADDRESS	x
Select the chip-select minimum granularity (16MB).	GPMC_CONFIG7_i[11-8] MASKADDRESS	x

**Table 12-354. Asynchronous Read and Write Operations**

Subprocess Name	Register/Bit Field	Value
Configure adequate timing parameters in asynchronous modes	See <a href="#">Section 12.3.4.5.6, GPMC Timing Parameters</a> .	

**Table 12-355. ECC Engine**

Subprocess Name	Register/Bit Field	Value
Select the ECC result register where the first ECC computation is stored (applies only to Hamming).	GPMC_ECC_CONTROL[3-0] ECCPOINTER	x <sup>(2)</sup>
Clear all ECC result registers.	GPMC_ECC_CONTROL[8] ECCCLEAR	Write 1 to clear.
Define ECCSIZE0 and ECCSIZE1.	GPMC_ECC_SIZE_CONFIG[21-12] ECCSIZE0 and [31-22] ECCSIZE1	x <sup>(1)</sup>
Select the size of each of the 9 result registers (size specified by ECCSIZE0 or ECCSIZE1).	GPMC_ECC_SIZE_CONFIG[j-1] ECCjRESULTSIZ where j = 1 to 9	x

**Table 12-355. ECC Engine (continued)**

Subprocess Name	Register/Bit Field	Value
Select the chip-select where ECC is computed.	GPMC_ECC_CONFIG[3-1] ECCCS	x
Select the Hamming code or BCH code ECC algorithm in use.	GPMC_ECC_CONFIG[16] ECCALGORITHM	x
Select word size for ECC calculation.	GPMC_ECC_CONFIG[7] ECC16B	x
If the BCH code is used, Set an error correction capability and Select a number of sectors to process.	GPMC_ECC_CONFIG[13-12] ECCBCHTSEL and GPMC_ECC_CONFIG[6-4] ECCTOPSECTOR	x
Enable the ECC computation.	GPMC_ECC_CONFIG[0] ECCENABLE	0x1

(1) Depends on the size of each sector in the NAND page

(2) This parameter depends on the numbers of sectors in a page.

**Table 12-356. Prefetch and Write-Posting Engine**

Subprocess Name	Register/Bit Field	Value
Disable the engine before configuration.	GPMC_PREFETCH_CONTROL[0] STARTENGINE	0x0
Select the chip-select associated with a NAND device where the prefetch engine is active.	GPMC_PREFETCH_CONFIG1[26-24] ENGINECSSELECTOR	x
Select access direction through prefetch engine, read or write.	GPMC_PREFETCH_CONFIG1[0] ACCESSMODE	x
Select the threshold used to issue an interrupt request.	GPMC_PREFETCH_CONFIG1[14-8] FIFOTHRESHOLD	x
Select interrupt synchronization mode.	GPMC_PREFETCH_CONFIG1[2] DMAMODE	x
Select if the engine immediately starts accessing the memory upon STARTENGINE assertion or if hardware synchronization based on a WAIT signal is used.	GPMC_PREFETCH_CONFIG1[3] SYNCHROMODE	x
Select which WAIT pin edge detector should start the engine in synchronized mode.	GPMC_PREFETCH_CONFIG1[5-4] WAITPINSELECTOR	x
Enter a number of clock cycles removed to timing parameters (for all back-to-back accesses to the NAND flash except the first one).	GPMC_PREFETCH_CONFIG1[30-28] CYCLEOPTIMIZATION	x
Enable the prefetch postwrite engine.	GPMC_PREFETCH_CONFIG1[7] ENABLEENGINE	0x1
Select the number of bytes to be read or written by the engine to the selected chip-select.	GPMC_PREFETCH_CONFIG2[13-0] TRANSFERCOUNT	x
Start the prefetch engine.	GPMC_PREFETCH_CONTROL[0] STARTENGINE	0x1

**Table 12-357. WAIT Pin Configuration**

Subprocess Name	Register/Bit Field	Value
Selects when the engine starts the access to chip-select.	GPMC_PREFETCH_CONFIG1[3] SYNCHROMODE	x
Select which WAIT pin edge detector should start the engine in synchronized mode.	GPMC_PREFETCH_CONFIG1[5-4] WAITPINSELECTOR	x

**Table 12-358. Enable Chip-Select**

Subprocess Name	Register/Bit Field	Value
When all parameters are configured, enable the chip-select.	GPMC_CONFIG7_i[6] CSVALID	x

#### 12.3.4.5.5 Set Memory Access

##### Note

Some of the GPMC features described in this section may not be supported on this family of devices. For more information, see *GPMC Not Supported Features*.

This section describes the bit field to configure to set the GPMC in various memory modes. [Table 12-359](#) and [Table 12-360](#) provide check lists for mode parameters and access type parameters, respectively.

**Table 12-359. Mode Parameters Check List**

Register	Bit	Name	Asynchronous				Synchronous			
			Single Read Access	Single Write Access	Multiple Read (Page) Access	Multiple Write (Page) Access	Single Read Access	Single Write Access	Multiple Read (Burst) Access	Multiple Write (Burst) Access
GPMC_CONFIG1_i	30	READMULTIPLE	0x0	Don't care	0x1 <sup>(1)</sup>	Not Supported	0x0	Don't care	0x1	Don't care
GPMC_CONFIG1_i	29	READTYPE	0x0	Don't care	0x0 <sup>(1)</sup>	Not Supported	0x1	Don't care	0x1	Don't care
GPMC_CONFIG1_i	28	WRITEMULTIPLE	Don't care	0x0	- <sup>(1)</sup>	Not Supported	Don't care	0x0	Don't care	0x1
GPMC_CONFIG1_i	27	WRITETYPE	Don't care	0x0	- <sup>(1)</sup>	Not Supported	Don't care	0x1	Don't care	0x1

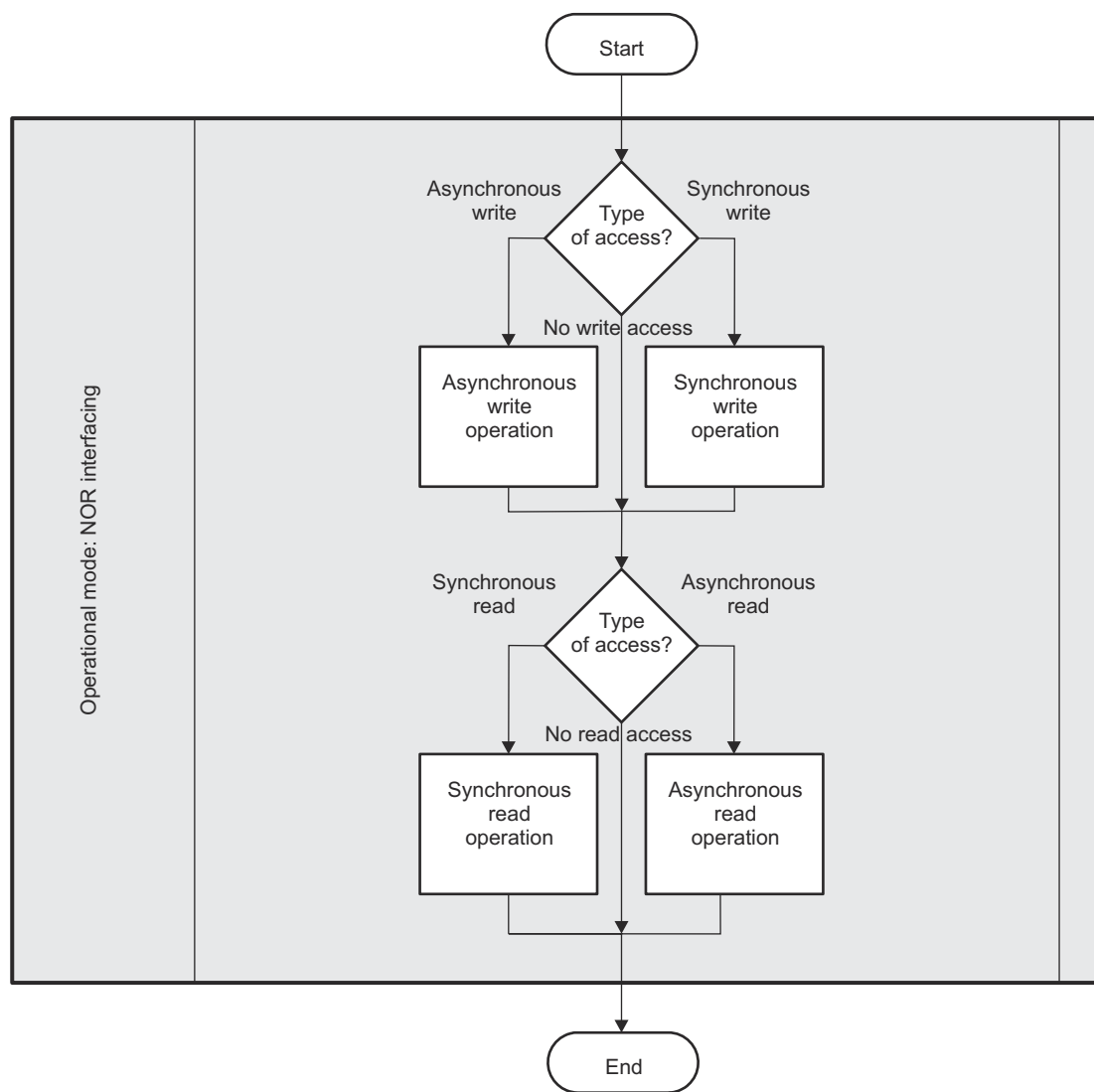
(1) Multiple read is not supported in address/data-multiplexed and AAD-multiplexed modes. Multiple read is supported in non-multiplexed mode.

**Table 12-360. Access Type Parameters Check List**

Register	Bit	Name	Access Type		
			non-multiplexed	Address/ Data-Multiplexed	AAD-Multiplexed
GPMC_CONFIG1_i	9-8	MUXADDDATA	0x0	0x2	0x1

#### 12.3.4.5.6 GPMC Timing Parameters

Figure 12-279 shows a programming model diagram for the NOR interfacing timing parameters.



gpmc-uc-002

**Figure 12-279. NOR Interfacing Timing Parameters Diagram**

Table 12-361 lists the bit fields to configure adequate timing parameters in various memory modes.

**Table 12-361. Timing Parameters**

Register	Bit	Name	Asynchronous			Synchronous				Non-multiplexed	Access/ Data-Multiplexed	Access Type
			Single Read Accesses	Single Write Accesses	Multiplexed Read (Page) Accesses	Single Read Accesses	Single Write Accesses	Multiplexed Read (Burst) Accesses	Multiplexed Write (Burst) Accesses			
GPMC_CONFIG1_i	9	MUXADDDATA	y	y	y	y	y	y	y	y	y	y
GPMC_CONFIG1_i	29	READTYPE	y		y	y		y		y	y	y
GPMC_CONFIG1_i	30	READMULTIPLE	y		y	y		y		y	y	y
GPMC_CONFIG1_i	27	WRITETYPE		y			y		y	y	y	y
GPMC_CONFIG1_i	28	WRITEMULTIPLE		y			y		y	y	y	y
GPMC_CONFIG1_i	31	WRAPBURST						y	y	y	y	y



**Table 12-361. Timing Parameters (continued)**

			Asynchronous			Synchronous				Access Type		
GPMC_CONFIG1_i	26-25	CLKACTIVATIONTIME				y	y	y	y	y	y	y
GPMC_CONFIG1_i	19-18	WAITMONITORINGTIME	y	y	y	y	y	y	y	y	y	y
GPMC_CONFIG1_i	4	TIMEPARAGRANULARITY	y	y	y	y	y	y	y	y	y	y
GPMC_CONFIG2_i	20-16	CSWROFFTIME		y			y		y	y	y	y
GPMC_CONFIG2_i	12-8	CSRDOFFTIME	y		y	y		y		y	y	y
GPMC_CONFIG2_i	7	CSEXTRADELAY	y	y	y	y	y	y	y	y	y	y
GPMC_CONFIG2_i	3-0	CSONTIME	y	y	y	y	y	y	y	y	y	y
GPMC_CONFIG3_i	30-28	ADVAADMUXWROFFTIME		y			y		y			y
GPMC_CONFIG3_i	30-29	ADVAADMUXRDOFFTIME	y		y	y		y				y
GPMC_CONFIG3_i	6-4	ADVAADMUXONTIME	y	y	y	y	y	y	y			y
GPMC_CONFIG3_i	20-16	ADVWROFFTIME		y			y		y	y	y	y
GPMC_CONFIG3_i	12-8	ADVRDOFFTIME	y		y	y		y		y	y	y
GPMC_CONFIG3_i	7	ADVEXTRADELAY	y	y	y	y	y	y	y	y	y	y
GPMC_CONFIG3_i	3-0	ADVONTIME	y	y	y	y	y	y	y	y	y	y
GPMC_CONFIG4_i	15-13	OEAADMUXOFFTIME	y	y	y	y	y	y	y			y
GPMC_CONFIG4_i	6-4	OEAADMUXONTIME	y	y	y	y	y	y	y			y
GPMC_CONFIG4_i	28-24	WEOFFTIME		y			y		y	y	y	y
GPMC_CONFIG4_i	23	WEEXTRADELAY		y			y		y	y	y	y
GPMC_CONFIG4_i	19-16	WEONTIME		y			y		y	y	y	y
GPMC_CONFIG4_i	12-8	OEOFFTIME	y		y	y		y		y	y	y
GPMC_CONFIG4_i	7	OEEXTRADELAY	y		y	y		y		y	y	y
GPMC_CONFIG4_i	3-0	OEONTIME	y		y	y		y		y	y	y
GPMC_CONFIG5_i	27-24	PAGEBURSTACCESSTIME			y			y	y	y	y	y
GPMC_CONFIG5_i	20-16	RDACCESSTIME	y		y	y		y		y	y	y
GPMC_CONFIG5_i	12-8	WRCYCLETIME		y			y		y	y	y	y
GPMC_CONFIG5_i	4-0	RDCYCLETIME	y		y	y		y		y	y	y
GPMC_CONFIG6_i	28-24	WRACCESSTIME		y			y		y	y	y	y
GPMC_CONFIG6_i	19-16	WRDATAONADMUXBUS		y			y		y		y	y
GPMC_CONFIG6_i	11-8	CYCLE2CYCLEDELAY	y	y	y	y	y	y	y	y	y	y
GPMC_CONFIG6_i	7	CYCLE2CYCLESAMECSSEN	y	y	y	y	y	y	y	y	y	y
GPMC_CONFIG6_i	6	CYCLE2CYCLEDIFFCSSEN	y	y	y	y	y	y	y	y	y	y
GPMC_CONFIG6_i	3-0	BUSTURNAROUND	y	y	y	y	y	y	y	y	y	y
GPMC_CONFIG7_i	6	CSVALID	y	y	y	y	y	y	y	y	y	y

### 12.3.4.5.6.1 GPMC Timing Parameters Formulas

This section is intended to help the user calculate the GPMC timing bit field values. Formulas are not listed exhaustively.

The section describes:

- NAND flash interface timing parameters formulas
- Synchronous NOR flash timing parameters formulas
- Asynchronous NOR flash timing parameters formulas

For complete information, such as OPP and board effects on timings, see the device-specific Datasheet.

#### 12.3.4.5.6.1.1 NAND Flash Interface Timing Parameters Formulas

This section lists formulas to calculate NAND timing parameters. This is the case when GPMC\_CONFIG1\_i[11-10] DEVICETYPE = 0x2. [Table 12-362](#) describes the NAND timing parameters.

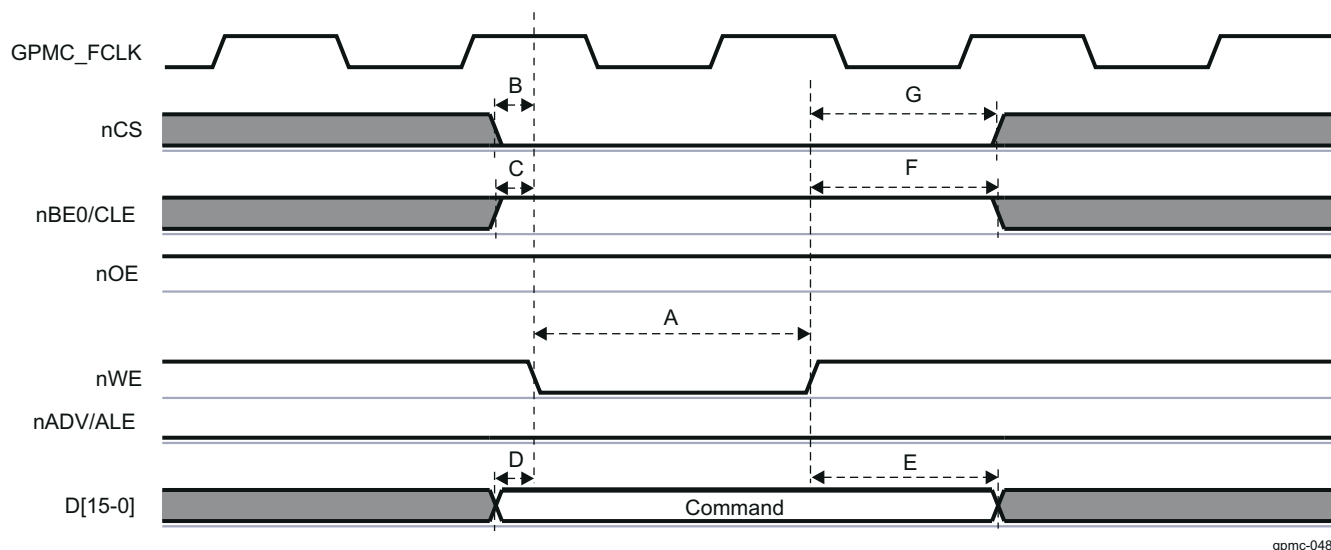
**Table 12-362. NAND Formulas Description**

Configuration Parameter	Unit	Description
A	ns	Pulse duration – GPMC_WEn valid time
B	ns	Delay time – GPMC_CS valid to GPMC_WEn valid
C	ns	Delay time – GPMC_BE0n_CLE/GPMC_ADVn_ALE high to GPMC_WEn valid
D	ns	Delay time – GPMC_AD[15-0] valid to GPMC_WEn valid
E	ns	Delay time – GPMC_WEn invalid to GPMC_AD[15-0] invalid
F	ns	Delay time – GPMC_WEn invalid to GPMC_BE0n_CLE/GPMC_ADVn_ALE invalid
G	ns	Delay time – GPMC_WEn invalid to GPMC_CS invalid
H	ns	Cycle time – Write cycle time
I	ns	Delay time – GPMC_CS valid to GPMC_OEn_REn valid
J	ns	Setup time – GPMC_AD[15-0] valid to GPMC_OEn_REn invalid
K	ns	Pulse duration – GPMC_OEn_REn valid time
L	ns	Cycle time – Read cycle time
M	ns	Delay time – GPMC_OEn_REn invalid to GPMC_CS invalid

The configuration parameters are calculated through the following formulas. For more information, see the device-specific Datasheet.

$$\begin{aligned}
 A &= (\text{WEOffTime} - \text{WEOntime}) * (\text{TimeParaGranularity} + 1) * \text{GPMC\_FCLK period} \\
 B &= ((\text{WEOntime} - \text{CSOnTime}) * (\text{TimeParaGranularity} + 1) + 0.5 * (\text{WEEExtraDelay} - \text{CSEExtraDelay})) * \text{GPMC\_FCLK period} \\
 C &= ((\text{WEOntime} - \text{ADVOnTime}) * (\text{TimeParaGranularity} + 1) + 0.5 * (\text{WEEExtraDelay} - \text{ADVExtraDelay})) * \text{GPMC\_FCLK period} \\
 D &= (\text{WEOntime} * (\text{TimeParaGranularity} + 1) + 0.5 * \text{WEEExtraDelay}) * \text{GPMC\_FCLK period} \\
 E &= (\text{WrCycleTime} - \text{WEOffTime} * (\text{TimeParaGranularity} + 1) - 0.5 * \text{WEEExtraDelay}) * \text{GPMC\_FCLK period} \\
 F &= (\text{ADVWrOffTime} - \text{WEOffTime} * (\text{TimeParaGranularity} + 1) + 0.5 * (\text{ADVExtraDelay} - \text{WEEExtraDelay})) * \text{GPMC\_FCLK period} \\
 G &= (\text{CSWrOffTime} - \text{WEOffTime} * (\text{TimeParaGranularity} + 1) + 0.5 * (\text{CSEExtraDelay} - \text{WEEExtraDelay})) * \text{GPMC\_FCLK period} \\
 H &= \text{WrCycleTime} * (1 + \text{TimeParaGranularity}) * \text{GPMC\_FCLK period} \\
 I &= ((\text{OEOnTime} - \text{CSOnTime}) * (\text{TimeParaGranularity} + 1) + 0.5 * (\text{OEEExtraDelay} - \text{CSEExtraDelay})) * \text{GPMC\_FCLK period} \\
 J &= ((\text{RdAccessTime} - \text{OEOffTime}) * (\text{TimeParaGranularity} + 1) - 0.5 * \text{OEEExtraDelay}) * \text{GPMC\_FCLK period} \\
 K &= (\text{OEOffTime} - \text{OEOnTime}) * (1 + \text{TimeParaGranularity}) * \text{GPMC\_FCLK period} \\
 L &= \text{RdCycleTime} * (1 + \text{TimeParaGranularity}) * \text{GPMC\_FCLK period} \\
 M &= (\text{CSRdOffTime} - \text{OEOffTime} * (\text{TimeParaGranularity} + 1) + 0.5 * (\text{CSEExtraDelay} - \text{OEEExtraDelay})) * \text{GPMC\_FCLK period}
 \end{aligned}$$

[Figure 12-280](#) shows a simplified example of command latch cycle timing where formulas are associated with signal waves.



**Figure 12-280. NAND Command Latch Cycle Timing Simplified Example**

#### 12.3.4.5.6.1.2 Synchronous NOR Flash Timing Parameters Formulas

This section lists all formulas to calculate synchronous NOR timing parameters. This is the case when GPMC\_CONFIG1\_i[11-10] DEVICETYPE = 0x0 and when READTYPE or WRITETYPE are set to synchronous mode. [Table 12-363](#) describes the synchronous NOR formulas.

**Table 12-363. Synchronous NOR Formulas Description**

Configuration Parameter	Unit	Description
A	ns	Pulse duration – nCS low
B	ns	Delay time – address bus valid to CLK first edge Delay time – nBE0 / nBE1 valid to CLK first edge
C	ns	Pulse duration – nBE0 / nBE1 low
D	ns	Delay time – CLK rising edge to nBE0 / nBE1 invalid Delay time – CLK rising edge to nADV/ALE invalid
E	ns	Delay time – CLK rising edge to nCS invalid Delay time – CLK rising edge to nOE/nRE invalid
F	ns	Delay time – CLK rising edge to nCS transition
G	ns	Delay time – CLK rising edge to nADV/ALE transition
H	ns	Delay time – CLK rising edge to nOE/nRE transition
I	ns	Delay time – CLK rising edge to nWE transition
J	ns	Delay time – CLK rising edge to A[16-1]/D[15-0] data bus transition Delay time – CLK rising edge to nBE0 / nBE1 transition
K	ns	Pulse duration – nADV/ALE low
L	ns	Delay time – WAIT invalid to first data latching CLK edge

The configuration parameters are calculated through the following formulas. For more information, see the device-specific Datasheet.

1. For single read accesses:

$$A = (\text{CSRDOFFTIME} - \text{CSONTIME}) * (\text{TIMEPARAGRANULARITY} + 1) * \text{GPMC\_FCLK period}$$

$$C = \text{RDCYCLETIME} * (\text{TIMEPARAGRANULARITY} + 1) * \text{GPMC\_FCLK period}$$

$$D = (\text{RDCYCLETIME} - \text{RDACCESSTIME}) * \text{GPMC\_FCLK period}$$

$$E = (\text{CSRDOFFTIME} - \text{RDACCESSTIME}) * \text{GPMC\_FCLK period}$$

2. For burst read accesses (where n is the page burst access number):
 
$$A = (CSRDOFFTIME - CSONTIME + (n - 1) * PAGEBURSTACCESSTIME) * (TIMEPARAGRANULARITY + 1) * GPMC\_FCLK \text{ period}$$

$$C = (RDCYCLETIME + (n - 1) * PAGEBURSTACCESSTIME) * (TIMEPARAGRANULARITY + 1) * GPMC\_FCLK \text{ period}$$

$$D = (RDCYCLETIME - (RDACCESSTIME + (n - 1) * PAGEBURSTACCESSTIME)) * GPMC\_FCLK \text{ period}$$

$$E = (CSRDOFFTIME - (RDACCESSTIME + (n - 1) * PAGEBURSTACCESSTIME)) * GPMC\_FCLK \text{ period}$$
3. For burst write accesses (where n is the page burst access number):
 
$$A = (CSWROFFTIME - CSONTIME + (n - 1) * PAGEBURSTACCESSTIME) * (TIMEPARAGRANULARITY + 1) * GPMC\_FCLK \text{ period}$$

$$C = (WRCYCLETIME + (n - 1) * PAGEBURSTACCESSTIME) * (TIMEPARAGRANULARITY + 1) * GPMC\_FCLK \text{ period}$$

$$D = (WRCYCLETIME - (RDACCESSTIME + (n - 1) * PAGEBURSTACCESSTIME)) * GPMC\_FCLK \text{ period}$$

$$E = (CSWROFFTIME - (RDACCESSTIME + (n - 1) * PAGEBURSTACCESSTIME)) * GPMC\_FCLK \text{ period}$$
4. For all accesses:
 

For nCS falling edge (chip-select activated):

  - Case where GPMC\_CONFIG1\_i[1-0] GPMCFCLKDIVIDER = 0x0:
 
$$F = 0.5 * CSEXTRADELAY * GPMC\_FCLK \text{ period}$$
  - Case where GPMCFCLKDIVIDER = 0x1:
 
$$F = 0.5 * CSEXTRADELAY * GPMC\_FCLK \text{ period, when (CLKACTIVATIONTIME and CSONTIME are odd) or (CLKACTIVATIONTIME and CSONTIME are even)}$$

$$F = (1 + 0.5 * CSEXTRADELAY) * GPMC\_FCLK \text{ period otherwise.}$$
  - Case where GPMCFCLKDIVIDER = 0x2:
 
$$F = 0.5 * CSEXTRADELAY * GPMC\_FCLK \text{ period, when (CSONTIME - CLKACTIVATIONTIME) is a multiple of 3}$$

$$F = (1 + 0.5 * CSEXTRADELAY) * GPMC\_FCLK \text{ period, when (CSONTIME - CLKACTIVATIONTIME - 1) is a multiple of 3}$$

$$F = (2 + 0.5 * CSEXTRADELAY) * GPMC\_FCLK \text{ period, when (CSONTIME - CLKACTIVATIONTIME - 2) is a multiple of 3}$$

For nCS rising edge (chip-select deactivated) in reading mode:

  - Case where GPMC\_CONFIG1\_i[1-0] GPMCFCLKDIVIDER = 0x0:
 
$$F = 0.5 * CSEXTRADELAY * GPMC\_FCLK \text{ period}$$
  - Case where GPMCFCLKDIVIDER = 0x1:
 
$$F = 0.5 * CSEXTRADELAY * GPMC\_FCLK \text{ period, when (CLKACTIVATIONTIME and CSRDOFFTIME are odd) or (CLKACTIVATIONTIME and CSRDOFFTIME are even)}$$

$$F = (1 + 0.5 * CSEXTRADELAY) * GPMC\_FCLK \text{ period otherwise.}$$
  - Case where GPMCFCLKDIVIDER = 0x2:
 
$$F = 0.5 * CSEXTRADELAY * GPMC\_FCLK \text{ period, when (CSRDOFFTIME - CLKACTIVATIONTIME) is a multiple of 3}$$

$$F = (1 + 0.5 * CSEXTRADELAY) * GPMC\_FCLK \text{ period, when (CSRDOFFTIME - CLKACTIVATIONTIME - 1) is a multiple of 3}$$

$$F = (2 + 0.5 * CSEXTRADELAY) * GPMC\_FCLK \text{ period, when (CSRDOFFTIME - CLKACTIVATIONTIME - 2) is a multiple of 3}$$

For nCS rising edge (chip-select deactivated) in writing mode:

  - Case where GPMC\_CONFIG1\_i[1-0] GPMCFCLKDIVIDER = 0x0:
 
$$F = 0.5 * CSEXTRADELAY * GPMC\_FCLK \text{ period}$$
  - Case where GPMCFCLKDIVIDER = 0x1:
 
$$F = 0.5 * CSEXTRADELAY * GPMC\_FCLK \text{ period, when (CLKACTIVATIONTIME and CSWROFFTIME are odd) or (CLKACTIVATIONTIME and CSWROFFTIME are even)}$$

$$F = (1 + 0.5 * CSEXTRADELAY) * GPMC\_FCLK \text{ period otherwise.}$$
  - Case where GPMCFCLKDIVIDER = 0x2:
 
$$F = 0.5 * CSEXTRADELAY * GPMC\_FCLK \text{ period, when (CSWROFFTIME - CLKACTIVATIONTIME) is a multiple of 3}$$

$F = (1 + 0.5 * CSEXTRADELAY) * GPMC\_FCLK \text{ period, when } (CSWROFFTIME - CLKACTIVATIONTIME - 1) \text{ is a multiple of } 3$   
 $F = (2 + 0.5 * CSEXTRADELAY) * GPMC\_FCLK \text{ period, when } (CSWROFFTIME - CLKACTIVATIONTIME - 2) \text{ is a multiple of } 3$

For nADV falling edge (nADV activated):

- Case where GPMC\_CONFIG1\_i[1-0] GPMCFCLKDIVIDER = 0x0:  
 $G = 0.5 * ADVEXTRADELAY * GPMC\_FCLK \text{ period}$
- Case where GPMCFCLKDIVIDER = 0x1:  
 $G = 0.5 * ADVEXTRADELAY * GPMC\_FCLK \text{ period, when } (CLKACTIVATIONTIME \text{ and } ADVONTIME \text{ are odd}) \text{ or } (CLKACTIVATIONTIME \text{ and } ADVONTIME \text{ are even})$   
 $G = (1 + 0.5 * ADVEXTRADELAY) * GPMC\_FCLK \text{ period otherwise.}$
- Case where GPMCFCLKDIVIDER = 0x2:  
 $G = 0.5 * ADVEXTRADELAY * GPMC\_FCLK \text{ period, when } (ADVONTIME - CLKACTIVATIONTIME) \text{ is a multiple of } 3$   
 $G = (1 + 0.5 * ADVEXTRADELAY) * GPMC\_FCLK \text{ period, when } (ADVONTIME - CLKACTIVATIONTIME - 1) \text{ is a multiple of } 3$   
 $G = (2 + 0.5 * ADVEXTRADELAY) * GPMC\_FCLK \text{ period, when } (ADVONTIME - CLKACTIVATIONTIME - 2) \text{ is a multiple of } 3$

For nADV rising edge (nADV deactivated) in reading mode:

- Case where GPMC\_CONFIG1\_i[1-0] GPMCFCLKDIVIDER = 0x0:  
 $G = 0.5 * ADVEXTRADELAY * GPMC\_FCLK \text{ period}$
- Case where GPMCFCLKDIVIDER = 0x1:  
 $G = 0.5 * ADVEXTRADELAY * GPMC\_FCLK \text{ period, when } (CLKACTIVATIONTIME \text{ and } ADVRDOFFTIME \text{ are odd}) \text{ or } (CLKACTIVATIONTIME \text{ and } ADVRDOFFTIME \text{ are even})$   
 $G = (1 + 0.5 * ADVEXTRADELAY) * GPMC\_FCLK \text{ period otherwise.}$
- Case where GPMCFCLKDIVIDER = 0x2:  
 $G = 0.5 * ADVEXTRADELAY * GPMC\_FCLK \text{ period, when } (ADVRDOFFTIME - CLKACTIVATIONTIME) \text{ is a multiple of } 3$   
 $G = (1 + 0.5 * ADVEXTRADELAY) * GPMC\_FCLK \text{ period, when } (ADVRDOFFTIME - CLKACTIVATIONTIME - 1) \text{ is a multiple of } 3$   
 $G = (2 + 0.5 * ADVEXTRADELAY) * GPMC\_FCLK \text{ period, when } (ADVRDOFFTIME - CLKACTIVATIONTIME - 2) \text{ is a multiple of } 3$

For nADV rising edge (nADV deactivated) in writing mode:

- Case where GPMC\_CONFIG1\_i[1-0] GPMCFCLKDIVIDER = 0x0:  
 $G = 0.5 * ADVEXTRADELAY * GPMC\_FCLK \text{ period}$
- Case where GPMCFCLKDIVIDER = 0x1:  
 $G = 0.5 * ADVEXTRADELAY * GPMC\_FCLK \text{ period, when } (CLKACTIVATIONTIME \text{ and } ADVWROFFTIME \text{ are odd}) \text{ or } (CLKACTIVATIONTIME \text{ and } ADVWROFFTIME \text{ are even})$   
 $G = (1 + 0.5 * ADVEXTRADELAY) * GPMC\_FCLK \text{ period otherwise.}$
- Case where GPMCFCLKDIVIDER = 0x2:  
 $G = 0.5 * ADVEXTRADELAY * GPMC\_FCLK \text{ period, when } (ADVWROFFTIME - CLKACTIVATIONTIME) \text{ is a multiple of } 3$   
 $G = (1 + 0.5 * ADVEXTRADELAY) * GPMC\_FCLK \text{ period, when } (ADVWROFFTIME - CLKACTIVATIONTIME - 1) \text{ is a multiple of } 3$   
 $G = (2 + 0.5 * ADVEXTRADELAY) * GPMC\_FCLK \text{ period, when } (ADVWROFFTIME - CLKACTIVATIONTIME - 2) \text{ is a multiple of } 3$

For nOE falling edge (nOE activated):

- Case where GPMC\_CONFIG1\_i[1-0] GPMCFCLKDIVIDER = 0x0:  
 $H = 0.5 * OEEXTRADELAY * GPMC\_FCLK \text{ period}$
- Case where GPMCFCLKDIVIDER = 0x1:  
 $H = 0.5 * OEEXTRADELAY * GPMC\_FCLK \text{ period, when } (CLKACTIVATIONTIME \text{ and } OEONTIME \text{ are odd}) \text{ or } (CLKACTIVATIONTIME \text{ and } OEONTIME \text{ are even})$

- $$H = (1 + 0.5 * OEEXTRADELAY) * GPMC\_FCLK \text{ period otherwise.}$$
- Case where  $GPMCFCLKDIVIDER = 0x2$ :  
 $H = 0.5 * OEEXTRADELAY * GPMC\_FCLK \text{ period, when } (OEONTIME - CLKACTIVATIONTIME) \text{ is a multiple of } 3$   
 $H = (1 + 0.5 * OEEXTRADELAY) * GPMC\_FCLK \text{ period, when } (OEONTIME - CLKACTIVATIONTIME - 1) \text{ is a multiple of } 3$   
 $H = (2 + 0.5 * OEEXTRADELAY) * GPMC\_FCLK \text{ period, when } (OEONTIME - CLKACTIVATIONTIME - 2) \text{ is a multiple of } 3$

For nOE rising edge (nOE deactivated):

- Case where  $GPMC\_CONFIG1\_i[1-0] GPMCFCLKDIVIDER = 0x0$ :  
 $H = 0.5 * OEEXTRADELAY * GPMC\_FCLK \text{ period}$
- Case where  $GPMCFCLKDIVIDER = 0x1$ :  
 $H = 0.5 * OEEXTRADELAY * GPMC\_FCLK \text{ period, when } (CLKACTIVATIONTIME \text{ and } OEOFFTIME \text{ are odd}) \text{ or } (CLKACTIVATIONTIME \text{ and } OEOFFTIME \text{ are even})$   
 $H = (1 + 0.5 * OEEXTRADELAY) * GPMC\_FCLK \text{ period otherwise.}$
- Case where  $GPMCFCLKDIVIDER = 0x2$ :  
 $H = 0.5 * OEEXTRADELAY * GPMC\_FCLK \text{ period, when } (OEOFFTIME - CLKACTIVATIONTIME) \text{ is a multiple of } 3$   
 $H = (1 + 0.5 * OEEXTRADELAY) * GPMC\_FCLK \text{ period, when } (OEOFFTIME - CLKACTIVATIONTIME - 1) \text{ is a multiple of } 3$   
 $H = (2 + 0.5 * OEEXTRADELAY) * GPMC\_FCLK \text{ period, when } (OEOFFTIME - CLKACTIVATIONTIME - 2) \text{ is a multiple of } 3$

For nWE falling edge (nWE activated):

- Case where  $GPMC\_CONFIG1\_i[1-0] GPMCFCLKDIVIDER = 0x0$ :  
 $I = 0.5 * WEEXTRADELAY * GPMC\_FCLK \text{ period}$
- Case where  $GPMCFCLKDIVIDER = 0x1$ :  
 $I = 0.5 * WEEXTRADELAY * GPMC\_FCLK \text{ period, when } (CLKACTIVATIONTIME \text{ and } WEONTIME \text{ are odd}) \text{ or } (CLKACTIVATIONTIME \text{ and } WEONTIME \text{ are even})$   
 $I = (1 + 0.5 * WEEXTRADELAY) * GPMC\_FCLK \text{ period otherwise.}$
- Case where  $GPMCFCLKDIVIDER = 0x2$ :  
 $I = 0.5 * WEEXTRADELAY * GPMC\_FCLK \text{ period, when } (WEONTIME - CLKACTIVATIONTIME) \text{ is a multiple of } 3$   
 $I = (1 + 0.5 * WEEXTRADELAY) * GPMC\_FCLK \text{ period, when } (WEONTIME - CLKACTIVATIONTIME - 1) \text{ is a multiple of } 3$   
 $I = (2 + 0.5 * WEEXTRADELAY) * GPMC\_FCLK \text{ period, when } (WEONTIME - CLKACTIVATIONTIME - 2) \text{ is a multiple of } 3$

For nWE rising edge (nWE deactivated):

- Case where  $GPMC\_CONFIG1\_i[1-0] GPMCFCLKDIVIDER = 0x0$ :  
 $I = 0.5 * WEEXTRADELAY * GPMC\_FCLK \text{ period}$
- Case where  $GPMCFCLKDIVIDER = 0x1$ :  
 $I = 0.5 * WEEXTRADELAY * GPMC\_FCLK \text{ period, when } (CLKACTIVATIONTIME \text{ and } WEOFFTIME \text{ are odd}) \text{ or } (CLKACTIVATIONTIME \text{ and } WEOFFTIME \text{ are even})$   
 $I = (1 + 0.5 * WEEXTRADELAY) * GPMC\_FCLK \text{ period otherwise.}$
- Case where  $GPMCFCLKDIVIDER = 0x2$ :  
 $I = 0.5 * WEEXTRADELAY * GPMC\_FCLK \text{ period, when } (WEOFFTIME - CLKACTIVATIONTIME) \text{ is a multiple of } 3$   
 $I = (1 + 0.5 * WEEXTRADELAY) * GPMC\_FCLK \text{ period, when } (WEOFFTIME - CLKACTIVATIONTIME - 1) \text{ is a multiple of } 3$   
 $I = (2 + 0.5 * WEEXTRADELAY) * GPMC\_FCLK \text{ period, when } (WEOFFTIME - CLKACTIVATIONTIME - 2) \text{ is a multiple of } 3$

For nADV low pulse duration:

- Read operation:

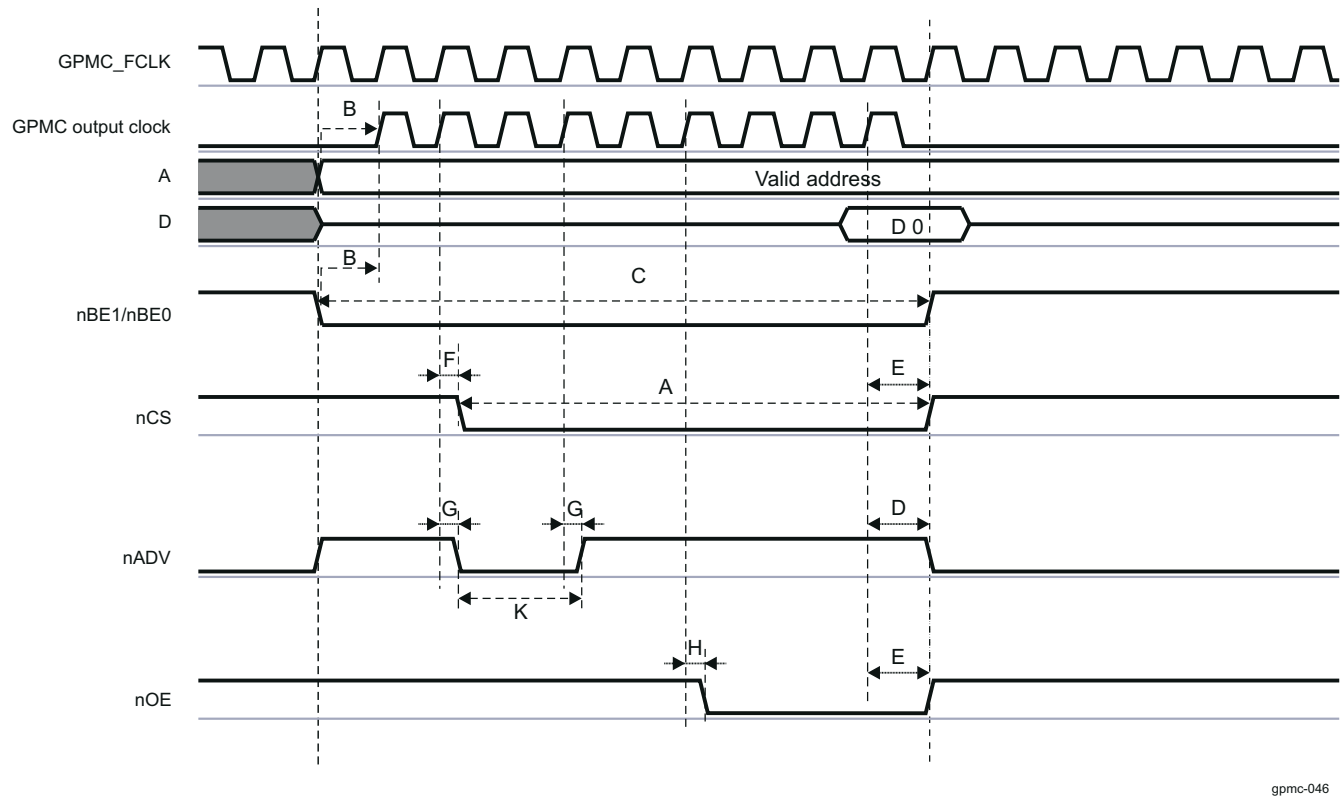


- $K = (ADVRDOFFTIME - ADVONTIME) * (TIMEPARAGRANULARITY + 1) * GPMC\_FCLK \text{ period}$
- Write operation:  
 $K = (ADVWROFFTIME - ADVONTIME) * (TIMEPARAGRANULARITY + 1) * GPMC\_FCLK \text{ period}$

For WAIT invalid to first data latching GPMC output clock edge:

- $L = WAITMONITORINGTIME * (GPMCFCLKDIVIDER + 1) * GPMC\_FCLK \text{ period} + GPMC \text{ output clock period}$

Figure 12-281 shows a simplified example of a synchronous NOR single read where formulas are associated with signal waves.



gpmc-046

**Figure 12-281. Synchronous NOR Single Read Simplified Example**

#### 12.3.4.5.6.1.3 Asynchronous NOR Flash Timing Parameters Formulas

This section lists all the formulas to calculate asynchronous NOR timing parameters. This is the case when GPMC\_CONFIG1\_i[11-10] DEVICETYPE = 0x0 and when READTYPE or WRITETYPE are set to asynchronous mode. Table 12-364 describes the asynchronous NOR formulas.

**Table 12-364. Asynchronous NOR Formulas Description**

Configuration Parameter	Unit	Description
A	ns	Pulse duration – nCS low
B	ns	Delay time – nCS valid to nADV/ALE invalid
C	ns	Delay time – nCS valid to nOE/nRE invalid (single read)
D	ns	Pulse duration – address bus valid - 2nd, 3rd and 4th accesses
E	ns	Delay time – nCS valid to nWE valid
F	ns	Delay time – nCS valid to nWE invalid
G	ns	Address invalid duration between two successive R/W accesses
H	ns	Setup time – read data valid before nOE/nRE high

**Table 12-364. Asynchronous NOR Formulas Description (continued)**

Configuration Parameter	Unit	Description
I	ns	Delay time – nCS valid to nOE/nRE invalid (burst read)
J	ns	Delay time – address bus valid to nCS valid
		Delay time – data bus valid to nCS valid
		Delay time – nBE0 / nBE1 valid to nCS valid
K	ns	Delay time – nCS valid to nADV/ALE valid
L	ns	Delay time – nCS valid to nOE/nRE valid
M	ns	Delay time – nCS valid to first data latching edge
N	ns	Pulse duration – nBE0 / nBE1 valid time
O	ns	Delay time – nCS valid to nADV/ALE valid

The configuration parameters are calculated through the following formulas. These formulas are not exhaustive. For more information, see the device-specific Datasheet.

- nCS low pulse:  
For single read:  $A = (\text{CSRDOFFTIME} - \text{CSONTIME}) * (\text{TIMEPARAGRANULARITY} + 1) * \text{GPMC\_FCLK period}$   
For burst read:  $A = (\text{CSRDOFFTIME} - \text{CSONTIME} + (N - 1) * \text{PAGEBURSTACCESSTIME}) * (\text{TIMEPARAGRANULARITY} + 1) * \text{GPMC\_FCLK period}$ , where N = page burst access number  
For single write:  $A = (\text{CSWROFFTIME} - \text{CSONTIME}) * (\text{TIMEPARAGRANULARITY} + 1) * \text{GPMC\_FCLK period}$   
For burst write:  $A = (\text{CSWROFFTIME} - \text{CSONTIME} + (N - 1) * \text{PAGEBURSTACCESSTIME}) * (\text{TIMEPARAGRANULARITY} + 1) * \text{GPMC\_FCLK period}$ , where N = page burst access number
- nCS valid to nADV/ALE invalid delay:  
For reading:  $B = ((\text{ADVROFFTIME} - \text{CSONTIME}) * (\text{TIMEPARAGRANULARITY} + 1) + 0.5 * (\text{ADVEXTRADELAY} - \text{CSEXTRADELAY})) * \text{GPMC\_FCLK period}$   
For writing:  $B = ((\text{ADVWROFFTIME} - \text{CSONTIME}) * (\text{TIMEPARAGRANULARITY} + 1) + 0.5 * (\text{ADVEXTRADELAY} - \text{CSEXTRADELAY})) * \text{GPMC\_FCLK period}$
- $C = ((\text{OEOFFTIME} - \text{CSONTIME}) * (\text{TIMEPARAGRANULARITY} + 1) + 0.5 * (\text{OEEXTRADELAY} - \text{CSEXTRADELAY})) * \text{GPMC\_FCLK period}$
- $D = \text{PAGEBURSTACCESSTIME} * (\text{TIMEPARAGRANULARITY} + 1) * \text{GPMC\_FCLK period}$
- $E = ((\text{WEONTIME} - \text{CSONTIME}) * (\text{TIMEPARAGRANULARITY} + 1) + 0.5 * (\text{WEEXTRADELAY} - \text{CSEXTRADELAY})) * \text{GPMC\_FCLK period}$
- $F = ((\text{WEOFFTIME} - \text{CSONTIME}) * (\text{TIMEPARAGRANULARITY} + 1) + 0.5 * (\text{WEEXTRADELAY} - \text{CSEXTRADELAY})) * \text{GPMC\_FCLK period}$
- $G = \text{CYCLE2CYCLEDELAY} * \text{GPMC\_FCLK period}$
- $H = ((\text{OEOFFTIME} - \text{RDACCESSTIME}) * (\text{TIMEPARAGRANULARITY} + 1) + 0.5 * \text{OEEXTRADELAY}) * \text{GPMC\_FCLK period}$
- $I = ((\text{OEOFFTIME} + (N - 1) * \text{PAGEBURSTACCESSTIME} - \text{CSONTIME}) * (\text{TIMEPARAGRANULARITY} + 1) + 0.5 * (\text{OEEXTRADELAY} - \text{CSEXTRADELAY})) * \text{GPMC\_FCLK period}$ , where N = page burst access number
- $J = (\text{CSONTIME} * (\text{TIMEPARAGRANULARITY} + 1) + 0.5 * \text{CSEXTRADELAY}) * \text{GPMC\_FCLK period}$
- $K = ((\text{ADVONTIME} - \text{CSONTIME}) * (\text{TIMEPARAGRANULARITY} + 1) + 0.5 * (\text{ADVEXTRADELAY} - \text{CSEXTRADELAY})) * \text{GPMC\_FCLK period}$
- $L = ((\text{OEONTIME} - \text{CSONTIME}) * (\text{TIMEPARAGRANULARITY} + 1) + 0.5 * (\text{OEEXTRADELAY} - \text{CSEXTRADELAY})) * \text{GPMC\_FCLK period}$
- $M = ((\text{RDACCESSTIME} - \text{CSONTIME}) * (\text{TIMEPARAGRANULARITY} + 1) - 0.5 * \text{CSEXTRADELAY}) * \text{GPMC\_FCLK period}$
- nBE0 /nBE1 pulse:  
For single read:  $N = \text{RDCYCLETIME} * (\text{TIMEPARAGRANULARITY} + 1) * \text{GPMC\_FCLK period}$

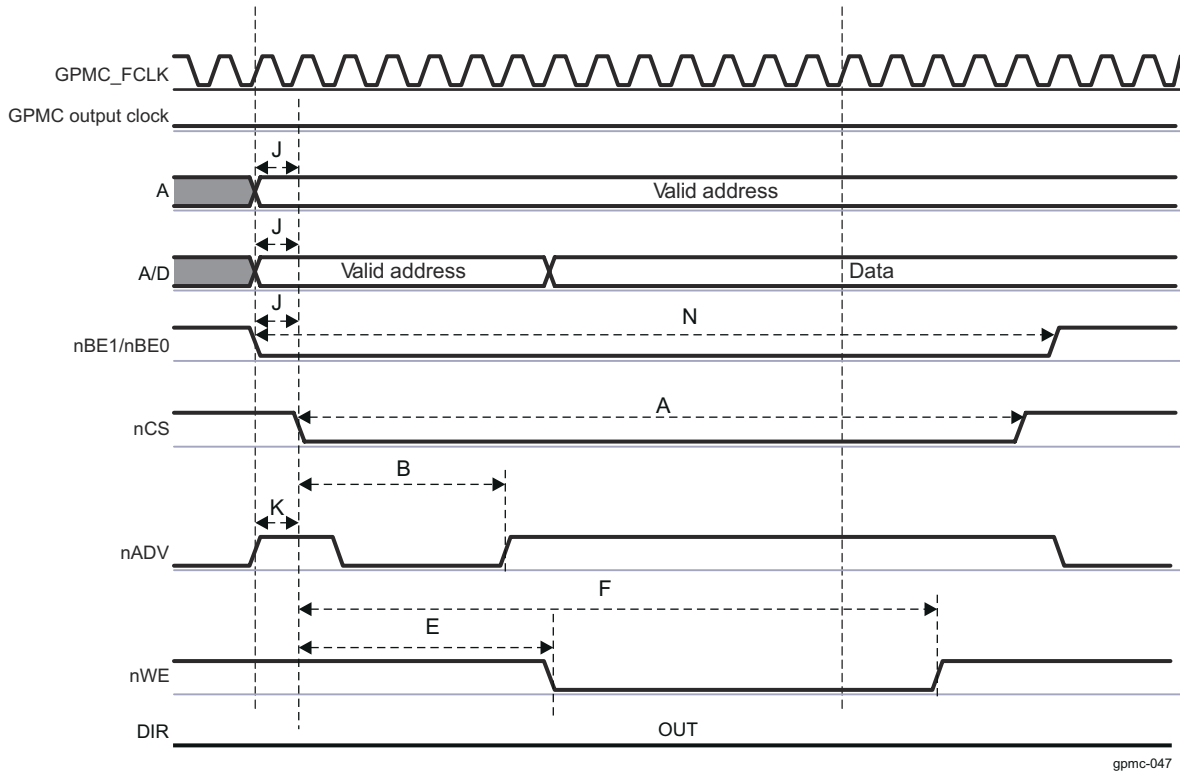


For burst read:  $N = (RDCYCLETIME + (N - 1) * PAGEBURSTACCESSTIME) * (TIMEPARAGRANULARITY + 1) * GPMC\_FCLK \text{ period}$ , where N = page burst access number

For burst write:  $N = (WRCYCLETIME + (N - 1) * PAGEBURSTACCESSTIME) * (TIMEPARAGRANULARITY + 1) * GPMC\_FCLK \text{ period}$ , where N = page burst access number

- $O = ((WRCYCLETIME + (N - 1) * PAGEBURSTACCESSTIME - CSONTIME) * (TIMEPARAGRANULARITY + 1) + 0.5 * (ADVEXTRADELAY - CSEXTRADELAY)) * GPMC\_FCLK \text{ period}$

Figure 12-282 shows a simplified example of an asynchronous NOR single write where formulas are associated with signal waves.



**Figure 12-282. Asynchronous NOR Single Write Simplified Example**

#### Note

Write multiple access is not supported in asynchronous mode. If WRITEMULTIPLE is enabled with WRITETYPE as asynchronous, the GPMC processes single asynchronous accesses.

### 12.3.5 Error Location Module (ELM)

This section describes the Error Location Module (ELM) for the device.

#### 12.3.5.1 ELM Overview

The ELM extracts error addresses from generated syndrome polynomials.

The ELM is used with the GPMC. Syndrome polynomials generated on-the-fly when reading a NAND flash page and stored in GPMC registers are passed to the ELM. A host processor can then correct the data block by flipping the bits to which the ELM error-location outputs point.

When reading from NAND flash memories, some level of error-correction is required. In the case of NAND modules with no internal correction capability, sometimes referred to as *bare NANDs*, the correction process is delegated to the memory controller. ELM can be also used to support parallel NOR flash or NAND flash.

The General-Purpose Memory Controller (GPMC) probes data read from an external NAND flash and uses this to compute checksum-like information, called syndrome polynomials, on a per-block basis. Each syndrome polynomial gives a status of the read operations for a full block, including 512 bytes of data, parity bits, and an optional spare-area data field, with a maximum block size of 1023 bytes. Computation is based on a Bose-Chaudhuri-Hocquenghem (BCH) algorithm. The ELM extracts error addresses from these syndrome polynomials.

Based on the syndrome polynomial value, the ELM can detect errors, compute the number of errors, and give the location of each error bit. The actual data is not required to complete the error-correction algorithm. Errors can be reported anywhere in the NAND flash block, including in the parity bits.

The maximum acceptable number of errors that can be corrected depends on a programmable configuration parameter. 4-, 8-, and 16-bit error-correction levels are supported. The ELM depends on a static and fixed definition of the generator polynomial for each error-correction level that corresponds to the generator polynomials defined in the GPMC (there are three fixed polynomial for the three correction error levels). A larger number of errors than the programmed error-correction level may be detected, but the ELM cannot correct them all. The offending block is then tagged as *uncorrectable* in the associated computation exit status register. If the computation is successful, that is, if the number of errors detected does not exceed the maximum value authorized for the chosen correction capability, the exit status register contains the information on the number of detected errors.

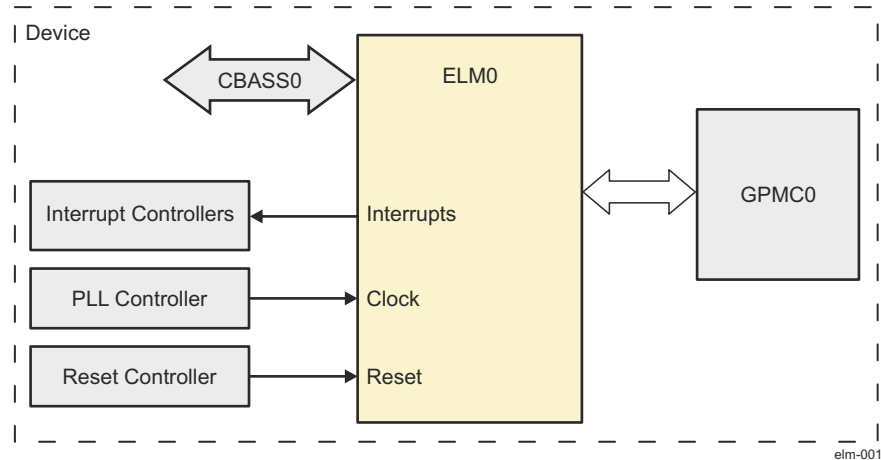
When the error-location process completes, an interrupt is triggered to inform the software that its status can be checked. The number of detected errors and their locations in the NAND block can be retrieved from the module through register accesses.

Table 12-365 shows the ELM allocation across device domains.

**Table 12-365. ELM Allocation Across Device Domains**

Instance	WKUP	MCU	MAIN
	Domain		
ELM0	-	-	✓

Figure 12-283 shows the ELM0 module overview.



**Figure 12-283. ELM0 Overview**

#### 12.3.5.1.1 ELM Features

The ELM has the following features:

- 4, 8, and 16 bits per 512-byte block error-location, based on BCH algorithms
- Eight simultaneous processing contexts
- Page-based and continuous modes
- Interrupt generation on error-location process completion:
  - When the full page has been processed in page mode
  - For each syndrome polynomial in continuous mode.

#### 12.3.5.1.2 ELM Not Supported Features

The following features are not supported on this family of devices:

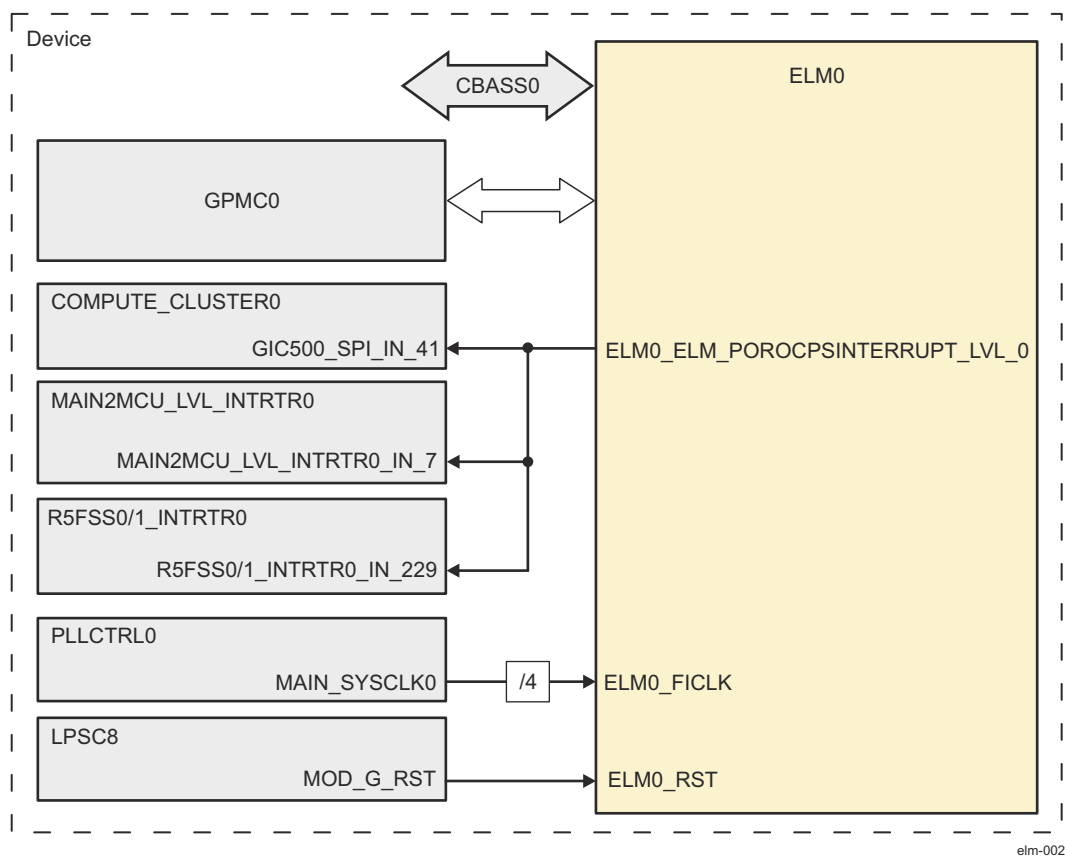
- Local power management of clock activity

### 12.3.5.2 ELM Integration

This section describes module integration in the device, including information about clocks, resets, and hardware requests.

### 12.3.5.2.1 ELM Integration in MAIN Domain

A single ELM module is integrated in the device MAIN domain - ELM0. Figure 12-284 shows the ELM0 integration.



**Figure 12-284. ELM0 Integration**

Table 12-366 through Table 12-368 summarize the integration of ELM0 in device MAIN domain.

**Table 12-366. ELM0 Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
ELM0	PSC0	PD0	LPSC8	CBASS0

**Table 12-367. ELM0 Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
ELM0	ELM0_FICLK	MAIN_SYSCCLK0/4	PLLCTRL0	ELM0 functional and interface clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
ELM0	ELM0_RST	MOD_G_RST	LPSC8	ELM0 hardware reset

**Table 12-368. ELM0 Hardware Requests**

Interrupt Requests				
--------------------	--	--	--	--

**Table 12-368. ELM0 Hardware Requests (continued)**

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
ELM0	ELM0_ELM_POROCPSINTERR UPT_LVL_0	GIC500_SPI_IN_41	COMPUTE_CLUSTER0	Error-location process complete interrupt	Level
		R5FSS0_INTRTR0_IN_29	R5FSS0_INTRTR0	Error-location process complete interrupt	Level
		R5FSS1_INTRTR0_IN_29	R5FSS1_INTRTR0	Error-location process complete interrupt	Level
		MAIN2MCU_LVL_INTRTR0_IN_7	MAIN2MCU_LVL_INTRTR0	Error-location process complete interrupt	Level

### 12.3.5.3 ELM Functional Description

The ELM0 module is hereinafter referred to as ELM.

The ELM is designed around the error-location engine, which handles the computation based on the input syndrome polynomials.

The ELM maps the error-location engine to a standard interconnect interface by using a set of registers to control inputs and outputs.

#### 12.3.5.3.1 ELM Software Reset

To perform a software reset, set the ELM\_SYSCONFIG[1] SOFTRESET bit to 1. The ELM\_SYSSTATUS[0] RESETDONE bit indicates that the software reset is complete when its value is 1. When the software reset completes, the ELM\_SYSCONFIG[1] SOFTRESET bit is automatically reset.

#### 12.3.5.3.2 ELM Power Management

##### Note

Some of the ELM features described in this section may not be supported on this family of devices. For more information, see *ELM Not Supported Features*.

Table 12-369 describes the power-management features available to the ELM.

**Table 12-369. Local Power-Management Features**

Feature	Registers	Description
Clock autogating	ELM_SYSCONFIG[0] AUTOGATING	This bit allows a local power optimization inside the module by gating the ELM_FICLK clock upon the interface activity.
Idle modes	ELM_SYSCONFIG[4-3] SIDLEMODE	Force-idle, no-idle, and smart-idle modes are available.
Clock activity	ELM_SYSCONFIG[8] CLOCKACTIVITY	The clock can be switched off or maintained.

#### 12.3.5.3.3 ELM Interrupt Requests

Table 12-370 lists the event flags, and their masks, that can cause module interrupts asserting the ELM0\_ELM\_POROCPSINTERRUPT\_LVL\_0 signal.

**Table 12-370. ELM Events**

Event Flag	Event Mask	Description
ELM_IRQSTATUS[8] PAGE_VALID	ELM_IRQENABLE[8] PAGE_MASK	Page interrupt
ELM_IRQSTATUS[7] LOC_VALID_7	ELM_IRQENABLE[7] LOCATION_MASK_7	Error-location interrupt for syndrome polynomial 7
ELM_IRQSTATUS[6] LOC_VALID_6	ELM_IRQENABLE[6] LOCATION_MASK_6	Error-location interrupt for syndrome polynomial 6
ELM_IRQSTATUS[5] LOC_VALID_5	ELM_IRQENABLE[5] LOCATION_MASK_5	Error-location interrupt for syndrome polynomial 5
ELM_IRQSTATUS[4] LOC_VALID_4	ELM_IRQENABLE[4] LOCATION_MASK_4	Error-location interrupt for syndrome polynomial 4
ELM_IRQSTATUS[3] LOC_VALID_3	ELM_IRQENABLE[3] LOCATION_MASK_3	Error-location interrupt for syndrome polynomial 3
ELM_IRQSTATUS[2] LOC_VALID_2	ELM_IRQENABLE[2] LOCATION_MASK_2	Error-location interrupt for syndrome polynomial 2
ELM_IRQSTATUS[1] LOC_VALID_1	ELM_IRQENABLE[1] LOCATION_MASK_1	Error-location interrupt for syndrome polynomial 1
ELM_IRQSTATUS[0] LOC_VALID_0	ELM_IRQENABLE[0] LOCATION_MASK_0	Error-location interrupt for syndrome polynomial 0

### 12.3.5.3.4 ELM Processing Initialization

ELM\_LOCATION\_CONFIG global setting parameters must be set before using the error-location engine. The ELM\_LOCATION\_CONFIG[1-0] ECC\_BCH\_LEVEL bit field defines the error-correction level used (4-, 8-, or 16-bit error correction). The ELM\_LOCATION\_CONFIG[26-16] ECC\_SIZE bit field defines the maximum buffer length beyond which the engine processing no longer looks for errors.

Software can choose to use the ELM in continuous mode or page mode. If all ELM\_PAGE\_CTRL[i] SECTOR\_i bits (i is the syndrome polynomial number, where i = 0 to 7) are reset, continuous mode is used. In any other case, page mode is implicitly selected.

- Continuous mode: Each syndrome polynomial is processed independently. Results for a syndrome can be retrieved and acknowledged at any time, regardless of the status of the other seven processing contexts.
- Page mode: Syndrome polynomials are grouped into atomic entities: only one page can be processed at any given time, even if all eight contexts are not used for this page. Unused contexts are lost and cannot be affected to any other processing. The full page must be acknowledged and cleared before moving to the next page.

For completion interrupts to be generated correctly, all ELM\_IRQENABLE[i] LOCATION\_MASK\_i bits (where i = 0 to 7) must be forced to 0 when in page mode, and set to 1 in continuous mode. Additionally, the ELM\_IRQENABLE[8] PAGE\_MASK bit must be set to 1 when in page mode.

Software initiates error-location processing by writing a syndrome polynomial into one of the eight possible register sets. Each of these register sets includes seven registers: ELM\_SYNDROME\_FRAGMENT\_0\_i to ELM\_SYNDROME\_FRAGMENT\_6\_i. The first six registers can be written in any order, but ELM\_SYNDROME\_FRAGMENT\_6\_i must be written last because it includes the validity bit, which instructs the ELM that this syndrome polynomial must be processed (the ELM\_SYNDROME\_FRAGMENT\_6\_i[16] SYNDROME\_VALID bit).

As soon as one validity bit is asserted (ELM\_SYNDROME\_FRAGMENT\_6\_i[16] SYNDROME\_VALID = 0x1, where i = 0 to 7), error-location processing can start for the corresponding syndrome polynomial. The associated ELM\_LOCATION\_STATUS\_i and ELM\_ERROR\_LOCATION\_0\_i to ELM\_ERROR\_LOCATION\_15\_i registers (where i = 0 to 7) are not reset. Software must not consider them until the corresponding ELM\_IRQSTATUS[i] LOC\_VALID\_i bit is set.

### 12.3.5.3.5 ELM Processing Sequence

While the error-location engine is busy processing one syndrome polynomial, further syndrome polynomials can be written. They are processed when the current processing completes.

The engine completes early when:

- No error is detected; that is, when the ELM\_LOCATION\_STATUS\_i[8] ECC\_CORRECTABLE bit is set to 1 and the ELM\_LOCATION\_STATUS\_i[4-0] ECC\_NB\_ERRORS bit field is set to 0x0.
- Too many errors are detected; that is, when the ELM\_LOCATION\_STATUS\_i[8] ECC\_CORRECTABLE bit is set to 0 while the ELM\_LOCATION\_STATUS\_i[4-0] ECC\_NB\_ERRORS bit field is set with the value output by the error-location engine. The reported number of errors is not ensured if ECC\_CORRECTABLE is 0.

If the engine completes early, the associated error-location registers ELM\_ERROR\_LOCATION\_0\_i to ELM\_ERROR\_LOCATION\_15\_i (where i = 0 to 7) are not updated.

In all other cases, the engine goes through the entire error-location process. Each time an error location is found, it is logged in the associated ECC\_ERROR\_LOCATION bit field. The first error detected is logged in the ELM\_ERROR\_LOCATION\_0\_i[12-0] ECC\_ERROR\_LOCATION bit field; the second is logged in the ELM\_ERROR\_LOCATION\_1\_i[12-0] ECC\_ERROR\_LOCATION bit field, and so on.

Table 12-371 describes the ELM\_LOCATION\_STATUS\_i value decoding.

**Table 12-371. ELM\_LOCATION\_STATUS\_i Value Decoding**

ECC_CORRECTABLE Value	ECC_NB_ERRORS Value	Status	Number of Errors Detected	Action Required
-----------------------	---------------------	--------	---------------------------	-----------------



**Table 12-371. ELM\_LOCATION\_STATUS\_i Value Decoding (continued)**

1	0	OK	0	None
1	≠ 0	OK	ECC_NB_ERRORS	Correct the data buffer read based on the ELM_ERROR_LOCATION_0_i to ELM_ERROR_LOCATION_15_i results.
0	Any	Failed	Unknown	Software-dependent

### 12.3.5.3.6 ELM Processing Completion

When the processing for a given syndrome polynomial completes, its ELM\_SYNDROME\_FRAGMENT\_6\_i[16] SYNDROME\_VALID bit is reset. It must not be set again until the exit status registers, ELM\_LOCATION\_STATUS\_i (where i = 0 to 7) for this processing are checked. Failure to comply with this rule leads to potential loss of the first polynomial process data output.

The error-location engine signals the process completion to the ELM. When this event is detected, the corresponding ELM\_IRQSTATUS[i] LOC\_VALID\_i bit (where i = 0 to 7) is set. The processing exit status is available from the associated ELM\_LOCATION\_STATUS\_i register, and error locations are stored in order in the ECC\_ERROR\_LOCATION bit fields. Software must read only valid error-location registers based on the number of errors detected and located.

Immediately after the error-location engine completes, a new syndrome polynomial can be processed, if any is available, as reported by the ELM\_SYNDROME\_FRAGMENT\_6\_i[16] SYNDROME\_VALID bit, depending on the configured error-correction level. If several syndrome polynomials are available, a round-robin arbitration is used to select one for processing.

In continuous mode (that is, all bits in ELM\_PAGE\_CTRL are reset), an interrupt is triggered whenever a ELM\_IRQSTATUS[i] LOC\_VALID\_i bit is asserted. Software must read the ELM\_IRQSTATUS register to determine which polynomial is processed and retrieve the exit status and error locations (ELM\_LOCATION\_STATUS\_i and ELM\_ERROR\_LOCATION\_0\_i to ELM\_ERROR\_LOCATION\_15\_i). When done, software must clear the corresponding ELM\_IRQSTATUS[i] LOC\_VALID\_i bit by setting it to 1. Other status bits must be set to 0 so that other interrupts are not unintentionally cleared. When using this mode, the ELM\_IRQSTATUS[8] PAGE\_VALID interrupt is never triggered.

In page mode, the module does not trigger interrupts for the processing completion of each polynomial because the ELM\_IRQENABLE[i] LOCATION\_MASK\_i bits are cleared. A page is defined using the ELM\_PAGE\_CTRL register. Each SECTOR\_i bit set means the corresponding polynomial i is part of the page processing. A page is fully processed when all tagged polynomials have been processed, as logged in the ELM\_IRQSTATUS[i] LOC\_VALID\_i bits. The module triggers an ELM\_IRQSTATUS[8] PAGE\_VALID interrupt whenever it detects that the full page has been processed. To make sure the next page can be correctly processed, all status bits in the ELM\_IRQSTATUS register must be cleared by using a single atomic-write access.

### Note

Do not modify page setting parameters in the ELM\_PAGE\_CTRL register unless the engine is idle, no polynomial input is valid, and all interrupts have been cleared.

Because no polynomial-level interrupt is triggered in page mode, polynomials cleared in the ELM\_PAGE\_CTRL[i] SECTOR\_i bits (where i = 0 to 7) are processed as usual, but are essentially ignored. Software must manually poll the ELM\_IRQSTATUS bits to check for their status.

### 12.3.5.4 ELM Basic Programming Model

#### 12.3.5.4.1 ELM Low-Level Programming Model

##### 12.3.5.4.1.1 Processing Initialization

**Table 12-372. ELM Processing Initialization**

Step	Register/Bit Field/Programming Model	Value
Resets the module.	ELM_SYSCONFIG[1] SOFTRESET	0x1
Wait until reset is done.	ELM_SYSSTATUS[0] RESETDONE	0x1
Configure the target interface power management.	ELM_SYSCONFIG[4-3] SIDLEMODE	Set value
Defines the error-correction level used.	ELM_LOCATION_CONFIG[1-0] ECC_BCH_LEVEL	Set value
Defines the maximum buffer length.	ELM_LOCATION_CONFIG[26-16] ECC_SIZE	Set value
Sets the ELM in continuous mode or page mode.	ELM_PAGE_CTRL	Set value
<b>IF</b> continuous mode is used:	All ELM_PAGE_CTRL[i] SECTOR_i (where i = 0 to 7)	0x0
Enable interrupt for syndrome polynomial i.	ELM_IRQENABLE[i] LOCATION_MASK_i	0x1
<b>ELSE</b> (page mode is used):	One syndrome polynomial i is set ELM_PAGE_CTRL[i] SECTOR_i (where i = 0 to 7)	0x1
Disable all interrupts for syndrome polynomial and enable PAGE_MASK interrupt.	All ELM_IRQENABLE[i] LOCATION_MASK_i = 0x0 and ELM_IRQENABLE[8] PAGE_MASK = 0x1	Set value
<b>ENDIF</b>		Set value
Set the input syndrome polynomial i.	ELM_SYNDROME_FRAGMENT_0_i	Set value
	ELM_SYNDROME_FRAGMENT_1_i	Set value
	ELM_SYNDROME_FRAGMENT_5_i	Set value
	ELM_SYNDROME_FRAGMENT_6_i	Set value
Initiates the computation process.	ELM_SYNDROME_FRAGMENT_6_i[16] SYNDROME_VALID	0x1

##### 12.3.5.4.1.2 Read Results

The engine goes through the entire error-location process and results can be read. [Table 12-373](#) and [Table 12-374](#) describe the processing completion for continuous and page modes, respectively.

**Table 12-373. ELM Processing Completion for Continuous Mode**

Step	Register/Bit Field/Programming Model	Value
Wait until process is complete for syndrome polynomial i: Wait until the ELM0_ELM_POROCPSINTERRUPT_LVL_0 interrupt is generated, or poll the status register.		
Read for which i the error-location process is complete.	ELM_IRQSTATUS[i] LOC_VALID_i	0x1
<b>IF</b> the process fails (too many errors):	ELM_LOCATION_STATUS_i[8] ECC_CORRECTABLE	0x0
It is software dependant.		
<b>ELSE</b> (process successful, the engine completes):	ELM_LOCATION_STATUS_i[8] ECC_CORRECTABLE	0x1
Read the number of errors.	ELM_LOCATION_STATUS_i[4-0] ECC_NB_ERRORS	
Read the error-location bit addresses for syndrome polynomial i of the ECC_NB_ERRORS first registers. Software must correct errors in the data buffer.	ELM_ERROR_LOCATION_0_i[12-0] ECC_ERROR_LOCATION	
	ELM_ERROR_LOCATION_1_i[12-0] ECC_ERROR_LOCATION	
	...	
	ELM_ERROR_LOCATION_15_i[12-0] ECC_ERROR_LOCATION	
<b>ENDIF</b>		
Clear the corresponding i interrupt.	ELM_IRQSTATUS[i] LOC_VALID_i	0x1

A new syndrome polynomial can be processed after the end of processing (ELM\_SYNDROME\_FRAGMENT\_6\_i[16] SYNDROME\_VALID = 0x0) and after the exit status register check (ELM\_LOCATION\_STATUS\_i).

**Table 12-374. ELM Processing Completion for Page Mode**

Step	Register/Bit Field/Programming Model	Value
Wait until process is complete for syndrome polynomial i: Wait until the ELM0_ELM_POROCPSINTERRUPT_LVL_0 interrupt is generated, or poll the status register.		
Wait for page completed interrupt: All error locations are valid.	ELM_IRQSTATUS[8] PAGE_VALID	0x1
<b>Repeat</b> the following actions the necessary number of times. That is, once for each valid defined block in the page.		
Read the process exit status.	ELM_LOCATION_STATUS_i[8] ECC_CORRECTABLE	
<b>IF</b> the process fails (too many errors): It is software dependent.	ELM_LOCATION_STATUS_i[8] ECC_CORRECTABLE	0x0
<b>ELSE</b> (process successful, the engine completes):	ELM_LOCATION_STATUS_i[8] ECC_CORRECTABLE	0x1
Read the number of errors.	ELM_LOCATION_STATUS_i[4-0] ECC_NB_ERRORS	
Read the error-location bit addresses for syndrome polynomial i of the ECC_NB_ERRORS first registers.	ELM_ERROR_LOCATION_0_i[12-0] ECC_ERROR_LOCATION	
	ELM_ERROR_LOCATION_1_i[12-0] ECC_ERROR_LOCATION	
	...	
	ELM_ERROR_LOCATION_15_i[12-0] ECC_ERROR_LOCATION	
<b>ENDIF</b>		
<b>End Repeat</b>		
Clear the ELM_IRQSTATUS register.	ELM_IRQSTATUS	0x1FF

Next page can be correctly processed after a page is fully processed, when all tagged polynomials have been processed (ELM\_IRQSTATUS[i] LOC\_VALID\_i = 0x1 for all syndrome polynomials i used in the page).

#### 12.3.5.4.1.3

Next page can be correctly processed after a page is fully processed, when all tagged polynomials have been processed (ELM\_IRQSTATUS[i] LOC\_VALID\_i = 0x1 for all syndrome polynomials i used in the page).

#### 12.3.5.4.2 Use Case: ELM Used in Continuous Mode

In this example, the ELM is programmed for an 8-bit error-correction capability in continuous mode (see [Table 12-375](#)). After reading a 528-byte NAND flash sector (512B data plus 16B spare area) with a 16-bit interface, a nonzero polynomial syndrome is reported from the GPMC (polynomial syndrome 0 is used in the ELM):

- P = 0x0A16ABE115E44F767BFB0D0980.

**Table 12-375. Use Case: Continuous Mode**

Step	Register/Bit Field/Programming Model	Value
Reset the module.	ELM_SYSCONFIG[1] SOFTRESET	0x1
Wait until reset is done.	ELM_SYSSTATUS[0] RESETDONE	0x1
Configure the target interface power management: Smart idle is used.	ELM_SYSCONFIG[4-3] SIDLEMODE	0x2
Define the error-correction level used: 8 bits.	ELM_LOCATION_CONFIG[1-0] ECC_BCH_LEVEL	0x1
Define the maximum buffer length: 528 bytes (2 × 528 = 1056).	ELM_LOCATION_CONFIG[26-16] ECC_SIZE	0x420
Set the ELM in continuous mode.	ELM_PAGE_CTRL	0
Enable interrupt for syndrome polynomial 0.	ELM_IRQENABLE[0] LOCATION_MASK_0	0x1

**Table 12-375. Use Case: Continuous Mode (continued)**

Step	Register/Bit Field/Programming Model	Value
Set the input syndrome polynomial 0.	ELM_SYNDROME_FRAGMENT_0_i (where i = 0)	0xFB0D0980
	ELM_SYNDROME_FRAGMENT_1_i (where i = 0)	0xE44F767B
	ELM_SYNDROME_FRAGMENT_2_i (where i = 0)	0x16ABE115
	ELM_SYNDROME_FRAGMENT_3_i (where i = 0)	0x0000000A
Initiate the computation process.	ELM_SYNDROME_FRAGMENT_6_i[16] SYNDROME_VALID (where i = 0)	0x1
Wait until process is complete for syndrome polynomial 0: ELM0_ELM_POROCPSINTERRUPT_LVL_0 is generated or poll the status register.		
Read that error-location process is complete for syndrome polynomial 0.	ELM_IRQSTATUS[0] LOC_VALID_0	0x1
Read the process exit status: All errors were successfully located.	ELM_LOCATION_STATUS_i[8] ECC_CORRECTABLE (where i = 0)	0x1
Read the number of errors: Four errors detected.	ELM_LOCATION_STATUS_i[4-0] ECC_NB_ERRORS (where i = 0)	0x4
Read the error-location bit addresses for syndrome polynomial 0 of the first four registers: Errors are located in the data buffer at decimal addresses 431, 1062, 1909, 3452.	ELM_ERROR_LOCATION_0_i (where i = 0)	0x1AF
	ELM_ERROR_LOCATION_1_i (where i = 0)	0x426
	ELM_ERROR_LOCATION_2_i (where i = 0)	0x775
	ELM_ERROR_LOCATION_3_i (where i = 0)	0xD7C
Clear the corresponding interrupt for polynomial 0.	ELM_IRQSTATUS[0] LOC_VALID_0	0x1

The NAND flash data in the sector are seen as a polynomial of degree 4223 (number of bits in a 528 byte buffer minus 1), with each data bit being a coefficient in the polynomial. When reading from a NAND flash using the GPMC module, computation of the polynomial syndrome assumes that the first NAND word read at address 0x0 contains the highest-order coefficient in the message. Furthermore, in the 16-bit NAND word, bits are ordered from bit 7 to bit 0, and then from bit 15 to bit 8. Based on this convention, an address table of the data buffer can be built. NAND memory addresses in [Table 12-376](#) are given in decimal format.

**Table 12-376. 16-Bit NAND Sector Buffer Address Map**

NAND Memory Address	Message Bit Addresses in Memory Word															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	4215	4214	4213	4212	4211	4210	4209	4208	4223	4222	4221	4220	4219	4218	4217	4216
1	4175	4174	4173	4172	4171	4170	4169	4168	4183	4182	4181	4180	4179	4178	4177	4176
...																
47	3463	3462	3461	3460	3459	3458	3457	3456	3471	3470	3469	3468	3467	3466	3465	3464
48	3447	3446	3445	3444	3443	3442	3441	3440	3455	3454	3453	3452	3451	3450	3449	3448
49	3431	3430	3429	3428	3427	3426	3425	3424	3439	3438	3437	3436	3435	3434	3433	3432
50	3415	3414	3413	3412	3411	3410	3409	3408	3423	3422	3421	3420	3419	3418	3417	3416
...																
255	135	134	133	132	131	130	129	128	143	142	141	140	139	138	137	136
256	119	118	117	116	115	114	113	112	127	126	125	124	123	122	121	120
257	103	102	101	100	99	98	97	96	111	110	109	108	107	106	105	104
258	87	86	85	84	83	82	81	80	95	94	93	92	91	90	89	88
259	71	70	69	68	67	66	65	64	79	78	77	76	75	74	73	72
260	55	54	53	52	51	50	49	48	63	62	61	60	59	58	57	56
261	39	38	37	36	35	34	33	32	47	46	45	44	43	42	41	40
262	23	22	21	20	19	18	17	16	31	30	29	28	27	26	25	24

**Table 12-376. 16-Bit NAND Sector Buffer Address Map (continued)**

263	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8
-----	---	---	---	---	---	---	---	---	----	----	----	----	----	----	---	---

The table can now be used to determine which bits in the buffer were incorrect and must be flipped. In this example, the first bit to be flipped is bit 4 from the 49th byte read from memory. It is up to the processor to correctly map this word to the copied buffer and flip this bit. The same process must be repeated for all detected errors.

#### 12.3.5.4.3 Use Case: ELM Used in Page Mode

In this example, the ELM module is programmed for a 16-bit error-correction capability in page mode (see [Table 12-377](#)). After reading a 528-byte NAND flash sector (512B data plus 16B spare area) with a 16-bit interface, four non-zero polynomial syndromes are reported from the GPMC (polynomial syndrome 0, 1, 2, and 3 are used in the ELM):

- P0 = 0xE8B0 12ADDB5A318E05BE B0693DB28330B5CC A329AA05E0B718EF
- P1 = 0xBAD0 49A0D932C22E6669 0948DF08BE093336 79C6BA10E5F935EB
- P2 = 0x69D9 B86ABCD5EC3697FA A6498FEE54556EA0 1579EF7D60BA3189
- P3 = 0x0

**Table 12-377. Use Case: Page Mode**

Step	Register/Bit Field/Programming Model	Value
Reset the module	ELM_SYSCONFIG[1] SOFTRESET	0x1
Wait until reset is done.	ELM_SYSSTATUS[0] RESETDONE	0x1
Configure the target interface power management: Smart idle is used.	ELM_SYSCONFIG[4-3] SIDLEMODE	0x2
Define the error-correction level used: 16 bits	ELM_LOCATION_CONFIG[1-0] ECC_BCH_LEVEL	0x2
Define the maximum buffer length: 528 bytes	ELM_LOCATION_CONFIG[26-16] ECC_SIZE	0x420
Set the ELM in page mode (four blocks in a page)	ELM_PAGE_CTRL[0] SECTOR_0	0x1
	ELM_PAGE_CTRL[1] SECTOR_1	0x1
	ELM_PAGE_CTRL[2] SECTOR_2	0x1
	ELM_PAGE_CTRL[3] SECTOR_3	0x1
Disable all interrupts for syndrome polynomial and enable PAGE_MASK interrupt.	ELM_IRQENABLE	0x100
Set the input syndrome polynomial 0.	ELM_SYNDROME_FRAGMENT_0_i (where i = 0)	0xE0B718EF
	ELM_SYNDROME_FRAGMENT_1_i (where i = 0)	0xA329AA05
	ELM_SYNDROME_FRAGMENT_2_i (where i = 0)	0x8330B5CC
	ELM_SYNDROME_FRAGMENT_3_i (where i = 0)	0xB0693DB2
	ELM_SYNDROME_FRAGMENT_4_i (where i = 0)	0x318E05BE
	ELM_SYNDROME_FRAGMENT_5_i (where i = 0)	0x12ADDB5A
	ELM_SYNDROME_FRAGMENT_6_i (where i = 0)	0xE8B0
Set the input syndrome polynomial 1.	ELM_SYNDROME_FRAGMENT_0_i (where i = 1)	0xE5F935EB
	ELM_SYNDROME_FRAGMENT_1_i (where i = 1)	0x79C6BA10
	ELM_SYNDROME_FRAGMENT_2_i ((where i = 1)	0xBE093336
	ELM_SYNDROME_FRAGMENT_3_i (where i = 1)	0x0948DF08
	ELM_SYNDROME_FRAGMENT_4_i (where i = 1)	0xC22E6669
	ELM_SYNDROME_FRAGMENT_5_i (where i = 1)	0x49A0D932
	ELM_SYNDROME_FRAGMENT_6_i (where i = 1)	0xBAD0

**Table 12-377. Use Case: Page Mode (continued)**

Step	Register/Bit Field/Programming Model	Value
Set the input syndrome polynomial 2.	ELM_SYNDROME_FRAGMENT_0_i (where i = 2)	0x60BA3189
	ELM_SYNDROME_FRAGMENT_1_i (where i = 2)	0x1579EF7D
	ELM_SYNDROME_FRAGMENT_2_i (where i = 2)	0x54556EA0
	ELM_SYNDROME_FRAGMENT_3_i (where i = 2)	0xA6498FEE
	ELM_SYNDROME_FRAGMENT_4_i (where i = 2)	0xEC3697FA
	ELM_SYNDROME_FRAGMENT_5_i (where i = 2)	0xB86ABCD5
	ELM_SYNDROME_FRAGMENT_6_i (where i = 2)	0x69D9
Set the input syndrome polynomial 3.	ELM_SYNDROME_FRAGMENT_0_i (where i = 3)	0x0
	ELM_SYNDROME_FRAGMENT_1_i (where i = 3)	0x0
	ELM_SYNDROME_FRAGMENT_2_i (where i = 3)	0x0
	ELM_SYNDROME_FRAGMENT_3_i (where i = 3)	0x0
	ELM_SYNDROME_FRAGMENT_4_i (where i = 3)	0x0
	ELM_SYNDROME_FRAGMENT_5_i (where i = 3)	0x0
	ELM_SYNDROME_FRAGMENT_6_i (where i = 3)	0x0
Initiate the computation process for syndrome polynomial 0	ELM_SYNDROME_FRAGMENT_6_i[16] SYNDROME_VALID (where i = 0)	0x1
Initiate the computation process for syndrome polynomial 1	ELM_SYNDROME_FRAGMENT_6_i[16] SYNDROME_VALID (where i = 1)	0x1
Initiate the computation process for syndrome polynomial 2	ELM_SYNDROME_FRAGMENT_6_i[16] SYNDROME_VALID (where i = 2)	0x1
Initiate the computation process for syndrome polynomial 3	ELM_SYNDROME_FRAGMENT_6_i[16] SYNDROME_VALID (where i = 3)	0x1
Wait until process is complete for syndrome polynomial 0, 1, 2, and 3: Wait until the ELM0_ELM_POROCPSINTERRUPT_LVL_0 interrupt is generated or poll the status register.		
Wait for page completed interrupt: All error locations are valid.	ELM_IRQSTATUS[8] PAGE_VALID	0x1
Read the process exit status for syndrome polynomial 0: All errors were successfully located.	ELM_LOCATION_STATUS_i[8] ECC_CORRECTABLE (where i = 0)	0x1
Read the process exit status for syndrome polynomial 1: All errors were successfully located.	ELM_LOCATION_STATUS_i[8] ECC_CORRECTABLE (where i = 1)	0x1
Read the process exit status for syndrome polynomial 2: All errors were successfully located.	ELM_LOCATION_STATUS_i[8] ECC_CORRECTABLE (where i = 2)	0x1
Read the process exit status for syndrome polynomial 3: All errors were successfully located.	ELM_LOCATION_STATUS_i[8] ECC_CORRECTABLE (where i = 3)	0x1
Read the number of errors for syndrome polynomial 0: four errors detected.	ELM_LOCATION_STATUS_i[4-0] ECC_NB_ERRORS (where i = 0)	0x4
Read the number of errors for syndrome polynomial 1: two errors detected.	ELM_LOCATION_STATUS_i[4-0] ECC_NB_ERRORS (where i = 1)	0x2
Read the number of errors for syndrome polynomial 2: one error detected.	ELM_LOCATION_STATUS_i[4-0] ECC_NB_ERRORS (where i = 2)	0x1
Read the number of errors for syndrome polynomial 3: no errors detected.	ELM_LOCATION_STATUS_i[4-0] ECC_NB_ERRORS (where i = 3)	0x0

**Table 12-377. Use Case: Page Mode (continued)**

Step	Register/Bit Field/Programming Model	Value
Read the error-location bit addresses for syndrome polynomial 0 of the first four registers:	ELM_ERROR_LOCATION_0_i (where i = 0)	0x1FE
	ELM_ERROR_LOCATION_1_i (where i = 0)	0x617
	ELM_ERROR_LOCATION_2_i (where i = 0)	0x650
	ELM_ERROR_LOCATION_3_i (where i = 0)	0xA83
Read the error-location bit addresses for syndrome polynomial 1 of the first two registers:	ELM_ERROR_LOCATION_0_i (where i = 1)	0x4
	ELM_ERROR_LOCATION_1_i (where i = 1)	0x1036
Read the errors location bit addresses for syndrome polynomial 2 of the first registers:	ELM_ERROR_LOCATION_0_i (where i = 1)	0x3E8
Clear the ELM_IRQSTATUS register.	ELM_IRQSTATUS	0x1FF



### 12.3.6 Multimedia Card Secure Digital (MMCSD) Interface

This section describes the MMCSD modules in the device.

#### 12.3.6.1 MMCSD Overview

There are three MMCSD modules inside the device - MMCSD0, MMCSD1 and MMCSD2. Each MMCSD module includes one MMCSD Host Controller.

Each controller has the following data bus width:

- MMCSD0 - 8-bit wide data bus
- MMCSD1 - 4-bit wide data bus
- MMCSD2 - 4-bit wide data bus

Table 12-378 shows MMCSD allocation across device domains.

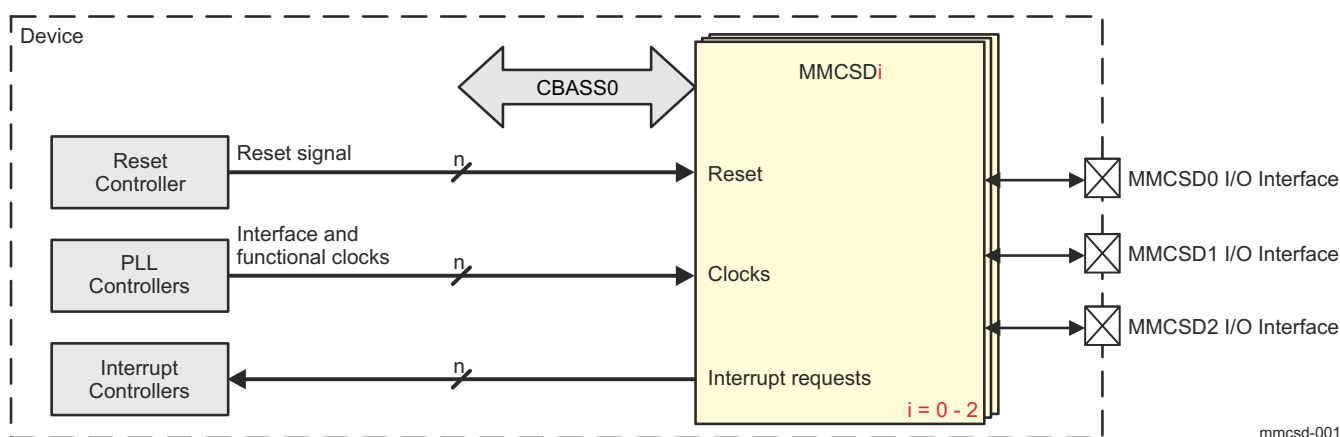
**Table 12-378. MMCSD Allocation Across Device Domains**

Instance	Domain		
	WKUP	MCU	MAIN
MMCSD0	-	-	✓
MMCSD1	-	-	✓
MMCSD2	-	-	✓

The MMCSD Host Controller provides an interface to eMMC 5.1 (embedded MultiMedia Card), SD 4.10 (Secure Digital), and SDIO 4.0 (Secure Digital IO) devices. The MMCSD Host Controller deals with MMC/SD/SDIO protocol at transmission level, data packing, adding cyclic redundancy checks (CRCs), start/end bit insertion, and checking for syntactical correctness.

The MMCSD Host Controller provides accessibility to external MMC/SD/SDIO devices using a Programmed IO method or DMA data transfer method. In programmed IO method, the device CPU transfers data using the Buffer Data Port register (MMCSD0\_DATA\_PORT / MMCSD12\_DATA\_PORT). In DMA data transfer method, the MMCSD Host Controller can read or write memory without device CPU intervention.

Figure 12-285 shows the MMCSDi module overview (where i = 0 to 2).



**Figure 12-285. MMCSDi Module Overview**

mmcsd-001



### 12.3.6.1.1 MMCSD Features

Each MMCSD Host Controller supports:

- Integrated DMA controller supporting SD Advanced DMA - ADMA2 and ADMA3 (for more information about ADMA support, see [Section 12.3.6.4.5, Advanced DMA](#))
- System Bus Interface:
  - 64-bit data width (master interface)
  - 64-bit address
  - Clock asynchronous to MMCSD clock (MMCi\_CLK)
  - Little endian only
- Configuration Bus Interface:
  - 32-bit data width (slave interface)
  - Linear incrementing addressing mode
  - 32-bit aligned accesses only
  - Little endian only
- Muxing of other LVCMOS interfaces onto the MMCSD interface at the SoC level (MMCSD1/MMCSD2 only)

MMCSD0 Host Controller (eMMC interface):

- MultiMedia Card Support:
  - eMMC Electrical Standard 5.1 (JESD84-B51)
  - Backward compatible with earlier eMMC standards
  - Legacy MMC SDR:
    - 1.8 V, 8-bit bus width, 0-25 MHz, 25 MBps
    - 1.8 V, 4-bit bus width, 0-25 MHz, 12.5 MBps
    - 1.8 V, 1-bit bus width, 0-25 MHz, 3.125 MBps
  - High Speed SDR:
    - 1.8 V, 8-bit bus width, 0-50 MHz, 50 MBps
    - 1.8 V, 4-bit bus width, 0-50 MHz, 25 MBps
    - 1.8 V, 1-bit bus width, 0-50 MHz, 6.25 MBps
  - High Speed DDR:
    - 1.8 V, 8-bit bus width, 0-50 MHz, 100 MBps
    - 1.8 V, 4-bit bus width, 0-50 MHz, 50 MBps
  - HS200 SDR:
    - 1.8 V, 0-200 MHz, 8-bit bus width, 200 MBps
    - 1.8 V, 0-200 MHz, 4-bit bus width, 100 MBps

MMCSD1/MMCSD2 Host Controller (SD/SDIO interface):

- 
- Secure Digital Card Support:
  - Backward compatible with earlier SD card specifications
  - SD Host Controller Standard Specification 4.10 and SD Physical Layer Specification v3.01
  - SDIO Specification v3.00
  - Default Speed mode: 3.3 V signaling, frequency up to 25 MHz, up to 12.5 MBps
  - High Speed mode: 3.3 V signaling, frequency up to 50 MHz, up to 25 MBps
  - SDR12: UHS-I 1.8 V signaling, frequency up to 25 MHz, up to 12.5 MBps
  - SDR25: UHS-I 1.8 V signaling, frequency up to 50 MHz, up to 25 MBps
  - SDR50: UHS-I 1.8 V signaling, frequency up to 100 MHz, up to 50 MBps
  - DDR50: UHS-I 1.8 V signaling, frequency up to 50 MHz, up to 50 MBps

### 12.3.6.1.2 MMCSD Not Supported Features

MMCSD0 Host Controller:

- MultiMedia Card:

- 3.3 V, 3.0 V and 1.2 V
- Secure Digital Card and SDIO
- Muxing of other LVCMOS interfaces onto the MMCSD interface at the SoC level
- HS400 DDR:
  - 1.8 V, 0-200 MHz, 8-bit bus width, 400 MBps

MMCSD1/MMCSD2 Host Controller:

- MultiMedia Card
- Secure Digital Card:
  - UHS-II
  - SDR104: UHS-I 1.8 V signaling, frequency up to 200 MHz, up to 100 MBps

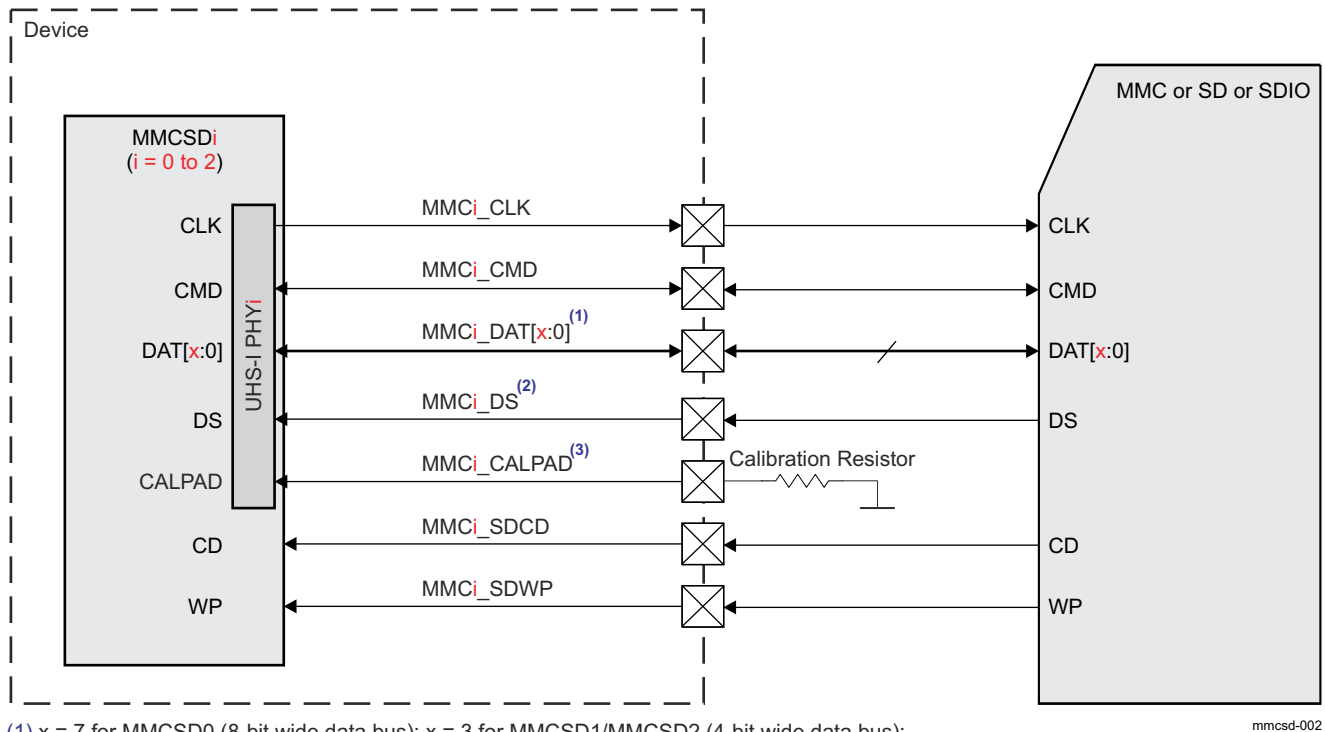
### 12.3.6.2 MMCSD Environment

The MMCSD0, MMCSD1 and MMCSD2 modules are hereinafter referred to as MMCSDi module.

This section describes the MMCSDi external connections (environment).

The MMCSD0 can be used for connection to 1, 4, or 8-bit devices (dedicated for connection to eMMC devices). The MMCSD1/MMCSD2 can be used for connection to 1 or 4-bit devices (dedicated for connection to SD or SDIO devices). For each MMCSD an integrated UHS-I PHY provides interface to external device.

Figure 12-286 shows the MMCSDi (where i = 0 to 2) connected to MMC, SD, or SDIO device.



**Figure 12-286. MMCSDi Connected to MMC, SD, or SDIO Device**

Table 12-379 describes the MMCSDi I/O signals.

**Table 12-379. MMCSDi I/O Signals**

Module Pin	Device Level Signal	I/O <sup>(1)</sup>	Description	Module Pin Reset Value
<b>MMCSD0</b>				
CLK	MMC0_CLK <sup>(3)</sup>	O	External Clock	0x0
CMD	MMC0_CMD	I/O	Command Line	0x0
DAT[7:0]	MMC0_DAT[7:0]	I/O	Data Signals	0x0
DS	MMC0_DS	I	Data Strobe	0x0
CALPAD	MMC0_CALPAD <sup>(2)</sup>	A	PHY Calibration Resistor	HiZ
<b>MMCSD1</b>				
CLK	MMC1_CLK	O	External Clock	0x0
CMD	MMC1_CMD	I/O	Command Line	0x0
DAT[3:0]	MMC1_DAT[3:0]	I/O	Data Signals	0x0
WP	MMC1_SDWP	I	SD Card Write Protect	0x1
CD	MMC1_SDCD	I	SD Card Detect	0x0
<b>MMCSD2</b>				
CLK	MMC2_CLK	O	External Clock	0x0
CMD	MMC2_CMD	I/O	Command Line	0x0
DAT[3:0]	MMC2_DAT[3:0]	I/O	Data Signals	0x0
WP	MMC2_SDWP	I	SD Card Write Protect	0x1
CD	MMC2_SDCD	I	SD Card Detect	0x0

(1) I = Input; O = Output; HiZ = High Impedance; A = Analog

(2) An external 10 kΩ ±1% resistor must be connected between this pin and VSS. No external voltage should be applied to this pin.

(3) This output signal is also used as a retiming input. It is also recommended to place a 33 Ω resistor in series (close to the SoC) to avoid signal reflections.

#### Note

For MMC1\_CLK signal to work properly, the RXACTIVE bit of the CTRLMMR0\_PADCONFIG171 register should be set to 0x1 because of retiming purposes.

For MMC2\_CLK signal to work properly, the RXACTIVE bit of the CTRLMMR0\_PADCONFIG172 register should be set to 0x1 because of retiming purposes.

#### Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), refer to the device-specific Datasheet.

### 12.3.6.2.1 Protocol and Data Format

The bus protocol between the MMCSD Host Controller and the card is message-based. Each message is represented by one of the following parts:

- **Command:** A command starts an operation. The command is transferred serially from the MMCSD Host Controller to the card on the CMD line.
- **Response:** A response is an answer to a command. The response is sent from the card to the MMCSD Host Controller. It is transferred serially on the CMD line.
- **Data:** Data are transferred from the MMCSD Host Controller to the card or from the card to the MMCSD Host Controller using the data lines.
- **Busy:** The DAT[0] signal is maintained low by the card as far as it is programming the data received.
- **CRC status:** The CRC result is sent by the card through the DAT[0] line when executing a write transfer. In the case of transmission error, occurring on any of the active data lines, the card sends a negative CRC status on DAT[0]. In the case of successful transmission, over all active data lines, the card sends a positive CRC status on DAT[0] and starts the data programming procedure.

#### 12.3.6.2.1.1 Protocol

There are two types of data transfer:

- Sequential operation
- Block-oriented operation

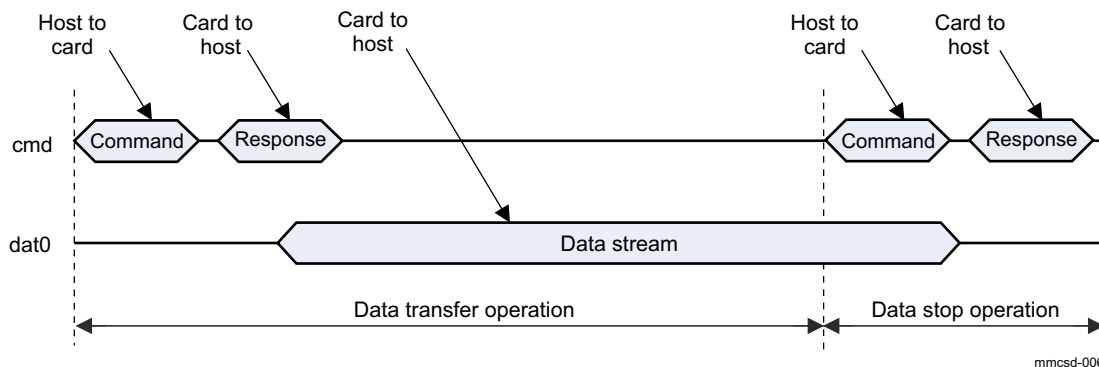
There are specific commands for each type of operation (sequential or block-oriented).

For information about commands and programming sequences supported by the MMC, SD, and SDIO devices, see the *Multimedia Card System Specification*, *SD Memory Card Specifications*, and *SDIO Card Specification (Part E1)*.

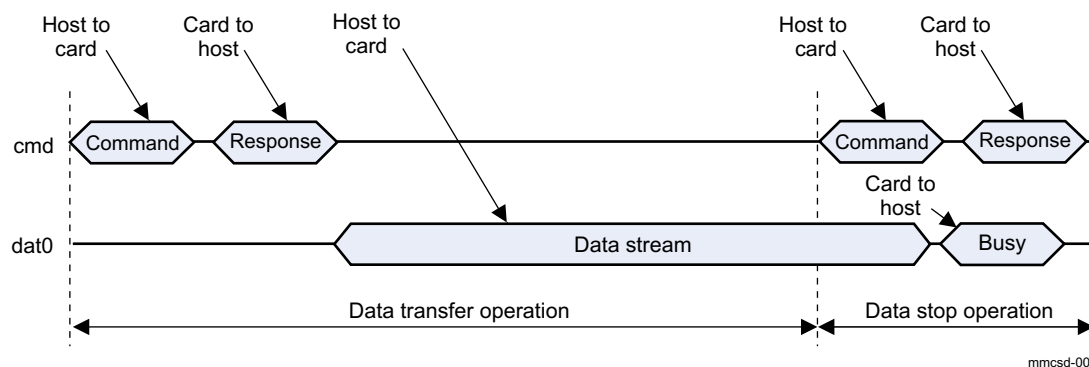
Figure 12-287 and Figure 12-288 show how sequential operations are defined. Sequential operation is only for 1-bit transfer and initiates a continuous data stream. The transfer terminates when a stop command follows on the CMD line.

#### Note

Stream commands are supported only by MMCs.

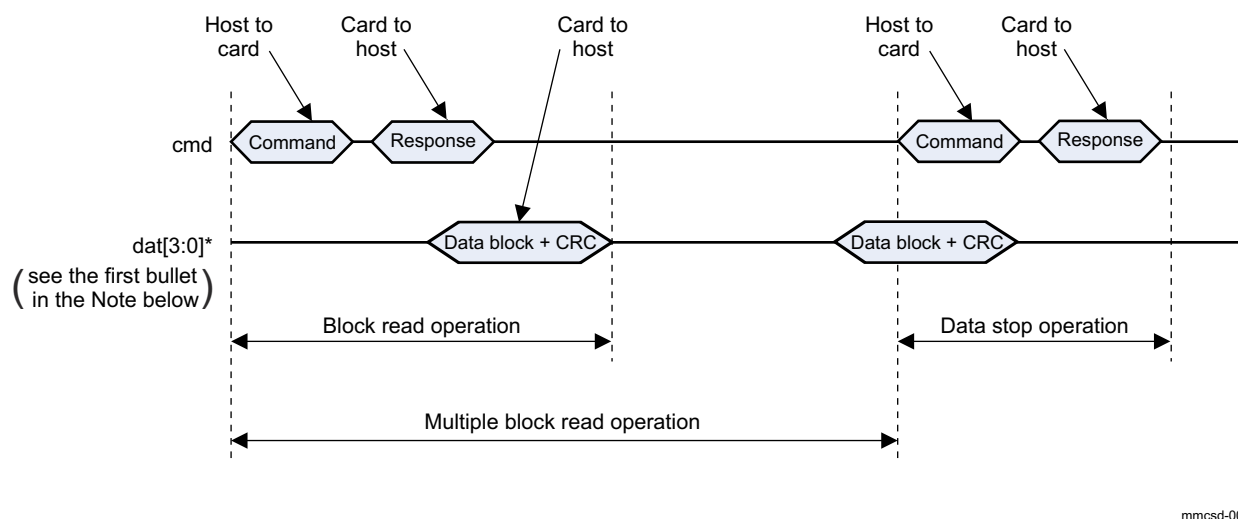


**Figure 12-287. Sequential Read Operation (MMCs Only)**

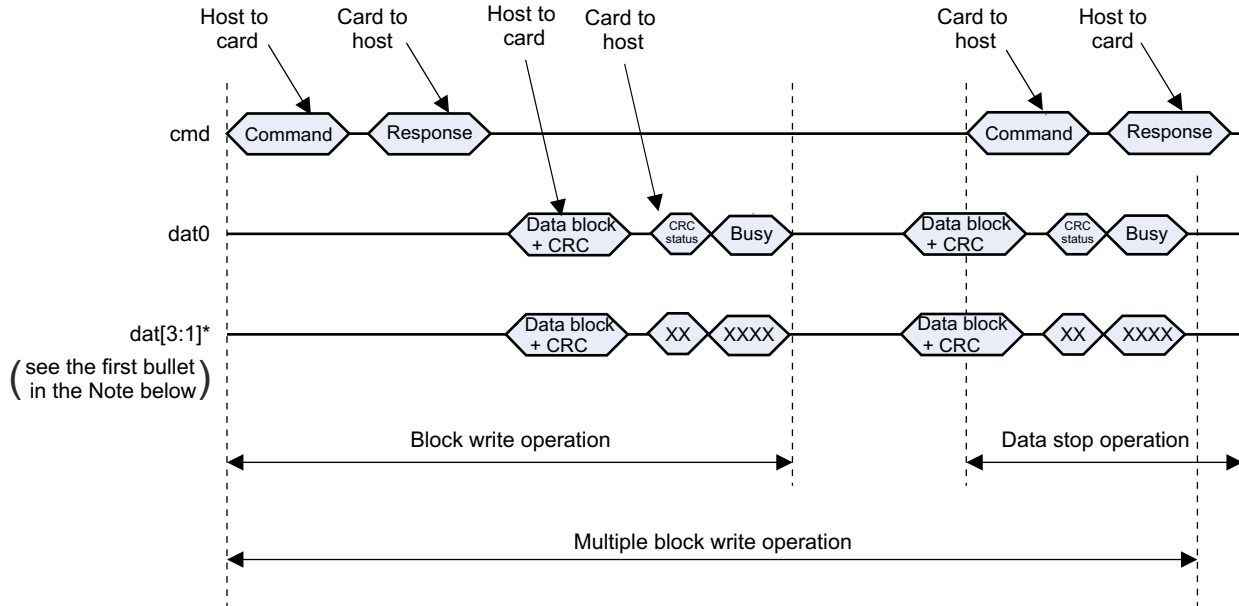


**Figure 12-288. Sequential Write Operation (MMCs Only)**

Figure 12-289 and Figure 12-290 show how multiple block-oriented operations are defined. A multiple block-oriented operation sends a data block plus CRC bits. The transfer terminates when a stop command follows on the CMD line. These operations are available for all kinds of devices (MMC, SD, SDIO).



**Figure 12-289. Multiple Block Read Operation**



mmcsd-009

**Figure 12-290. Multiple Block Write Operation With Card Busy Signal**

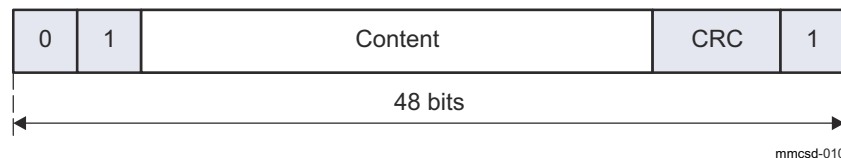
**Note**

- The card busy signal is not always generated by the card. Refer to [Figure 12-289](#) and [Figure 12-290](#), that show a particular case.
- Software must perform a software reset (see MMCS00\_SOFTWARE\_RESET / MMCS12\_SOFTWARE\_RESET register) after a data time-out to ensure that MMCi\_CLK is stopped.
- For multiblock transfer, and especially for MMC devices, a transfer can be aborted without using a stop command. If a CMD23 is used before data transfer to define the number of blocks that will be transferred, then the transfer stops automatically after the last block (if the MMCs supports this feature).

**12.3.6.2.1.2 Data Format**

**12.3.6.2.1.2.1 Coding Scheme for Command Token**

Command tokens always start with 0 and end with 1. The second bit is a transmitter bit: 1 for a host command. The content is the command index (coded by 6 bits) and an argument (for example, an address), coded by 32 bits. The content is protected by 7-bit CRC checksum (see [Figure 12-291](#)).



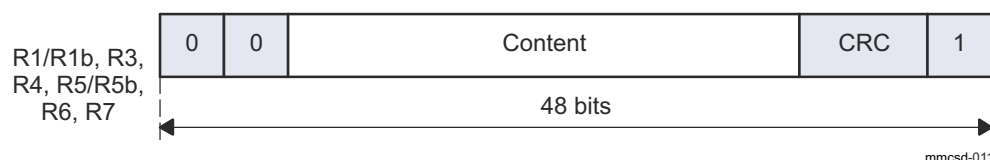
mmcsd-010

**Figure 12-291. Command Token Format**

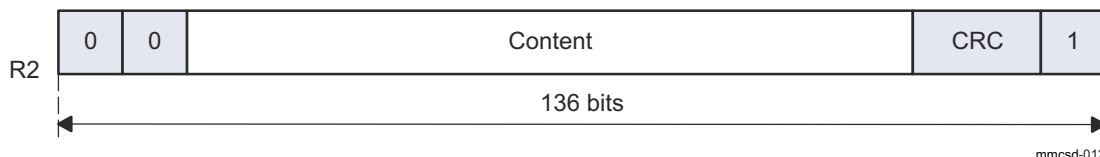
**12.3.6.2.1.2.2 Coding Scheme for Response Token**

Response tokens always start with 0 and end with 1. The second bit is a transmitter bit: 0 for a card response. The content is different for each type of response (R1, R2, R3, R4, and R5, R6, R7 [for SD]) and the content is protected by 7-bit CRC checksum (see [Figure 12-292](#) and [Figure 12-293](#)). Depending on the type of commands sent to the card, the MMCS00\_COMMAND / MMCS12\_COMMAND register must be configured differently to

avoid false CRC or index errors to be flagged on command response (see [Table 12-380](#)). For more information about response types, see the *Multimedia Card System Specification*, *SD Memory Card Specifications*, and *SDIO Card Specification (Part E1)*.



**Figure 12-292. Response Token Format (R1/R1b, R3, R4, R5/R5b, R6, R7)**



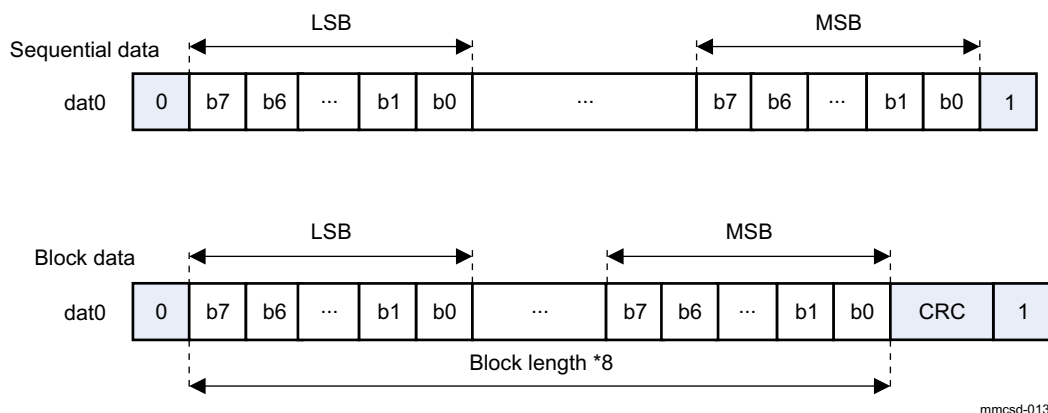
**Figure 12-293. Response Token Format (R2)**

**Table 12-380. Relationship Between Configuration and Name of Response Type**

Response Type MMCSD0_COMMAND[1-0] RESP_TYPE_SEL / MMCSD12_COMMAND[1-0] RESP_TYPE_SEL	Index Check Enable MMCSD0_COMMAND[4] CMD_INDEX_CHK_ENA / MMCSD12_COMMAND[4] CMD_INDEX_CHK_ENA	CRC Check Enable MMCSD0_COMMAND[3] CMD_CRC_CHK_ENA / MMCSD12_COMMAND[3] CMD_CRC_CHK_ENA	Name of Response Type
00	0	0	No response
01	0	1	R2
10	0	0	R3 (R4 for SD cards)
10	1	1	R1, R6, R5, (R7 for SD cards)
11	1	1	R1b, R5b

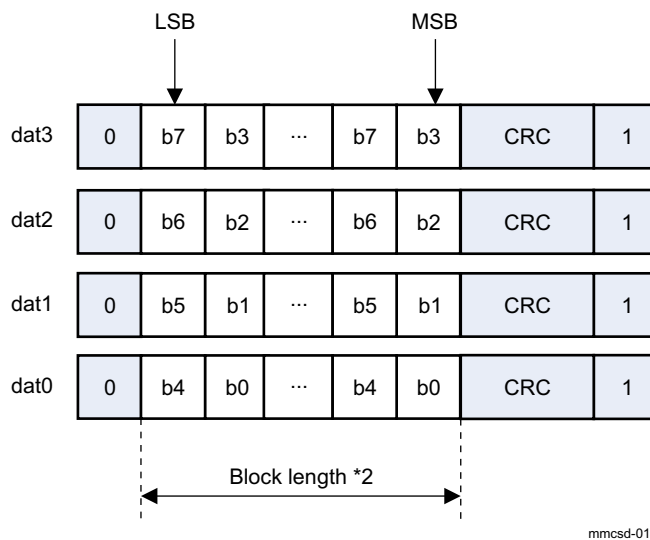
### 12.3.6.2.1.2.3 Coding Scheme for Data Token

Data tokens always start with 0 and end with 1 (see [Figure 12-294](#) through [Figure 12-296](#)).

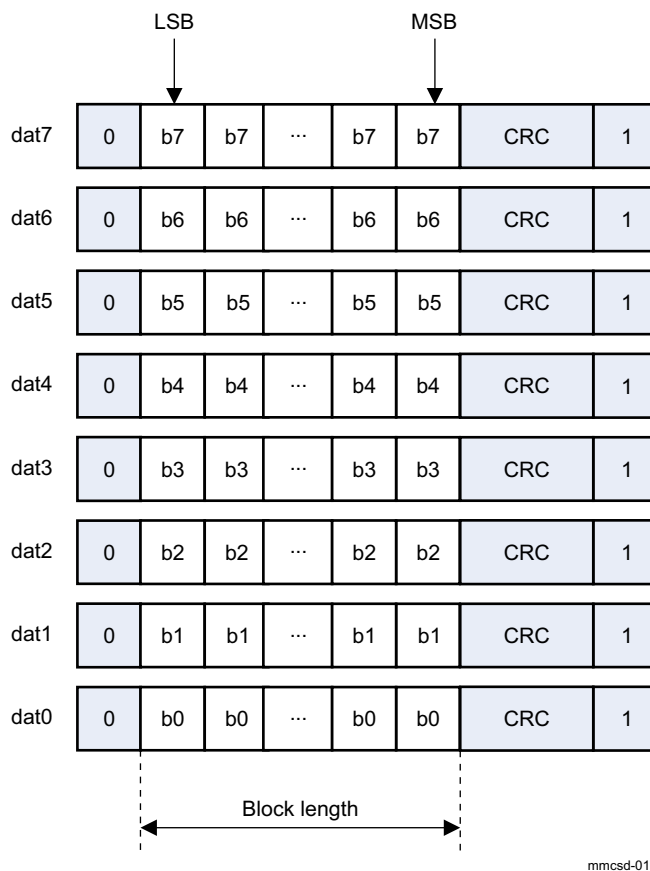


**Figure 12-294. Data Token Format for 1-Bit Transfers**





**Figure 12-295. Data Token Format for 4-Bit Transfers**



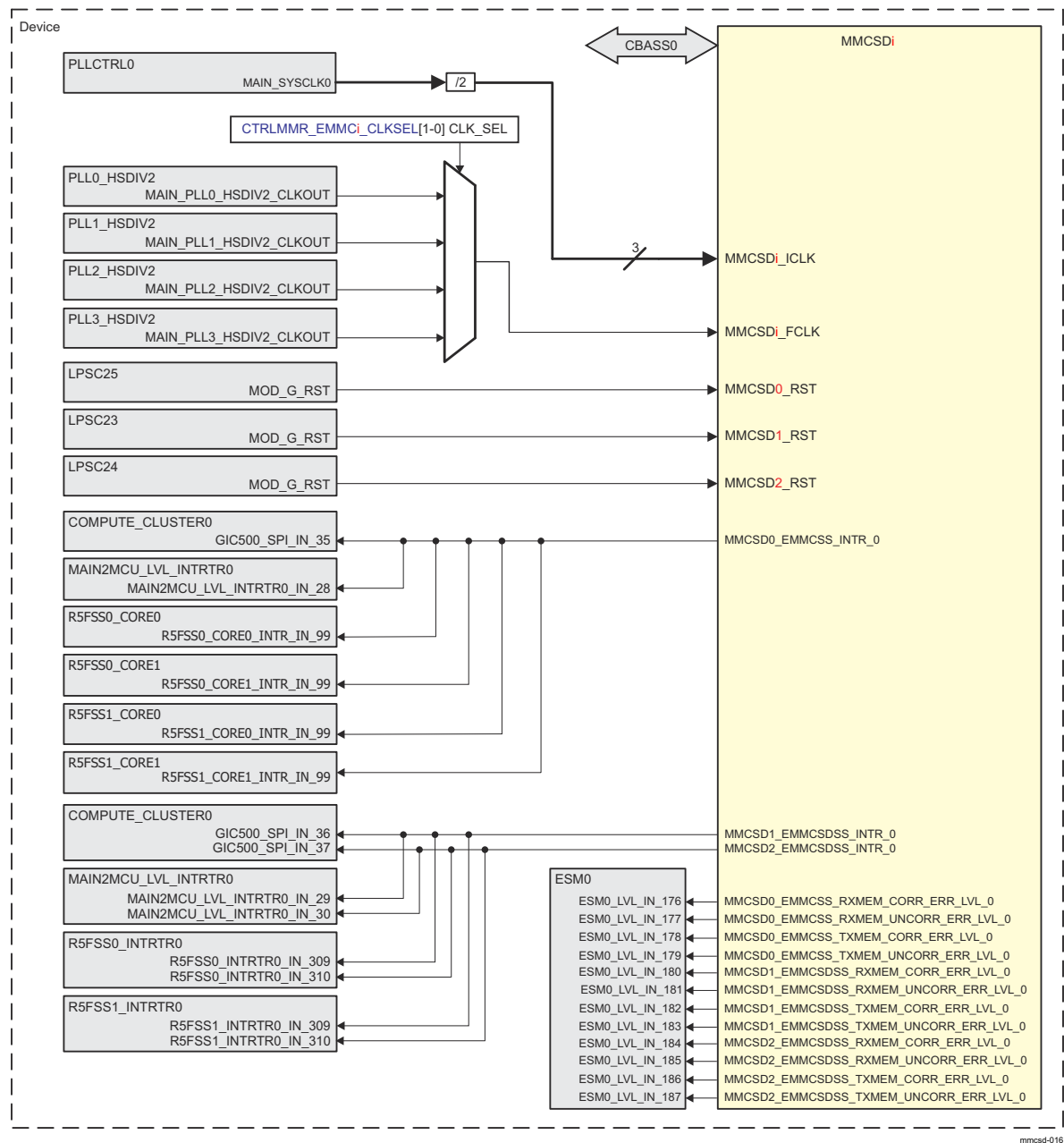
**Figure 12-296. Data Token Format for 8-Bit Transfers**

### 12.3.6.3 MMCSD Integration

This section describes the MMCSD integration in the device, including information about clocks, resets, and hardware requests.

#### 12.3.6.3.1 MMCSD Integration in MAIN Domain

There are three MMCSD modules integrated in the device MAIN domain - MMCSD0, MMCSD1 and MMCSD2. [Figure 12-297](#) shows the integration of MMCSD0, MMCSD1 and MMCSD2.



mmcscd-016

A.  $i = 0 \text{ to } 2$

**Table 12-381. MMCSD Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
MMCSD0	PSC0	PD0	LPSC25	CBASS0
MMCSD1	PSC0	PD0	LPSC23	CBASS0
MMCSD2	PSC0	PD0	LPSC24	CBASS0

**Table 12-382. MMCSD Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
MMCSD0	MMCSD0_ICLK	MAIN_SYSCLK0/2	PLLCTRL0	MMCSD0 Interface Clock
	MMCSD0_FCLK	MAIN_PLL0_HSDIV2_CLKOUT (MMCSD0 default selection)	PLL0_HSDIV2	MMCSD0 Functional Clock (for more information about clock multiplexing, see CTRLMMR_EMMC0_CLKSEL[1-0] CLK_SEL bit field in <i>Control Module (CTRL_MMR)</i> . Default: CTRLMMR_EMMC0_CLKSEL[1-0] CLK_SEL = 0, MAIN_PLL0_HSDIV2_CLKOUT is selected)
		MAIN_PLL1_HSDIV2_CLKOUT (MMCSD1 default selection)	PLL1_HSDIV2	
		MAIN_PLL2_HSDIV2_CLKOUT (MMCSD2 default selection)	PLL2_HSDIV2	
		MAIN_PLL3_HSDIV2_CLKOUT (can be used by any MMCSD)	PLL3_HSDIV2	
MMCSD1	MMCSD1_ICLK	MAIN_SYSCLK0/2	PLLCTRL0	MMCSD1 Interface Clock
	MMCSD1_FCLK	MAIN_PLL0_HSDIV2_CLKOUT (MMCSD0 default selection)	PLL0_HSDIV2	MMCSD1 Functional Clock (for more information about clock multiplexing, see CTRLMMR_EMMC1_CLKSEL[1-0] CLK_SEL bit field in <i>Control Module (CTRL_MMR)</i> . Default: CTRLMMR_EMMC1_CLKSEL[1-0] CLK_SEL = 0, MAIN_PLL0_HSDIV2_CLKOUT is selected)
		MAIN_PLL1_HSDIV2_CLKOUT (MMCSD1 default selection)	PLL1_HSDIV2	
		MAIN_PLL2_HSDIV2_CLKOUT (MMCSD2 default selection)	PLL2_HSDIV2	
		MAIN_PLL3_HSDIV2_CLKOUT (can be used by any MMCSD)	PLL3_HSDIV2	
MMCSD2	MMCSD2_ICLK	MAIN_SYSCLK0/2	PLLCTRL0	MMCSD2 Interface Clock
	MMCSD2_FCLK	MAIN_PLL0_HSDIV2_CLKOUT (MMCSD0 default selection)	PLL0_HSDIV2	MMCSD2 Functional Clock (for more information about clock multiplexing, see CTRLMMR_EMMC2_CLKSEL[1-0] CLK_SEL bit field in <i>Control Module (CTRL_MMR)</i> . Default: CTRLMMR_EMMC2_CLKSEL[1-0] CLK_SEL = 0, MAIN_PLL0_HSDIV2_CLKOUT is selected)
		MAIN_PLL1_HSDIV2_CLKOUT (MMCSD1 default selection)	PLL1_HSDIV2	
		MAIN_PLL2_HSDIV2_CLKOUT (MMCSD2 default selection)	PLL2_HSDIV2	
		MAIN_PLL3_HSDIV2_CLKOUT (can be used by any MMCSD)	PLL3_HSDIV2	
Resets				

**Table 12-382. MMCSD Clocks and Resets (continued)**

Module Instance	Module Reset Input	Source Reset Signal	Source	Description
MMCSD0	MMCSD0_RST	MOD_G_RST	LPSC25	MMCSD0 Asynchronous Reset
MMCSD1	MMCSD1_RST	MOD_G_RST	LPSC23	MMCSD1 Asynchronous Reset
MMCSD2	MMCSD2_RST	MOD_G_RST	LPSC24	MMCSD2 Asynchronous Reset

**Table 12-383. MMCSD Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MMCSD0	MMCSD0_EMMCSS_INTR_0	GIC500_SPI_IN_35	COMPUTE_CLUSTER0	MMCSD0 Interrupt Request	Level
		MAIN2MCU_LVL_INTRTR0_IN_28	MAIN2MCU_LVL_INTRTR0		
		R5FSS0_CORE0_INTR_IN_99	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_99	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_99	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_99	R5FSS1_CORE1		
	MMCSD0_EMMCSS_RXMEM_CORR_ERR_LVL_0	ESM0_LVL_IN_176	ESM0	MMCSD0 Receive ECC Correctable Error Interrupt Request	Level
	MMCSD0_EMMCSS_RXMEM_UNCORR_ERR_LVL_0	ESM0_LVL_IN_177	ESM0	MMCSD0 Receive ECC Uncorrectable Error Interrupt Request	
	MMCSD0_EMMCSS_TXMEM_CORR_ERR_LVL_0	ESM0_LVL_IN_178	ESM0	MMCSD0 Transmit ECC Correctable Error Interrupt Request	
	MMCSD0_EMMCSS_TXMEM_UNCORR_ERR_LVL_0	ESM0_LVL_IN_179	ESM0	MMCSD0 Transmit ECC Uncorrectable Error Interrupt Request	
MMCSD1	MMCSD1_EMMCSDSS_INTR_0	GIC500_SPI_IN_36	COMPUTE_CLUSTER0	MMCSD1 Interrupt Request	Level
		MAIN2MCU_LVL_INTRTR0_IN_29	MAIN2MCU_LVL_INTRTR0		
		R5FSS0_INTRTR0_IN_309	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_309	R5FSS1_INTRTR0		

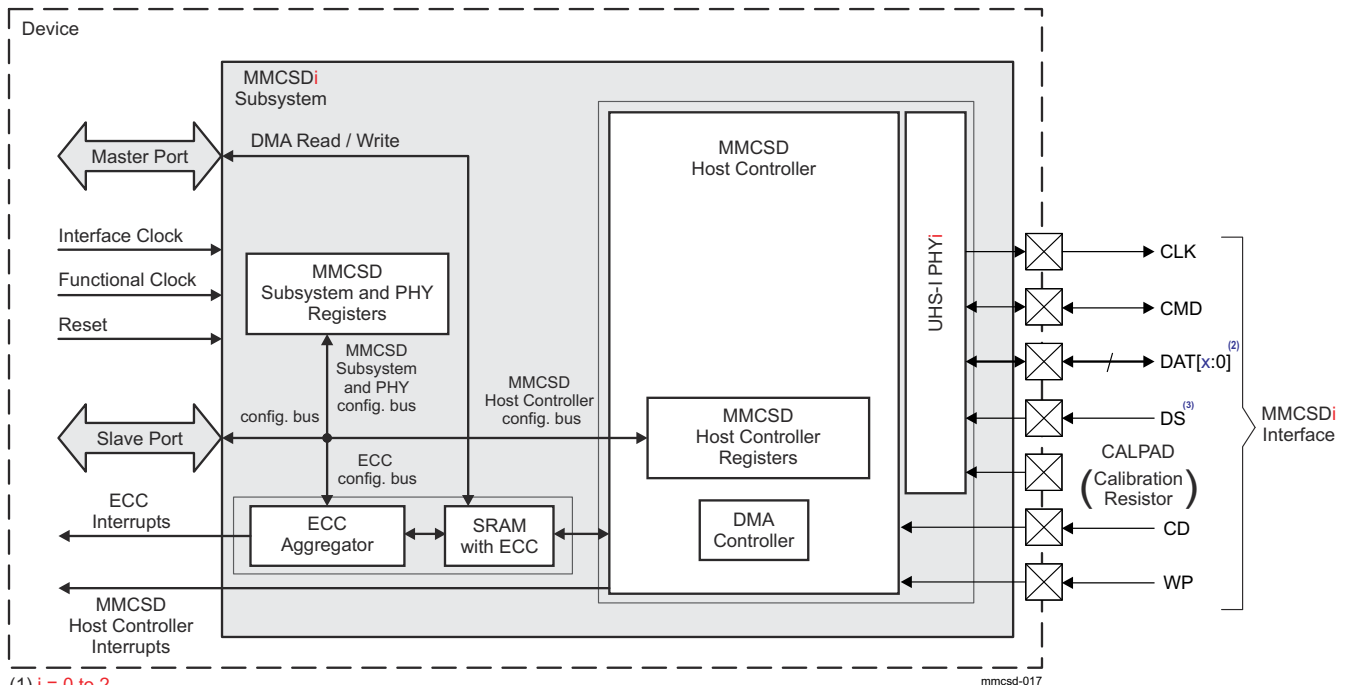
**Table 12-383. MMCSD Hardware Requests (continued)**

MMCSD1	MMCSD1_EMMCSDSS_RXMEM_CORR_ERR_LVL_0	ESM0_LVL_IN_180	ESM0	MMCSD1 Receive ECC Correctable Error Interrupt Request	Level
	MMCSD1_EMMCSDSS_RXMEM_UNCORR_ERR_LVL_0	ESM0_LVL_IN_181	ESM0	MMCSD1 Receive ECC Uncorrectable Error Interrupt Request	
	MMCSD1_EMMCSDSS_TXMEM_CORR_ERR_LVL_0	ESM0_LVL_IN_182	ESM0	MMCSD1 Transmit ECC Correctable Error Interrupt Request	
	MMCSD1_EMMCSDSS_TXMEM_UNCORR_ERR_LVL_0	ESM0_LVL_IN_183	ESM0	MMCSD1 Transmit ECC Uncorrectable Error Interrupt Request	
	MMCSD2_EMMCSDSS_INTR_0	GIC500_SPI_IN_37	COMPUTE_CLUSTER0	MMCSD2 Interrupt Request	Level
		MAIN2MCU_LVL_INTRTR0_IN_30	MAIN2MCU_LVL_INTRTR0		
		R5FSS0_INTRTR0_IN_310	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_310	R5FSS1_INTRTR0		
	MMCSD2_EMMCSDSS_RXMEM_CORR_ERR_LVL_0	ESM0_LVL_IN_184	ESM0	MMCSD2 Receive ECC Correctable Error Interrupt Request	Level
	MMCSD2_EMMCSDSS_RXMEM_UNCORR_ERR_LVL_0	ESM0_LVL_IN_185	ESM0	MMCSD2 Receive ECC Uncorrectable Error Interrupt Request	
	MMCSD2_EMMCSDSS_TXMEM_CORR_ERR_LVL_0	ESM0_LVL_IN_186	ESM0	MMCSD2 Transmit ECC Correctable Error Interrupt Request	
	MMCSD2_EMMCSDSS_TXMEM_UNCORR_ERR_LVL_0	ESM0_LVL_IN_187	ESM0	MMCSD2 Transmit ECC Uncorrectable Error Interrupt Request	

## 12.3.6.4 MMCSD Functional Description

### 12.3.6.4.1 Block Diagram

Figure 12-298 shows the MMCSDi module block diagram (where i = 0 to 2).



- (1) i = 0 to 2  
 (2) x = 7 for MMCSD0 (8-bit wide data bus); x = 3 for MMCSD1/MMCSD2 (4-bit wide data bus);  
 (3) Used for HS400 mode (MMCSD0 Only);

**Figure 12-298. MMCSDi Module Block Diagram**

#### Basic Blocks:

- MMCSD Host Controller:** The MMCSD Host Controller is situated in the MMCSD Subsystem and provides accessibility to external MMC/SD/SDIO devices using a Programmed IO method or DMA data transfer method.
- UHS-I PHY:** The integrated UHS-I PHY provides an interface between the MMCSD Host Controller and external MMC/SD/SDIO devices.
- MMCSDi Interface:** The MMCSDi Interface includes all used interface pins (for more information, see [Table 12-379](#)).
- Master Port:** Provides a 64-bit wide read/write interface between the device and the MMCSD Subsystem internal SRAM.
- Slave Port:** Provides a 32-bit wide interface between the device and the MMCSD Subsystem and MMCSD Host Controller parts.
- MMCSD Host Controller Registers:** This block includes set of all MMCSD Host Controller registers.
- MMCSD Subsystem and PHY Registers:** This block implements memory-mapped registers at the MMCSD Subsystem level and memory-mapped registers to control and program the MMCSD PHY.
- ECC Aggregator:** The ECC Aggregator block facilitates aggregating and reporting internal SRAM ECC errors (for more information, see [Section 12.3.6.4.4, ECC Support](#)).
- Internal SRAM with ECC:** The internal SRAM block is used for data storage during read/write transactions.
- DMA Controller:** Manages the data transfer between the device memory and the MMCSD Subsystem internal SRAM.
- System Clocks:** There are asynchronous relationship between the interface (system) clock and functional clock (for more information, see [Table 12-382](#)).
- System Reset:** The reset to the MMCSD Subsystem provides reset to the all MMCSD Subsystem parts (for more information, see [Table 12-382](#)).

- Interrupts: The MMCSD Subsystem sources one MMCSD Host Controller interrupt and four ECC Aggregator interrupts (for more information, see [Table 12-383](#) and [Section 12.3.6.4.3, Interrupt Requests](#)).

For more information about all MMCSD registers, see *MMCSD Registers*.

The MMCSD Host Controller can use a Programmed IO method (PIO) or DMA data transfer method to access external MMC/SD/SDIO devices.

The slave port is used for device CPU access to the MMCSD Host Controller register set. The MMCSD Host Controller register set provides the communication between the device CPU and the MMCSD Host Controller. In PIO method the device CPU transfers data using the MMCSD0\_DATA\_PORT / MMCSD12\_DATA\_PORT register. The data flow from the device memory to the external MMC/SD/SDIO device (and vice versa) is through the slave port, the MMCSD0\_DATA\_PORT / MMCSD12\_DATA\_PORT register in the MMCSD Host Controller, and the PHY.

The master port is used for connection to the DMA controller (for more information about ADMA support, see [Section 12.3.6.4.5, Advanced DMA](#) and MMCSD0\_CAPABILITIES / MMCSD12\_CAPABILITIES register). In DMA data transfer method the DMA controller uses the master port to transfer data between the device memory and the MMCSD Subsystem internal SRAM. The other side of the internal SRAM is connected to the MMCSD Host Controller. The data flow from the device memory to the external MMC/SD/SDIO device (and vice versa) is through the master port, internal SRAM, MMCSD Host Controller, and the PHY. The DMA controller manages the read/write operations from/to the internal SRAM without device CPU intervention.

#### 12.3.6.4.2 Memory Regions

[Table 12-384](#) shows MMCSDi module memory map.

**Table 12-384. MMCSDi Module Memory Map**

Address Range			Size	Description
MMCSD0	MMCSD1	MMCSD2		
0x04F80000 - 0x04F80FFF	0x04FB0000 - 0x04FB0FFF	0x04F98000 - 0x04F98FFF	4 KB	MMCSD Host Controller registers region
0x04F88000 - 0x04F883FF	0x04FB8000 - 0x04FB83FF	0x04F90000 - 0x04F903FF	1 KB	MMCSD Subsystem registers region
0x02A24000 - 0x02A243FF	0x02A26000 - 0x02A263FF	0x02A71000 - 0x02A713FF	1 KB	ECC Aggregator Receive RAM registers region
0x02A25000 - 0x02A253FF	0x02A27000 - 0x02A273FF	0x02A70000 - 0x02A703FF	1 KB	ECC Aggregator Transmit RAM registers region

#### 12.3.6.4.3 Interrupt Requests

Each MMCSD instance sources one active high level MMCSD Host Controller interrupt and four active high level ECC Aggregator interrupts (see [Table 12-383](#)). The MMCSD Host Controller interrupt is generated based on the bit values in the MMCSD0\_NORMAL\_INTR\_STS / MMCSD12\_NORMAL\_INTR\_STS and MMCSD0\_NORMAL\_INTR\_STS\_ENA / MMCSD12\_NORMAL\_INTR\_STS\_ENA registers. The ECC Aggregator interrupts are generated based on the ECC errors - single and double bit errors (for more information about ECC Aggregator interrupts, see [Section 12.3.6.4.4, ECC Support](#)).

#### 12.3.6.4.4 ECC Support

The Error Correcting Code (ECC) is a mechanism for providing increased system reliability via reduction of memory software errors by allowing single bit errors to be detected and corrected (SEC) and double bit errors to be detected (DED).

There are two SRAM memories (transmit SRAM and receive SRAM) within each MMCSD instance. These two memories are ECC protected.

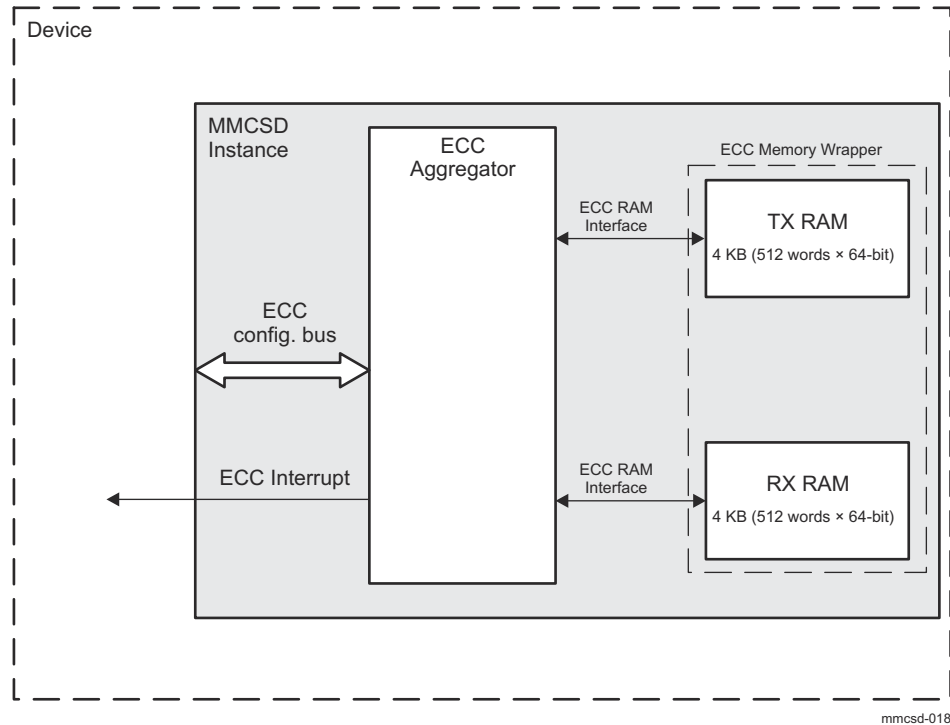
The SEC logic detects and corrects a single bit error (single bit error per ECC word or per ECC data segment). The DED logic only detects (does not correct) double errors (double bit errors per ECC word or per ECC data segment).



#### 12.3.6.4.4.1 ECC Aggregator

There are two SRAMs (each with size 4 KB - 512 words × 64-bit) in each MMCSD Subsystem. One SRAM dedicated for transmit and one SRAM dedicated for receive operations.

Figure 12-299 shows the ECC Aggregator block diagram.



**Figure 12-299. ECC Aggregator Block Diagram**

#### Note

For more information about ECC Aggregator, refer to *ECC Aggregator*.

For more information about ECC Aggregator Registers, refer to *MMCSD Registers*.

#### 12.3.6.4.5 Advanced DMA

The MMCSD modules support SD Advanced DMA – ADMA2 and ADMA3.

#### Note

For more information, refer to the SD Association specification titled "*SD Host Controller Simplified Specification, Part A2, Version 4.10*".

#### 12.3.6.4.6 eMMC PHY BIST

##### 12.3.6.4.6.1 BIST Overview

To verify the eMMC5.1 PHY functionality, the BIST function is built on top of the eMMC5.1 PHY that can either be controlled from SOC registers or from special Vendor Registers defined in the TI's eMMC5.1 Host Controller. The functionality of the BIST is to be able to test various functions of the PHY (without using the eMMC protocol/ software stack) and provide the results of the testing to the diagnostic software.

The 128-byte Test Pattern is as follows:

[illegible]

In non-HS200 Modes, the testing is performed for one time with programmed Tx Phase and Rx Phase values. In HS200 Mode, the testing is performed 32-times and it is expected that some of the iterations will have data mismatch because of the RxPhase Timing. The BIST engine iterates through all 32-phases of Rx Clock and provides the result of data/cmd line compare for each of the phases in a 32-bit status vector.

This is the High Speed (HS) mode where the eMMC CLK is set to 50 MHz. To emulate the Interface timing, a Hold time is inserted on Transmit data lines by using the Phase shifted Tx Clock. The amount of phase shift can be from 1 to 16 Taps. In this mode, the TX Clock Phase shift is being performed by using one of the first 16-taps of the DLL TXCLK Phases. The Phase shifting on the RX path is disabled in this mode.

#### **12.3.6.4.6.2.3 HS Mode with TXDLY using Delay Chain**

This is the High Speed (HS) mode where the eMMC CLK is set to 50 MHz. To emulate the Interface timing, a Hold time is inserted on Transmit data lines by using the Phase shifted Tx Clock. The amount of phase shift can be from 1 to 16 Taps. In this mode, the TX Clock Phase shift is being performed by using one of the first 16-taps of the TX Delay Chain Phases. The Phase shifting on the RX path is disabled in this mode.

The DLL is disabled in this mode.

#### **12.3.6.4.6.2.4 DDR50 Mode with TXDLY using DLL**

This is the DDR50 mode where the eMMC CLK is set to 50 MHz. The data is driven on both the edges of the clock. To emulate the Interface timing, a Hold time is inserted on Transmit data lines by using the Phase shifted TX Clock. The amount of phase shift can be from 1 to 16 Taps. In this mode, the TX Clock Phase shift is being performed by using one of the first 16-taps of the DLL TXCLK Phases. The Phase shifting on the RX path is disabled in this mode.

#### **12.3.6.4.6.2.5 DDR50 Mode with TXDLY using Delay Chain**

This is the DDR50 mode where the eMMC CLK is set to 50 MHz. The data is driven on both the edges of the clock. To emulate the Interface timing, a Hold time is inserted on Transmit data lines by using the Phase shifted Tx Clock. The amount of phase shift can be from 1 to 16 Taps. In this mode, the Tx Clock Phase shift is being performed by using one of the first 16-taps of the Tx Delay Chain Phases. The Phase shifting on the RX path is disabled in this mode. The DLL is disabled in this mode.

#### **12.3.6.4.6.2.6 HS200 Mode with TX/RXDLY using DLL**

This is the HS200 mode where the eMMC CLK is set to 200 MHz. To emulate the Interface timing, a small Hold time is inserted on Transmit data lines by using the Phase shifted TX Clock. The amount of phase shift can be from 1 to 16 Taps. The RxClock Phase is adjusted from 1 to 32 Phases to check the data at different phases. In this mode, the TX Phase shift is being performed by using one of the first 16-taps of the DLL TXCLK Phases and the RX Phase shift is being performed by using one of the 32 taps of the DLL RXCLK Phases.

#### **12.3.6.4.6.2.7 HS200 Mode with TX/RXDLY using Delay Chain**

This is the HS200 mode where the eMMC CLK is set to 200 MHz. To emulate the Interface timing, a small Hold time is inserted on Transmit data lines by using the Phase shifted Tx Clock. The amount of phase shift can be from 1 to 16 Taps. The Rx Clock Phase is adjusted from 1 to 32 Phases to check the data at different phases. In this mode, the TX Phase shift is being performed by using one of the first 16-taps of the Delay Chain Phases and the RX Phase shift is being performed by using one of the 32 taps of the Delay Chain Phases.

The DLL is disabled in this mode.

#### **12.3.6.4.6.2.8 HS400 Mode**

In this mode, the Receive data is captured using the STRB line. To support this mode, the AFE is configured such that the STRB is toggled during the time when the fixed pattern transmission (including the START/END). The Receive STRB line is phase shifted by 90 and 180 degrees by the DLL. As the data is captured using the STRB, there is no need to enable the RX Clock Phases, and these can be disabled. To maintain the timing, the TX Phase can be adjusted accordingly.

#### **12.3.6.4.6.3 BIST Functionality**

The BIST engine inside the eMMC PHY leverages the Tuning Pattern to verify the functionality of the PHY (including the IOs and the DLL). The BIST engine is controlled from SOC and the eMMC Host Controller should be disabled during the BIST Mode. The BIST can be run at different modes (Clock Speeds) and the result of the BIST operation is presented to the SOC.

In normal mode, the PHY Transmit Data path operates on the MMCS0\_CLOCK\_CONTROL[15:8].SDCLK\_FRQSEL. This clock is provided by the eMMC Host Controller and the frequency of this clock is varied based on the operating mode. The following are the various frequencies used in various modes.

- Default Speed: 25 MHz
- High Speed: 25 MHz
- DDR52: 50 MHz
- HS200: 200 MHz
- HS400: 200 MHz

This clock is generated by the eMMC Host Controller and under direct control of the software through the Clock Control Register @0x2C of the Host Controller Register Set. The eMMC Host Controller uses the XIN\_CLK (the reference clock to the Host Controller, which is usually 200 MHz) and uses the Clock Divider Value and the Clock Enable Control.

The BIST Engine runs on the same MMCSD0\_CLOCK\_CONTROL[15:8].SDCLK\_FRQSEL. Though the Host Controller is disabled during the BIST operation, the software should program the Clock Control Register so that the MMCSD0\_CLOCK\_CONTROL[15:8].SDCLK\_FRQSEL is generated for the BIST engine.

#### 12.3.6.4.6.4 Signal Interface

The BIST logic is controlled by a set of signals which can be from a set of registers that are software accessible. The following are the signals for BIST Functions.

**Table 12-385. BIST Logic Signals**

Signal	Direction	Description
MMCSD0_SS_PHY_CTRL_6_REG[31].BISTENABLE	Input	<b>BIST Enable</b> When set the BIST Controller controls the DFE and overwrites the data path signals from the Host Controller Interface. When this is cleared, the Host Controller signals are connected to the DFE and BIST is disabled.
MMCSD0_SS_PHY_CTRL_6_REG[27:24].BISTMODE	Input	<b>BIST Mode</b> This selects one of the specified BIST Modes for running the BIST sequence. The following are the encoding for the mode: 4'b0000: HS200 Mode with DLL 4'b0001: HS200 Mode with DLY_CHAIN 4'b0010: HS400 Mode (STROBE Mode) 4'b0011: DDR52 Mode with DLL 4'b0100: DDR52 Mode with DLY_CHAIN 4'b0101: HS Mode with DLL 4'b0110: HS Mode with DLY_CHAIN 4'b0111: DS Mode (with DLY_CHAIN)
MMCSD0_SS_PHY_CTRL_6_REG[30].BISTSTART	Input	<b>BIST Start</b> The rising edge of this signal is used to start the BIST function on the specified BIST Mode. When the BIST is complete the MMCSD0_SS_PHY_STAT_1_REG[3].BISTDONE output is asserted. Before setting this signal, the SOC must make sure that the MMCSD0_SS_PHY_STAT_1_REG[3].BISTDONE is low (deasserted).
MMCSD0_SS_PHY_STAT_2_REG[31:0].BISTSTATUS	Output	<b>BIST Status</b> This is the status of the BIST Testing. In non HS200 Mode, the Bit[0] indicates whether the BIST Passed or not. In HS200 modes, the 32-bit indicates the Pattern Match at each of the 32-taps on RxClk Path.

**Table 12-385. BIST Logic Signals (continued)**

Signal	Direction	Description
MMCSDB0_SS_PHY_STAT_1_REG[3].BISTDONE	Output	<b>BIST Done</b> This is asserted when the BIST completes its sequence. The result of the test is presented in the MMCSDB0_SS_PHY_STAT_2_REG[31:0].BISTSTATUS output. This is asserted until the MMCSDB0_SS_PHY_CTRL_6_REG[30].BISTSTART is deasserted at which point the MMCSDB0_SS_PHY_STAT_1_REG[3].BISTDONE is also cleared.

#### 12.3.6.4.6.5 Programming Flow

For the BIST operation, the OD\_EN/REN/PU/OD controls to the CMD/STRB/DAT lines should be programmed as follows:

- OD\_EN: 1'b0
- REN: 1'b0
- PU: 1'b1
- OD: 1'b0

##### 12.3.6.4.6.5.1 DS Mode

The following are the various settings required for performing the BIST in Default Speed timing.

##### 12.3.6.4.6.5.1.1 Configuration

- MMCSDB0\_CLOCK\_CONTROL[15:8].SDCLK\_FRQSEL should be set to 25 MHz.
- PHY's DLL should be disabled.
- MMCSDB0\_SS\_PHY\_CTRL\_6\_REG[27:24].BISTMODE should be set to 4'b0111.
- MMCSDB0\_SS\_PHY\_CTRL\_6\_REG[31].BISTENABLE should be set to 1'b1.
- MMCSDB0\_SS\_PHY\_CTRL\_4\_REG[20].OTAPDLYENA to 1'b0.
- MMCSDB0\_SS\_PHY\_CTRL\_4\_REG[15:12].OTAPDLYSEL to 4'b0000.
- MMCSDB0\_SS\_PHY\_CTRL\_4\_REG[8].ITAPDLYENA to 1'b0
- MMCSDB0\_SS\_PHY\_CTRL\_4\_REG[4:0].ITAPDLYSEL to 5'b00000
- MMCSDB0\_SS\_PHY\_CTRL\_4\_REG[9].ITAPCHGWIN to 1'b0

##### 12.3.6.4.6.5.1.2 BIST Programming

1. Set the MMCSDB0\_SS\_PHY\_CTRL\_6\_REG[30].BISTSTART to 1'b1.
2. Wait until the MMCSDB0\_SS\_PHY\_STAT\_1\_REG[3].BISTDONE is set to 1'b1.
3. Read the MMCSDB0\_SS\_PHY\_STAT\_2\_REG[31:0].BISTSTATUS[0] Bit and check the value.

If (MMCSDB0\_SS\_PHY\_STAT\_2\_REG[31:0].BISTSTATUS\*0+==1'b1) then

BIST PASS

else

BIST FAIL;

endif

4. Clear the MMCSDB0\_SS\_PHY\_CTRL\_6\_REG[30].BISTSTART to 1'b0.

##### 12.3.6.4.6.5.2 HS Mode with DLY\_CHAIN

The following are the various settings required for performing the BIST in HS Mode with DLY Chain.

##### 12.3.6.4.6.5.2.1 Configuration

- MMCSDB0\_CLOCK\_CONTROL[15:8].SDCLK\_FRQSEL should be set to 50 MHz.
- PHY's DLL should be disabled.

- MMCSD0\_SS\_PHY\_CTRL\_6\_REG[27:24].BISTMODE should be set to 4'b0110.
- MMCSD0\_SS\_PHY\_CTRL\_6\_REG[31].BISTENABLE should be set to 1'b1.
- MMCSD0\_SS\_PHY\_CTRL\_4\_REG[20].OTAPDLYENA to 1'b1.
- MMCSD0\_SS\_PHY\_CTRL\_4\_REG[15:12].OTAPDLYSEL to on of the tap Value based on Timing Closure.
- MMCSD0\_SS\_PHY\_CTRL\_4\_REG[8].ITAPDLYENA\* to 1'b0.
- MMCSD0\_SS\_PHY\_CTRL\_4\_REG[4:0].ITAPDLYSEL\* to 5'b00000
- MMCSD0\_SS\_PHY\_CTRL\_4\_REG[9].ITAPCHGWIN to 1'b0

\* If Timing closure requires usage of Input Tap Enable and Tap Select should be set appropriately.

#### 12.3.6.4.6.5.2.2 BIST Programming

1. Set the MMCSD0\_SS\_PHY\_CTRL\_6\_REG[30].BISTSTART to 1'b1.
2. Wait until the MMCSD0\_SS\_PHY\_STAT\_1\_REG[3].BISTDONE is set to 1'b1.
3. Read the MMCSD0\_SS\_PHY\_STAT\_2\_REG[31:0].BISTSTATUS[0] Bit and check the value.

If (MMCSD0\_SS\_PHY\_STAT\_2\_REG[31:0].BISTSTATUS\*0+==1'b1) then

BIST PASS

else

BIST FAIL;

endif

4. Clear the MMCSD0\_SS\_PHY\_CTRL\_6\_REG[30].BISTSTART to 1'b0.

#### 12.3.6.4.6.5.3 HS Mode with DLL

The following are the various settings required for performing the BIST in HS Mode with DLL.

##### 12.3.6.4.6.5.3.1 Configuration

- MMCSD0\_CLOCK\_CONTROL[15:8].SDCLK\_FRQSEL should be set to 50 MHz.
- PHY's DLL should be Enabled.
- MMCSD0\_SS\_PHY\_CTRL\_6\_REG[27:24].BISTMODE should be set to 4'b0101.
- MMCSD0\_SS\_PHY\_CTRL\_6\_REG[31].BISTENABLE should be set to 1'b1.
- MMCSD0\_SS\_PHY\_CTRL\_4\_REG[20].OTAPDLYENA to 1'b1.
- MMCSD0\_SS\_PHY\_CTRL\_4\_REG[15:12].OTAPDLYSEL to on of the tap Value based on Timing Closure.
- MMCSD0\_SS\_PHY\_CTRL\_4\_REG[8].ITAPDLYENA\* to 1'b0.
- MMCSD0\_SS\_PHY\_CTRL\_4\_REG[4:0].ITAPDLYSEL\* to 5'b00000
- MMCSD0\_SS\_PHY\_CTRL\_4\_REG[9].ITAPCHGWIN to 1'b0
- MMCSD0\_SS\_PHY\_CTRL\_5\_REG[10:8].FRQSEL should be set for 50 MHz Mode.

\* If Timing closure requires usage of Input Tap Enable and Tap Select should be set appropriately.

##### 12.3.6.4.6.5.3.2 BIST Programming

1. Wait for MMCSD0\_SS\_PHY\_STAT\_1\_REG[0].DLLRDY to be 1'b1 (DLL is Ready)
2. Set the MMCSD0\_SS\_PHY\_CTRL\_6\_REG[30].BISTSTART to 1'b1.
3. Wait until the MMCSD0\_SS\_PHY\_STAT\_1\_REG[3].BISTDONE is set to 1'b1.
4. Read the MMCSD0\_SS\_PHY\_STAT\_2\_REG[31:0].BISTSTATUS[0] Bit and check the value.

If (MMCSD0\_SS\_PHY\_STAT\_2\_REG[31:0].BISTSTATUS\*0+==1'b1) then

BIST PASS

else

BIST FAIL;

endif

5. Clear the MMCSD0\_SS\_PHY\_CTRL\_6\_REG[30].BISTSTART to 1'b0.



#### 12.3.6.4.6.5.4 DDR52 Mode with DLY\_CHAIN

The following are the various settings required for performing the BIST in DDR52 Mode with DLY Chain.

##### 12.3.6.4.6.5.4.1 Configuration

- MMCSO0\_CLOCK\_CONTROL[15:8].SDCLK\_FRQSEL should be set to 50 MHz.
- PHY's DLL should be disabled.
- MMCSO0\_SS\_PHY\_CTRL\_6\_REG[27:24].BISTMODE should be set to 4'b0100.
- MMCSO0\_SS\_PHY\_CTRL\_6\_REG[31].BISTENABLE should be set to 1'b1.
- MMCSO0\_SS\_PHY\_CTRL\_4\_REG[20].OTAPDLYENA to 1'b1.
- MMCSO0\_SS\_PHY\_CTRL\_4\_REG[15:12].OTAPDLYSEL to on of the tap Value based on Timing Closure.
- MMCSO0\_SS\_PHY\_CTRL\_4\_REG[8].ITAPDLYENA\* to 1'b0.
- MMCSO0\_SS\_PHY\_CTRL\_4\_REG[4:0].ITAPDLYSEL\* to 5'b00000
- MMCSO0\_SS\_PHY\_CTRL\_4\_REG[9].ITAPCHGWIN to 1'b0

If Timing closure requires usage of Input Tap Enable and Tap Select should be set appropriately.

##### 12.3.6.4.6.5.4.2 BIST Programming

1. Set the MMCSO0\_SS\_PHY\_CTRL\_6\_REG[30].BISTSTART to 1'b1.
2. Wait until the MMCSO0\_SS\_PHY\_STAT\_1\_REG[3].BISTDONE is set to 1'b1.
3. Read the MMCSO0\_SS\_PHY\_STAT\_2\_REG[31:0].BISTSTATUS[0] Bit and check the value.

If (MMCSO0\_SS\_PHY\_STAT\_2\_REG[31:0].BISTSTATUS\*0+==1'b1) then

BIST PASS

else

BIST FAIL;

endif

4. Clear the MMCSO0\_SS\_PHY\_CTRL\_6\_REG[30].BISTSTART to 1'b0.

#### 12.3.6.4.6.5.5 DDR52 Mode with DLL

The following are the various settings required for performing the BIST in DDR52 Mode with DLL.

##### 12.3.6.4.6.5.5.1 Configuration

- MMCSO0\_CLOCK\_CONTROL[15:8].SDCLK\_FRQSEL should be set to 50 MHz
- PHY's DLL should be Enabled.
- MMCSO0\_SS\_PHY\_CTRL\_6\_REG[27:24].BISTMODE should be set to 4'b0011.
- MMCSO0\_SS\_PHY\_CTRL\_6\_REG[31].BISTENABLE should be set to 1'b1.
- MMCSO0\_SS\_PHY\_CTRL\_4\_REG[20].OTAPDLYENA to 1'b1.
- MMCSO0\_SS\_PHY\_CTRL\_4\_REG[15:12].OTAPDLYSEL to one of the tap values based on Timing Closure.
- MMCSO0\_SS\_PHY\_CTRL\_4\_REG[8].ITAPDLYENA\* to 1'b0.
- MMCSO0\_SS\_PHY\_CTRL\_4\_REG[4:0].ITAPDLYSEL\* to 5'b00000
- MMCSO0\_SS\_PHY\_CTRL\_4\_REG[9].ITAPCHGWIN to 1'b0
- MMCSO0\_SS\_PHY\_CTRL\_5\_REG[10:8].FRQSEL should be set for 50MHz Mode.

If Timing closure requires usage of Input Tap Enable and Tap Select should be set appropriately.

##### 12.3.6.4.6.5.5.2 BIST Programming

1. Wait for MMCSO0\_SS\_PHY\_STAT\_1\_REG[0].DLLRDY to be 1'b1 (DLL is Ready)
2. Set the MMCSO0\_SS\_PHY\_CTRL\_6\_REG[30].BISTSTART to 1'b1.
3. Wait until the MMCSO0\_SS\_PHY\_STAT\_1\_REG[3].BISTDONE is set to 1'b1.
4. Read the MMCSO0\_SS\_PHY\_STAT\_2\_REG[31:0].BISTSTATUS[0] Bit and check the value.

If (MMCSO0\_SS\_PHY\_STAT\_2\_REG[31:0].BISTSTATUS\*0+==1'b1) then

```
BIST PASS
```

```
else
```

```
BIST FAIL;
```

```
endif
```

5. Clear the MMCSD0\_SS\_PHY\_CTRL\_6\_REG[30].BISTSTART to 1'b0.

#### 12.3.6.4.6.5.6 HS200 Mode with DLY\_CHAIN

The following are the various settings required for performing the BIST in HS200 Mode with DLY Chain.

##### 12.3.6.4.6.5.6.1 Configuration

- MMCSD0\_CLOCK\_CONTROL[15:8].SDCLK\_FRQSEL should be set to 200 MHz.
- PHY's DLL should be disabled.
- MMCSD0\_SS\_PHY\_CTRL\_6\_REG[27:24].BISTMODE should be set to 4'b0001.
- MMCSD0\_SS\_PHY\_CTRL\_6\_REG[31].BISTENABLE should be set to 1'b1.
- MMCSD0\_SS\_PHY\_CTRL\_4\_REG[20].OTAPDLYENA to 1'b1.
- MMCSD0\_SS\_PHY\_CTRL\_4\_REG[15:12].OTAPDLYSEL to one of the tap values based on Timing Closure.
- MMCSD0\_SS\_PHY\_CTRL\_4\_REG[8].ITAPDLYENA to 1'b0
- MMCSD0\_SS\_PHY\_CTRL\_4\_REG[4:0].ITAPDLYSEL to 5'b00000
- MMCSD0\_SS\_PHY\_CTRL\_4\_REG[9].ITAPCHGWIN to 1'b0

\*In this mode, the BIST Engine cycles thru all the 32 taps, so there is no need of controlling the Input Tap Delay.

##### 12.3.6.4.6.5.6.2 BIST Programming

1. Set the MMCSD0\_SS\_PHY\_CTRL\_6\_REG[30].BISTSTART to 1'b1.
2. Wait until the MMCSD0\_SS\_PHY\_STAT\_1\_REG[3].BISTDONE is set to 1'b1.
3. Read the MMCSD0\_SS\_PHY\_STAT\_2\_REG[31:0].BISTSTATUS\*31:0+ Bit and run the "HS200 Bist Result Check Procedure"
4. Clear the MMCSD0\_SS\_PHY\_CTRL\_6\_REG[30].BISTSTART to 1'b0.

#### 12.3.6.4.6.5.7 HS200 Mode with DLL

The following are the various settings required for performing the BIST in HS200 Mode with DLL.

##### 12.3.6.4.6.5.7.1 Configuration

- MMCSD0\_CLOCK\_CONTROL[15:8].SDCLK\_FRQSEL should be set to 200 MHz.
- PHY's DLL should be disabled.
- MMCSD0\_SS\_PHY\_CTRL\_6\_REG[27:24].BISTMODE should be set to 4'b0000.
- MMCSD0\_SS\_PHY\_CTRL\_6\_REG[31].BISTENABLE should be set to 1'b1.
- MMCSD0\_SS\_PHY\_CTRL\_4\_REG[20].OTAPDLYENA to 1'b1.
- MMCSD0\_SS\_PHY\_CTRL\_4\_REG[15:12].OTAPDLYSEL to one of the tap values based on Timing Closure.
- MMCSD0\_SS\_PHY\_CTRL\_4\_REG[8].ITAPDLYENA\* to 1'b0.
- MMCSD0\_SS\_PHY\_CTRL\_4\_REG[4:0].ITAPDLYSEL\* to 5'b00000
- MMCSD0\_SS\_PHY\_CTRL\_4\_REG[9].ITAPCHGWIN to 1'b0
- MMCSD0\_SS\_PHY\_CTRL\_5\_REG[10:8].FRQSEL should be set for 200 MHz Mode.

\* If Timing closure requires usage of Input Tap Enable and Tap Select should be set appropriately.

##### 12.3.6.4.6.5.7.2 BIST Programming

1. Wait for MMCSD0\_SS\_PHY\_STAT\_1\_REG[0].DLLRDY to be 1'b1 (DLL is Ready)
2. Set the MMCSD0\_SS\_PHY\_CTRL\_6\_REG[30].BISTSTART to 1'b1.
3. Wait until the MMCSD0\_SS\_PHY\_STAT\_1\_REG[3].BISTDONE is set to 1'b1.
4. Read the MMCSD0\_SS\_PHY\_STAT\_2\_REG[31:0].BISTSTATUS\*31:0+ Bit and run the "HS200 Bist Result Check Procedure".
5. Clear the MMCSD0\_SS\_PHY\_CTRL\_6\_REG[30].BISTSTART to 1'b0.



### 12.3.6.4.6.5.8 HS400 Mode with DLL

The following are the various settings required for performing the BIST in HS400 Mode with DLL.

#### 12.3.6.4.6.5.8.1 Configuration

- MMCSO0\_CLOCK\_CONTROL[15:8].SDCLK\_FRQSEL should be set to 200 MHz.
- PHY's DLL should be enabled.
- MMCSO0\_SS\_PHY\_CTRL\_6\_REG[27:24].BISTMODE should be set to 4'b0010.
- MMCSO0\_SS\_PHY\_CTRL\_6\_REG[31].BISTENABLE should be set to 1'b1.
- MMCSO0\_SS\_PHY\_CTRL\_4\_REG[20].OTAPDLYENA to 1'b1.
- MMCSO0\_SS\_PHY\_CTRL\_4\_REG[15:12].OTAPDLYSEL to one of the tap values based on Timing Closure.
- MMCSO0\_SS\_PHY\_CTRL\_4\_REG[8].ITAPDLYENA to 1'b0
- MMCSO0\_SS\_PHY\_CTRL\_4\_REG[4:0].ITAPDLYSEL to 5'b00000
- MMCSO0\_SS\_PHY\_CTRL\_4\_REG[9].ITAPCHGWIN to 1'b0
- MMCSO0\_SS\_PHY\_CTRL\_5\_REG[10:8].FRQSEL should be set for 200 MHz Mode.
- MMCSO0\_SS\_PHY\_CTRL\_4\_REG[31:24].STRBSEL should be set to appropriate values based on based on STRB timing closure.

\* If Timing closure requires usage of Input Tap Enable and Tap Select should be set appropriately.

#### 12.3.6.4.6.5.8.2 BIST Programming

1. Wait for MMCSO0\_SS\_PHY\_STAT\_1\_REG[0].DLLRDY to be 1'b1 (DLL is Ready)
2. Set the MMCSO0\_SS\_PHY\_CTRL\_6\_REG[30].BISTSTART to 1'b1.
3. Wait until the MMCSO0\_SS\_PHY\_STAT\_1\_REG[3].BISTDONE is set to 1'b1.
4. Read the MMCSO0\_SS\_PHY\_STAT\_2\_REG[31:0].BISTSTATUS[0] Bit and check the value.

If (MMCSO0\_SS\_PHY\_STAT\_2\_REG[31:0].BISTSTATUS\*0+==1'b1) then

BIST PASS

else

BIST FAIL;

endif

5. Clear the MMCSO0\_SS\_PHY\_CTRL\_6\_REG[30].BISTSTART to 1'b0.

#### 12.3.6.4.6.6 HS200 BIST Result Check Procedure

In HS200 Mode, the BIST engine in the PHY runs Tuning Pattern test 32 times, with different TAP values selected, for each iteration. The result from the 32 iterations is presented on the MMCSO0\_SS\_PHY\_STAT\_2\_REG[31:0].BISTSTATUS output. It is expected that some of the TAP values will have timing issues and results in pattern mismatch when compared with the received pattern for those taps. The PASS/FAIL for the HS200 Mode is based on the following theory.

Of the 32 taps, the Data Capture EYE for certain taps will result in mismatches, but these will be sequential tap values. Thus, in a full 32 iterations, a sequence of taps results in data mismatches. Also, based on the timing closure, the number of taps with pattern mismatch should be approximately 1/4th of the number of taps with pattern match. This 1/4th count is not absolute, as this involves various factors such as the timing closure, PVT variation, and so forth. Anywhere from 4 to 16 invalid sequential taps is normal.

For example, if 8 of the 32 sequential taps have the data mismatches, the following would be the resulting MMCSO0\_SS\_PHY\_STAT\_2\_REG[31:0].BISTSTATUS output based which of the 8 sequential taps the mismatches are. The following is the notation used:

- 1: Pattern matched at this tap
- 0: Pattern match failed at this tap

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0		
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1	
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1	
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	
1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	
1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	
1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	
1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	
1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	
1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	
1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	
0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	

### 12.3.6.5 MMCSD Programming Guide

#### 12.3.6.5.1 Sequences

This section defines basic sequence flow chart divided into several sub sequences.

#### Note

This section is applicable for MMCSD0, MMCSD1 and MMCSD2 but MMCSD0 registers are used in the content.

#### Note

UHSII is not supported. For more information, see [Section 12.3.6.1.2, Not Supported Features](#).

"Wait for interrupts" is used in the flow chart. This means the Host Driver waits until specified interrupts are asserted. If already asserted, then fall through that step in the flow chart. Timeout checking shall be always required to detect no interrupt generated but this is not described in the flow chart.

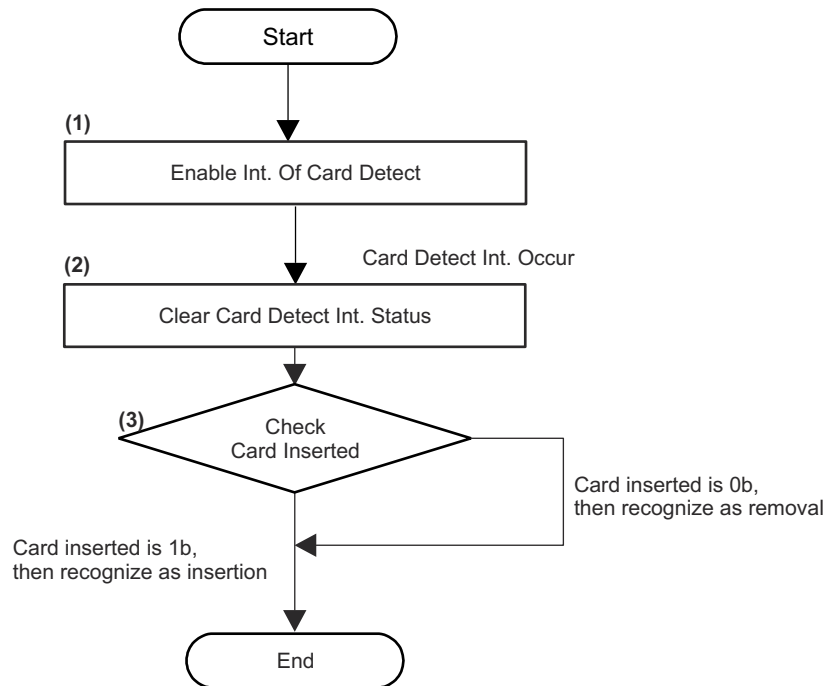
This specification uses the double box like [Figure 12-301](#), (the step (1) in [Figure 12-305](#)), it means that the other flows, which already are shown, shall be performed. Therefore, the interrupt may be included in the other flows.



mmcsd-019

**Figure 12-301. Double Box Notation**

#### 12.3.6.5.1.1 SD Card Detection



mmcsd-020

**Figure 12-302. SD Card Detect Sequence**

(1) The host driver should wait for 1 second after power is applied to the Host Controller before executing card initialization sequence.

The flow chart for detecting a SD card is shown in [Figure 12-302](#).

Each step is executed as follows:

To enable interrupt for card detection, write 1 to the following bits:

- MMCSD0\_NORMAL\_INTR\_STS\_ENA[6] CARD\_INSERTION
- MMCSD0\_NORMAL\_INTR\_SIG\_ENA[6] CARD\_INSERTION
- MMCSD0\_NORMAL\_INTR\_STS\_ENA[7] CARD\_REMOVAL
- MMCSD0\_NORMAL\_INTR\_SIG\_ENA[7] CARD\_REMOVAL

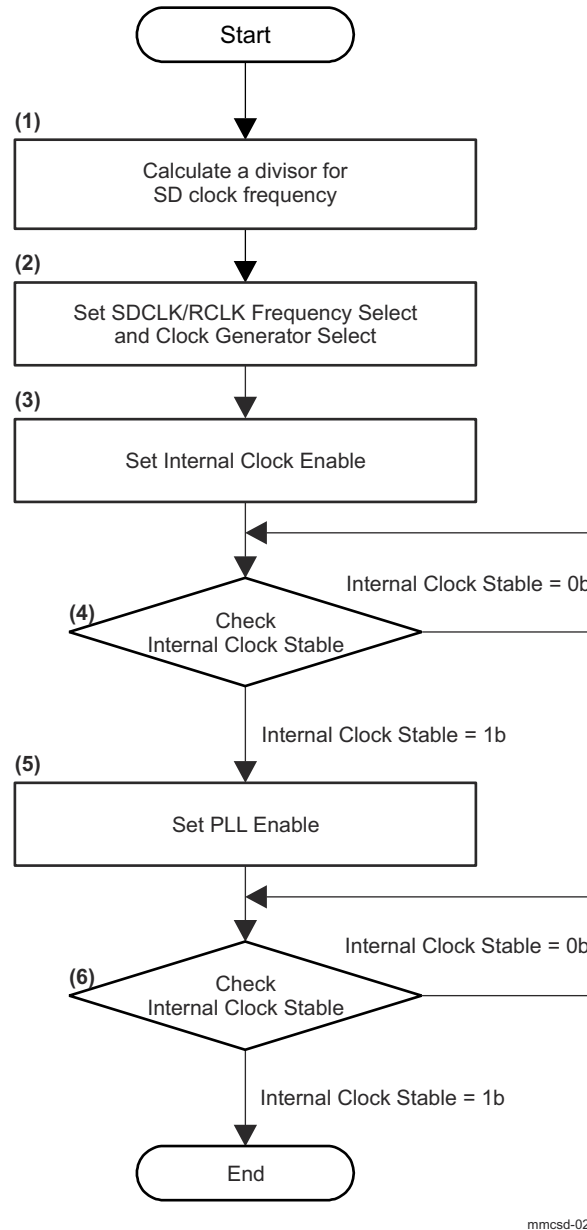
(1) When the Host Driver detects the card insertion or removal, clear its interrupt statuses. If Card Insertion interrupt is generated, write 1 to MMCSD0\_NORMAL\_INTR\_STS\_ENA[6] CARD\_INSERTION bit. If Card Removal interrupt is generated, write 1 to MMCSD0\_NORMAL\_INTR\_STS\_ENA[7] CARD\_REMOVAL bit.

(2) Check MMCSD0\_PRESENTSTATE[16] CARD\_INSERTED bit. In the case where MMCSD0\_PRESENTSTATE[16] CARD\_INSERTED bit is 1, the Host Driver can supply the power and the clock to the SD card. In the case where MMCSD0\_PRESENTSTATE[16] CARD\_INSERTED bit is 0, the other executing processes of the Host Driver shall be immediately closed.

If miniSD adaptor is used for standard SD slot and miniSD card is inserted or extracted from the adaptor, card detect interrupt may not be generated. When host does not receive response to any commands, miniSD card is extracted or in idle state, the host should try to re-initialize the card. In this case, all card information shall be re-loaded.

### 12.3.6.5.1.2 SD Clock Control

#### 12.3.6.5.1.2.1 Internal Clock Setup Sequence



mmcsd-021

**Figure 12-303. Internal Clock Setup Sequence**

The sequence for supplying SD Clock to a SD card is described in [Figure 12-303](#). From Version 4.10, MMCS0\_CLOCK\_CONTROL[3] PLL\_ENA bit is added. This sequence is also applicable to prior versions which do not support MMCS0\_CLOCK\_CONTROL[3] PLL\_ENA bit.

(1) Calculate a divisor to determine SD Clock frequency for legacy IF or RCLK frequency for UHS-II IF by reading MMCS0\_CAPABILITIES[15-8] BASE\_CLK\_FREQ and MMCS0\_CAPABILITIES[55-48] CLOCK\_MULTIPLIER bit fields. If non-zero value is set to MMCS0\_CAPABILITIES[55-48] CLOCK\_MULTIPLIER bit field, Programmable Clock Mode can be used (for more information, see MMCS0\_CLOCK\_CONTROL[5] CLKGEN\_SEL bit). If MMCS0\_CAPABILITIES[15-8] BASE\_CLK\_FREQ bit field is 00 0000b, the Host System shall provide this information to the Host Driver by another method.

(2) Set MMCSD0\_CLOCK\_CONTROL[15-8] SDCLK\_FRQSEL bit field and MMCSD0\_CLOCK\_CONTROL[5] CLKGEN\_SEL bit in accordance with the calculated result of step (1).

If MMCSD0\_HOST\_CONTROL2[15] PRESET\_VALUE\_ENA bit is set, these bits are set by Host Controller automatically as specified in the Preset Value register (MMCSD0\_PRESET\_VALUE0 - MMCSD0\_PRESET\_VALUE10).

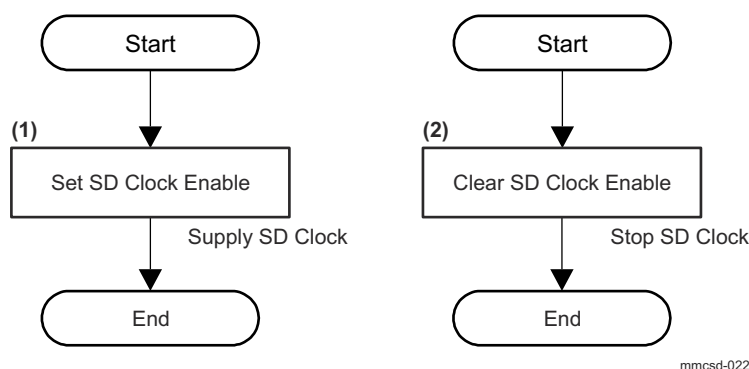
(3) Set MMCSD0\_CLOCK\_CONTROL[0] INT\_CLK\_ENA bit.

(4) Check MMCSD0\_CLOCK\_CONTROL[1] INT\_CLK\_STABLE bit. Repeat this step until this status is 1. Clock will be stable in shorter time but timeout of this loop is defined as 150 ms.

(5) Set MMCSD0\_CLOCK\_CONTROL[3] PLL\_ENA bit. This step does not affect Host Controllers which do not support MMCSD0\_CLOCK\_CONTROL[3] PLL\_ENA bit.

(6) If MMCSD0\_CLOCK\_CONTROL[3] PLL\_ENA bit is supported, PLL locked may be checked by this status (if MMCSD0\_CLOCK\_CONTROL[3] PLL\_ENA bit is not supported, this status is supposed to indicate 1 by step (3)). Clock will be stable in shorter time but timeout of this loop is defined as 150 ms.

#### 12.3.6.5.1.2.2 SD Clock Supply and Stop Sequence



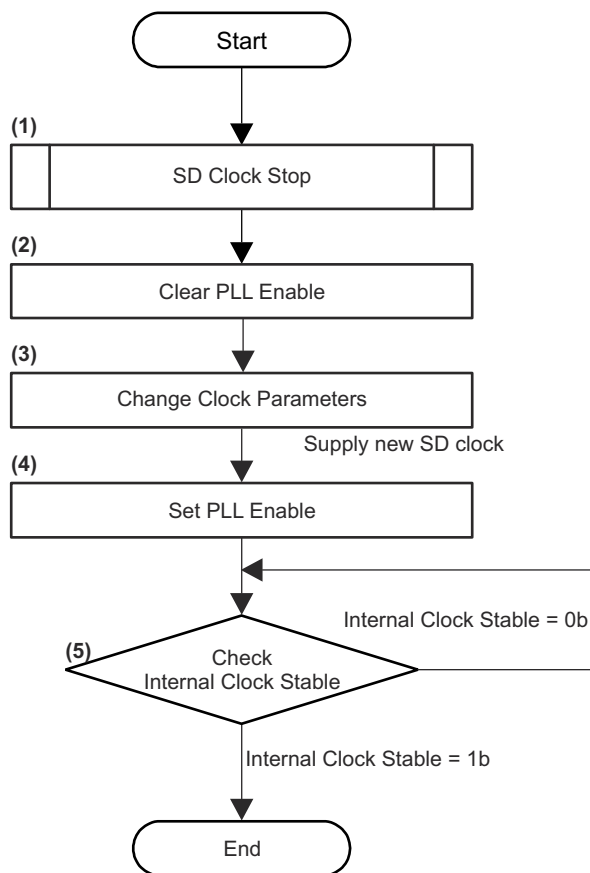
**Figure 12-304. SD Clock Supply and Stop Sequence**

The flow chart for stopping the SD Clock is shown in [Figure 12-304](#). The Host Driver shall not stop the SD Clock when a SD transaction is occurring on the SD Bus -- namely, when either MMCSD0\_PRESENTSTATE[1] INHIBIT\_DAT or MMCSD0\_PRESENTSTATE[0] INHIBIT\_CMD bit is set to 1.

(1) Setup Internal Clock (see [Figure 12-303](#)). Then set MMCSD0\_CLOCK\_CONTROL[2] SD\_CLK\_ENA bit to 1. Then, the Host Controller starts supplying the SD Clock.

(2) Set MMCSD0\_CLOCK\_CONTROL[2] SD\_CLK\_ENA bit to 0. Then, the Host Controller stops supplying the SD Clock. Internal Clock is still oscillating.

### 12.3.6.5.1.2.3 SD Clock Frequency Change Sequence



mmcsd-023

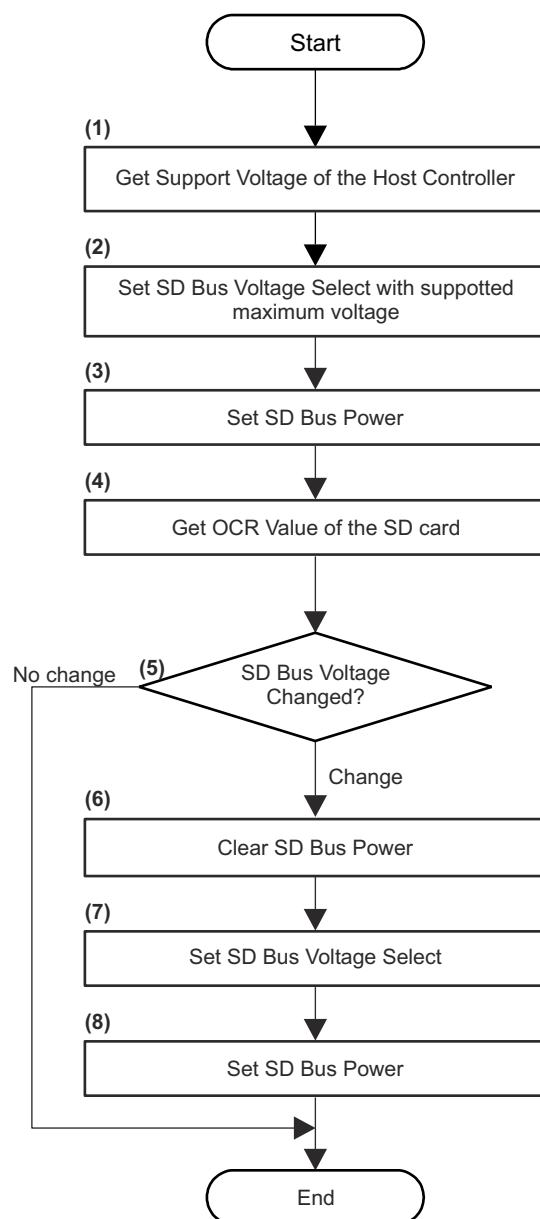
**Figure 12-305. SD Clock Change Sequence**

The sequence for changing SD Clock frequency is shown in [Figure 12-305](#).

- (1) Execute the SD Clock Stop Sequence (see [Figure 12-304](#)). If SD Clock supply to card is already stopped, skip this step.
- (2) Clear MMCSD0\_CLOCK\_CONTROL[3] PLL\_ENA bit to 0. If MMCSD0\_CLOCK\_CONTROL[3] PLL\_ENA bit is not supported, this step has no effect and may be skipped.
- (3) When MMCSD0\_HOST\_CONTROL2[15] PRESET\_VALUE\_ENA bit is set to 0, change clock parameters in the MMCSD0\_CLOCK\_CONTROL register. When MMCSD0\_HOST\_CONTROL2[15] PRESET\_VALUE\_ENA bit is set to 1, select Bus Speed Mode as described.
- (4) Set MMCSD0\_CLOCK\_CONTROL[3] PLL\_ENA bit to 1. If MMCSD0\_CLOCK\_CONTROL[3] PLL\_ENA bit is not supported, this step this step has no effect and may be skipped.
- (5) Wait until MMCSD0\_CLOCK\_CONTROL[1] INT\_CLK\_STABLE bit is set to 1. Clock will be stable in shorter time but timeout of this loop is defined as 150 ms. SD Clock Supply Sequence is required to provide clock to device (see [Figure 12-304](#)).

### 12.3.6.5.1.3 SD Bus Power Control

The sequence for controlling the SD Bus Power is described in [Figure 12-306](#).



mmcsd-024

**Figure 12-306. SD Bus Power Control Sequence**

- (1) By reading the MMCSD0\_CAPABILITIES register, get the support voltage of the Host Controller.
- (2) Set MMCSD0\_POWER\_CONTROL[3-1] SD\_BUS\_VOLTAGE bit field with maximum voltage that the Host Controller supports.
- (3) Set MMCSD0\_POWER\_CONTROL[0] SD\_BUS\_POWER bit to 1.
- (4) Get the OCR value of all function internal of SD card.
- (5) Judge whether SD Bus voltage needs to be changed or not. In case where SD Bus voltage needs to be changed, go to step (6). In case where SD Bus voltage does not need to be changed, go to "End".
- (6) Set MMCSD0\_POWER\_CONTROL[0] SD\_BUS\_POWER bit to 0 for clearing this bit. The card requires voltage rising from 0 volt to detect it correctly. The Host Driver shall



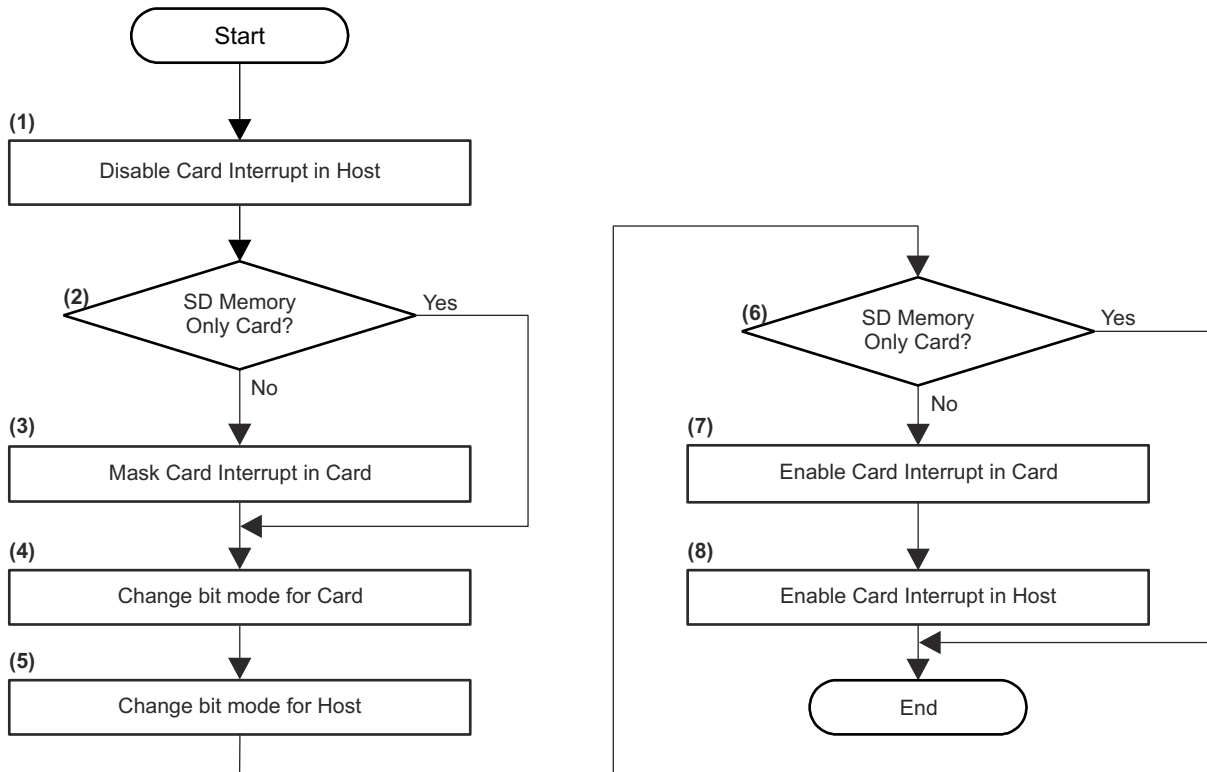
clear MMCSD0\_POWER\_CONTROL[0] SD\_BUS\_POWER bit before changing voltage by setting MMCSD0\_POWER\_CONTROL[3-1] SD\_BUS\_VOLTAGE bit field.

(7) Set MMCSD0\_POWER\_CONTROL[3-1] SD\_BUS\_VOLTAGE bit field.

(8) Set MMCSD0\_POWER\_CONTROL[0] SD\_BUS\_POWER to 1.

**Note:** Step (2) and step (3) can be executed at same time. And also, step (7) and step (8) can be executed at same time.

#### 12.3.6.5.1.4 Changing Bus Width



mmscd-025

**Figure 12-307. Change Bus Width Sequence**

The sequence for changing bit mode on SD Bus is shown in [Figure 12-307](#).

(1) Set MMCSD0\_NORMAL\_INTR\_STS\_ENA[8] CARD\_INTERRUPT bit to 0 for masking incorrect interrupts that may occur while changing the bus width.

(2) In case of SD memory only card, go to step (4). In case of other card, go to step (3).

(3) Set "IENM" of the CCCR in a SDIO or SD combo card to 0 by CMD52.

(4) Change the bus width mode for a SD card. SD Memory Card bus width is changed by ACMD6 and SDIO card bus width is changed by setting Bus Width of Bus Interface Control register in CCCR.

(5) In case of changing to 4-bit mode, set MMCSD0\_HOST\_CONTROL1[1] DATA\_WIDTH bit to 1. In another case (1-bit mode), set this bit to 0.

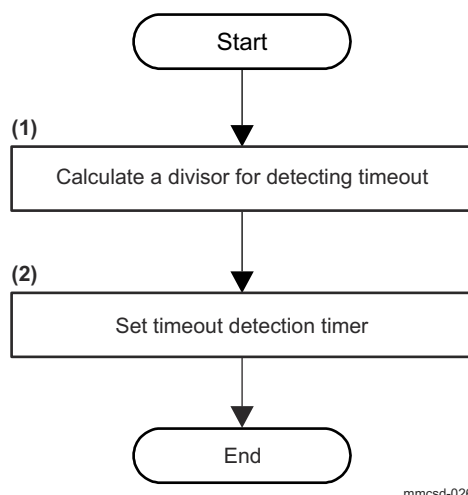
(6) In case of SD memory only card, go to the "End". In case of other card, go to step (7).

(7) Set "IENM" of the CCCR in a SDIO or SD combo card to 1 by CMD52.

(8) Set MMCSD0\_NORMAL\_INTR\_STS\_ENA[8] CARD\_INTERRUPT bit to 1.

Note that if the card is locked, bus width cannot be changed. Unlock the card is required before changing bus width.

#### 12.3.6.5.1.5 Timeout Setting on DAT Line



mmcsd-026

**Figure 12-308. Timeout Setting Sequence**

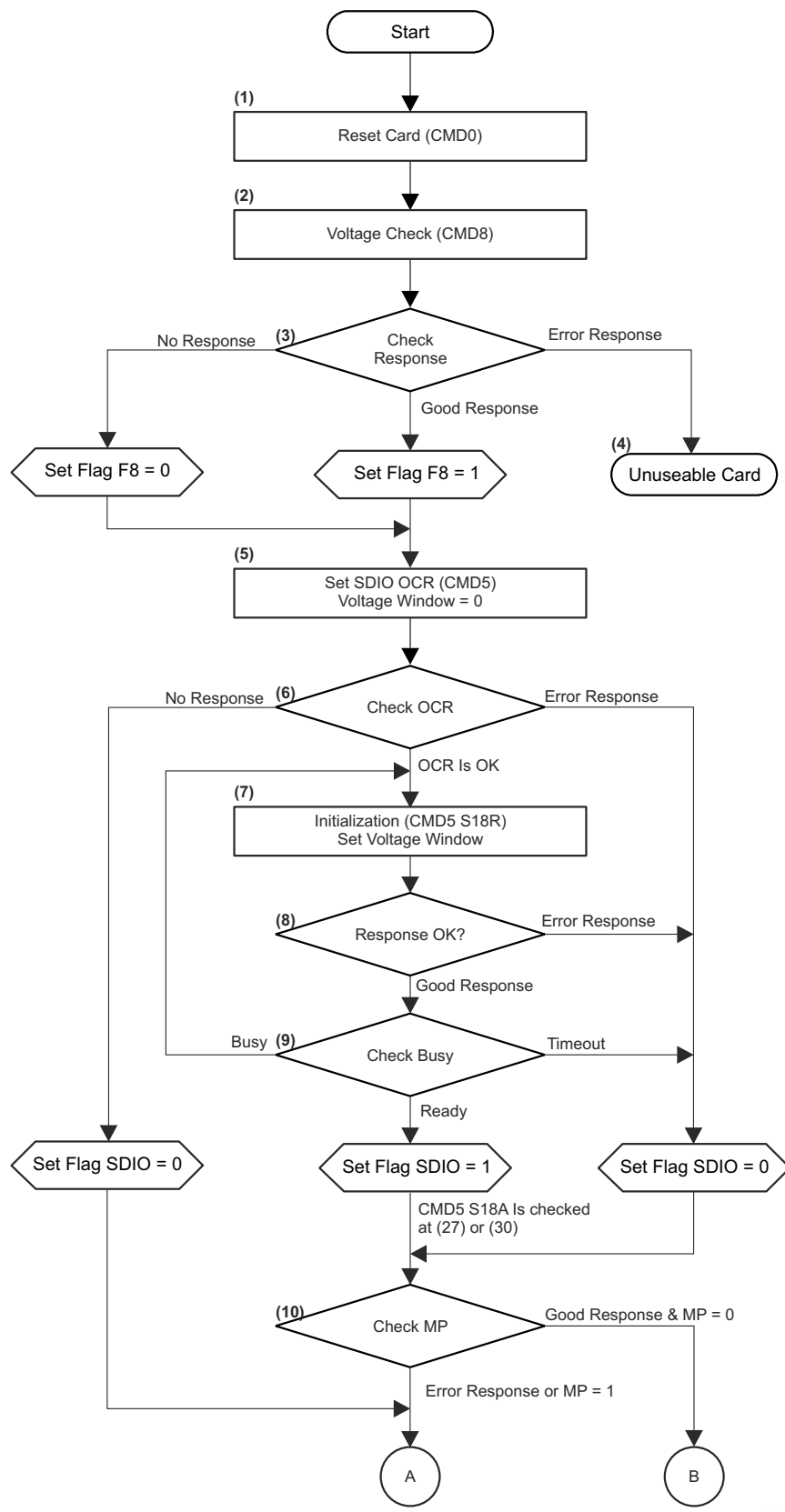
In order to detect timeout errors on DAT line, the Host Driver shall execute the following two steps before any SD transaction. For more information regarding SD transactions:

(1) Calculate a divisor to detect timeout errors by reading MMCSD0\_CAPABILITIES[5-0] TIMEOUT\_CLK\_FREQ bit field and MMCSD0\_CAPABILITIES[7] TIMEOUT\_CLK\_UNIT bit. If MMCSD0\_CAPABILITIES[5-0] TIMEOUT\_CLK\_FREQ bit field is 00 0000b, the Host System shall provide this information to the Host Driver by another method.

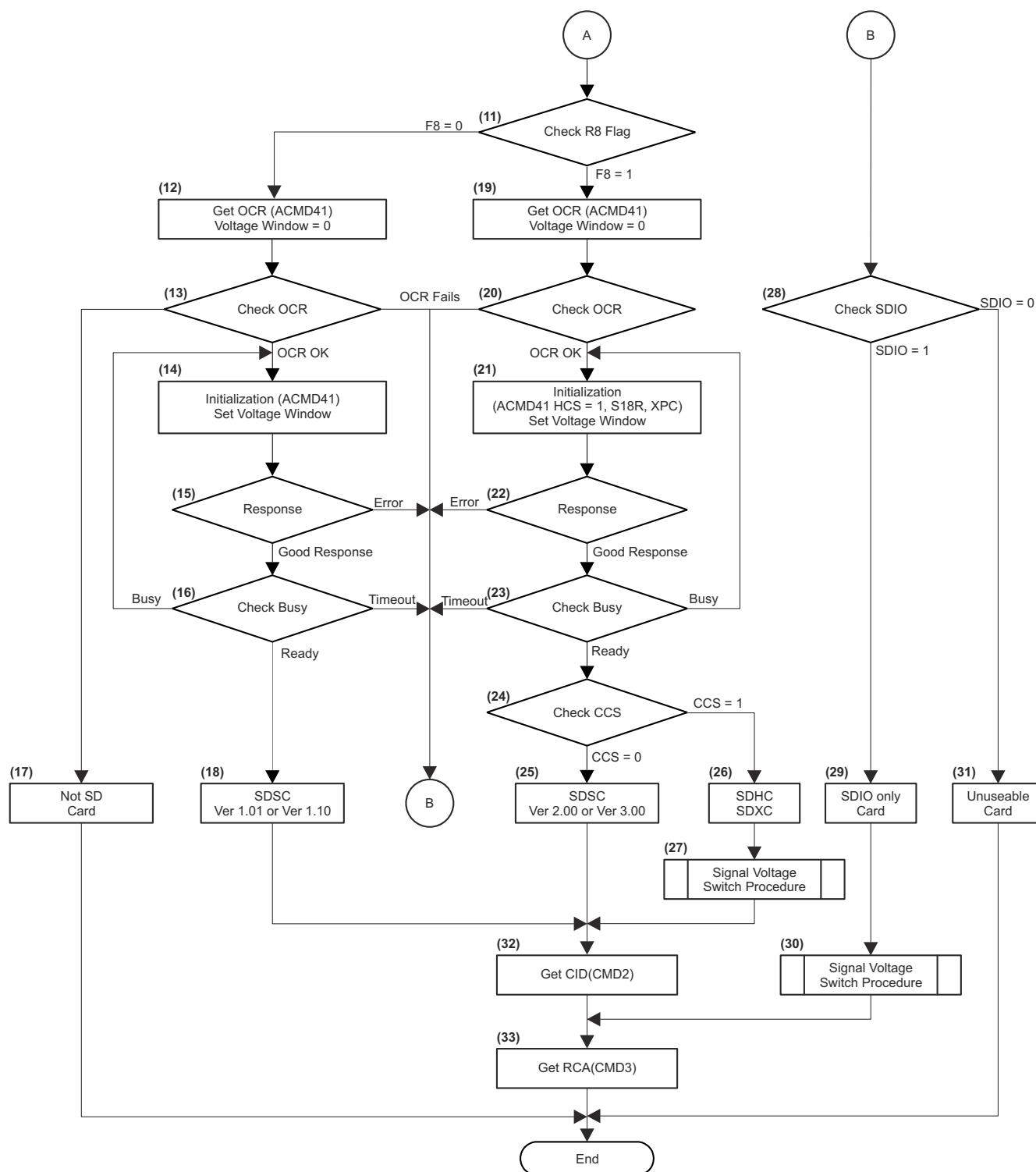
(2) Set MMCSD0\_TIMEOUT\_CONTROL[3-0] COUNTER\_VALUE bit field in accordance with the value from step (1) above.

#### 12.3.6.5.1.6 Card Initialization and Identification (for SD I/F)

Figure 12-309 and Figure 12-310 shows initialization and card identification sequence for the Standard Capacity SD Memory Card (SDSC), the High Capacity SD Memory Card (SDHC) and the Extended Capacity SD Memory Card (SDXC) that was based on the Physical Layer Version 3.01. Refer to the latest sequence.



**Figure 12-309. Card Initialization and Identification (1)**



mmcsd-028

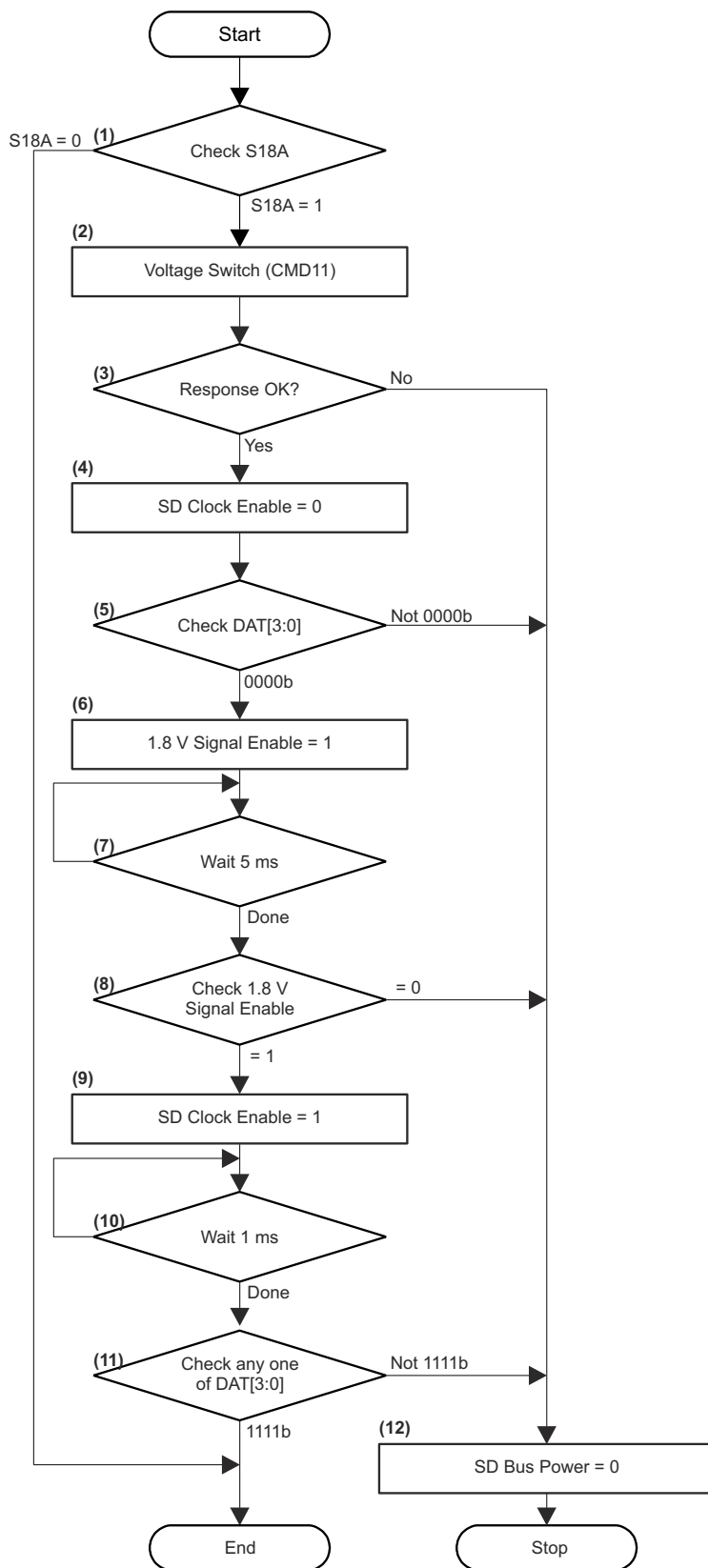
**Figure 12-310. Card Initialization and Identification (2)**

- (1) SD Bus mode is selected by CMD0 (Keep Pin 1 to high during CMD0 execution).
- (2) New CMD8 shall be issued after CMD0 to support High Capacity SD Memory Card.

- (3) Voltage check command enables the Hosts to support future low voltage specification. However, at this time, only one voltage is defined. Legacy cards and Not SD cards do not respond to CMD8. In this case, set F8 to 0 (F8 is CMD8 valid flag used in step (11)) and go to Step (5). Only Version 2.00 or higher cards can respond to CMD8. The host needs to check whether CRC of the response is valid and whether VHS and check pattern in the argument are equal to VCA and check pattern in the response. Passing all these checks results in CMD8 response OK. In this case, set F8 to 1 and go to step (5). If one of the checks is failed, go to step (4).
- (4) Initialization is stopped by CMD8 fails. The Host Driver should retry step (1) to (3) one more time (this is not described in the figure).
- (5) SDIO OCR is available by issuing CMD5 with setting voltage window (bit 23 to 0) in the argument to 0. SDIO initialization is not started.
- (6) No response means the card does not have SDIO function. Set SDIO flag to 0 and go to step (11). If the card responds to CMD5 and the response is OK, go to step (7). If the response is error, set SDIO flag to 0 and go to step (10). SDIO flag indicates whether SDIO functions are initialized or not.
- (7) The SDIO portion starts initialization by CMD5 with setting the supply voltage to the voltage window. UHS-I supported host sets S18R to 1. If the supplied voltage is not matched with voltage window of card, the card goes into inactive state and does not return the response.
- (8) If no response or error response is receive, set SDIO flag to 0 and go to step (10). If good response is received, go to step (9).
- (9) Check busy status in the response. If busy is released, set SDIO flag to 1 and go to step (10). Repeat from step (7) while busy is indicated. Detecting timeout of 1 second exits the loop. In this case, set SDIO flag to 0 and go to step (10).
- (10) Good response in this step means that all responses received at (6) and (8) are valid. When response is good, MP (memory present) flag in the response can be checked. If the response valid and MP = 0, go to step (28). Otherwise, go to step (11).
- (11) Check F8 flag set in step (3). If CMD8 is executed correctly (F8 = 1), go to step (19). Otherwise, go to step (12).
- (12) OCR is available by issuing ACMD41 with the voltage window (bit 23 to 0) in the argument is set to 0. Memory initialization is not started. The response of CMD55 (ACMD41) may indicate illegal command error due to some SD cards do not recognized CMD8. The Host Driver should ignore this error or issue CMD0 before ACMD41 to clear this error status.
- (13) If response of CMD55 is not received, the card is not SD cards and goes to (17). If the card responds to CMD55, it may also respond to CMD41. If the responses of ACMD41 are OK, go to Step (14). Otherwise, go to step (28). Locked card can be detected by the card status in the response of CMD55.
- (14) The memory portion starts initialization by Issuing ACMD41 with setting the supply voltage to the voltage window. If the supplied voltage is not matched with voltage window of card, the card goes into inactive state and does not return the response.
- (15) If no response or error response is received, go to step (28). If good response is received, go to step (16).
- (16) Check busy status in the response. If busy is released, go to step (18). Repeat from step (14) while busy is indicated. The interval of ACMD41 shall be less than 50 ms. Detecting timeout of 1 second exits the loop and go to step (28).
- (17) The host recognizes that the card is not SD memory card and quits SD card initialization.
- (18) The host recognizes that the card is Version 1.xx Standard Capacity SD Memory Card. Go to Step (30).
- (19) OCR is available by issuing ACMD41 with setting the voltage window (bit 23 to 0) in the argument is set to 0. Memory initialization is not started. Setting of HCS does not affect this operation.

- (20) If the card responds to CMD55, it may also respond to CMD41. If the responses of ACMD41 are OK, go to Step (21). Otherwise, go to step (28). Locked card can be detected by the card status in the response of CMD55.
- (21) The memory portion starts initialization by Issuing ACMD41 with setting the supply voltage to the voltage window. UHS-I supported host sets S18R to 1. If the host can supply more than 150mA, XPC is set to 1. HCS in the argument is set to 1, which indicates supporting High Capacity Memory Card. If the supplied voltage is not matched with voltage window of card, the card goes into inactive state and does not return the response.
- (22) If no response or error response is received, go to step (28). If good response is received, go to step (23).
- (23) Check busy status in the response. If busy is released, go to step (24). Repeat from step (21) while busy is indicated. The interval of ACMD41 shall be less than 50 ms. Detecting timeout of 1 second exits the loop and go to step (28).
- (24) CCS in the response is valid after busy is released. If CCS = 0, it indicates the Standard Capacity SD Memory Card and go to step (25). If CCS = 1, it indicates the High Capacity SD Memory Card or Extended Capacity Memory Card and go to Step (26).
- (25) The host recognizes that the card is Ver2.00 or Ver3.00 Standard Capacity SD Memory Card. Optimal functions defined in Version 2.00 or higher are available. Go to Step (32).
- (26) The host recognizes that the card is the High Capacity SD Memory Card or Extended Capacity Memory Card.
- (27) Perform the signal voltage switch procedure and go to step (32).
- (28) Check SDIO flag. If SDIO = 1, go to step (28). Otherwise, go to step (31).
- (29) The host recognizes that the card is SDIO only card and go to step (30).
- (30) Perform the signal voltage switch procedure and go to step (33).
- (31) The host recognizes that the card is unusable.
- (32) In case of memory card, CMD2 is issued to get CID and Go to Step (31).
- (33) CMD3 is issued to get RCA. If the RCA number is 0, the Host should issue CMD3 again.

### 12.3.6.5.1.6.1 Signal Voltage Switch Procedure (for UHS-I)



mmcsd-029

**Figure 12-311. Signal Voltage Switch Procedure**

- (1) If S18A of CMD5 or S18A of ACMD41 is set to 1, signal voltage switch is performed according to the following steps. Otherwise, exits from this procedure.
- (2) Issue CMD11.
- (3) Check response and if an error is detected, go to step (12).
- (4) Stop providing SD clock to the card.
- (5) Check DAT[3:0] level. If the level is 0000b, the card is ready to start voltage switch sequence. Otherwise, go to (12) to quit the sequence.
- (6) Set MMCSD0\_HOST\_CONTROL2[3] V1P8\_SIGNAL\_ENA bit.
- (7) Wait 5 ms. 1.8 V voltage regulator shall be stable within this period.
- (8) If MMCSD0\_HOST\_CONTROL2[3] V1P8\_SIGNAL\_ENA bit is cleared by Host Controller, go to step (12).
- (9) Provide SD Clock to the card again.
- (10) Wait 1 ms.
- (11) Check DAT[3:0] level. If the level is 1111b, switch to 1.8 V signal level is completed successfully. Otherwise, go to (12).
- (12) If an error occurs during voltage switch procedure, stop providing the power to the card. In this case, Host Driver should retry initialization procedure by setting S18R to 0 at step (7) and (21) in [Figure 12-309](#) and [Figure 12-310](#).

#### **12.3.6.5.1.7 SD Transaction Generation**

This section describes the sequences how to generate and control various kinds of SD transactions. SD transactions are classified into three cases:

- (1) Transactions that do not use the DAT line.
- (2) Transactions that use the DAT line only for the busy signal.
- (3) Transactions that use the DAT line for transferring data.

In this specification the first and the second case's transactions are classified as "Transaction Control without Data Transfer using DAT Line", the third case's transaction is classified as "Transaction Control with Data Transfer using DAT Line". Refer to the latest SD Physical Layer Specification and SDIO Specification for more detail about SD commands specification.

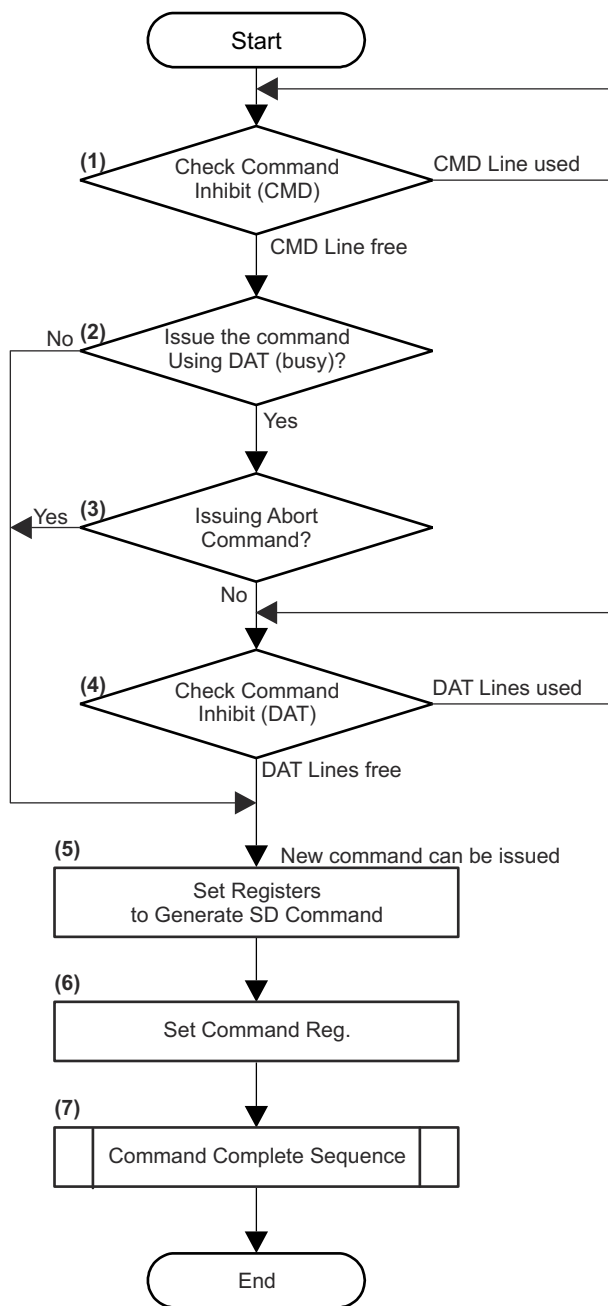
##### **12.3.6.5.1.7.1 Transaction Control without Data Transfer Using DAT Line**

In this section, the sequence for how to issue SD Command and how to complete SD Command is explained. [Figure 12-312](#) shows the sequence to issue a SD Command and [Figure 12-313](#) shows the sequence to finalize a SD Command.

##### **12.3.6.5.1.7.1.1 The Sequence to Issue a SD Command**

The sequence to issue the SD Command is detailed in [Figure 12-312](#).





mmcscd-030

**Figure 12-312. SD Command Issue Sequence**

(1) Check MMCSD0\_PRESENTSTATE[0] INHIBIT\_CMD bit. Repeat this step until MMCSD0\_PRESENTSTATE[0] INHIBIT\_CMD bit is 0. That is, when MMCSD0\_PRESENTSTATE[0] INHIBIT\_CMD bit is 1, the Host Driver shall not issue a SD Command.

(2) If the Host Driver issues a SD Command using DAT lines including busy signal, go to step (3). If without using DAT lines including busy signal, go to step (5).

(3) If the Host Driver is issuing an abort command, go to step (5). In the case of nonabort command, go to step (4).

(4) Check MMCSD0\_PRESENTSTATE[1] INHIBIT\_DAT bit. Repeat this step until MMCSD0\_PRESENTSTATE[1] INHIBIT\_DAT bit is set to 0.

(5) Set registers except the MMCSD0\_COMMAND register.

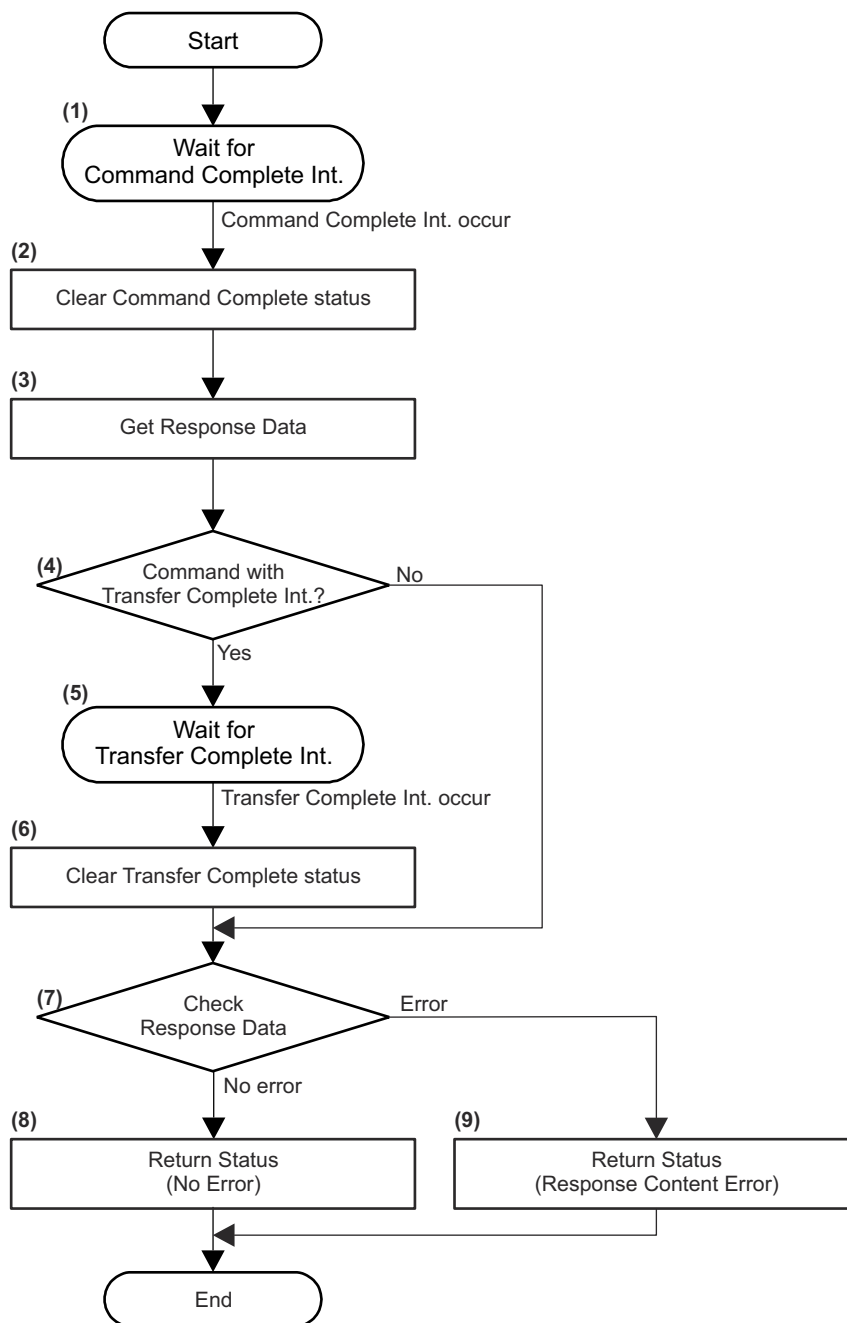
(6) Set the MMCSD0\_COMMAND register.

**Note:** Writing the upper byte [3] in the MMCSD0\_COMMAND register causes the Host Controller to issue a SD command to the SD card.

(7) Perform Command Completion Sequence in accordance.

#### **12.3.6.5.1.7.1.2 The Sequence to Finalize a Command**

[Figure 12-313](#) shows the sequence to finalize a SD Command when response check is disabled. There is a possibility that some errors (Command Index/End bit/CRC/Timeout Error) occur during this sequence. If response check is enabled, error is indicated by Response Error Interrupt.



mmcsd-031

**Figure 12-313. Command Complete Sequence**

#### 12.3.6.5.1.7.1.3

(1) If MMCS0\_TRANSFER\_MODE[8] RESP\_INTR\_DIS bit is set to 1 (response check is enabled), go to stop (4) else wait for the Command Complete Interrupt (MMCS0\_NORMAL\_INTR\_STS[0] CMD\_COMPLETE). If the Command Complete Interrupt has occurred, go to step (2).

(2) Write 1 to MMCS0\_NORMAL\_INTR\_STS[0] CMD\_COMPLETE bit to clear this bit.

(3) Read the Response register (see MMCS0\_RESPONSE\_0 - MMCS0\_RESPONSE\_7) and get necessary information of the issued command.

- (4) Judge whether the command uses the Transfer Complete Interrupt (MMCSD0\_NORMAL\_INTR\_STS\_ENA[1] XFER\_COMPLETE) or not. If it uses Transfer Complete, go to step (5). If not, go to step (7).
- (5) Wait for the Transfer Complete Interrupt. If the Transfer Complete Interrupt has occurred, go to step (6).
- (6) Write 1 to MMCSD0\_NORMAL\_INTR\_STS\_ENA[1] XFER\_COMPLETE bit to clear this bit.
- (7) Check for errors in Response Data (MMCSD0\_RESPONSE\_0 - MMCSD0\_RESPONSE\_7). If there is no error, go to step (8). If there is an error, go to step (9).
- (8) Return Status of "No Error".
- (9) Return Status of "Response Contents Error".

**Note1:** While waiting for the Transfer Complete interrupt, the Host Driver shall only issue commands that do not use the busy signal.

**Note2:** The Host Driver shall judge the Auto CMD12 complete by monitoring Transfer Complete.

**Note3:** When the last block of un-protected area is read using memory multiple block read command (CMD18), OUT\_OF\_RANGE error may occur even if the sequence is correct. The Host Driver should ignore it. This error will appear in the response of Auto CMD12 or in the response of the next memory command.

#### **12.3.6.5.1.7.2 Transaction Control with Data Transfer Using DAT Line**

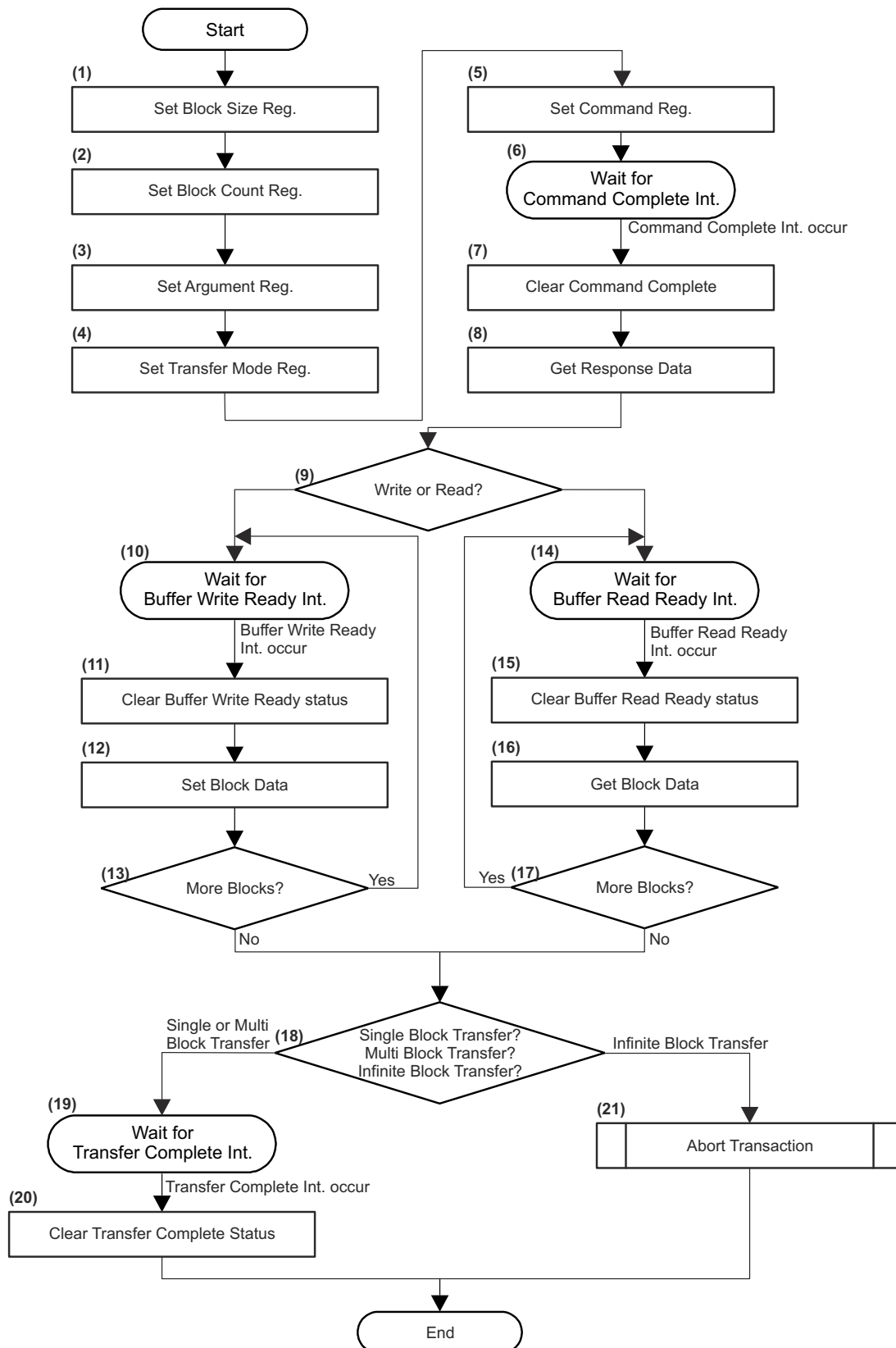
Depending on whether DMA (optional) is used or not, there are two execution methods.

In addition, the sequences for SD transfers are classified into following three kinds according to how the number of blocks is specified:

- (1) Single Block Transfer: The number of blocks is specified to the Host Controller before the transfer. The number of blocks specified is always one.
- (2) Multiple Block Transfer: The number of blocks is specified to the Host Controller before the transfer. The number of blocks specified shall be one or more.
- (3) Infinite Block Transfer: The number of blocks is not specified to the Host Controller before the transfer. This transfer is continued until an abort transaction is executed. This abort transaction is performed by CMD12 in the case of a SD memory card and by CMD52 in the case of a SDIO card.

##### **12.3.6.5.1.7.2.1 Not using DMA**

The sequence for not using DMA is shown in [Figure 12-314](#).



mmscd-032

**Figure 12-314. Transaction Control with Data Transfer Using DAT Line Sequence (Not using DMA)**

- (1) Set the value corresponding to the executed data byte length of one block to the MMCSD0\_BLOCK\_SIZE register.
- (2) Set the value corresponding to the executed data block count to the MMCSD0\_BLOCK\_COUNT register in accordance with [Table 12-386](#).

**Table 12-386. Determination of Transfer Type**

Multi/Single Block Select	Block Count Enable	Block Count	Function
0	Don't Care	Don't Care	Single Transfer
1	0	Don't Care	Infinite Transfer
1	1	Not Zero	Multiple Transfer
1	1	Zero	Stop Multiple Transfer

- (3) Set the argument value to Argument register (MMCSD0\_ARGUMENT1\_LO and MMCSD0\_ARGUMENT1\_HI).

- (4) Set the value to the MMCSD0\_TRANSFER\_MODE register. The Host Driver determines Multi/Single Block Select, Block Count Enable, Data Transfer Direction, Auto CMD12 Enable and DMA Enable in the MMCSD0\_TRANSFER\_MODE register. Multi/Single Block Select and Block Count Enable are determined according to [Table 12-386](#).

If response check is enabled (MMCSD0\_TRANSFER\_MODE[7] RESP\_ERR\_CHK\_ENA = 1), set MMCSD0\_TRANSFER\_MODE[8] RESP\_INTR\_DIS bit to 1 and select Response Type R1/R5 (MMCSD0\_TRANSFER\_MODE[6] RESP\_TYPE).

- (5) Set the value to MMCSD0\_COMMAND register.

**Note:** When writing the upper byte [3] of MMCSD0\_COMMAND register, SD command is issued.

- (6) If response check is enabled, go to stop (9) else wait for the Command Complete Interrupt (MMCSD0\_NORMAL\_INTR\_STS[0] CMD\_COMPLETE bit).

- (7) Write 1 to the MMCSD0\_NORMAL\_INTR\_STS[0] CMD\_COMPLETE bit for clearing this bit.

- (8) Read Response register (MMCSD0\_RESPONSE\_0 - MMCSD0\_RESPONSE\_7) and get necessary information of the issued command.

- (9) In the case where this sequence is for write to a card, go to step (10). In case of read from a card, go to step (14).

- (10) Then wait for Buffer Write Ready Interrupt (MMCSD0\_NORMAL\_INTR\_STS\_ENA[4] BUF\_WR\_READY).

- (11) Write 1 to the MMCSD0\_NORMAL\_INTR\_STS\_ENA[4] BUF\_WR\_READY bit for clearing this bit.

- (12) Write block data (in according to the number of bytes specified at the step (1)) to MMCSD0\_DATA\_PORT register.

- (13) Repeat until all blocks are sent and then go to step (18).

- (14) Then wait for the Buffer Read Ready Interrupt (MMCSD0\_NORMAL\_INTR\_STS\_ENA[5] BUF\_RD\_READY).

- (15) Write 1 to the MMCSD0\_NORMAL\_INTR\_STS\_ENA[5] BUF\_RD\_READY bit for clearing this bit.

- (16) Read block data (in according to the number of bytes specified at the step (1)) from the MMCSD0\_DATA\_PORT register.

- (17) Repeat until all blocks are received and then go to step (18).

- (18) If this sequence is for Single or Multiple Block Transfer, go to step (19). In case of Infinite Block Transfer, go to step (21).

- (19) Wait for Transfer Complete Interrupt (MMCSD0\_NORMAL\_INTR\_STS\_ENA[1] XFER\_COMPLETE).

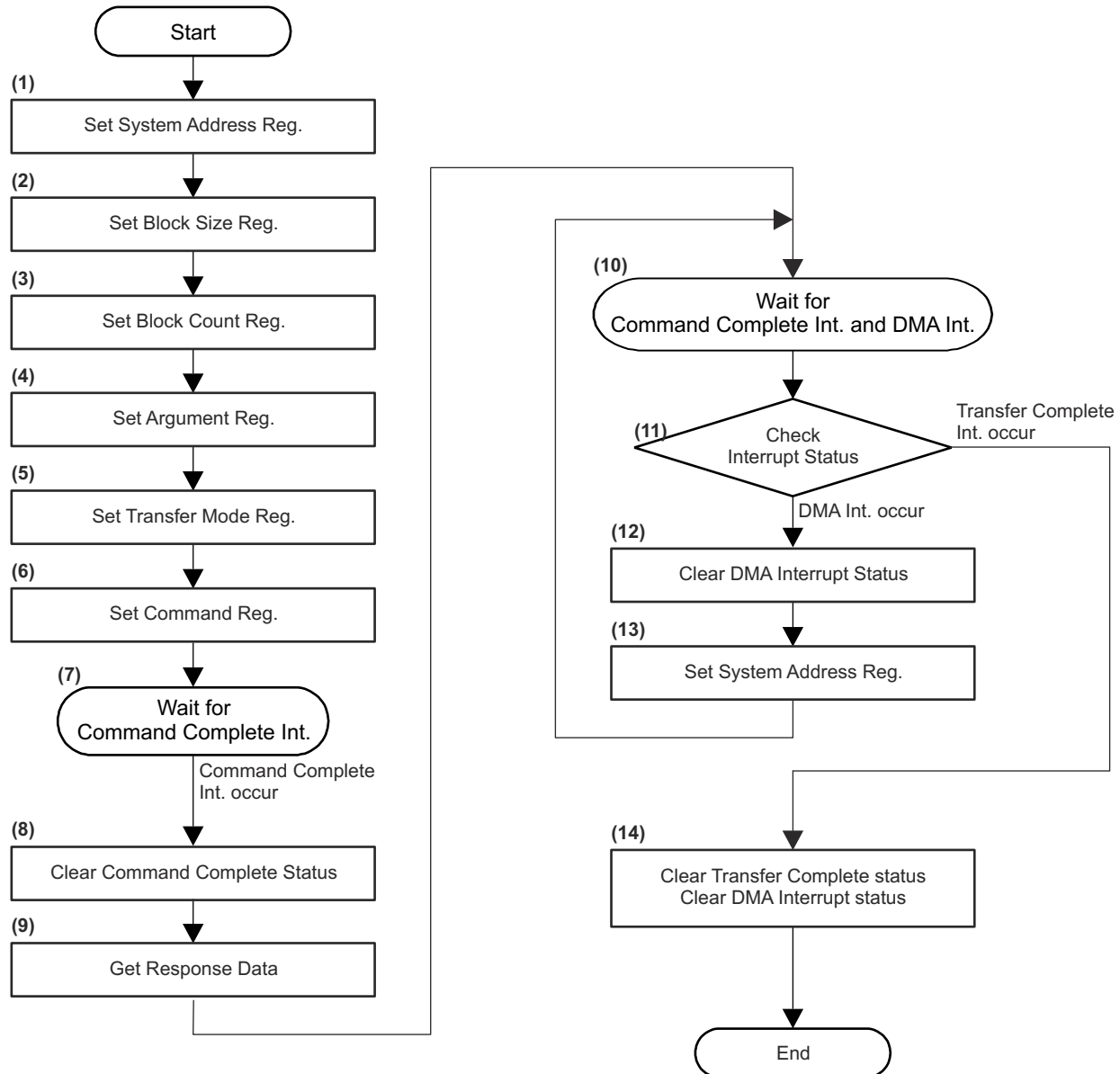
(20) Write 1 to the MMCSD0\_NORMAL\_INTR\_STS\_ENA[1] XFER\_COMPLETE bit for clearing this bit.

(21) Perform the sequence for Abort Transaction in accordance with [Section 12.3.6.5.1.8, Abort Transaction](#).

**Note:** Step (1) and Step (2) can be executed at same time. Step (4) and Step (5) can be executed at same time.

#### 12.3.6.5.1.7.2.2 Using SDMA

The sequence for using SDMA is shown in [Figure 12-315](#).



mmcsd-033

**Figure 12-315. Transaction Control with Data Transfer Using DAT Line Sequence (Using SDMA)**

(1) Data location of system memory is set to the SDMA System Address register if MMCSD0\_HOST\_CONTROL2[12] HOST\_VER40\_ENA = 0 or set to the MMCSD0\_ADMA\_SYS\_ADDRESS register if MMCSD0\_HOST\_CONTROL2[12] HOST\_VER40\_ENA = 1.

(2) Set the value corresponding to the executed data byte length of one block in the MMCSD0\_BLOCK\_SIZE register.

(3) Set the value corresponding to the executed data block count in the MMCSDB0\_BLOCK\_COUNT register in accordance with [Table 12-386](#).

(4) Set the argument value to the Argument register (MMCSDB0\_ARGUMENT1\_LO and MMCSDB0\_ARGUMENT1\_HI).

(5) Set the value to the MMCSDB0\_TRANSFER\_MODE register. The Host Driver determines Multi/Single Block Select, Block Count Enable, Data Transfer Direction, Auto CMD12 Enable and DMA Enable in the MMCSDB0\_TRANSFER\_MODE register. Multi/Single Block Select and Block Count Enable are determined according to [Table 12-386](#). If response check is enabled (MMCSDB0\_TRANSFER\_MODE[7] RESP\_ERR\_CHK\_ENA = 1), set MMCSDB0\_TRANSFER\_MODE[8] RESP\_INTR\_DIS bit to 1 and select Response Type R1/R5.

(6) Set the value to the MMCSDB0\_COMMAND register.

**Note:** When writing to the upper byte [3] of the MMCSDB0\_COMMAND register, the SD command is issued and SDMA is started.

(7) If response check is enabled, go to stop (10) else wait for the Command Complete Interrupt (MMCSDB0\_NORMAL\_INTR\_STS[0] CMD\_COMPLETE bit).

(8) Write 1 to the MMCSDB0\_NORMAL\_INTR\_STS[0] CMD\_COMPLETE bit to clear this bit.

(9) Read Response register (MMCSDB0\_RESPONSE\_0 - MMCSDB0\_RESPONSE\_7) and get necessary information of the issued command.

(10) Wait for the Transfer Complete Interrupt (MMCSDB0\_NORMAL\_INTR\_STS\_ENA[1] XFER\_COMPLETE) and DMA Interrupt (MMCSDB0\_NORMAL\_INTR\_STS\_ENA[3] DMA\_INTERRUPT).

(11) If MMCSDB0\_NORMAL\_INTR\_STS\_ENA[1] XFER\_COMPLETE bit is set to 1, go to Step (14) else if MMCSDB0\_NORMAL\_INTR\_STS\_ENA[3] DMA\_INTERRUPT bit is set to 1, go to Step (12). The MMCSDB0\_NORMAL\_INTR\_STS\_ENA[1] XFER\_COMPLETE bit is higher priority than the MMCSDB0\_NORMAL\_INTR\_STS\_ENA[3] DMA\_INTERRUPT bit.

(12) Write 1 to the MMCSDB0\_NORMAL\_INTR\_STS\_ENA[3] DMA\_INTERRUPT bit to clear this bit.

(13) Set the next system address of the next data position to the System Address register (MMCSDB0\_ADMA\_SYS\_ADDRESS) and go to Step (10).

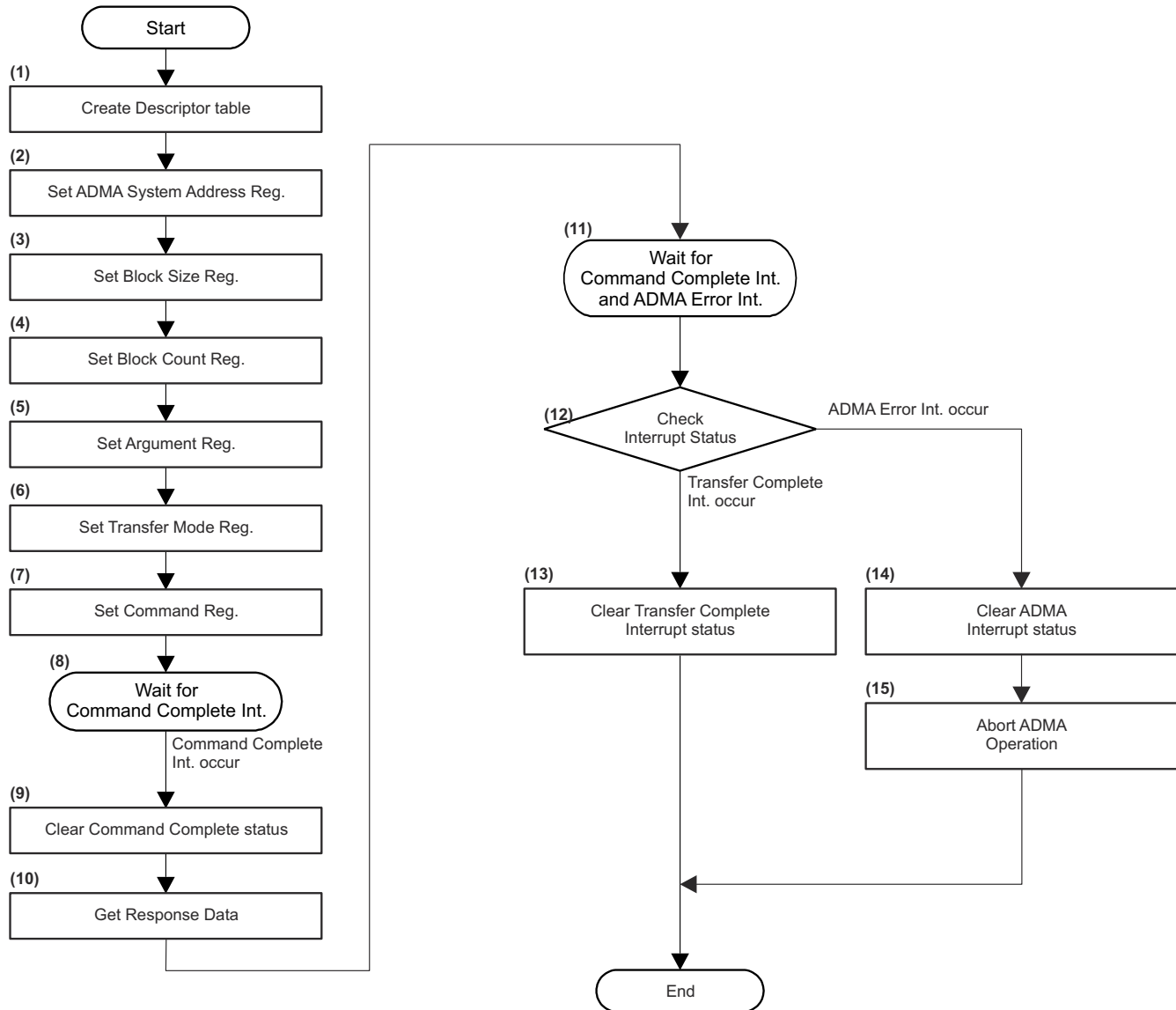
(14) Write 1 to the MMCSDB0\_NORMAL\_INTR\_STS\_ENA[1] XFER\_COMPLETE bit and MMCSDB0\_NORMAL\_INTR\_STS\_ENA[3] DMA\_INTERRUPT bit to clear this bit.

**Note:** Step (2) and Step (3) can be executed simultaneously. Step (5) and Step (6) can also be executed simultaneously.

#### 12.3.6.5.1.7.2.3 Using ADMA

The sequence for using ADMA is shown in [Figure 12-316](#).





mmcsd-034

**Figure 12-316. Transaction Control with Data Transfer Using DAT Line Sequence (Using ADMA)**

- (1) Create Descriptor table for ADMA in the system memory.
- (2) Set the Descriptor address for ADMA in the MMCSDD0\_ADMA\_SYS\_ADDRESS register.
- (3) Set the value corresponding to the executed data byte length of one block in the MMCSDD0\_BLOCK\_SIZE register.
- (4) Set the value corresponding to the executed data block count in the MMCSDD0\_BLOCK\_COUNT register in accordance with [Table 12-386](#).
- (5) Set the argument value to the Argument register (MMCSDD0\_ARGUMENT1\_LO and MMCSDD0\_ARGUMENT1\_HI).
- (6) Set the value to the MMCSDD0\_TRANSFER\_MODE register. The Host Driver determines Multi/Single Block Select, Block Count Enable, Data Transfer Direction, Auto CMD12 Enable and DMA Enable in the MMCSDD0\_TRANSFER\_MODE register. Multi/Single Block Select and Block Count Enable are determined according to [Table 12-386](#). If response check is enabled (MMCSDD0\_TRANSFER\_MODE[7]

RESP\_ERR\_CHK\_ENA = 1), set MMCSD0\_TRANSFER\_MODE[8] RESP\_INTR\_DIS bit to 1 and select Response Type R1/R5 (MMCSD0\_TRANSFER\_MODE[6] RESP\_TYPE).

(7) Set the value to the MMCSD0\_COMMAND register.

**Note:** When writing to the upper byte [3] of the MMCSD0\_COMMAND register, the SD command is issued and DMA is started.

(8) If response check is enabled, go to stop (11) else wait for the Command Complete Interrupt (MMCSD0\_NORMAL\_INTR\_STS[0] CMD\_COMPLETE bit).

(9) Write 1 to the MMCSD0\_NORMAL\_INTR\_STS[0] CMD\_COMPLETE bit to clear this bit.

(10) Read Response register (MMCSD0\_RESPONSE\_0 - MMCSD0\_RESPONSE\_7) and get necessary information of the issued command.

(11) Wait for the Transfer Complete Interrupt (MMCSD0\_NORMAL\_INTR\_STS\_ENA[1] XFER\_COMPLETE) and ADMA Error Interrupt (MMCSD0\_ADMA\_ERR\_STATUS[1-0] ADMA\_ERR\_STATE).

(12) If MMCSD0\_NORMAL\_INTR\_STS\_ENA[1] XFER\_COMPLETE bit is set to 1, go to Step (13) else if MMCSD0\_ADMA\_ERR\_STATUS[1-0] ADMA\_ERR\_STATE bit field is set to 1, go to Step (14).

(13) Write 1 to the MMCSD0\_NORMAL\_INTR\_STS\_ENA[1] XFER\_COMPLETE bit to clear this bit.

(14) Write 1 to the MMCSD0\_ADMA\_ERR\_STATUS[1-0] ADMA\_ERR\_STATE bit field to clear this bit.

(15) Abort ADMA operation. SD card operation should be stopped by issuing abort command. If necessary, the Host Driver checks MMCSD0\_ADMA\_ERR\_STATUS register to detect why ADMA error is generated.

**Note:** Step (3) and Step (4) can be executed simultaneously. Step (6) and Step (7) can also be executed simultaneously.

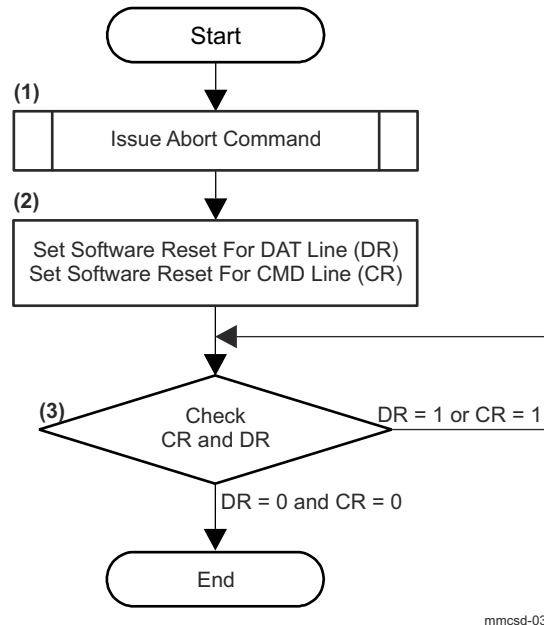
#### 12.3.6.5.1.8 Abort Transaction

An abort transaction is performed by issuing CMD12 for a SD memory card and by issuing CMD52 for a SDIO card. There are two cases where the Host Driver needs to do an Abort Transaction. The first case is when the Host Driver stops Infinite Block Transfers. The second case is when the Host Driver stops transfers while a Multiple Block Transfer is executing.

There are two ways to issue an Abort Command. The first is an asynchronous abort. The second is a synchronous abort. In an asynchronous abort sequence, the Host Driver can issue an Abort Command at any time unless MMCSD0\_PRESENTSTATE[0] INHIBIT\_CMD bit is set to 1. In a synchronous abort, the Host Driver shall issue an Abort Command after the data transfer stopped by using MMCSD0\_BLOCK\_GAP\_CONTROL[0] STOP\_AT\_BLK\_GAP bit.

##### 12.3.6.5.1.8.1 Asynchronous Abort

The sequence for Asynchronous Abort is shown in [Figure 12-317](#).

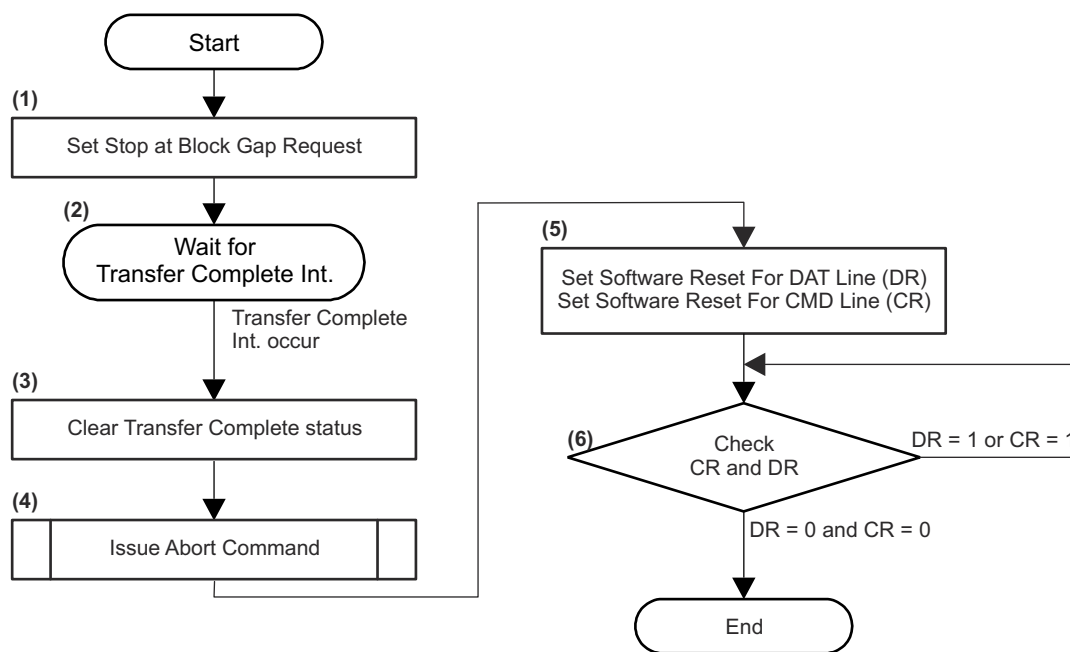


mmcsd-035

**Figure 12-317. Asynchronous Abort Sequence**

- (1) Issue Abort Command in accordance.
- (2) Set both MMCSD0\_SOFTWARE\_RESET[2] SWRST\_FOR\_DAT bit and MMCSD0\_SOFTWARE\_RESET[1] SWRST\_FOR\_CMD bit to 1 to do software reset.
- (3) Check MMCSD0\_SOFTWARE\_RESET[2] SWRST\_FOR\_DAT bit and MMCSD0\_SOFTWARE\_RESET[1] SWRST\_FOR\_CMD bit. If both MMCSD0\_SOFTWARE\_RESET[2] SWRST\_FOR\_DAT bit and MMCSD0\_SOFTWARE\_RESET[1] SWRST\_FOR\_CMD bit are 0, go to "End". If either MMCSD0\_SOFTWARE\_RESET[2] SWRST\_FOR\_DAT bit or MMCSD0\_SOFTWARE\_RESET[1] SWRST\_FOR\_CMD bit is 1, go to step (3).

### 12.3.6.5.1.8.2 Synchronous Abort



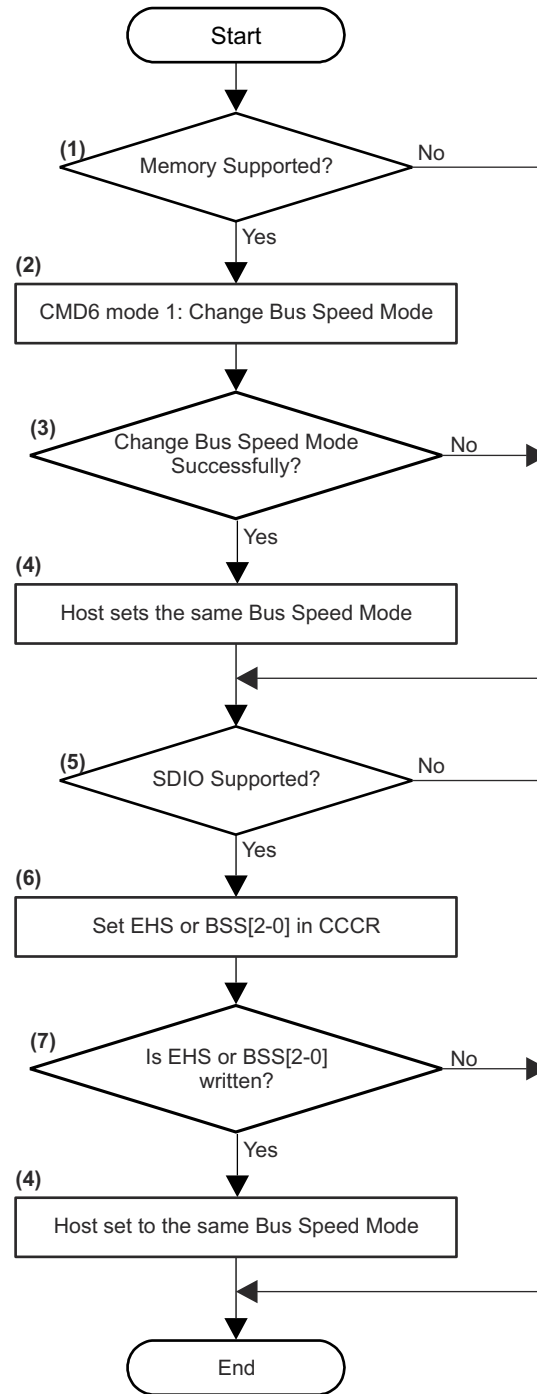
mmcsd-036

**Figure 12-318. Synchronous Abort Sequence**

- (1) Set the MMCSD0\_BLOCK\_GAP\_CONTROL[0] STOP\_AT\_BLK\_GAP bit to 1 to stop SD transactions.
- (2) Wait for the Transfer Complete Interrupt (MMCSD0\_NORMAL\_INTR\_STS\_ENA[1] XFER\_COMPLETE).
- (3) Set the MMCSD0\_NORMAL\_INTR\_STS\_ENA[1] XFER\_COMPLETE bit to 1 to clear this bit.
- (4) Issue the Abort Command
- (5) Set both MMCSD0\_SOFTWARE\_RESET[2] SWRST\_FOR\_DAT bit and MMCSD0\_SOFTWARE\_RESET[1] SWRST\_FOR\_CMD bit to 1 to do software reset.
- (6) Check both MMCSD0\_SOFTWARE\_RESET[2] SWRST\_FOR\_DAT bit and MMCSD0\_SOFTWARE\_RESET[1] SWRST\_FOR\_CMD bit. If both MMCSD0\_SOFTWARE\_RESET[2] SWRST\_FOR\_DAT bit and MMCSD0\_SOFTWARE\_RESET[1] SWRST\_FOR\_CMD bit are 0, go to "End". If either MMCSD0\_SOFTWARE\_RESET[2] SWRST\_FOR\_DAT bit or MMCSD0\_SOFTWARE\_RESET[1] SWRST\_FOR\_CMD bit is 1, go to step (6).

### 12.3.6.5.1.9 Changing Bus Speed Mode

This section describes the sequence for switching the bus speed mode: Default Speed, High Speed mode and UHS-I mode. The switch command (CMD6) is used to change memory card bus speed mode. The EHS bit (SDIO Version 2.00) or BSS[2-0] bits (SDIO Version 3.00) in the CCCR register is used to change the bus speed mode for SDIO card. In case of Combo card, either of the switch method changes both memory and IO bus speed mode. This means the first switch is effective. Refer to the Physical Layer Specification Version 3.0x and the SDIO Specification Version 3.00 for more information about switching bus speed. [Figure 12-319](#) shows the sequence for switching bus speed mode for Combo Card. Note that if the card is locked, bus width cannot be changed. Unlock the card is required before changing bus width.



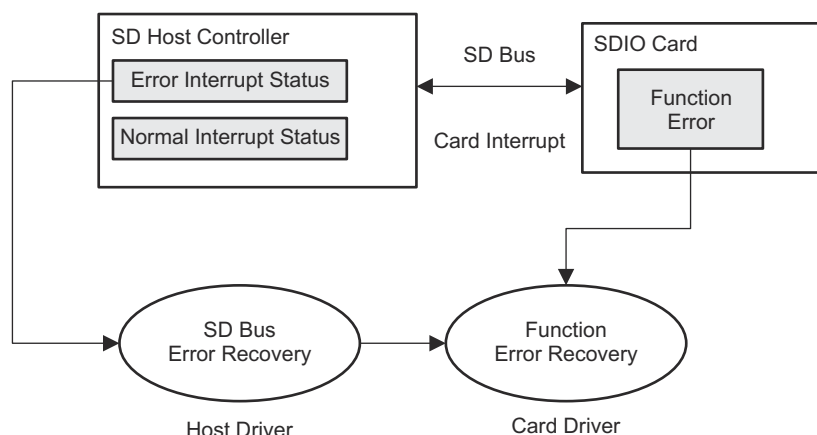
mmcsd-037

**Figure 12-319. Changing Bus Speed Mode**

- (1) The Host Driver checks if the card supports memory. If not supported, go to (5).
- (2) Issue CMD6 with mode 1 to change bus speed mode (Default, High Speed mode or UHS-I mode).
- (3) Check the response of CMD6. If the card does not supports CMD6 (no response) or bus speed is not changed successfully, go to step (5). In this case, the card is in Default Speed mode.
- (4) The Host Driver changes the Host Controller bus speed mode to the same mode.

- (5) The Host Driver checks if the card supports SDIO. If not supported, go to the end.
- (6) Issue CMD52 to write EHS bit or BSS[2-0] bits in CCCR to change bus speed mode (the same bus speed mode of (2) shall be set).
- (7) If EHS or BSS[2-0] are not changed successfully, go to the end.
- (8) The Host Driver changes the Host Controller bus speed to the same mode. In case of Combo card, bus speed is already changed at step (4) and this step does not affect changing bus speed.

#### 12.3.6.5.1.10 Error Recovery



mmcsd-038

**Figure 12-320. Error Report and Recovery**

Figure 12-320 shows concept of error report and its recovery. The Host Controller has 2 interrupt status registers. If an error occurs in the SD Bus transaction, one of the bits is set in the MMCSDB0\_ERROR\_INTR\_STS register. If the function errors occur in the SDIO card, the card interrupt informs these function errors and the Card interrupt is set in the MMCSDB0\_NORMAL\_INTR\_STS\_ENA register (the Card Interrupt is used to inform not only error statuses but also normal information. For example, to inform function ready). The Card Driver shall do function error recovery because the Host Driver does not know how to control the function. In the case that function error occurs due to SD Bus error, SD Bus error recovery is required before function error recovery. Abort command is used to recover SD Bus, and then the Host Driver should save error statuses related to SD Bus errors before issuing abort command and transfer these statuses to the Card Driver. These statuses may be used to recover function error. Following explanations are related to SD Bus error recovery. This specification does not specify the function error recovery.

When an error occurs during data transfer in 2L-HD UHS-II mode, there will be the case that Host Controller cannot drive D0 lane in input mode due to DIR LSS for retrieving lane direction is not detected. In this case, Host Driver cannot issue abort command for recovery. Then if DIR LSS is not detected, Host Controller sets MMCSDB0\_UHS2\_ERR\_INTR\_STS[17] DEADLOCK\_TIMEOUT bit. Host Driver should execute power cycle if MMCSDB0\_UHS2\_ERR\_INTR\_STS[17] DEADLOCK\_TIMEOUT bit is detected to recover from this error. Furthermore, if this type of error is detected several times in 2L-HD, Host Driver should use FD mode rather than using 2L-HD.

Implementation Note:

If the Card Driver cannot recover the function errors, the Host Driver should try following methods.

- (9) Using IOEx for SDIO card

IOEx may be used as the reset per function basis. Sequence is as follows:

Clear IOEx = 0 and wait until IORx = 0 and then set IOEx = 1 again. SDIO may be recovered when IORx = 1.

- (10) Using reset command for memory and SDIO card Re-initialization sequence is required.

(11) Off and on power supply for the SD Bus.

The card may be recovered by the power on reset. Re-initialization sequence is required.

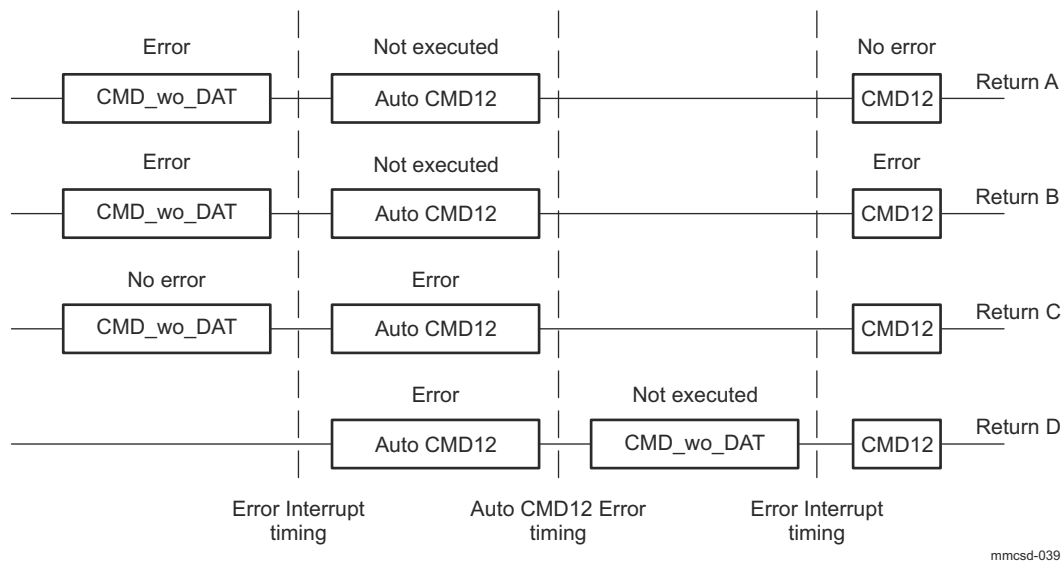
The two cases where the Host Driver needs the "Error Recovery" sequence are classified as follows:

(1) Error Interrupt Recovery:

If error interrupt is indicated by the MMCSD0\_ERROR\_INTR\_STS register, the Host Driver shall apply this sequence.

(2) Auto CMD12 Error Recovery:

If there are errors in Auto CMD12, the Host Driver shall apply this sequence. In terms of Return Status, Auto CMD12 Error Recovery is classified into 4 cases. It is shown in Figure 12-321. If error occurs during memory write transfer, strongly recommend using ACMD22 and then in the following recovery sequence, retry to send remaining blocks not written.



**Figure 12-321. Return Status of Auto CMD12 Error Recovery**

Implementation Note:

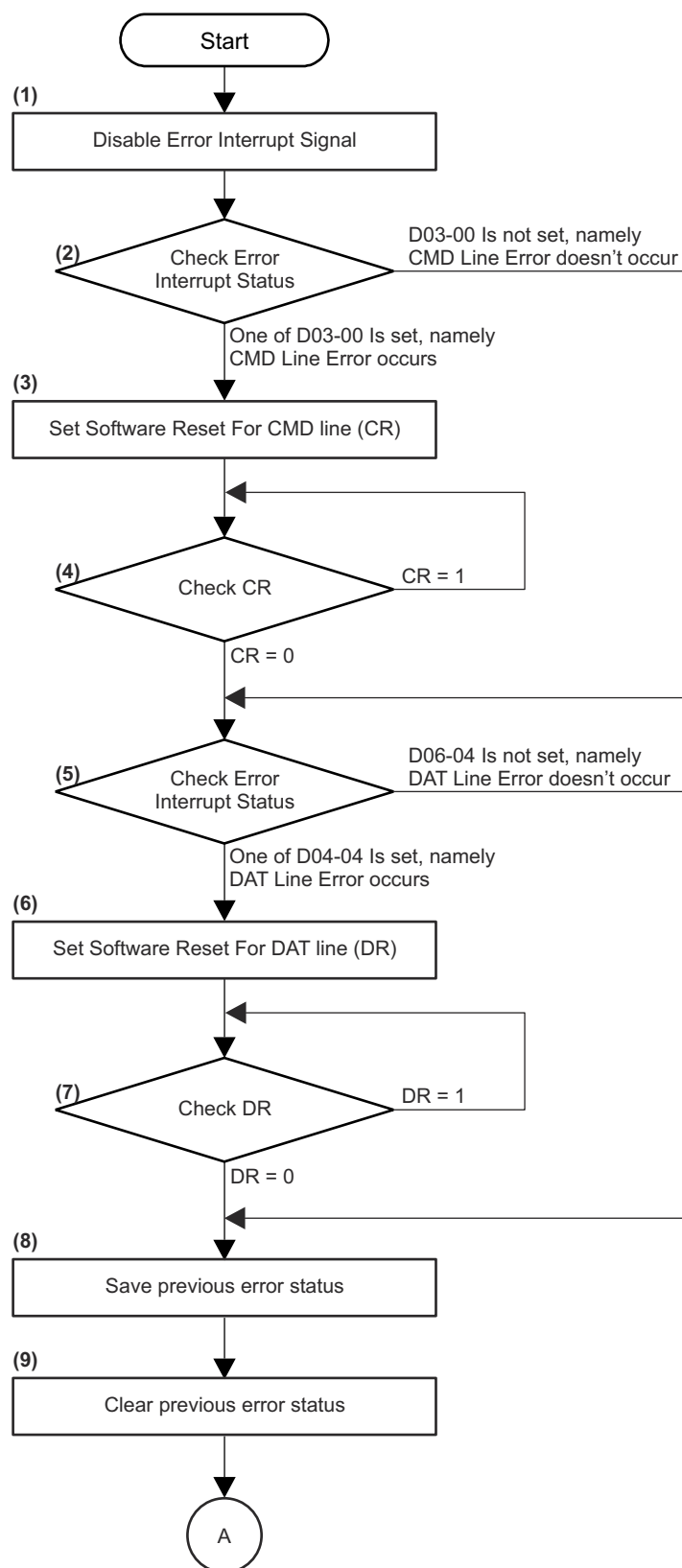
Abort command is used to recover from SD Bus error. SDIO transaction abort using CMD52 returns response but in the case of memory transaction abort using CMD12, response returns depending on the memory card state. If no response returns after issue CMD12, the Host Driver should check card state using CMD13. If the state is "tran" in the CURRENT\_STATE, consider CMD12 is successful.

Implementation Note:

The following sequence is one possible error recovery flow. There may be another methods, sometimes using interrupts or polling. It can be possible to use another flows, based on Host System requirements.

In these error recovery sequences, return statuses for the next sequence. When the Host Controller cannot issue the next command due to SD Bus error, the error recovery sequences return "Non-recoverable" status. In this case, the Host System may cut off power to the SD Bus and then power on SD Bus and initialize both the Host Controller and the SD card again.

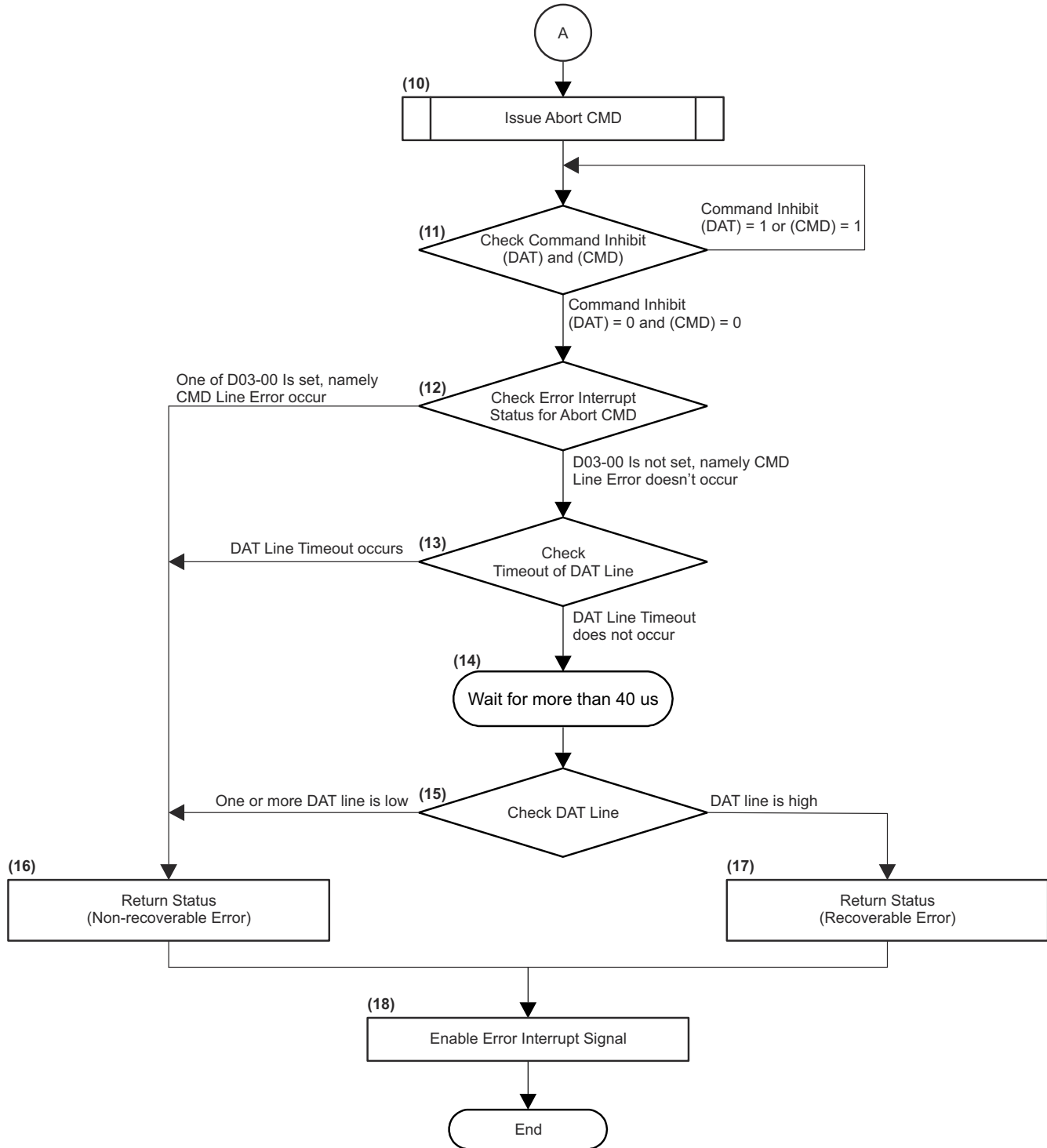
### 12.3.6.5.1.10.1 Error Interrupt Recovery



mmcsd-040

**Figure 12-322. Error Interrupt Recovery Sequence (1)**





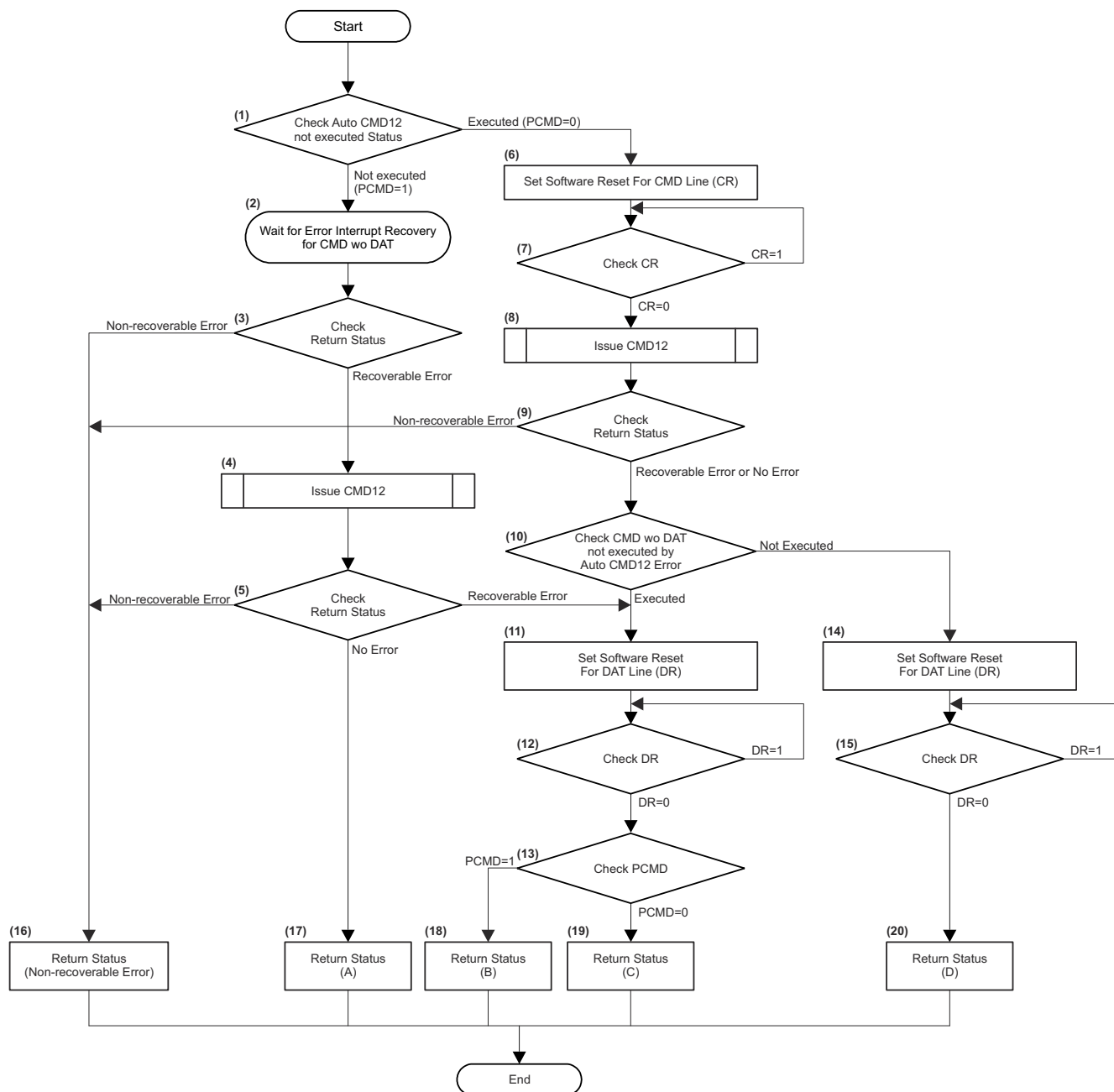
mmcsd-041

**Figure 12-323. Error Interrupt Recovery Sequence (2)**

- (1) Disable the Error Interrupt Signal (MMCSD0\_ERROR\_INTR\_SIG\_ENA).
- (2) Check bits D03-00 in the MMCSD0\_ERROR\_INTR\_STS register. If one of these bits (D03-00) is set to 1, go to step (3). If none are set to 1 (all are 0), go to step (5).
- (3) Set MMCSD0\_SOFTWARE\_RESET[1] SWRST\_FOR\_CMD bit to 1.

- (4) Check MMCSD0\_SOFTWARE\_RESET[1] SWRST\_FOR\_CMD bit. If MMCSD0\_SOFTWARE\_RESET[1] SWRST\_FOR\_CMD bit is 0, go to step (5). If it is 1, go to step (4).
- (5) Check bits D06-04 in the MMCSD0\_ERROR\_INTR\_STS register. If one of these bits (D06-04) is set to 1, go to step (6). If none are set to 1 (all are 0), go to step (8).
- (6) Set MMCSD0\_SOFTWARE\_RESET[2] SWRST\_FOR\_DAT bit to 1 for software reset of the DAT line.
- (7) Check MMCSD0\_SOFTWARE\_RESET[2] SWRST\_FOR\_DAT bit. If MMCSD0\_SOFTWARE\_RESET[2] SWRST\_FOR\_DAT bit is 0, go to step (8). If it is 1, go to step (7).
- (8) Save previous error status.
- (9) Clear previous error status with setting them to 1.
- (10) Issue Abort Command.
- (11) MMCSD0\_PRESENTSTATE[1] INHIBIT\_DAT bit and MMCSD0\_PRESENTSTATE[0] INHIBIT\_CMD bit. Repeat this step until both MMCSD0\_PRESENTSTATE[1] INHIBIT\_DAT bit and MMCSD0\_PRESENTSTATE[0] INHIBIT\_CMD bit are set to 0.
- (12) Check bits D03-00 in the MMCSD0\_ERROR\_INTR\_STS register for Abort Command. If one of these bits is set to 1, go to step (16). If none of these bits are set to 1 (all are 0), go to step (13).
- (13) Check MMCSD0\_ERROR\_INTR\_STS[4] DATA\_TIMEOUT bit. If this bit is set to 1, go to step (16). If it is 0, go to step (14).
- (14) Wait for more than 40 us.
- (15) By monitoring the DAT [3:0] Line Signal Level in the MMCSD0\_PRESENTSTATE register, judge whether the level of the DAT line is low or not. If one or more DAT lines are low, go to step (16). If the DAT lines are high, go to step (17).
- (16) Return Status of "Non-recoverable Error".
- (17) Return Status of "Recoverable Error".
- (18) Enable the Error Interrupt Signal (MMCSD0\_ERROR\_INTR\_SIG\_ENA).

### 12.3.6.5.1.10.2 Auto CMD12 Error Recovery



mmcsd-042

**Figure 12-324. Auto CMD12 Error Recovery**

The sequence for Auto CMD12 Error Recovery is shown in [Figure 12-324](#). Following four cases A-D shall be covered.

A: An error occurred in CMD\_wo\_DAT, but not in the SD memory transfer.

B: An error occurred in CMD\_wo\_DAT, and also occurred in the SD memory transfer.

C: An error did not occur in CMD\_wo\_DAT, but an error occurred in the SD memory transfer.

D: CMD\_wo\_DAT was not issued, and an error occurred in the SD memory transfer.

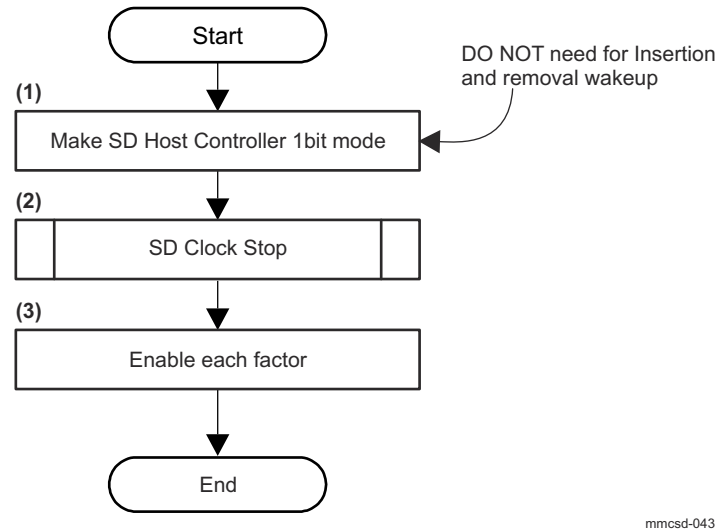
- (1) Check MMCSD0\_AUTOCMD\_ERR\_STS[0] ACMD12\_NOT\_EXEC bit. If this bit is set to 1, go to step (2). If this bit is set to 0, go to step (6). In addition, the Host Driver shall define PCMD flag, which changes to 1 if MMCSD0\_AUTOCMD\_ERR\_STS[0] ACMD12\_NOT\_EXEC bit is set to 1.
- (2) Wait for Error Interrupt Recovery for CMD\_wo\_DAT.
- (3) Check "Return Status". In the case of "Non-recoverable Error", go to step (16). In the case of "Recoverable Error", go to step (4).
- (4) Issue CMD12.
- (5) If the CMD line errors occur for the CMD12 (one of D03-00 is set in the MMCSD0\_ERROR\_INTR\_STS register), "Return Status" is "Non-recoverable Error" and go to step (16). If not CMD line error and busy timeout error occur (D04 is set in the MMCSD0\_ERROR\_INTR\_STS register), "Return Status" is "Recoverable Error" and go to step (11). Otherwise, "Return Status" is "No error" and go to step (17).
- (6) Set MMCSD0\_SOFTWARE\_RESET[1] SWRST\_FOR\_CMD bit to 1 for software reset of the CMD line.
- (7) Check MMCSD0\_SOFTWARE\_RESET[1] SWRST\_FOR\_CMD bit. If MMCSD0\_SOFTWARE\_RESET[1] SWRST\_FOR\_CMD bit is 0, go to step (8). If it is 1, go to step (7).
- (8) Issue CMD12. Acceptance of CMD12 depends on the state of the card. CMD12 may make the card to return to tran state. If the card is already in tran state, the card does not response to CMD12.
- (9) Check "Return Status" for CMD12. If "Return Status" returns "Non-recoverable Error", go to step (16). In the case of "Recoverable Error" or "No error", go to step (10).
- (10) Check the MMCSD0\_AUTOCMD\_ERR\_STS[0] ACMD12\_NOT\_EXEC bit. If this bit is 0, go to step (11). If it is 1, go to step (14).
- (11) Set MMCSD0\_SOFTWARE\_RESET[2] SWRST\_FOR\_DAT bit to 1 for software reset of the DAT line.
- (12) Check MMCSD0\_SOFTWARE\_RESET[2] SWRST\_FOR\_DAT bit. If MMCSD0\_SOFTWARE\_RESET[2] SWRST\_FOR\_DAT bit is 0, go to step (13). If it is 1, go to step (12).
- (13) Check the PCMD flag. If PCMD is 1, go to step (18). If it is 0, go to step (19).
- (14) Set MMCSD0\_SOFTWARE\_RESET[2] SWRST\_FOR\_DAT bit to 1 for software reset of the DAT line.
- (15) Check MMCSD0\_SOFTWARE\_RESET[2] SWRST\_FOR\_DAT bit. If MMCSD0\_SOFTWARE\_RESET[2] SWRST\_FOR\_DAT bit is 0, go to step (20). If it is 1, go to step (15).
- (16) Return Status of "Non-recoverable Error".
- (17) Return Status that an error has occurred in CMD\_wo\_DAT, but not in the SD memory transfer.
- (18) Return Status that an error has occurred in both CMD\_wo\_DAT, and the SD memory transfer.
- (19) Return Status that an error has not occurred in CMD\_wo\_DAT, but has occurred in the SD memory transfer.
- (20) Return Status that CMD\_wo\_DAT has not been issued, and an error has occurred in the SD memory transfer.

#### 12.3.6.5.1.11 Wakeup Control (Optional)

After the Host System goes into standby mode, the Host System can resume from standby via a wakeup event initiated by one of the following three events:

- (1) Interrupt from a SD card: If an SD card interrupt occurs, the Host System can resume from standby mode. If the Host System uses this wakeup factor, SD Bus power shall be kept on.
- (2) Insertion of SD card: If a SD card is inserted, the Host System can resume from standby mode.
- (3) Removal of SD card: If a SD card is removed, the Host System can resume from standby mode.

The sequence for preparing wakeup before the Host System goes into standby mode is shown in [Figure 12-325](#).



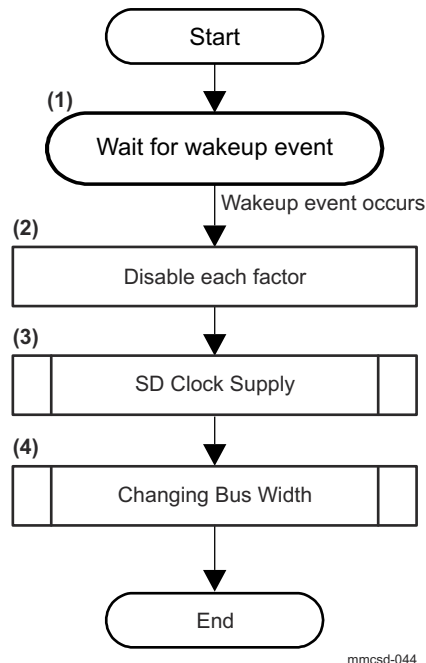
**Figure 12-325. Wakeup Control before Standby Mode**

(1) Set MMCSD0\_HOST\_CONTROL1[1] DATA\_WIDTH bit to 0.

(2) Execute SD Clock Stop Sequence as described in [Section 12.3.6.5.1.2.2](#), *SD Clock Supply and Stop Sequence*.

(3) Clear the MMCSD0\_NORMAL\_INTR\_STS register and the MMCSD0\_NORMAL\_INTR\_SIG\_ENA register, and then set the enable bits of each wakeup event factor to 1 in the MMCSD0\_WAKEUP\_CONTROL register and set the bits of MMCSD0\_ERROR\_INTR\_STS\_ENA register to use wakeup.

The sequence for wakeup once in standby mode is shown in [Figure 12-326](#).



**Figure 12-326. Wakeup from Standby**

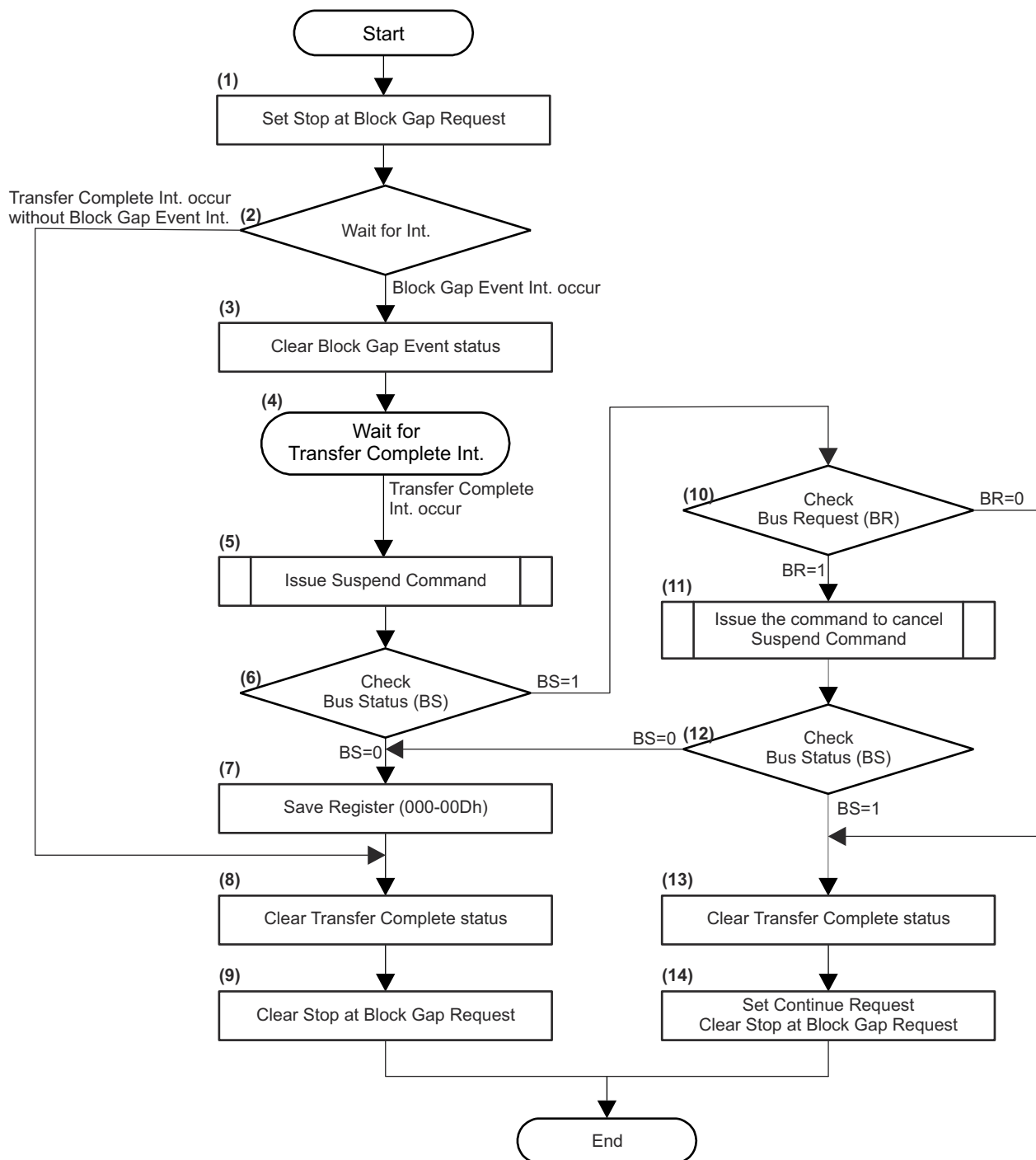
(1) Wait for wakeup event.

- (2) Set the enable bits of each wakeup event factor to 0 in the MMCSD0\_WAKEUP\_CONTROL register and then clear event statuses in the MMCSD0\_NORMAL\_INTR\_STS register. If necessary, set the MMCSD0\_NORMAL\_INTR\_SIG\_ENA register.
- (3) Execute SD Clock Supply Sequence (see [Section 12.3.6.5.1.2.2](#), *SD Clock Supply and Stop Sequence*).
- (4) Set the SD Bus width (see [Section 12.3.6.5.1.4](#), *Changing Bus Width*).

#### **12.3.6.5.1.12 Suspend/Resume (Optional, Not Supported from Version 4.00)**

If a SD card supports suspend and resume functionality, then the Host Controller can initiate suspend and resume. It is necessary for both the Host Controller and the SD card to support the function of "Read Wait". ADMA operation does not support this function.

### 12.3.6.5.1.12.1 Suspend Sequence



mmcsd-045

**Figure 12-327. The Sequence for Suspend**

(1) Set MMCSD0\_BLOCK\_GAP\_CONTROL[0] STOP\_AT\_BLK\_GAP bit to 1 to stop the SD transaction.

(2) Wait for an Interrupt. If MMCSD0\_NORMAL\_INTR\_STS[2] BLK\_GAP\_EVENT bit is set to 0 and MMCSD0\_NORMAL\_INTR\_STS[1] XFER\_COMPLETE bit is set to 1, go to step (8). If MMCSD0\_NORMAL\_INTR\_STS[2] BLK\_GAP\_EVENT bit is set to 1, go to step (3).

- (3) Set MMCSD0\_NORMAL\_INTR\_STS[2] BLK\_GAP\_EVENT bit to 1 to clear this bit.
- (4) Wait for the Transfer Complete Interrupt (MMCSD0\_NORMAL\_INTR\_STS\_ENA[1] XFER\_COMPLETE).
- (5) Issue the Suspend Command in accordance with [Section 12.3.6.5.1.7.1](#), *Transaction Control without Data Transfer Using DAT Line*.
- (6) Check the BS value of the response data. If BS is 0, go to step (7). If BS is 1, go to step (10).
- (7) Save the register .
- (8) Set MMCSD0\_NORMAL\_INTR\_STS[1] XFER\_COMPLETE bit to 1 to clear this bit.
- (9) Set MMCSD0\_BLOCK\_GAP\_CONTROL[0] STOP\_AT\_BLK\_GAP bit to 0 to clear this bit.
- (10) Check the BR value of the response data. If BR is 1, go to step (11). If BR is 0, go to step (13).
- (11) Issues the command to cancel the previous suspend command in accordance with [Section 12.3.6.5.1.7.1](#), *Transaction Control without Data Transfer Using DAT Line*.
- (12) Check the BS value of the response data. If BS is 0, go to step (7). If BS is 1, go to step (13).
- (13) Set MMCSD0\_NORMAL\_INTR\_STS[1] XFER\_COMPLETE bit to 1 to clear this bit.
- (14) Set MMCSD0\_BLOCK\_GAP\_CONTROL[1] CONTINUE bit to 1 to continue the transaction. At the same time, write 0 to MMCSD0\_BLOCK\_GAP\_CONTROL[0] STOP\_AT\_BLK\_GAP bit to clear this bit.

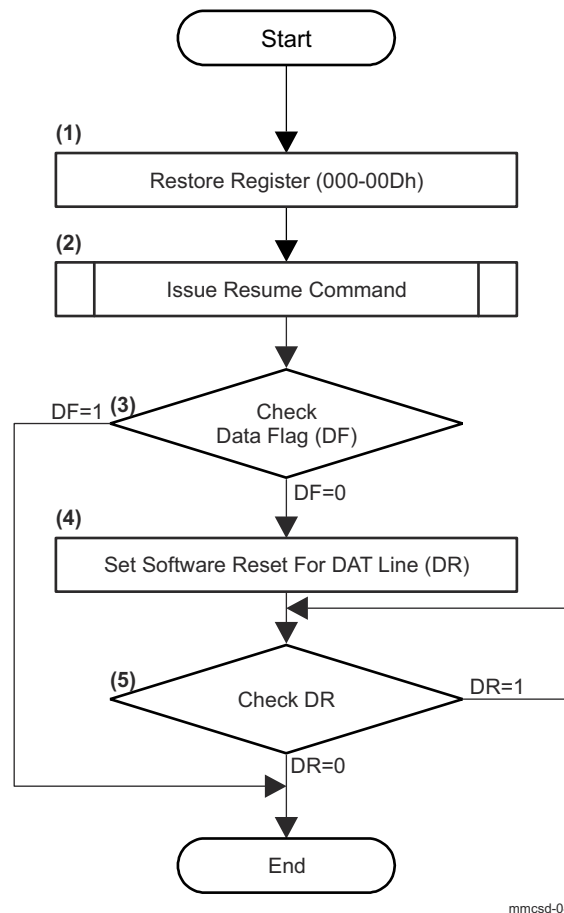
Conditions			Suspend/Resume Function	
Host Suspend/Resume Support	Card Suspend/Resume Support	Card Read Wait Support	Write Suspend/Resume	Read Suspend/Resume
Not supported	Don't care	Don't care	Cannot be used	Cannot be used
Supported	Not supported	Don't care	Cannot be used	Cannot be used
Supported	Supported	Not supported	Can be used	Cannot be used
Supported	Supported	Supported	Can be used	Can be used

mmcsd-046

**Figure 12-328. Suspend/Resume Condition**



### 12.3.6.5.1.12.2 Resume Sequence



mmcsd-047

**Figure 12-329. The Sequence for Resume**

- (1) Restore the register .
- (2) Issue the Resume Command .
- (3) Check the DF value of the response data. If DF is 0, go to step (4). If DF is 1, go to "End".
- (4) Set MMCSD0\_SOFTWARE\_RESET[2] SWRST\_FOR\_DAT bit to 1 for software reset of the DAT line.
- (5) Check MMCSD0\_SOFTWARE\_RESET[2] SWRST\_FOR\_DAT bit. If MMCSD0\_SOFTWARE\_RESET[2] SWRST\_FOR\_DAT bit is 0, go to "End". If it is 1, go to step (5).

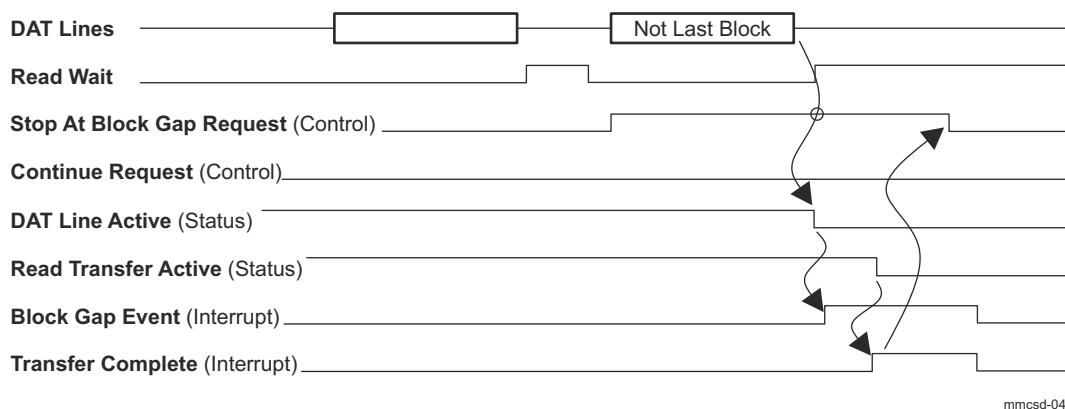
### 12.3.6.5.1.12.3 Stop At Block Gap/Continue Timing for Read Transaction

The timing of Stop At Block Gap Request and Continue Request. The Transfer Complete interrupt (MMCSD0\_NORMAL\_INTR\_STS[1] XFER\_COMPLETE) is always generated by setting MMCSD0\_BLOCK\_GAP\_CONTROL[0] STOP\_AT\_BLK\_GAP bit where data transfer is stopped. However, generation of the Block Gap Event interrupt (MMCSD0\_NORMAL\_INTR\_STS[2] BLK\_GAP\_EVENT) is dependent on whether the last data block is sent or not. Block Gap Event is not generated If all data blocks are transferred (the last block is transferred). It is not necessary to enable Block Gap Event interrupt. The status can be checked when transfer complete interrupt is detected. It is not necessary to use Continue Request (MMCSD0\_BLOCK\_GAP\_CONTROL[1] CONTINUE) if Block Gap Event status is not set because there is no further data to be transferred. If Read Wait is not supported, Host Controller stops SD Clock at the block gap.

Implementation Note:

The MMCSD0\_BLOCK\_GAP\_CONTROL[2] RDWAIT\_CTRL, MMCSD0\_PRESENTSTATE[2] DATA\_LINE\_ACTIVE, MMCSD0\_PRESENTSTATE[9] RD\_XFER\_ACTIVE bits shall be set and cleared by the Host Controller.

The MMCSD0\_BLOCK\_GAP\_CONTROL[0] STOP\_AT\_BLK\_GAP bit shall be set and cleared by the Host Driver. The MMCSD0\_BLOCK\_GAP\_CONTROL[1] CONTINUE bit shall be set by the Host Driver and be cleared by the Host Controller. The MMCSD0\_NORMAL\_INTR\_STS[2] BLK\_GAP\_EVENT bit and MMCSD0\_NORMAL\_INTR\_STS[1] XFER\_COMPLETE bit shall be set by the Host Controller and be cleared by the Host Driver.



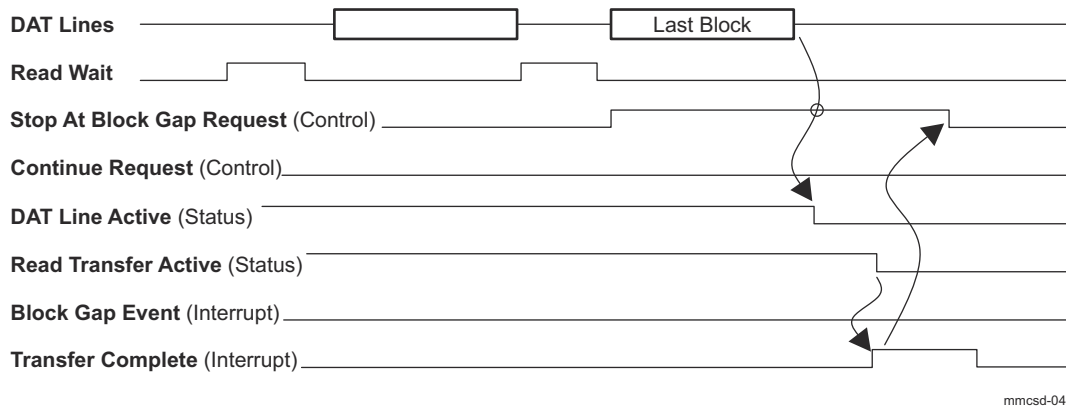
**Figure 12-330. Wait Read Transfer by Stop At Block Gap Request**

The Host Controller can accept a Stop At Block Gap Request (MMCSD0\_BLOCK\_GAP\_CONTROL[0] STOP\_AT\_BLK\_GAP) when all the following conditions are met.

- (1) It is at the block gap.
- (2) The Host Controller can assert read wait or it is already asserted.
- (3) The MMCSD0\_BLOCK\_GAP\_CONTROL[2] RDWAIT\_CTRL bit is set to 1.

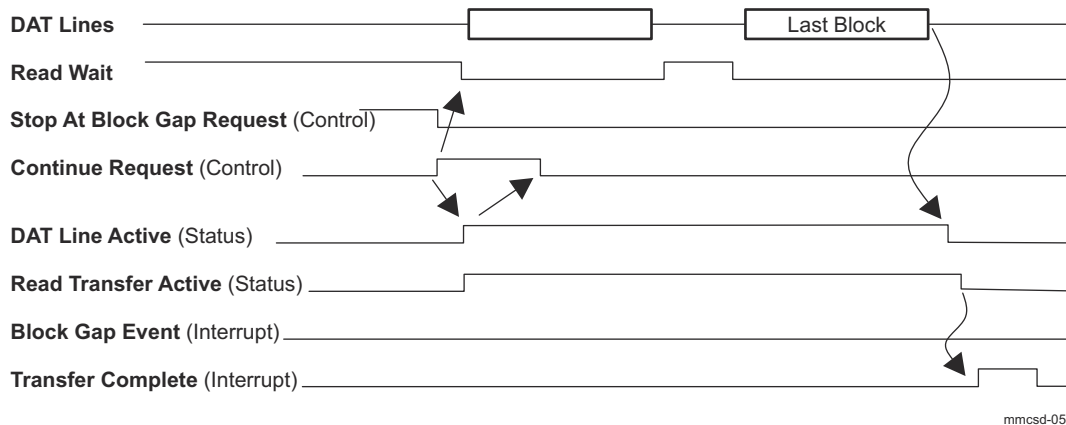
After accepting the MMCSD0\_BLOCK\_GAP\_CONTROL[0] STOP\_AT\_BLK\_GAP bit:

- (1) Clear MMCSD0\_PRESENTSTATE[2] DATA\_LINE\_ACTIVE bit and generate the Block Gap Event Interrupt (MMCSD0\_NORMAL\_INTR\_STS[2] BLK\_GAP\_EVENT).
- (2) After all valid data has been read (No valid read data remains in the Host Controller), clear the MMCSD0\_PRESENTSTATE[9] RD\_XFER\_ACTIVE bit and generate the Transfer Complete Interrupt (MMCSD0\_NORMAL\_INTR\_STS[1] XFER\_COMPLETE).
- (3) After accepting MMCSD0\_NORMAL\_INTR\_STS[1] XFER\_COMPLETE bit, clear the MMCSD0\_BLOCK\_GAP\_CONTROL[0] STOP\_AT\_BLK\_GAP bit.



**Figure 12-331. Stop At Block Gap Request is Not Accepted at the Last Block of the Read Transfer**

If the MMCSD0\_BLOCK\_GAP\_CONTROL[0] STOP\_AT\_BLK\_GAP bit is set to 1 during the last block transfer, the Host Controller shall not accept the MMCSD0\_BLOCK\_GAP\_CONTROL[0] STOP\_AT\_BLK\_GAP bit and stops the transaction normally. The Block Gap Event Interrupt (MMCSD0\_NORMAL\_INTR\_STS[2] BLK\_GAP\_EVENT) is not generated. When the Transfer Complete Interrupt (MMCSD0\_NORMAL\_INTR\_STS[1] XFER\_COMPLETE) is generated, and if the MMCSD0\_NORMAL\_INTR\_STS[2] BLK\_GAP\_EVENT bit status is not set to 1, the driver shall clear the MMCSD0\_BLOCK\_GAP\_CONTROL[0] STOP\_AT\_BLK\_GAP bit.



**Figure 12-332. Continue Read Transfer by Continue Request**

To restart a stopped data transfer, set the MMCSD0\_BLOCK\_GAP\_CONTROL[1] CONTINUE bit to 1 (the MMCSD0\_BLOCK\_GAP\_CONTROL[0] STOP\_AT\_BLK\_GAP bit shall be set to 0).

After accepting the MMCSD0\_BLOCK\_GAP\_CONTROL[1] CONTINUE bit:

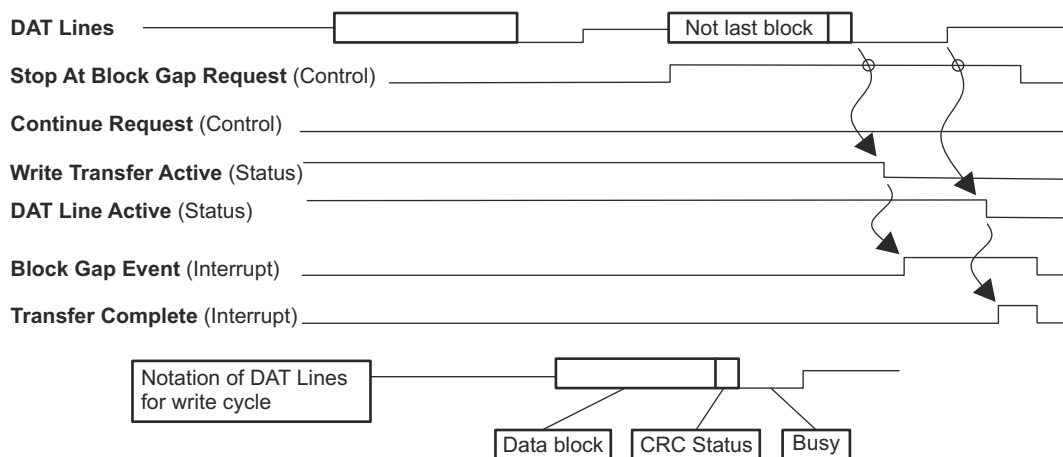
- (1) Release MMCSD0\_BLOCK\_GAP\_CONTROL[2] RDWAIT\_CTRL bit (if the data block can accept the next data).
- (2) Set the MMCSD0\_PRESENTSTATE[2] DATA\_LINE\_ACTIVE bit and the MMCSD0\_PRESENTSTATE[9] RD\_XFER\_ACTIVE bit.
- (3) The MMCSD0\_BLOCK\_GAP\_CONTROL[1] CONTINUE bit is automatically cleared by (2).

The end of the read transfer is specified by data length.

- (1) Clear the MMCSD0\_PRESENTSTATE[2] DATA\_LINE\_ACTIVE bit and do not generate the Block Gap Event Interrupt (MMCSD0\_NORMAL\_INTR\_STS[2] BLK\_GAP\_EVENT bit).

(2) After all valid data has been read (No valid read data remains in the Host Controller), clear the MMCSD0\_PRESENTSTATE[9] RD\_XFER\_ACTIVE bit and generate the Transfer Complete Interrupt (MMCSD0\_NORMAL\_INTR\_STS[1] XFER\_COMPLETE bit).

#### 12.3.6.5.1.2.4 Stop At Block Gap/Continue Timing for Write Transaction



mmcsd-051

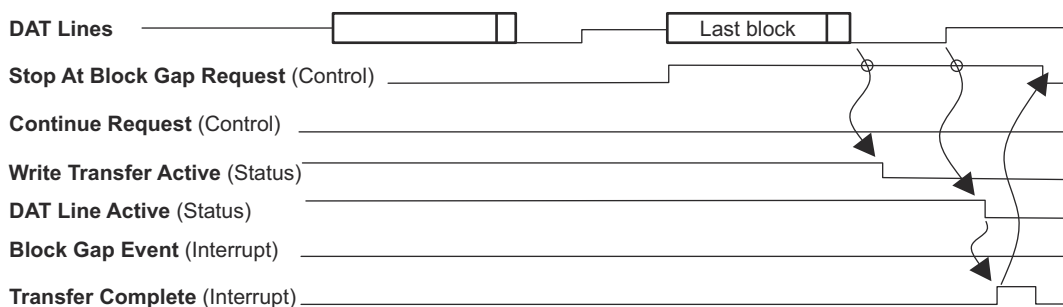
**Figure 12-333. Wait Write Transfer by Stop At Block Gap Request**

The Host Controller can accept the MMCSD0\_BLOCK\_GAP\_CONTROL[0] STOP\_AT\_BLK\_GAP bit when matches all following conditions:

- (1) It is at the block gap.
- (2) No valid write data remains in the Host Controller.

After accepting the MMCSD0\_BLOCK\_GAP\_CONTROL[0] STOP\_AT\_BLK\_GAP bit.

- (1) Clear the MMCSD0\_PRESENTSTATE[8] WR\_XFER\_ACTIVE bit and generate the Block Gap Event Interrupt (MMCSD0\_NORMAL\_INTR\_STS[2] BLK\_GAP\_EVENT bit).
- (2) After the busy signal is released, clear the MMCSD0\_PRESENTSTATE[2] DATA\_LINE\_ACTIVE bit and generate the Transfer Complete Interrupt (MMCSD0\_NORMAL\_INTR\_STS[1] XFER\_COMPLETE bit).
- (3) After accepting the Transfer Complete Interrupt (MMCSD0\_NORMAL\_INTR\_STS[1] XFER\_COMPLETE bit), clear the MMCSD0\_BLOCK\_GAP\_CONTROL[0] STOP\_AT\_BLK\_GAP bit.

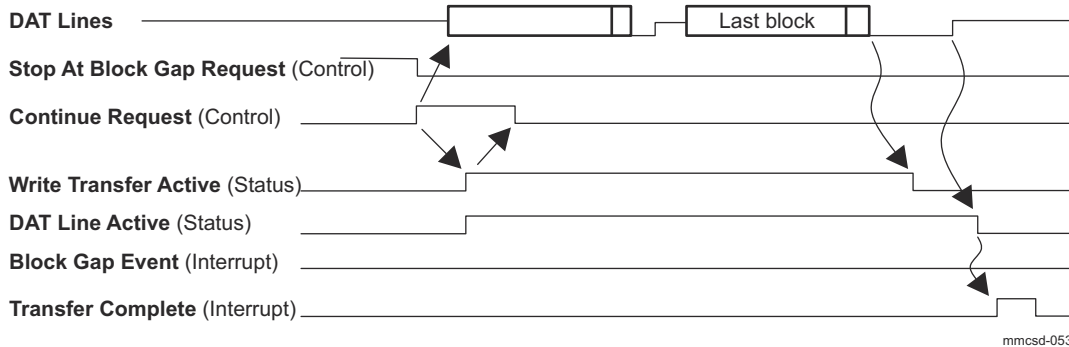


mmcsd-052

**Figure 12-334. Stop At Block Gap Request is Not Accepted at the Last Block of the Write Transfer**

If the MMCSD0\_BLOCK\_GAP\_CONTROL[0] STOP\_AT\_BLK\_GAP bit is set to 1 during the last block transfer, the Host Controller shall not accept the MMCSD0\_BLOCK\_GAP\_CONTROL[0] STOP\_AT\_BLK\_GAP bit and terminates the transaction normally. The Block Gap Event Interrupt (MMCSD0\_NORMAL\_INTR\_STS[2] BLK\_GAP\_EVENT) is not generated. When the Transfer

Complete Interrupt (MMCSD0\_NORMAL\_INTR\_STS[1] XFER\_COMPLETE) is generated, and if the MMCSD0\_NORMAL\_INTR\_STS[2] BLK\_GAP\_EVEN bit is not set to 1, the driver shall clear the MMCSD0\_BLOCK\_GAP\_CONTROL[0] STOP\_AT\_BLK\_GAP bit.



**Figure 12-335. Continue Write Transfer by Continue Request**

To restart a stopped data transfer, set the MMCSD0\_BLOCK\_GAP\_CONTROL[1] CONTINUE bit to 1 (the MMCSD0\_BLOCK\_GAP\_CONTROL[0] STOP\_AT\_BLK\_GAP bit shall be set to 0).

After accepting the MMCSD0\_BLOCK\_GAP\_CONTROL[1] CONTINUE bit:

(1) Set the MMCSD0\_PRESENTSTATE[2] DATA\_LINE\_ACTIVE bit and MMCSD0\_PRESENTSTATE[8] WR\_XFER\_ACTIVE bit.

(2) The MMCSD0\_BLOCK\_GAP\_CONTROL[1] CONTINUE bit is automatically cleared by (1).

The end of transfer is specified by data length.

(1) Clear the MMCSD0\_PRESENTSTATE[8] WR\_XFER\_ACTIVE bit, and do not generate the (MMCSD0\_NORMAL\_INTR\_STS[2] BLK\_GAP\_EVEN).

(2) After the busy signal is released, clear the MMCSD0\_PRESENTSTATE[2] DATA\_LINE\_ACTIVE bit and generates the Transfer Complete Interrupt (MMCSD0\_NORMAL\_INTR\_STS[1] XFER\_COMPLETE bit).

### 12.3.6.5.2 Driver Flow Sequence

#### Note

This section is applicable for MMCSD0, MMCSD1 and MMCSD2 but MMCSD0 registers are used in the content.

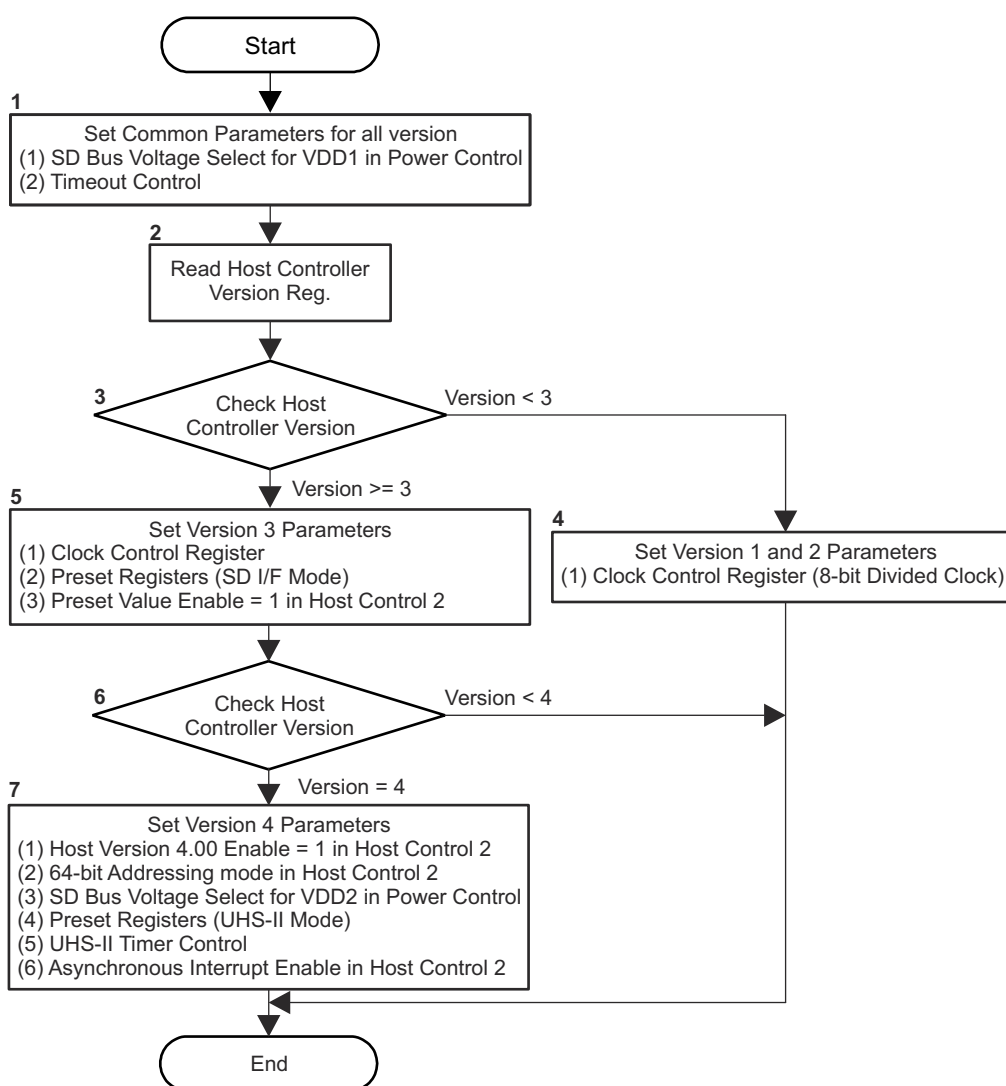
#### Note

UHSII is not supported. For more information, see [Section 12.3.6.1.2, Not Supported Features](#).

#### 12.3.6.5.2.1 Host Controller Setup and Card Detection

##### 12.3.6.5.2.1.1 Host Controller Setup Sequence

Figure 12-336 and Table 12-387 show a Host Controller setup sequence.



mmcsd-054

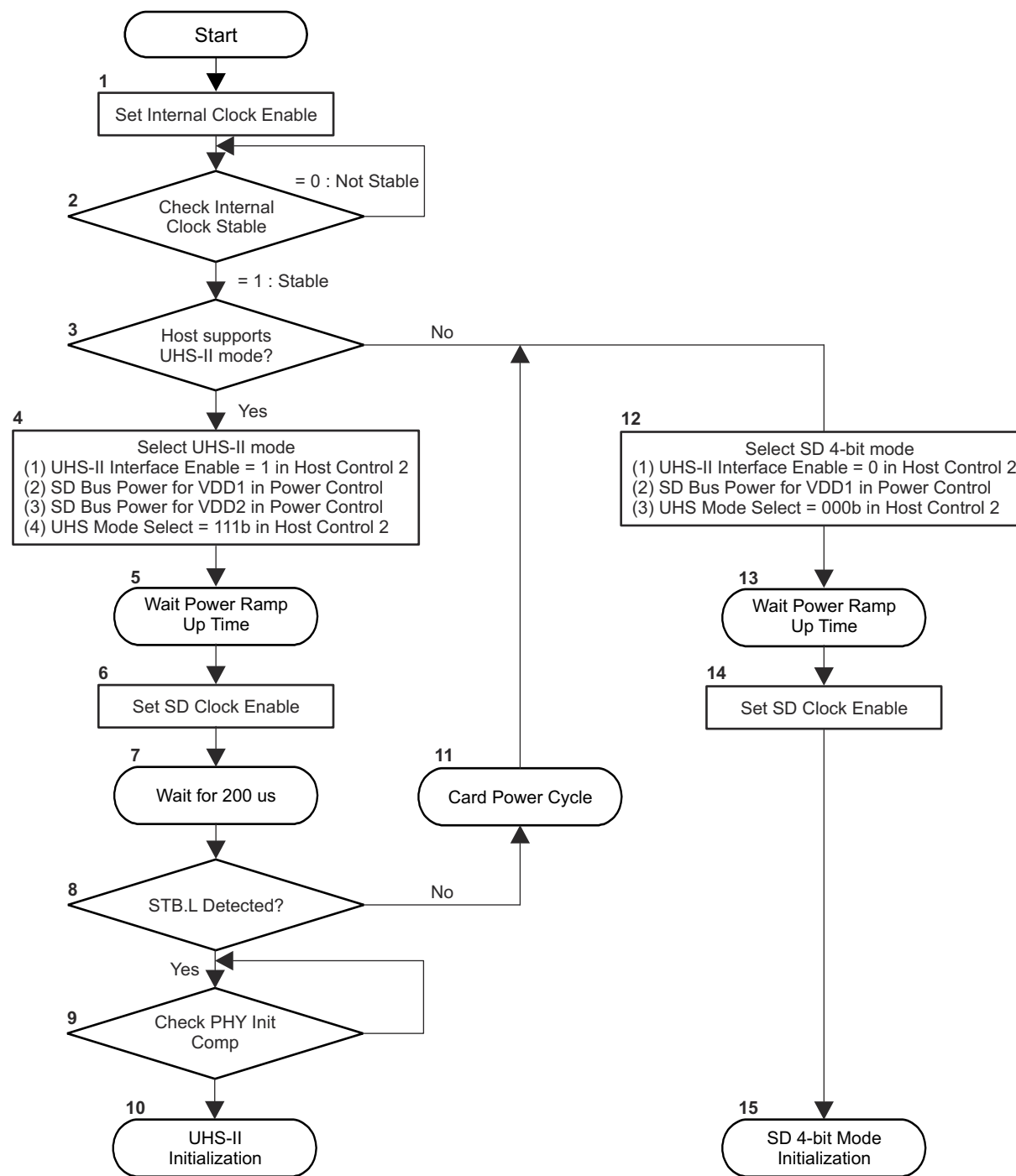
**Figure 12-336. Host Controller Setup Sequence**

**Table 12-387. Host Controller Setup Sequence**

Step	Description
1	Set parameters for all Host Controller version. Set the MMCSD0_POWER_CONTROL[3-1] SD_BUS_VOLTAGE and MMCSD0_TIMEOUT_CONTROL[3-0] COUNTER_VALUE bit fields.
2	Read the MMCSD0_HOST_CONTROLLER_VER[7-0] SPEC_VER_NUM bit field.
3	If Specification Version number (MMCSD0_HOST_CONTROLLER_VER[7-0] SPEC_VER_NUM) is less than version 3, go to step (4), if it is version 3 or later go to step (5).
4	Set the MMCSD0_CLOCK_CONTROL register using 8-bit Divided Clock mode.
5	Set Version 3 parameters. The MMCSD0_CLOCK_CONTROL register is sets in 10-bit Divided Clock Mode or Programmable Clock Mode. If Clock Multiplier in the MMCSD0_CAPABILITIES register is not zero, Programmable Clock Mode should be used. If Preset Value is used, set Preset Values of SD I/F Modes in the Preset Value register (MMCSD0_PRESET_VALUE0 - MMCSD0_PRESET_VALUE10) and set the MMCSD0_HOST_CONTROL2[15] PRESET_VALUE_ENA bit to 1.
6	If Specification Version number (MMCSD0_HOST_CONTROLLER_VER[7-0] SPEC_VER_NUM) is version 4, go to step (7), if it is less than version 4, exits.
7	Set Version 4 parameters. Set the MMCSD0_HOST_CONTROL2[12] HOST_VER40_ENA bit to 1. If the MMCSD0_CAPABILITIES[27] ADDR_64BIT_SUPPORT_V4 bit is set to 1, set the MMCSD0_HOST_CONTROL2[13] BIT64_ADDRESSING bit to 1. If UHS-II Support is set to 1 and 1.8 V VDD2 Support is set to 1 in the MMCSD0_CAPABILITIES register, set SD Bus Voltage Select for VDD2 to 1.8 V mode in the MMCSD0_POWER_CONTROL register, set preset value for UHS-II Mode to Preset Value register (MMCSD0_PRESET_VALUE0 - MMCSD0_PRESET_VALUE10) and set Timeout Counter Value for CMD_RES and Timeout Counter Value for Deadlock in the MMCSD0_UHS2_TIMER_CONTROL register based on Timeout Clock Frequency and Timeout Clock Unit in the MMCSD0_CAPABILITIES register. If Asynchronous Interrupt Support in the MMCSD0_CAPABILITIES register is set to 1, set Asynchronous Interrupt Enable to 1 in the MMCSD0_HOST_CONTROL2 register.

#### 12.3.6.5.2.1.2 Card Interface Detection Sequence

This procedure is invoked by the Card Insertion interrupt. [Figure 12-337](#) and [Table 12-388](#) show a card interface detection sequence.



mmcsd-055

**Figure 12-337. Card Interface Detection Sequence**
**Table 12-388. Card Interface Detection Sequence**

Step	Description
1	Set Internal Clock Enable to 1 in the MMCSDD0_CLOCK_CONTROL register.
2	Wait until Internal Clock Stable is set to 1 in the MMCSDD0_CLOCK_CONTROL register.
3	If Host Supports UHS-II, go to step (4) else go to step (12).
4	Try to initialize a card to UHS-II mode. Set UHS-II Interface Enable to 1 in the MMCSDD0_HOST_CONTROL2 register, set SD Bus Power for VDD1 and SD Bus Power for VDD2 to 1 in the MMCSDD0_POWER_CONTROL register, and set UHS Mode Select to 111b in the MMCSDD0_HOST_CONTROL2 register.

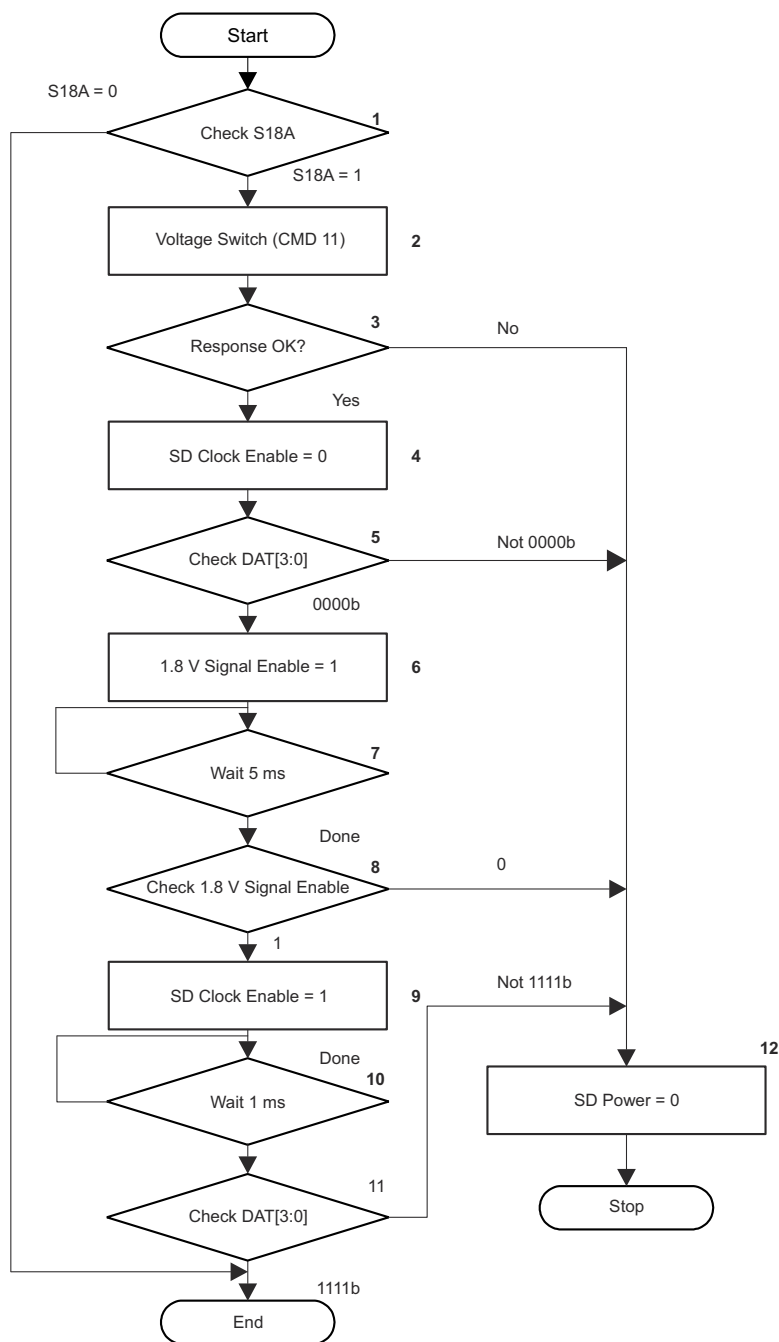


**Table 12-388. Card Interface Detection Sequence (continued)**

Step	Description
5	Wait power ramp up time. It is dependent on a Host System.
6	Set SD Clock Enable to 1 in the MMCSD0_CLOCK_CONTROL register.
7	Wait 200 $\mu$ s to check a card supports UHS-II mode.
8	Check STB.L Detection in the MMCSD0_PRESENTSTATE register. If UHS-II IF is detected, go to step (9). If UHS-II IF is not detected, go to step (11).
9	Wait until Lane Synchronization in the MMCSD0_PRESENTSTATE register is set to 1 (PHY Initialization is completed).
10	Perform UHS-II initialization.
11	Perform card power cycle.
12	Try to initialize a card to SD 4-bit mode. Set UHS-II Interface Enable to 0 in the MMCSD0_HOST_CONTROL2 register, set SD Bus Power for VDD1 to 1 in the MMCSD0_POWER_CONTROL register, and set UHS Mode Select to 000b in the MMCSD0_HOST_CONTROL2 register.
13	Wait power ramp up time. It is dependent on a Host System.
14	Set SD Clock Enable to 1 in the MMCSD0_CLOCK_CONTROL register.
15	Perform SD 4-bit mode initialization. Refer to <a href="#">Section 12.3.6.5.1.6</a> , <i>Card Initialization and Identification (for SD I/F)</i> .

### 12.3.6.5.2.2 Boot Operation

The boot operation sequence is shown in [Figure 12-338](#).



mmcsd-056

**Figure 12-338. Boot Operation Sequence**

#### 12.3.6.5.2.2.1 Normal Boot Operation: (For Legacy eMMC 5.0)

The Host driver writes the boot timeout value into MMCSD0\_BOOT\_TIMEOUT\_CONTROL register as per the eMMC4.3+ spec.

When the Host driver sets the "BOOT\_ENABLE" to 1 in MMCSD0\_BLOCK\_GAP\_CONTROL register and "DATA\_XFER\_DIR" bit to "1" in MMCSD0\_TRANSFER\_MODE register, the Host controller drives the CMD line to "0" for boot operation.

If the Host controller is configured to wait for boot acknowledgement from the eMMC4.3+ device, the controller receives the boot acknowledgement and asserts the "RCV\_BOOT\_ACK" interrupt to the driver

(MMCSD0\_NORMAL\_INTR\_STS[13] RCV\_BOOT\_ACK). If the Host controller doesn't receive the boot acknowledgement from the device within the timeout value, the Host controller will assert the data timeout error interrupt (MMCSD0\_ERROR\_INTR\_STS[4] DATA\_TIMEOUT).

After servicing the boot acknowledge interrupt or data timeout error interrupt, the driver programs the MMCSD0\_BOOT\_TIMEOUT\_CONTROL register with boot data timeout value.

The eMMC4.3+ device starts sending the boot data on the data line and Host controller sends the same to system whenever a block of data is received from device. The Host controller terminates the boot operation when the programmed number of blocks is transferred to the System. The Driver can write "BOOT\_ENABLE" as "0" in MMCSD0\_BLOCK\_GAP\_CONTROL register.

The boot operation can also be terminated in between the boot transfer (the Driver can set the "BOOT\_ENABLE" as "0" anytime to disable the boot operation). After the boot termination, the Host controller asserts soft reset for CMD line and DATA line internally once the driver clears the BOOT\_COMPLETE interrupt (MMCSD0\_NORMAL\_INTR\_STS[14] BOOT\_COMPLETE), to reset all the state machines to idle state.

There is no need to perform error recovery sequence in case data timeout interrupt occurs during boot operation (ignore sending abort command, soft reset and just clear the timeout interrupt and proceed with the boot flow).

If the device sends wrong acknowledgement to the Host, the Host controller will assert Data CRC Error interrupt (MMCSD0\_ERROR\_INTR\_STS[5] DATA\_CRC) on the Driver. In this case, the Host driver has to stop the boot mode operation by setting "BOOT\_ENABLE" as "0" in MMCSD0\_BLOCK\_GAP\_CONTROL register and write soft reset for CMD and data line.

If the device sends end bit as "0" in acknowledgement to the Host, the Host controller will assert Data End Bit Error interrupt (MMCSD0\_ERROR\_INTR\_STS[6] DATA\_ENDBIT) on the Driver.

The Host driver stops the boot mode operation by setting "BOOT\_ENABLE" as "0" in MMCSD0\_BLOCK\_GAP\_CONTROL register and write soft reset for CMD and data line.

**Note:** Enable the MMCSD0\_BLOCK\_GAP\_CONTROL[7] BOOT\_ACK\_ENA bit during boot operation. If failed, the controller will not wait for boot acknowledge from the card and send out data CRC error when the card sends boot acknowledgement first and followed by boot data.

#### 12.3.6.5.2.2.2 Alternate Boot Operation (For Legacy eMMC 5.0):

The Host driver writes the boot timeout value as per the eMMC4.3+ spec into MMCSD0\_BOOT\_TIMEOUT\_CONTROL register.

If the "ALT\_BOOT\_MODE" and "BOOT\_ENABLE" is set to 1 in MMCSD0\_BLOCK\_GAP\_CONTROL register and "DATA\_XFER\_DIR" bit is set to 1 in MMCSD0\_TRANSFER\_MODE register, the host controller drives the CMD0 (0xFFFFFFFF) on the CMD line.

The system shall wait for command complete interrupt before "RCV\_BOOT\_ACK" interrupt (MMCSD0\_NORMAL\_INTR\_STS[13] RCV\_BOOT\_ACK).

If the Host controller is configured to wait for boot acknowledgement from the eMMC4.3+ device, the controller receives the boot acknowledgement and asserts the "RCV\_BOOT\_ACK" interrupt to the driver.

The eMMC4.3+ device starts sending the boot data on the data line and Host controller sends the same to system whenever a block of data is received from device. The System needs to send CMD0 to inform the device about the boot operation complete. The system shall program MMCSD0\_COMMAND register with argument "00000000h" for the CMD0 and waits for command complete.

The Host controller terminates the boot operation when the programmed number of blocks are transferred to the System and the Driver shall send CMD0 to eMMC card and then program "BOOT\_ENABLE" as "0" in MMCSD0\_BLOCK\_GAP\_CONTROL register.

The boot operation can be terminated at any time (the Driver can send CMD0 and set the "BOOT\_ENABLE" as "0" anytime to disable the boot operation). After the boot termination, all state machines in the Host controller need to move to IDLE state. The Host controller asserts the soft reset for CMD and data line internally when the

driver clears the BOOT\_COMPLETE interrupt (MMCSD0\_NORMAL\_INTR\_STS[14] BOOT\_COMPLETE) and all the state machine goes to the idle state.

If the Host controller doesn't receive the acknowledgement from the device within timeout value the Host controller will assert the data timeout error interrupt (MMCSD0\_ERROR\_INTR\_STS[4] DATA\_TIMEOUT).

The driver programs the MMCSD0\_BOOT\_TIMEOUT\_CONTROL register with boot data timeout value.

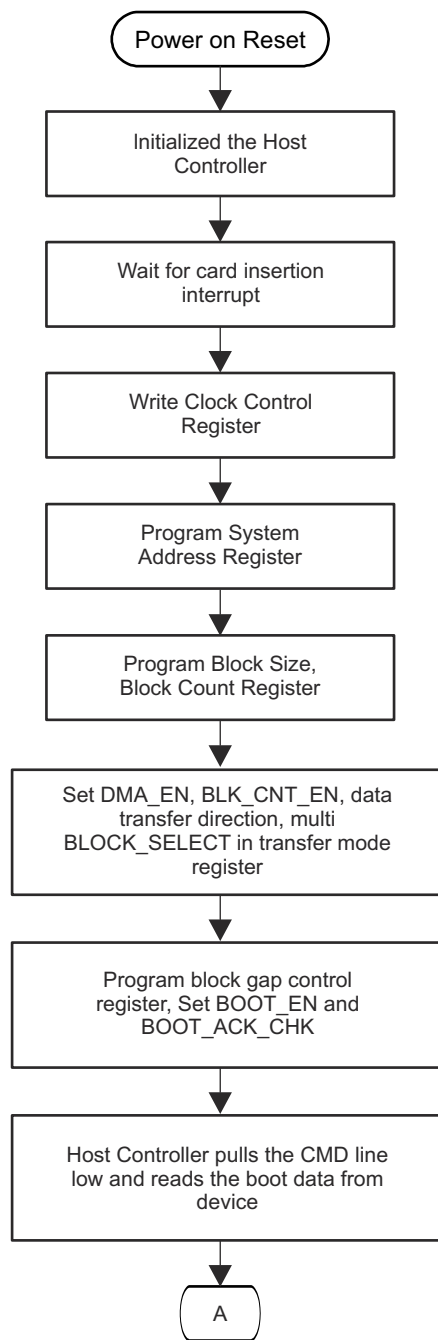
There is no need to perform error recovery sequence in case of data timeout interrupt as mentioned above. If the device sends wrong acknowledgement to the Host, then Host controller will assert Data CRC Error interrupt to the Driver (MMCSD0\_ERROR\_INTR\_STS[5] DATA\_CRC). The Host driver has to stop the boot mode operation by setting "ALT\_BOOT\_MODE" and "BOOT\_ENABLE" as "0" in MMCSD0\_BLOCK\_GAP\_CONTROL register and write soft reset for CMD and data line.

If the device sends end bit as "0" in acknowledgement to the Host, the Host controller asserts Data End Bit Error interrupt (MMCSD0\_ERROR\_INTR\_STS[6] DATA\_ENDBIT) on the Driver. The Host driver stops the boot mode operation by setting "ALT\_BOOT\_MODE" and "BOOT\_ENABLE" as "0" in MMCSD0\_BLOCK\_GAP\_CONTROL register and write soft reset for CMD and data line.

**Note:** Enable the MMCSD0\_BLOCK\_GAP\_CONTROL[7] BOOT\_ACK\_ENA bit during boot operation. If failed, the controller will not wait for boot acknowledge from the card and send out data CRC error when the card sends boot acknowledgement first and followed by boot data.

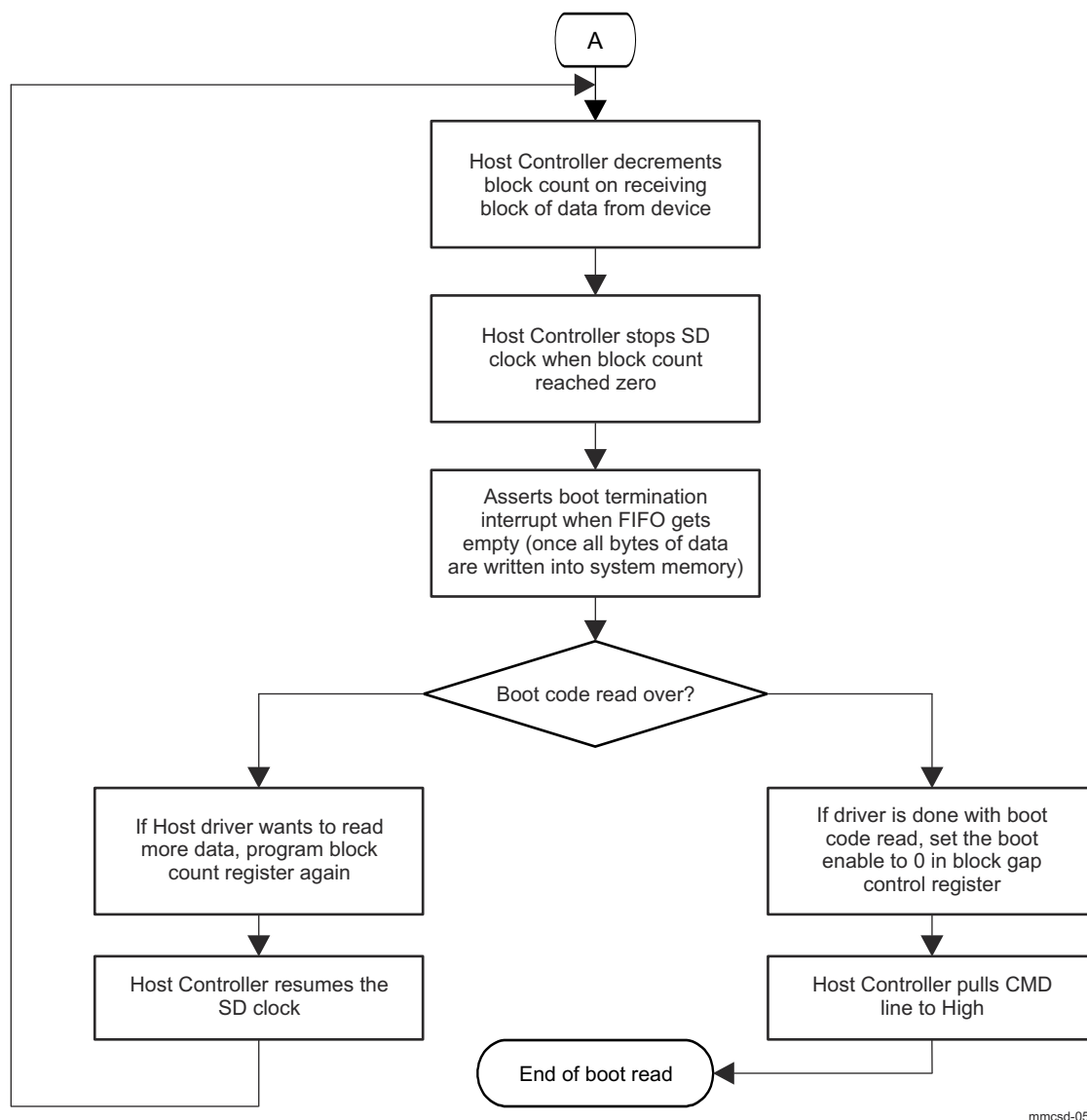
#### **12.3.6.5.2.2.3 Boot Code Chunk Read Operation (For Legacy eMMC 5.0):**

The following figure explains the boot code read done as chunks of data (the Driver can read the boot data as 2K, 4K, 1K and 8K etc. instead of 128K at a time)



mmcsd-057

**Figure 12-339. Boot Code Access Flow Diagram (1)**



**Figure 12-340. Boot Code Access Flow Diagram (2)**

#### 12.3.6.5.2.3 Retuning procedure (For Legacy Interface)

##### 12.3.6.5.2.3.1 Sampling Clock Tuning

The SD bus can be operating in high clock frequency mode and then the data window from the card on CMD and DAT[3:0] lines gets smaller. The position of the data window will vary depending on the card and host system implementation. Therefore, the Host Controller shall support a tuning circuit when SDR104 or SDR50.

The Host Controller shall support a tuning circuit when SDR104 or SDR50 (if Use Tuning for SDR50 is set to 1 in the MMCSDD0\_CAPABILITIES register) is supported by executing the tuning procedure and adjusting the sampling clock. Execute Tuning and Sampling Clock Select in the MMCSDD0\_HOST\_CONTROL2 register are used to control the tuning circuit.

##### 12.3.6.5.2.3.2 Tuning Modes

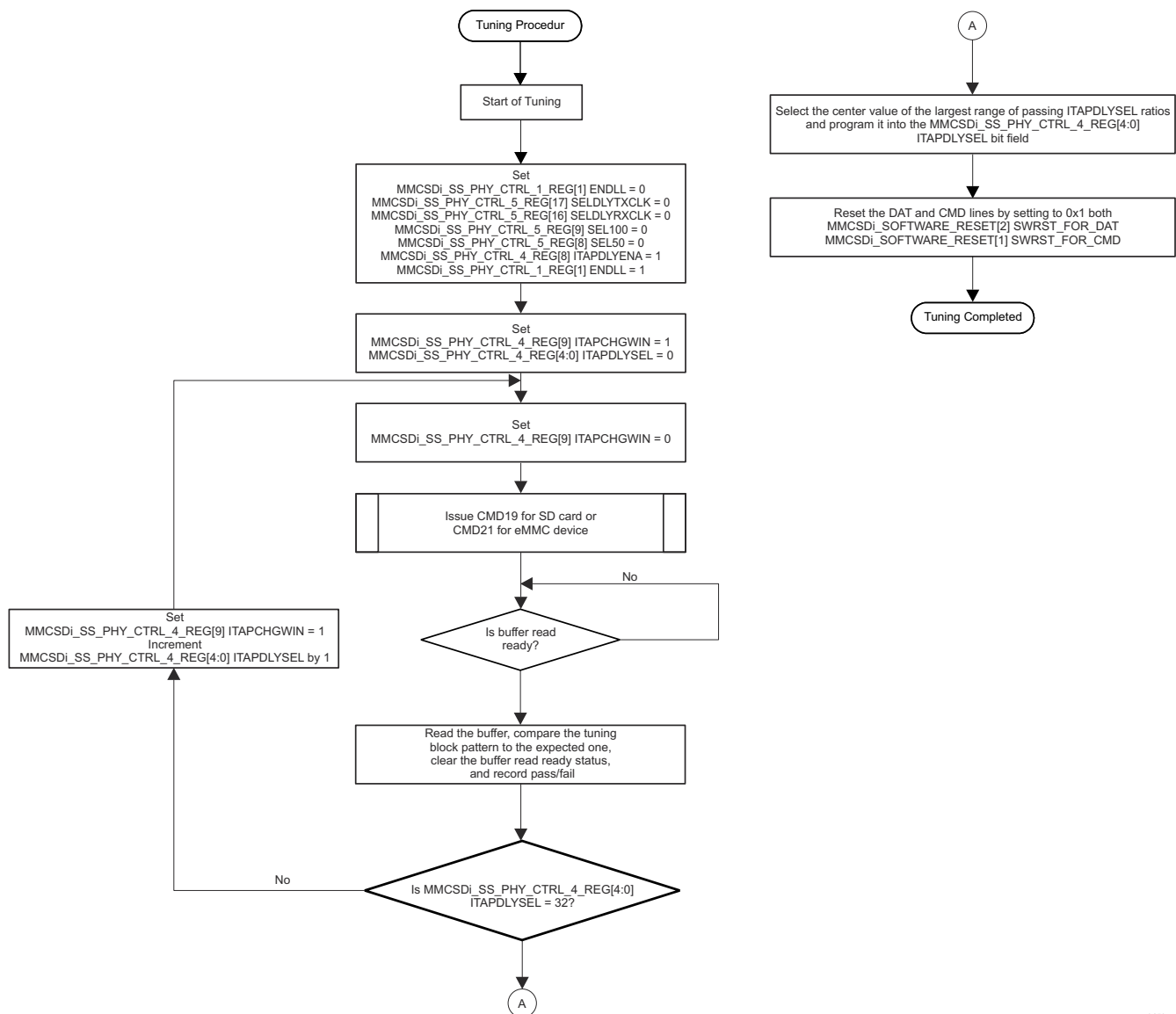
The re-tuning timing is specified by two methods: Re-Tuning Request generated by Host Controller and expiration of a re-tuning timer prepared by Host Driver.

(1) Re-Tuning Mode 1 The host controller does not have any internal logic to detect when the re-tuning needs to be performed. In this case, the Host Driver should maintain all re-tuning timings by using a Re-Tuning Timer. To enable inserting the re-tuning procedure during data transfers, the data length per read/write command shall be limited up to 4 MB.

(2) Re-Tuning Mode 2 The host controller has the capability to indicate the re-tuning timing by Re-Tuning Request during data transfers. Then the data length per read/write command shall be limited up to 4 MB. During non data transfer, re-tuning timing is determined by either Re-Tuning Request or Re-Tuning Timer. If Re-Tuning Request is used, Re-Tuning Timer should be disabled.

### 12.3.6.5.2.3.3 Re-Tuning Mode 2

The [Figure 12-341](#) defines sampling clock tuning procedure supported by Host Controller. In default, for lower frequency operation, fixed sampling clock is used to receive signals on CMD and DAT[3:0]. Before using SDR104, sampling clock tuning is required. Start of sampling clock tuning is requested by setting Execute Tuning to 1 and Sampling Clock Select to 0.



**Figure 12-341. MMCSD Clock Tuning**

mmcsd-059sm

Host driver issue CMD19 repeatedly until the host controller resets Execute Tuning to 0. Host Controller resets Execute Tuning to 0 when tuning is completed or tuning is not completed within 40 times. Host Driver can abort this loop by 40 times CMD19 issue or 150 ms time-out.

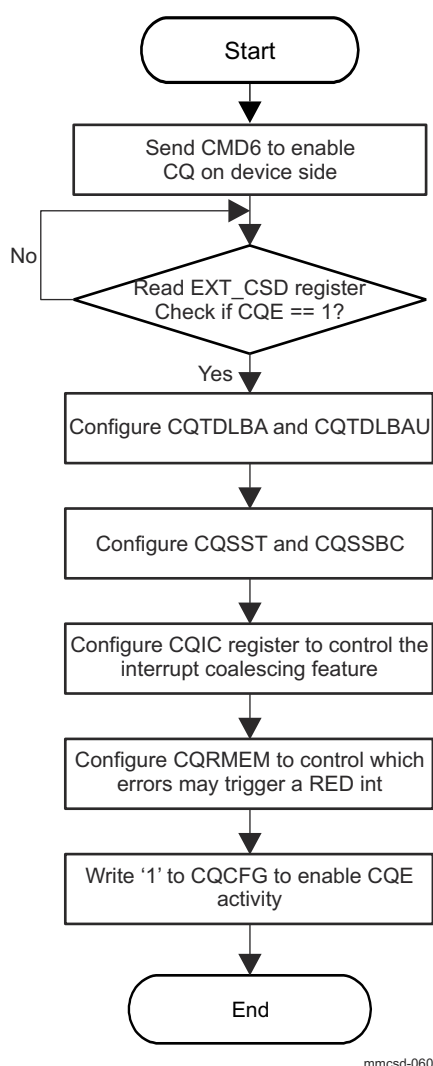
If tuning is completed successfully, Host Controller set Sampling Clock Select to 1 and this means the Host Controller start to use tuned sampling clock. If tuning is failed, Host Controller keeps Sampling Clock Select to 0. By writing Sampling Clock Select to 0, sampling clock is switched from tuned sampling clock to fixed sampling clock. Re-tuning time would be smaller than the first tuning time. CMD19 response errors are not indicated while tuning is performed.

The clock tuning tap delay values are selected using Variable sampling point detection. Fixed tap delay value is used for fixed tuning clock method.

#### 12.3.6.5.2.4 Command Queuing Driver Flow Sequence

##### 12.3.6.5.2.4.1 Command Queuing Initialization Sequence

Figure 12-342 and Table 12-389 show a Command Queuing initialization sequence.



**Figure 12-342. Command Queuing Initialization Sequence**

**Table 12-389. Command Queuing Initialization Sequence**

Step	Description
1	Initialize and enable Command Queueing in the device.

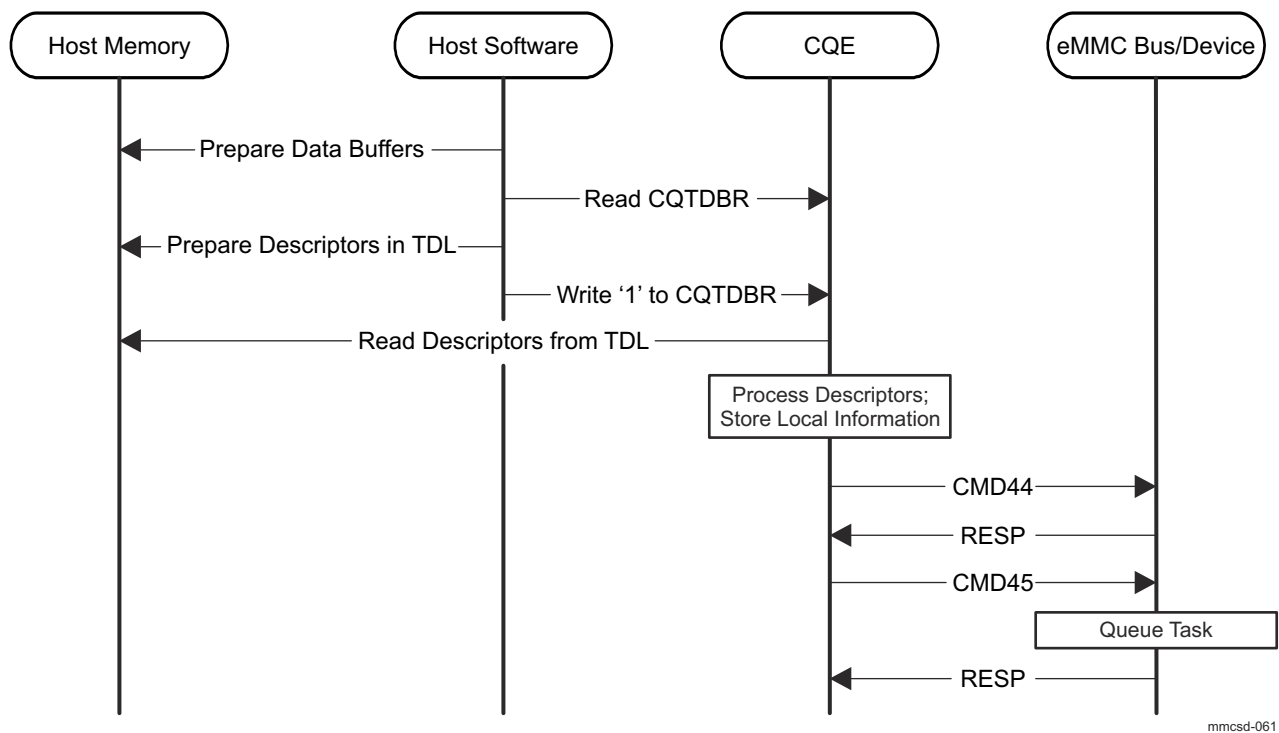


**Table 12-389. Command Queuing Initialization Sequence (continued)**

Step	Description
2	Configure Task Descriptor size in MMCSD0_CQ_CONFIG register.
3	Configure MMCSD0_CQ_TDL_BASE_ADDR and MMCSD0_CQ_TDL_BASE_ADDR_UPBITS registers to point to the memory location allocated to the TDL in host memory.
4	Configure CQSST and CQSSBC to control when SEND_QUEUE_STATUS commands are sent to the device by CQE.
5	Configure MMCSD0_CQ_INTR_COALESCING register to control the interrupt coalescing feature: enable/disable, set interrupt count and timer protection.
6	Configure MMCSD0_CQ_RESP_ERR_MASK register to control which errors may trigger a RED interrupt (if different from reset values).
7	Write '1' to MMCSD0_CQ_CONFIG register to enable CQE activity.

#### 12.3.6.5.2.4.2 Task Issuance Sequence

Figure 12-343 and Table 12-390 show a task issuance sequence.



mmcsd-061

**Figure 12-343. Task Issuance Sequence**
**Table 12-390. Task Issuance Sequence**

Step	Description
1	Find an empty transfer request slot by reading the MMCSD0_CQ_TASK_DOOR_BELL register. An empty transfer request slot has its respective bit cleared to '0' in the MMCSD0_CQ_TASK_DOOR_BELL register.

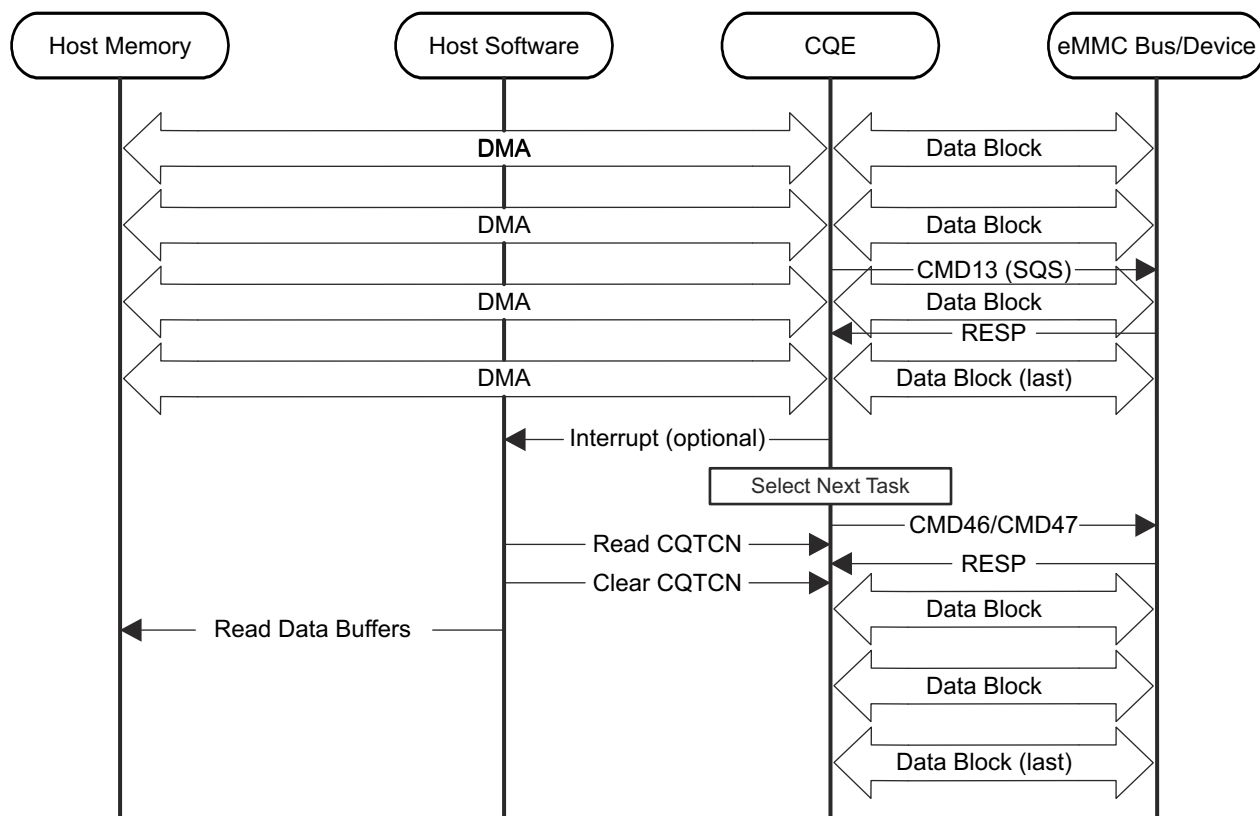
**Table 12-390. Task Issuance Sequence (continued)**

Step	Description
2	Build a Task Descriptor at the 1st entry of the empty slot. Task Descriptor field values: <ul style="list-style-type: none"> <li>1. Valid = 1, to indicate the descriptor is effective.</li> <li>2. End = 1, as required for Task Descriptors.</li> <li>3. Int = 1, if an interrupt on completion is required. Otherwise, Int = 0.</li> <li>4. Act = b101, to indicate a Task Descriptor .</li> <li>5. Data Direction = 1 for read commands, and 0 for write commands.</li> <li>6. Priority = 1 for high priority, and 0 for simple requests.</li> <li>7. QBR = 1 if Queue Barrier functionality is required. 0 otherwise.</li> <li>8. Forced Programming, Context ID, Tag Request, Reliable Write, Block Count, and Block Address programmed according to application requirements.</li> </ul>
3	Build a Transfer Descriptor at the 2nd entry of the empty slot. Transfer Descriptor field values: <ul style="list-style-type: none"> <li>1. Valid = 1, to indicate the descriptor is effective.</li> <li>2. End = 1, if a TRAN descriptor is used, to indicate the task only has one data buffer. End = 0 if LINK descriptor is used.</li> <li>3. Int = 0. Ignore in Command queueing.</li> <li>4. Act = b100, for TRAN descriptors, pointing directly to the task's single data buffer. Act = 401 b110, for LINK descriptors, point to a scatter/gather list (indirect).</li> <li>5. Address and Length programmed according to the data buffer supplied by the application.</li> </ul>
4	If more than one transfer is requested, repeat step 1-3 for all needed transfers.
5	Set MMCSD0_CQ_TASK_DOOR_BELL register to indicate to the CQE that one or more transfer requests are ready to be sent to the attached device. Host software shall only write a '1' to the bit position that corresponds to new tasks; all other bit positions within MMCSD0_CQ_TASK_DOOR_BELL register should be written with a '0', which indicates no change to their current values.

#### 12.3.6.5.2.4.3 Task Execution and Completion Sequence

The CQE is responsible for task execution, communication with the device and moving the data to the buffers in the host memory.

Figure 12-344 and Table 12-391 show a task execution and completion sequence.



mmcsd-062

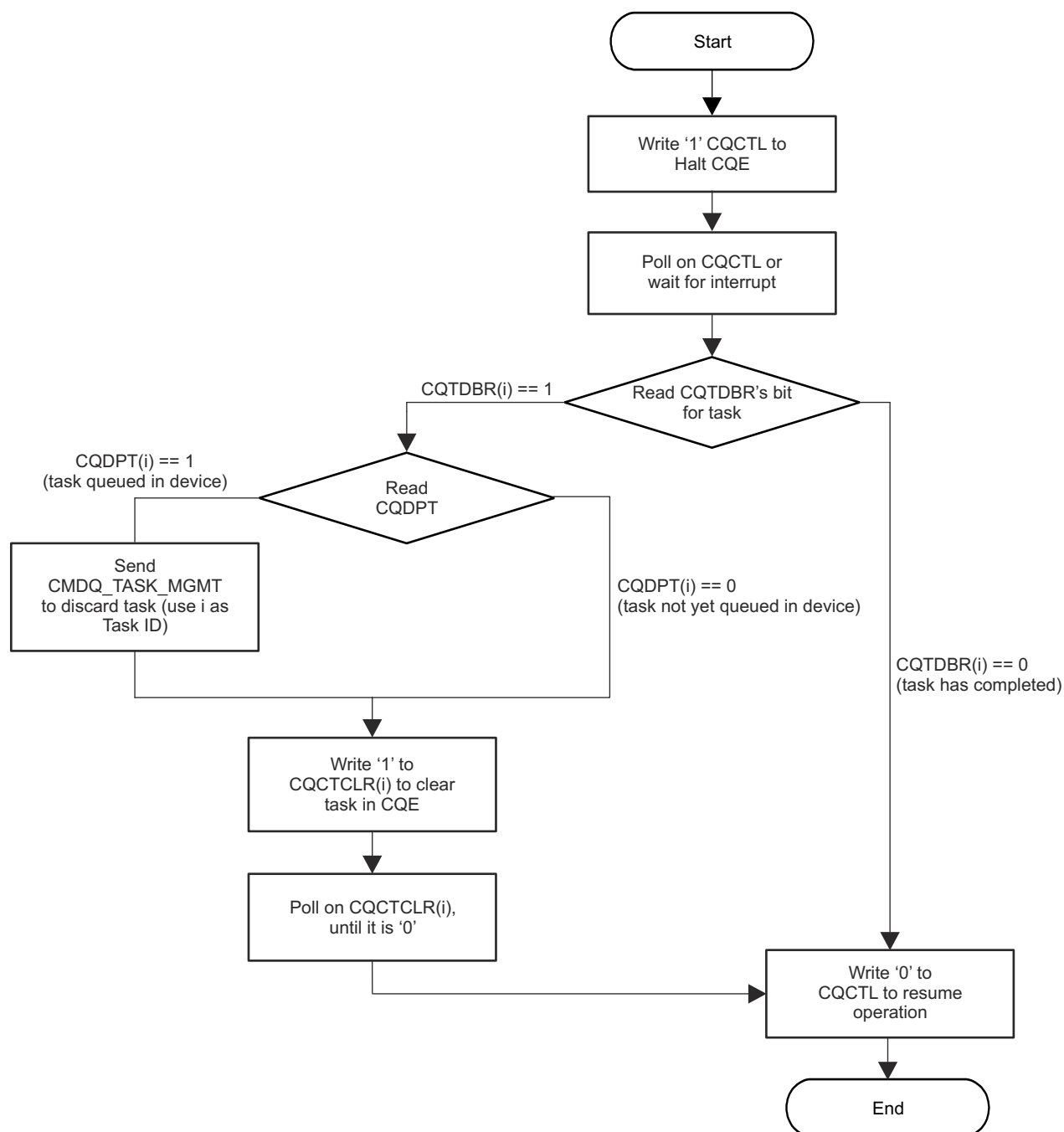
**Figure 12-344. Task Execution and Completion Sequence**

**Table 12-391. Task Execution and Completion Sequence**

Step	Description
1	Once the tasks are queued on the device side CQE sends CMD13 to determine the queue status. The queue status lets the CQE know which tasks are ready for execution.
2	The response to CMD13 may indicate no task is ready in which case the CQE is required to send CMD13 again at a later time as controlled CQSST register.
3	If a task is ready for execution then the host sends EXECUTE_READ_TASK (CMD46) or EXECUTE_WRITE_TASK (CMD47) to the device with the task id ordering it to execute a task.
4	When the task execution is completed, an interrupt may be generated, if requested, or as determined by Interrupt Coalescing mechanism.
5	The host software reads MMCSD0_CQ_TASK_COMP_NOTIF register to determine which task(s) has(have) been completed. Each bit set in MMCSD0_CQ_TASK_COMP_NOTIF register represents by index which task has completed but hasn't yet been served by software.
6	For every task completed clear the appropriate MMCSD0_CQ_TASK_COMP_NOTIF register bit.
7	Repeat steps 1-6 for the pending tasks.

#### 12.3.6.5.2.4.4 Task Discard and Clear Sequence

Figure 12-345 and Table 12-392 show a task discard and clear sequence.



mmscd-063

**Figure 12-345. Task Discard and Clear Sequence**
**Table 12-392. Task Discard and Clear Sequence**

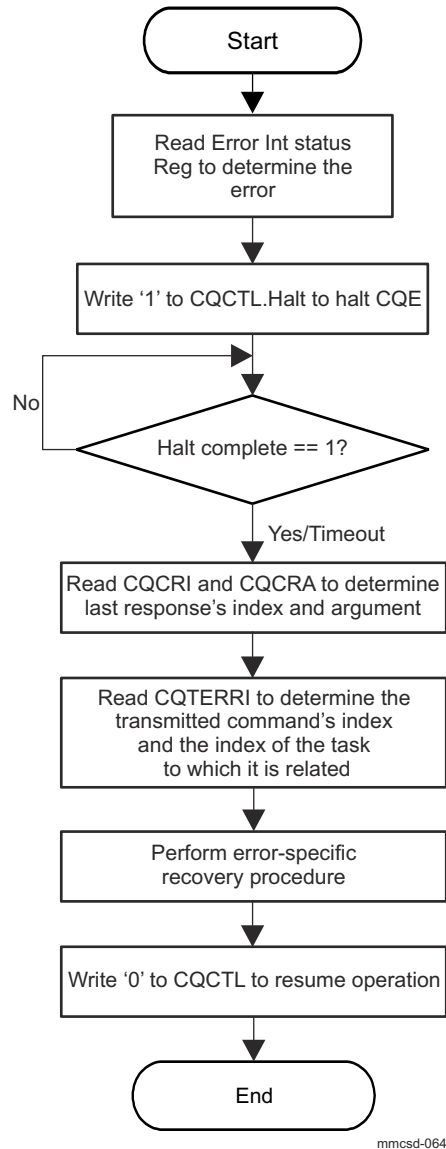
Step	Description
1	While CQE is enabled write '1' to MMCS0_CQ_CONTROL[0] HALT_BIT bit to halt CQE.
2	Poll on MMCS0_CQ_INTR_STS[0] HALT_COMPLETE bit.
3	Read MMCS0_CQ_TASK_DOOR_BELL register to determine if the task to be discarded is set to 1.
4	Read MMCS0_CQ_DEV_PENDING_TASKS register to check if the task is queued in the device.
5	Send CMDQ_TASK_MGMT(CMD48) to discard task using the task id as the argument.
6	Write '1' to CQCTCLR[i] to clear task in CQE.

**Table 12-392. Task Discard and Clear Sequence (continued)**

Step	Description
7	Poll on CQCTCLR[i] until it is '0'.
8	Write '0' to MMCSD0_CQ_CONTROL register to resume CQE.

#### 12.3.6.5.2.4.5 Error Detect and Recovery when CQ is enabled

Figure 12-346 and Table 12-393 show an error detect and recovery sequence.


**Figure 12-346. Error Detect and Recovery Sequence**
**Table 12-393. Error Detect and Recovery Sequence**

Step	Description
1	Read eMMC host controller MMCSD0_ERROR_INTR_STS register and determine error is related to CQE.
2	Write '1' to MMCSD0_CQ_CONTROL[0] HALT_BIT bit to halt CQE.
3	Wait for MMCSD0_CQ_CONTROL[0] HALT_BIT bit to read '1'. In some error cases, this may not happen, so software should proceed to the next step after a sufficient time-out.

**Table 12-393. Error Detect and Recovery Sequence (continued)**

Step	Description
4	Read MMCSD0_CQ_CMD_RESP_INDEX and MMCSD0_CQ_CMD_RESP_ARG registers to determine last response's index and argument.
5	Read MMCSD0_CQ_TASK_ERR_INFO register to determine the transmitted command's index and the index of the task to which it is related.
6	Perform error-specific recovery procedure.
7	Write '0' to MMCSD0_CQ_CONTROL register to resume operation.

### 12.3.7 Universal Flash Storage (UFS) Interface

This section describes the Universal Flash Storage (UFS) interface in the device.

#### 12.3.7.1 UFS Overview

The Universal Flash Storage (UFS) interface is a standard-based serial interface engine.

There is one UFS module inside the device - UFS0. The UFS module includes one UFS 2.1 host controller (HC) with an integrated M-PHY.

The UFS module complies with the standards as listed in [Table 12-394](#).

**Table 12-394. UFS Standards**

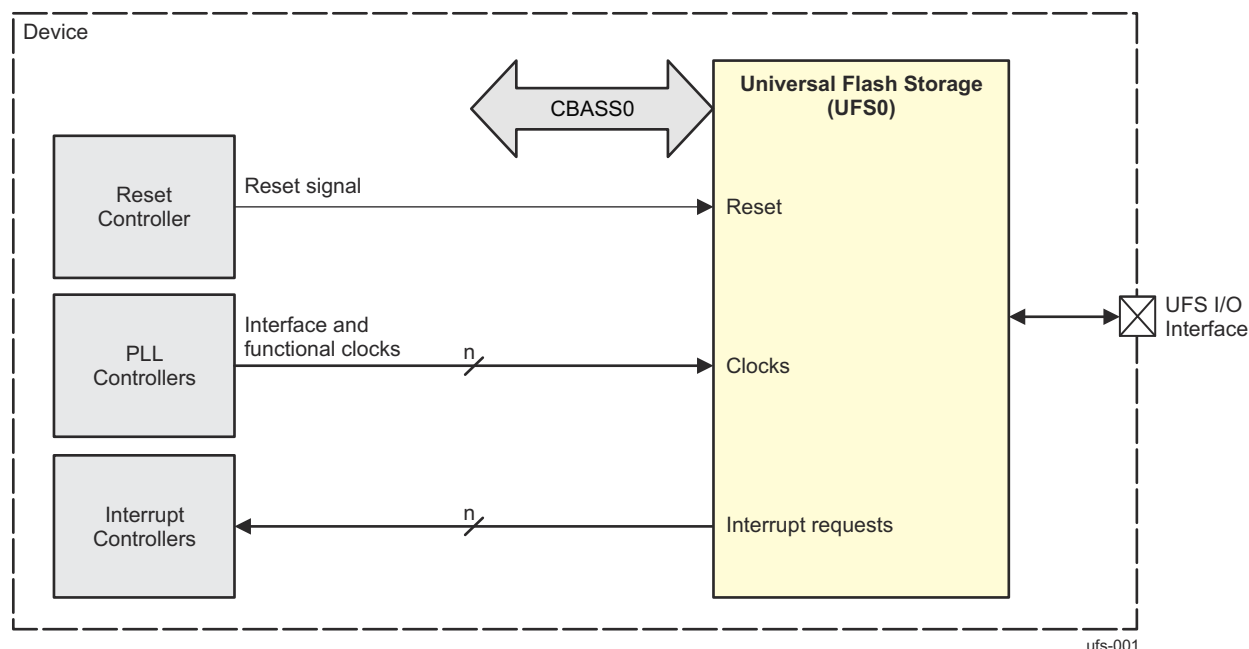
Document	Version	Description
JESD220-1A	v1.1	Universal Flash Storage (UFS) Unified Memory Extension
JESD220-2	v1.0	Universal Flash Storage (UFS) Card Extension
JESD220C	v2.1, March 2016	Universal Flash Storage (UFS)
JESD223-1B	v1.1A	Universal Flash Storage Host Controller Interface (UFSHCI) Unified Memory Extension
JESD223C	v2.1, March 2016	Universal Flash Storage Host Controller Interface (UFSHCI)
JESD224	March 2013	Universal Flash Storage (UFS) Test
	November, 2001	Federal Information Processing Standards (FIPS) 197 Advanced Encryption Standard (AES)
	v3.1, 2014	MIPI® Alliance Specification for M-PHY
	v1.60, 2013	MIPI Alliance Specification for Unified Protocol (UniProSM)
	Revision 24, August 2010	Small Computer System Interface (SCSI) Block Commands - 3
	Revision 27, October 2010	SCSI Primary Commands - 4

[Table 12-395](#) shows the UFS allocation across device domains.

**Table 12-395. UFS Allocation Across Device Domains**

Instance	Domain		
	WKUP	MCU	MAIN
UFS0	-	-	✓

[Figure 12-347](#) shows the UFS module overview.



**Figure 12-347. UFS Module Overview**

#### 12.3.7.1.1 UFS Features

##### Key Features:

- Supports Universal Flash Storage Host (UFS2.1, JESD220C)
- Supports Universal Flash Storage Host Controller Interface (UFSHCI, JESD223C)
- Supports UFS interconnect - MIPI UniPro and MIPI M-PHY
- Supports Gear 1: 1.46 Gbps, 2-lanes
- Supports Gear 2: 2.91 Gbps, 2-lanes
- Supports Gear 3: 5.83 Gbps, 2-lanes
- Supports all UFS mandatory SCSI commands required by JEDEC specification
- Supports power-on reset (POR), hardware (HW) reset, EndPoint reset, Logical Unit reset, and Warm reset

##### Master Bus Interfaces:

- Supports one 64-bit master interface (CBA4.0 VBUSM) for data transfers
- Supports 48-bit address width on the master interface
- Include embedded DMA controller within the host controller
- Supports little-endian mode only

##### Configuration Bus Interfaces:

- Supports one 32-bit slave interface (CBA4.0 VBUSP) for configuration access
- Supports linear incrementing addressing mode
- Supports aligned accesses only

Supports ECC on internal RAM's

##### M-PHY Features:

- Supports reference clock of 19.2 MHz and 26 MHz



### 12.3.7.2 UFS Environment

This section describes the UFS external connections (environment).

Figure 12-348 shows the UFS I/O interface signals.

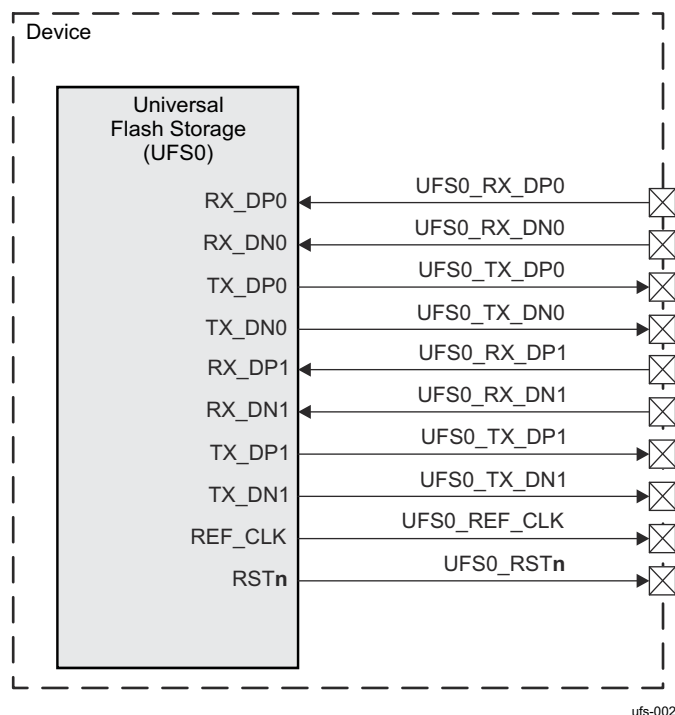


Figure 12-348. UFS I/O Interface Signals

Table 12-396 describes the UFS I/O signals.

Table 12-396. UFS I/O Signals

Module Pin	Device Level Signal	I/O <sup>(2)</sup>	Description	Module Pin Reset Value
<b>UFS0</b>				
RX_DP0	UFS0_RX_DP0	I	UFS Lane0 RX Data Positive	0x0
RX_DN0	UFS0_RX_DN0	I	UFS Lane0 RX Data Negative	0x0
TX_DP0	UFS0_TX_DP0	O	UFS Lane0 TX Data Positive	0x0
TX_DN0	UFS0_TX_DN0	O	UFS Lane0 TX Data Negative	0x0
RX_DP1	UFS0_RX_DP1	I	UFS Lane1 RX Data Positive	0x0
RX_DN1	UFS0_RX_DN1	I	UFS Lane1 RX Data Negative	0x0
TX_DP1	UFS0_TX_DP1	O	UFS Lane1 TX Data Positive	0x0
TX_DN1	UFS0_TX_DN1	O	UFS Lane1 TX Data Negative	0x0
REF_CLK	UFS0_REF_CLK	O	UFS Reference Clock Signal to Slave	0x0
RSTn	UFS0_RSTn <sup>(1)</sup>	O	UFS Reset Signal to Slave	0x0

(1) n - Active Low

(2) I = Input; O = Output

#### Note

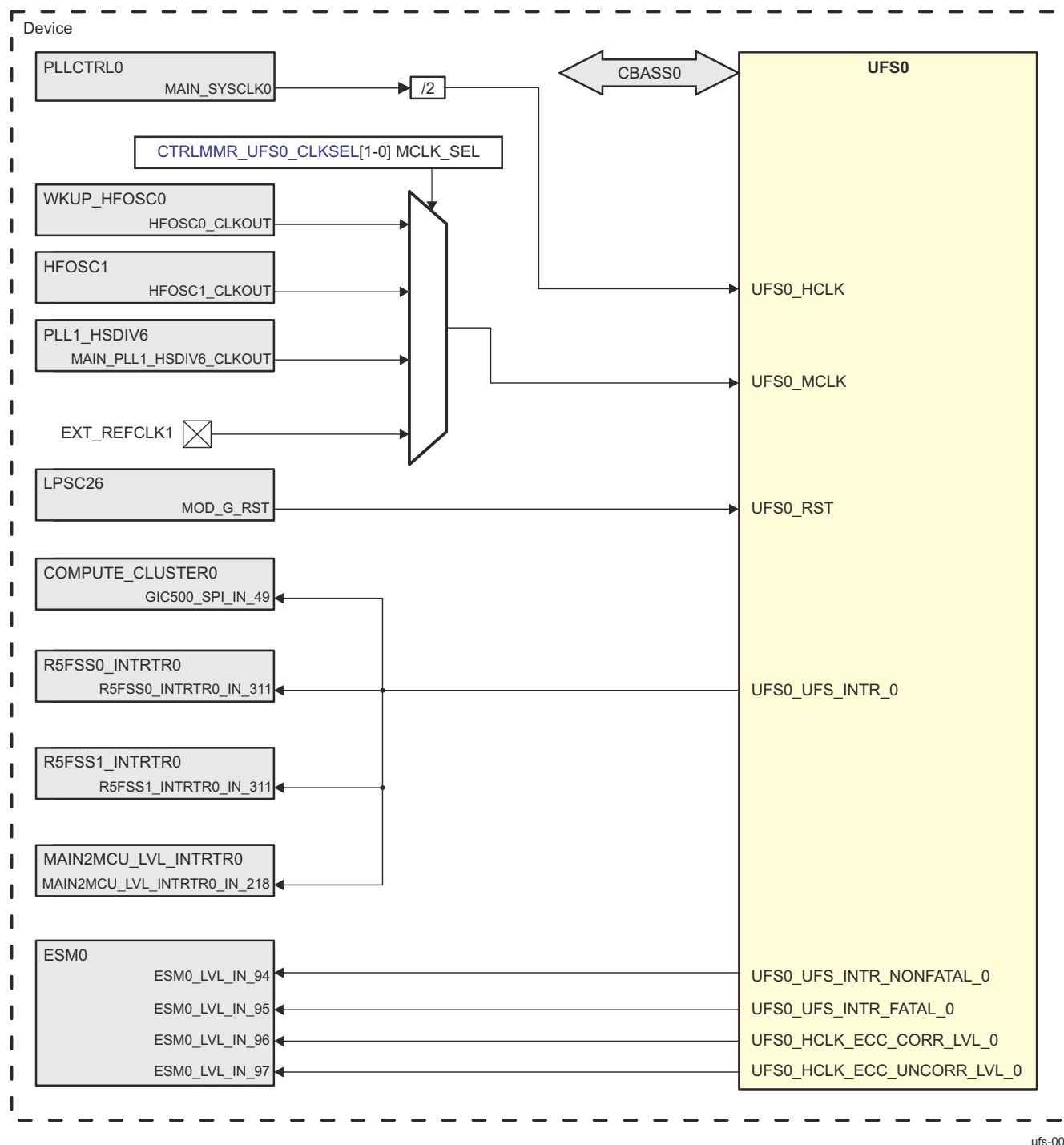
For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables Pin Attributes and Pin Multiplexing in the device-specific Datasheet.

### 12.3.7.3 UFS Integration

This section describes the UFS integration in the device, including information about clocks, resets, and hardware requests.

#### 12.3.7.3.1 UFS Integration in MAIN Domain

There is one UFS module integrated in the device MAIN domain - UFS0. Figure 12-349 shows the integration of UFS0.



ufs-003

**Figure 12-349. UFS0 Integration**

Table 12-397 through Table 12-399 summarize the integration of UFS0 in the device MAIN domain.

**Table 12-397. UFS0 Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
UFS0	PSC0	PD0	LPSC26	CBASS0

**Table 12-398. UFS0 Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
UFS0	UFS0_HCLK	MAIN_SYSCLK0/2	PLLCTRL0	UFS0 Interface Clock
	UFS0_MCLK	HFOSC0_CLKOUT	WKUP_HFOSC0	UFS0 Reference Clock for the M-PHY (for more information about clock multiplexing, see CTRLMMR_UFS0_CLKSEL[1-0] MCLK_SEL bit field in
		HFOSC1_CLKOUT	HFOSC1	Control Module (CTRL_MMR).
		MAIN_PLL1_HSDIV6_CLKOUT	PLL1_HSDIV6	Default: CTRLMMR_UFS0_CLKSEL[1-0] MCLK_SEL = 0h, HFOSC0_CLKOUT is selected)
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
UFS0	UFS0_RST	MOD_G_RST	LPSC26	UFS0 Module Reset

**Table 12-399. UFS0 Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
UFS0	UFS0_UFS_INTR_0	GIC500_SPI_IN_49	COMPUTE_CLUSTER0	UFS0 Interrupt Request	Level
		R5FSS0_INTRTR0_IN_311	R5FSS0_INTRTR0		Level
		R5FSS1_INTRTR0_IN_311	R5FSS1_INTRTR0		Level
		MAIN2MCU_LVL_INTRTR0_IN_218	MAIN2MCU_LVL_INTRTR0		Level
	UFS0_UFS_INTR_NONFATAL_0	ESM0_LVL_IN_94	ESM0	UFS0 Non-Fatal Interrupt Request	Level
	UFS0_UFS_INTR_FATAL_0	ESM0_LVL_IN_95	ESM0	UFS0 Fatal Interrupt Request	Level
	UFS0_HCLK_ECC_CORR_LVL_0	ESM0_LVL_IN_96	ESM0	UFS0 ECC Correctable Error Interrupt Request	Level
	UFS0_HCLK_ECC_UNCORR_LVL_0	ESM0_LVL_IN_97	ESM0	UFS0 ECC Uncorrectable Error Interrupt Request	Level

### 12.3.7.4 UFS Functional Description

#### 12.3.7.4.1 UFS Block Diagrams

The UFS subsystem includes one UFS 2.1 host controller with an integrated M-PHY. The UFS host controller is a standard-based serial interface engine.

The UFS host controller is compliant with the *Universal Flash Storage (UFS) Specification* and *UFS Host Controller Interface (UFSHCI) Specification* (see [Table 12-394](#)).

Figure 12-350 shows the UFS subsystem block diagram.

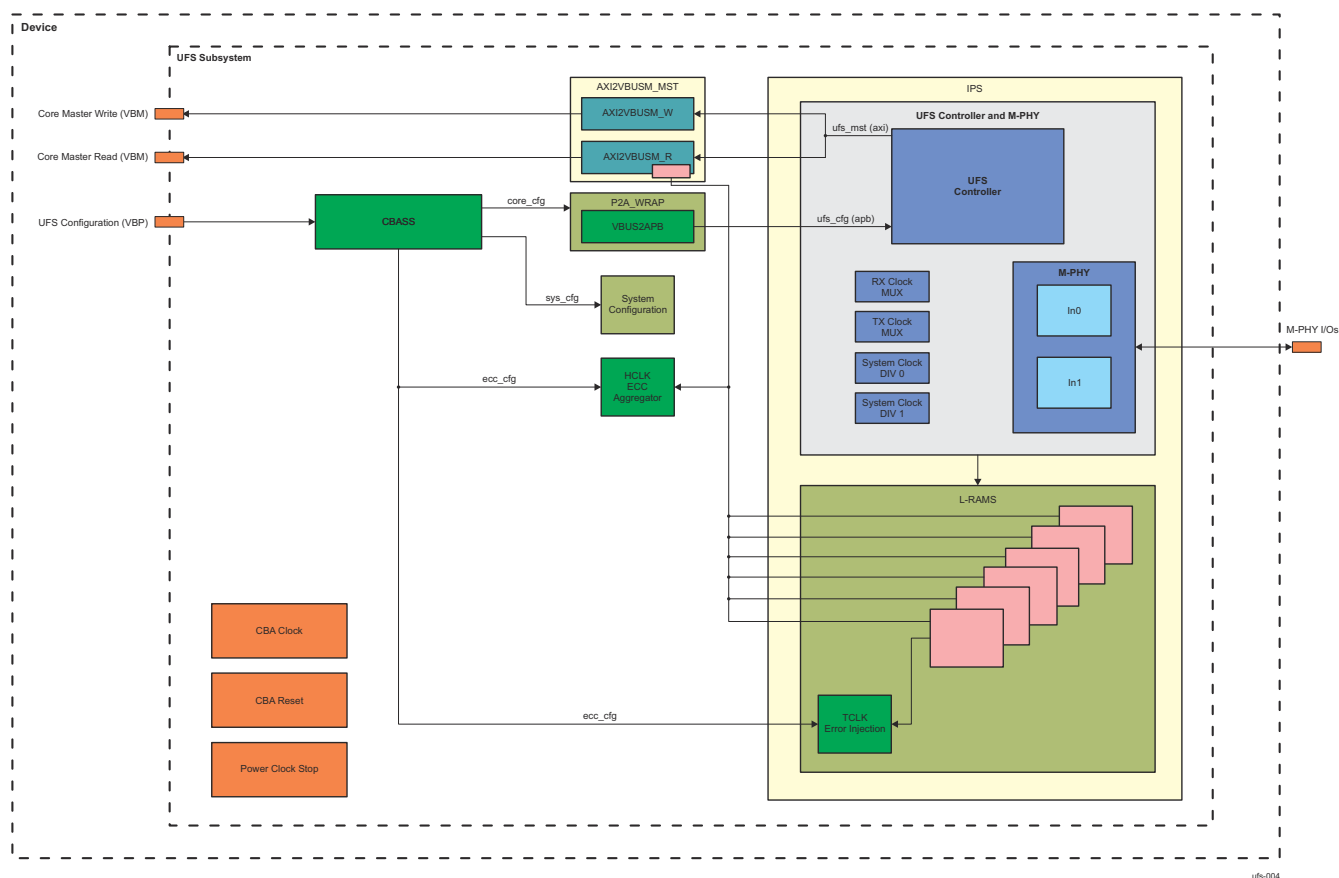
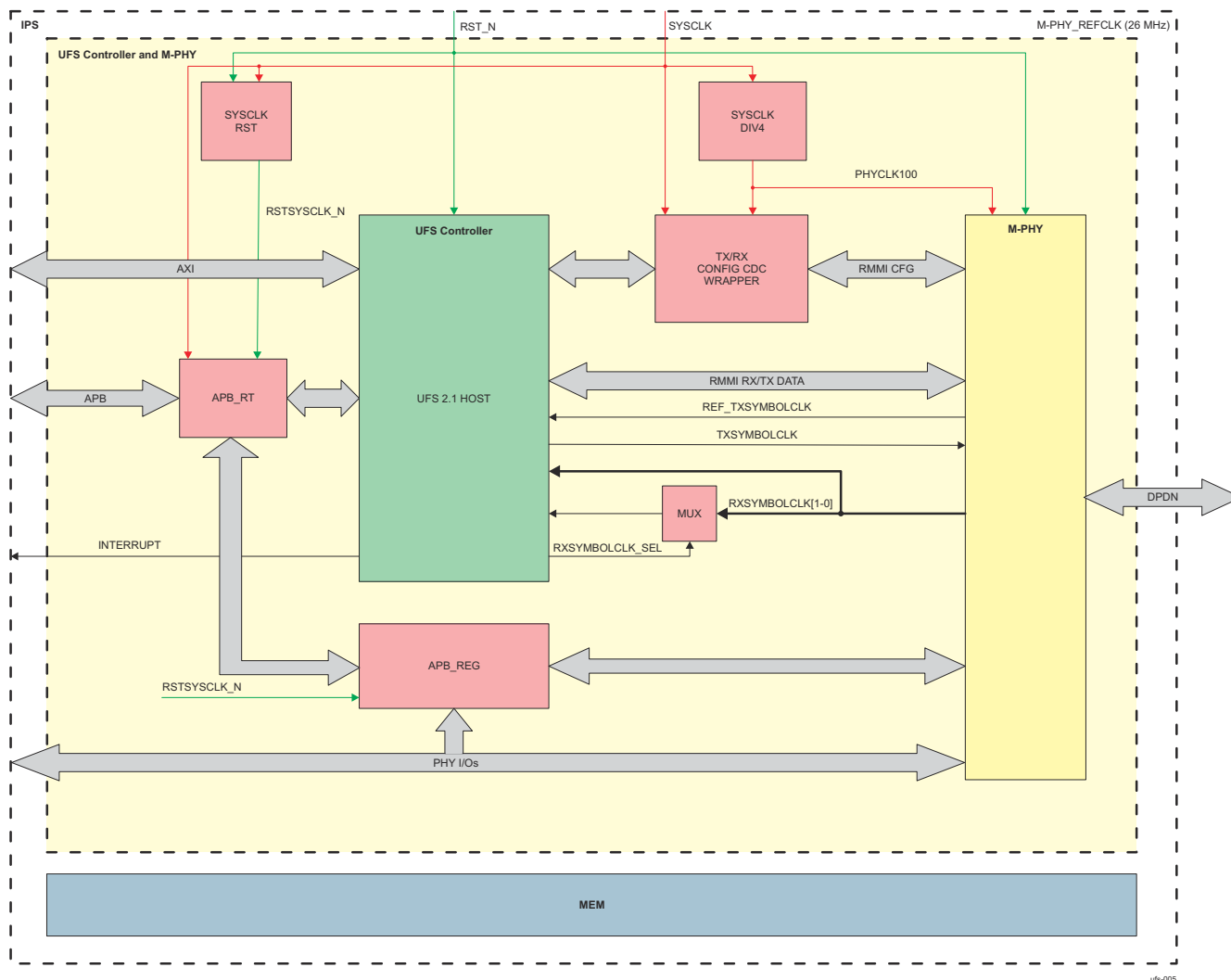


Figure 12-350. UFS Subsystem Block Diagram

Figure 12-351 presents an example of UFS Integrated Protocol Stack (IPS) subsystem.

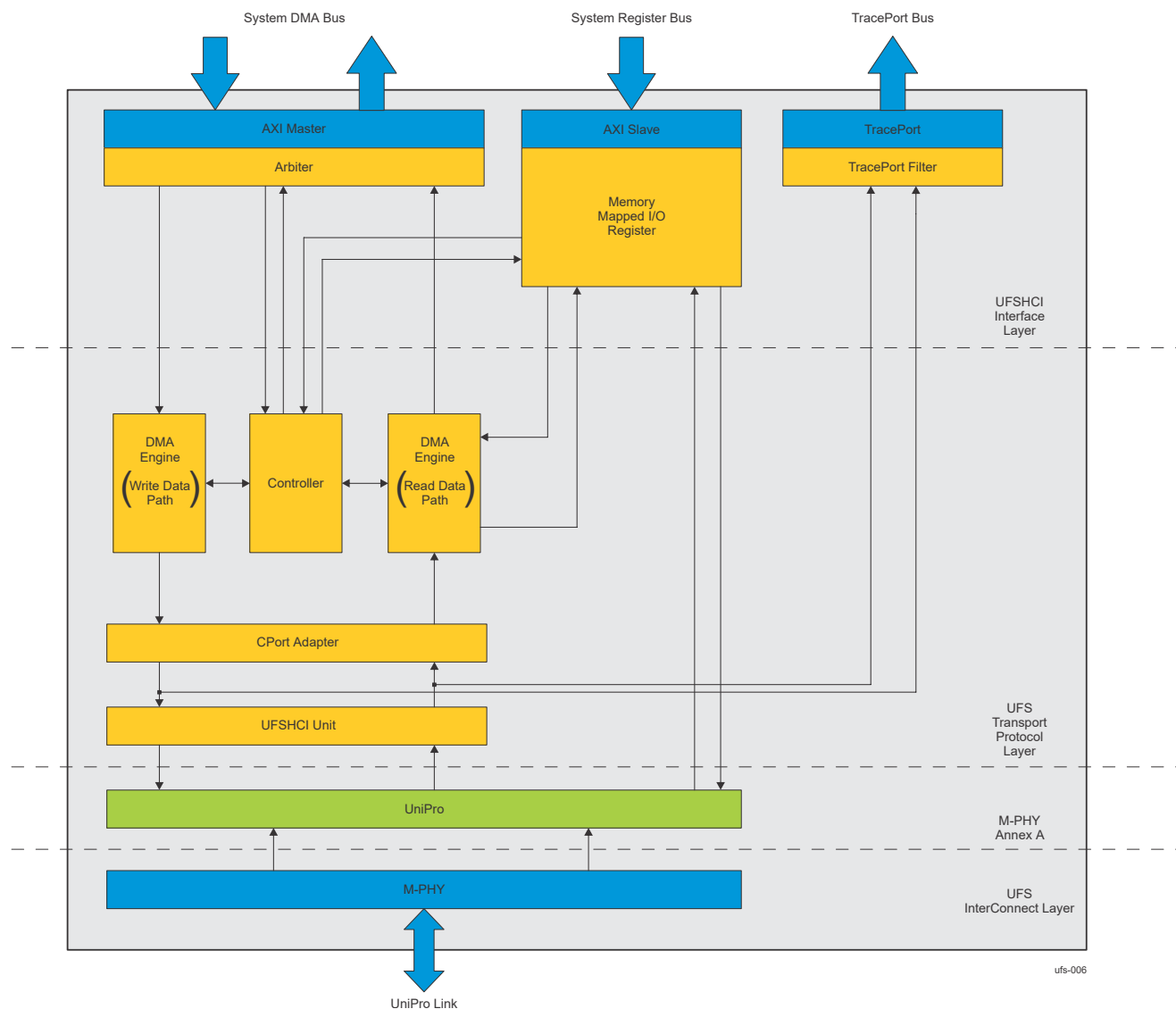
The UFS host controller is provided with additional hierarchy to support M-PHY integration.

These additional levels of hierarchy make up the IPS. Both the UFS host controller and the M-PHY share the same control interface (APB) and in further description is assumed they are mapped into single area in APB subsystem.



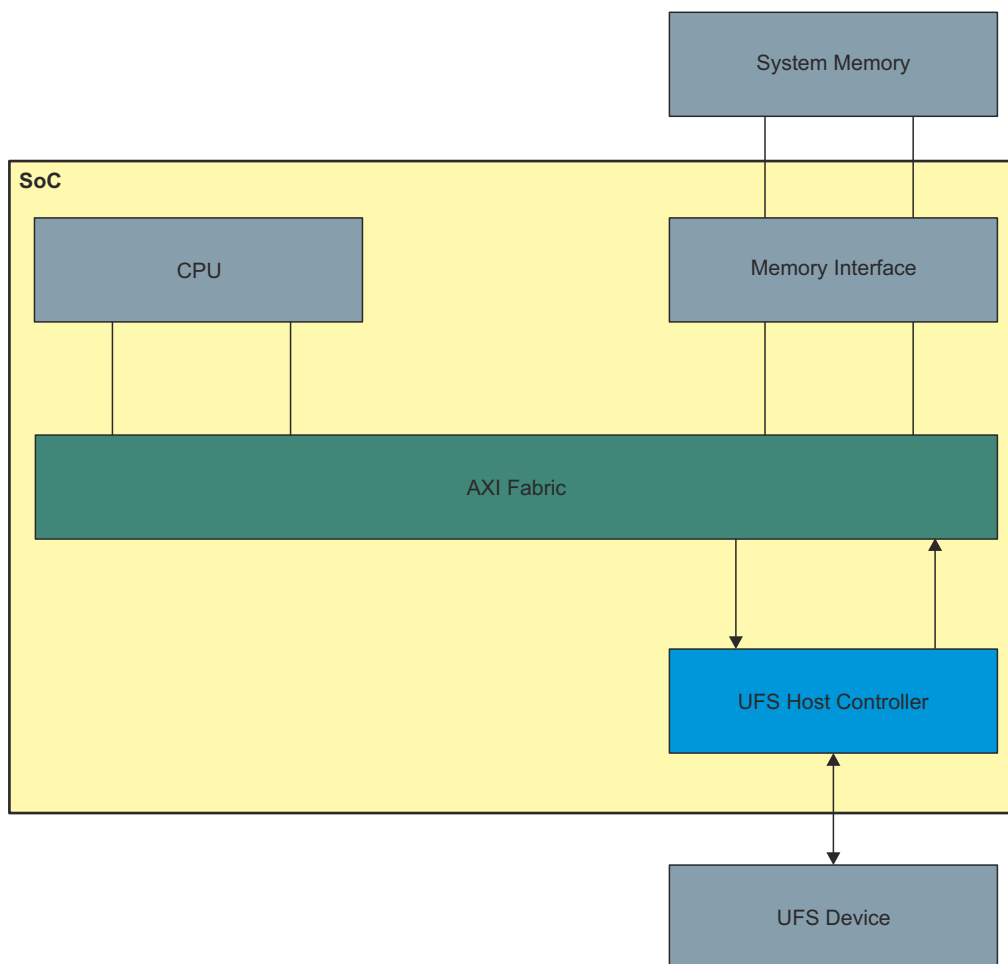
**Figure 12-351. UFS IPS Subsystem**

Figure 12-352 shows UFS controller block diagram.



**Figure 12-352. UFS Controller Block Diagram**

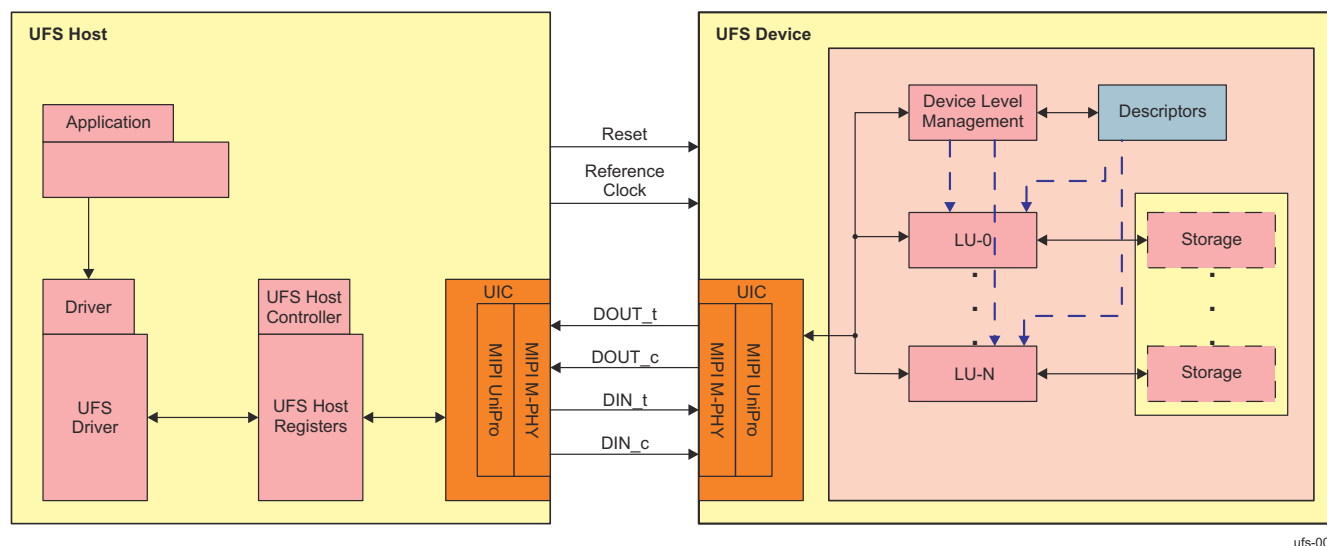
Figure 12-353 presents an example of UFS system level block diagram.



ufs-007

**Figure 12-353. UFS System Level Block Diagram**

Figure 12-354 shows UFS system model. The application layer of the UFS host controller connects to a UniPro protocol stack and M-PHY, both of which can be considered part of the UFS host controller.



**Figure 12-354. UFS System Model**

#### 12.3.7.4.2 UFS ECC Support

The UFS module supports Error Correcting Code (ECC). The ECC is a mechanism for providing increased system reliability via reduction of memory software errors by allowing single bit errors to be detected and corrected (SEC) and double bit errors to be detected (DED).

#### Note

For more information about ECC Aggregator, refer to *ECC Aggregator*.

For more information about UFS ECC Aggregator Registers, refer to *UFS0\_HCLK\_ECC\_AGGR\_CFG Registers*.

#### 12.3.7.5 UFS Programming Guide

#### Note

It is assumed that the reader of this section has an in-depth understanding of the following documents:

- JEDEC Standard Universal Flash Storage Unified Host Controller Interface
- JEDEC Standard Universal Flash Storage
- MIPI Alliance Specification for M-PHY
- MIPI Alliance Specification for Unified Protocol (UniPro®)
- SCSI Architecture Model

##### 12.3.7.5.1 UFS Start-Up Sequence

The initialization of the UFS host controller can be split up into the following four phases:

- Configuring and initializing the UniPro stack. This step can be further split up into the following tasks:
  - Initiate module enable and link startup
  - Set power mode related attributes
  - Start power mode change procedure
  - Set local and peer Layer 2 attributes
- Configuring the UFS host controller



- Initializing the UFS external device
- Refining the UFS host controller setup and configuring the host software driver

---

#### Note

While certain attributes and features of the external device may not be known from the start, commands and transaction will use default values known to work (minimal subset, probably lower performance).

---

#### 12.3.7.5.1.1 UniPro Initialization

##### 12.3.7.5.1.1.1 UniPro Layer 2 Configuration

Layer 2 configuration can be separated into two parts: thresholds and timers.

The calculation and selection of these attribute values have major influence on the performance of a UniPro system as they program the generation of AFC frames. Careful selection of these values is therefore very important. [Section 12.3.7.5.1.1.1.1](#) and [Section 12.3.7.5.1.1.1.5](#) describe these attributes in detail and give some advice on their configuration values.

---

#### Note

AFC frames for acknowledge generation are generated independently from AFC frames for credit exchange.

---

##### 12.3.7.5.1.1.1.1 Layer 2 Threshold Value Calculation

Layer 2 defines three configurable threshold values:

- DL\_TC0TXFCThreshold (see [Section 12.3.7.5.1.1.1.2](#))
- DL\_AFC0CreditThreshold (see [Section 12.3.7.5.1.1.1.3](#))
- DL\_C0OutAckThreshold (see [Section 12.3.7.5.1.1.1.4](#))

These threshold values are used to control AFC frame and credit transmission between the two devices on the link.

##### 12.3.7.5.1.1.1.2 DL\_TC0TXFCThreshold

A change in DL\_TC0TXFCThreshold influences the required level of available credits (1 credit = 32 bytes) in a transmitter before the Protection Timer is started to request an update of credit information.

For example, if this threshold value for TC0 is locally set to 10 credits, the local TX will start the Protection Timer when less than 10 credits are available for sending data to the peer device. If the Protection Timer expires, the local TX sends an AFC0 frame to the peer device with CReq set to 1 to request an update of the peer RX credit information. When the peer device answers with an AFC0 frame at any time, the Protection Timer is restarted if there are still 10 or fewer credits available or stopped otherwise.

This mechanism should prevent that the local TX runs out of credits and needs to stall its data transmissions, when for any reason the credit update was lost or not sent in time.

As a general guideline, this threshold value can be set to its maximum value (1 less than the maximum number of credits of the receive buffer for the corresponding TC, which is stored in the attribute DL\_PeerTC0RxInitCreditVal). With this setting the local TX will try to get an update of the local credit information as long as it has not all possible credits available. The overhead for sending the AFC frames depends on the counter value of the Protection Timer, but should not have a negative impact on the link performance.

**DLTCxTXFCThreshold = DLPeerTxTXInitCrediVal - 1;**

### 12.3.7.5.1.1.3 DL\_AFC0CreditThreshold

This threshold value is the counterpart of the DL\_TC0TXFCThreshold value in the receiver. It defines the amount of credits, which have become available since the last credit update but are not communicated to the peer device yet, before an AFC credit update is triggered.

For example, if this value is set to 5 credits, the device will not update its credit information to the peer device before at least 5×32 bytes are read from the receive buffer of the corresponding traffic class since the last credit update.

Changing this attribute to a lower value will increase the number of AFC frames sent for only updating credit information. A higher value however, might lead to a stall for data transmission on the peer device as not enough credits might be available. The maximum value is defined by the size of the RxBuffer in credits. A higher value for the threshold attribute would break credit handling.

As a guideline, setting this attribute to the credit value corresponding to the maximum size of a frame for this traffic class (MaxFrameSizeTC0), should give reasonable results. In this case, the receiver sends its credit update when at least one additional frame of MaxFrameSizeTC0 can be stored in its buffer.

**DL\_AFC0CreditThreshold = MaxFrameSizeTC0[credits];**

### 12.3.7.5.1.1.4 DL\_TC0OutAckThreshold

This attribute defines the number of frames (+1) to be received before an AFC frame is sent to the peer device.

For example, if this attribute is set to 2, the device will send an AFC frame back when three frames were received for the corresponding traffic class. Setting this attribute to a lower value will increase the number of AFC frames on the link. A higher value might have negative impact on the transmitter, if its own TX buffer reaches its limit as too many unacknowledged frames are stored.

As a guideline, this value should be set according to the two related buffers on each side of the link. The maximum number of frames that can be stored in the smaller buffer (peer TX or local RX) defines the upper bound for this attribute. A value at least 1 or two frames lower gives the AFC frame some time to traverse to the peer side.

**DL\_TC0OutAckThresholdmax = min(PeerTXBufferSizeTC0, LocalRxBufferSizeTC0)[frames];**

### 12.3.7.5.1.1.1.5 Layer 2 Timer Value Calculation

Layer 2 uses three timers to keep the link operational in case of wrong configuration or unstable communication with the peer device. These timers can be separately configured using three attributes:

- DL\_FC0ProtectionTimeOutVal (see [Section 12.3.7.5.1.1.1.6](#))
- DL\_TC0ReplayTimeOutVal (see [Section 12.3.7.5.1.1.1.7](#))
- DL\_AFC0ReqTimeOutVal (see [Section 12.3.7.5.1.1.1.7](#))

#### 12.3.7.5.1.1.1.6 DL\_FC0ProtectionTimeOutVal

The Protection Timer is used in conjunction with DL\_TC0TCXFCThreshold described in [Section 12.3.7.5.1.1.1.1](#). Its purpose is to "protect" the link against the loss of an AFC frame and the resulting stall in communication, if required credits were not delivered.

This timer is running whenever less credits than DL\_TC0TCXFCThreshold are available. After expiration, the local UniPro sends an AFC frame to the peer device requesting a credit information update (CReq set to 1). When this AFC request is sent, the timer is restarted. If the peer device does not respond within the second timer period, the link is automatically reinitialized, as the request seems to be lost.

The configuration of the Protection Timer has very little impact on the system performance. If the timer is set to a low value, the amount of AFC request frames may increase (depending on the threshold value). A too high value results in longer periods of inactivity if the link gets broken.

As a consequence setting this timeout value to any value between 512 and 1023 should result in a good compromise between additional AFC frame generation and latency in case of errors:

$$512 \leq \text{DL\_FC0ProtectionTimeOutVal} \leq 1023;$$

#### 12.3.7.5.1.1.1.7 DL\_TC0ReplayTimeOutVal and DL\_AFC0ReqTimeOut

The Replay Timer of one device is closely connected to the Request Timer on the other device of the link. In general, one Replay Timer for each traffic class is running when transmitted frames are not acknowledged yet and is reset every time a new frame is sent or one or more frames are acknowledged. This timer protects the protocol against lost AFC or NAC frames. When for a period of DL\_TC0ReplayTimeOutVal  $\mu$ s when no AFC is received and no new frame is sent for this traffic class, the timer expires and the link is re-initialized automatically and a retransmission is started.

An UFS device has two methods of generating an AFC frame for acknowledging data reception. The **DL\_TC0OutAck - Threshold value + 1** defines the number of received frames before an AFC is automatically generated. In addition, the AFC Request Timer at the receiver is used to trigger AFC generation before the Replay Timer at the transmitter is expired even when the DL\_TC0OutAckThreshold value is not reached.

To secure proper operation, the AFC Request Timer needs to be programmed to a significantly smaller value than the Replay Timer on the other side. The configuring application should take care that the Replay Timer value includes the AFC Request value, internal processing of a new AFC frame generation, and transmission over the link. To be on the safe side, the local Request Timer should be programmed with a value twice the remote AFC Request Timer value. With this rule the allowed accuracy of the timers of  $\pm 10\%$  should also be accounted for.

The AFC Request Timer should also have a quite high minimum value in order to not generate more AFC frames than actually needed. The following formula gives a general guideline for programming these two timer values.

$$1024 \leq 2 \times \text{DL\_AFC0RequestTimeOutVal} \leq \text{DL\_TC0ReplayTimeOutCal} \leq 4095;$$

#### 12.3.7.5.1.1.2 UniPro CPort Connection Management

For this particular case of UFS, as application layer, the connections are setup statically by having reset values for the UFS host controller and a UFS device predefined at reset. The predefined reset values are given in [Table 12-400](#).

Note that these reset values are different from the default reset values given by the *MIPI UniPro Specification* but are in line with the reset values given by the *JEDEC UFS Specification*. Furthermore, the reset values for the device are only assumed reset values.

**Table 12-400. Reset Values for Some UniPro Attributes**

UniPro Attribute	Reset Value of the UFS Host Controller	Expected UFS Device Reset Value
N_DeviceID	0	1
N_DeviceID_valid	TRUE	TRUE
T_PeerDeviceID	1	0
T_PeerCPortID	0	0
T_TrafficClass	0	0
T_CPortFlags	6 (E2E_FC off, CSD off, CSV off)	6 (E2E_FC off, CSD off, CSV off)
T_ConnectionState	1 (CONNECTED)	1 (CONNECTED)

At link startup and after exiting from the Auto-Hibernate state the UFS host controller will set the UniPro CPort attributes as shown in the following sequence:

CPort 0 Configuration:

- 4020h **T\_ConnectionState** 0h
- 4021h **T\_PeerDeviceID** 1h
- 4022h **T\_PeerCPortID** 0h
- 4023h **T\_TrafficClass** 0h
- 4025h **T\_CPortFlags** 6h
- 4020h **T\_ConnectionState** 1h

Additionally, if the value of the UFS\_XCNF[17-16] PCPCONFEX bit field is set to 3h, the UFS host controller will configure also the CPorts of the PEER device as follows:

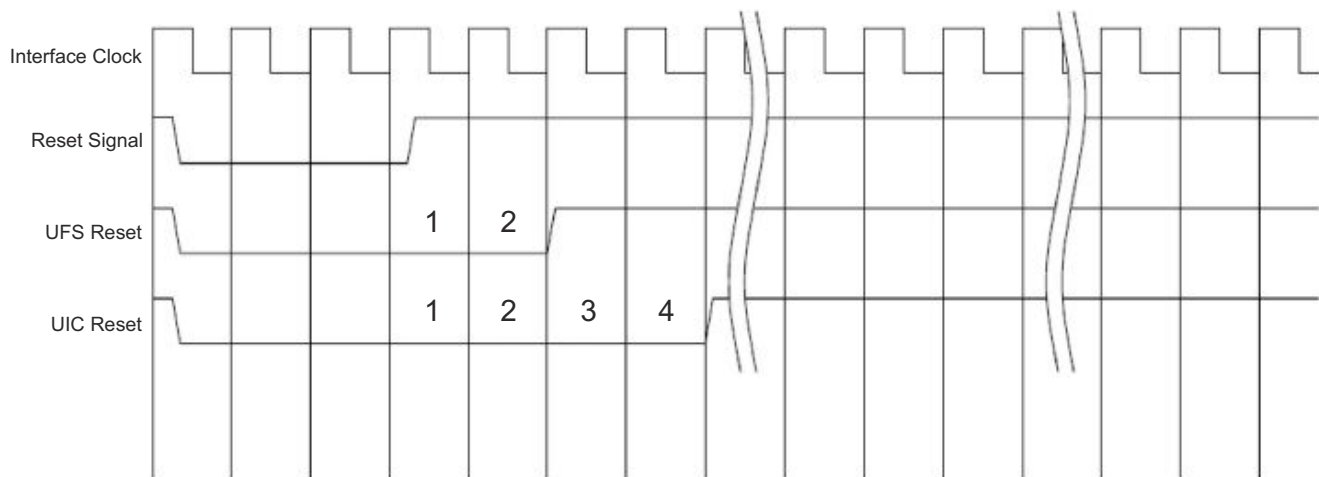
If the value of the UFS\_XCNF[17-16] PCPCONFEX bit field is set to 2h, the UFS host controller will poll the T\_ConnectionState attribute of CPort0 of the PEER device. Only CPort0 will be polled.

#### 12.3.7.5.1.2 UFS Host Controller Initialization

The following procedure shall be applied to initialize the UFS host controller:

1. Assert reset signal. For more information about the UFS reset signal, see *UFS0 Clocks and Resets*.
2. De-assert reset signal and wait for at least 4 interface clock cycles (see [Figure 12-355](#)). For more information about the UFS interface clock, see *UFS0 Clocks and Resets*.
3. Write appropriate value into the FS\_HCLKDIV register.
4. Write a 1h to the FS\_SS\_CTRL[0] RST\_N\_PCS bit to take UFS slave device out of reset.
5. Write a 1h to the UFS\_HCE register in order to enable the UFS host controller. This triggers an autonomous basic initialization of the local UIC layer. The initialization sequence shall consist of a *DME\_RESET* and a *DME\_ENABLE* command. Further commands, such as *DME\_SET* commands may be added, depending on the implementation needs. During the basic initialization sequence, the UFS\_HCE register is read as 0h.
6. Wait until the UFS\_HCE register is read as 1h before continuing. This indicates that the basic initialization sequence is completed.
7. Additional commands, such as *DME\_SET* commands may be sent from the system host to the UFS host controller to provide configuration flexibility.
8. Optionally set the UFS\_IE[10] UCCE bit to 1h in order to enable the interrupt (UFS\_IS[10] UCCS).
9. Send *DME\_LINKSTARTUP* command to start the link startup procedure.
10. Completion of the *DME\_LINKSTARTUP* command sets the UFS\_IS[10] UCCS bit and may flag an interrupt to the system host if the UFS\_IE[10] UCCE bit is set. This interrupt will be flagged independently from the GenericErrorCode.
11. In case the GenericErrorCode of the completed *DME\_LINKSTARTUP* command is *SUCCESS*, the UFS\_HCS[0] DP bit is set in addition to the UFS\_IS[10] UCCS bit.

12. Check the value of the UFS\_HCS[0] DP bit and make sure that there is a device attached to the Link. If presence of a device is detected, go to step **13**; otherwise, resend the *DME\_LINKSTARTUP* command after the UFS\_IS[7] ULLS bit has been set to 1h (go to step **8**). The UFS\_IS[7] ULLS bit equal 1h indicates that the UFS Device is ready for a link startup.
13. Enable additional interrupts by programming the UFS\_IE register.
14. Initialize the Interrupt Aggregation Control register (UFS\_UTRIACR) with the desired values for the threshold (UFS\_UTRIACR[12-8] IACTH) and timeout (UFS\_UTRIACR[7-0] IATOVAL).
15. For example, write value 81010664h to initialize with the following parameters:
  - The UFS\_UTRIACR[31] IAEN bit is set.
  - The UFS\_UTRIACR[24] IAPWEN bit is set.
  - The UFS\_UTRIACR[16] CTR bit is set.
  - The UFS\_UTRIACR[12-8] IACTH bit field is set to 6h.
  - The UFS\_UTRIACR[7-0] IATOVAL bit field is set to 64h (4.0 ms).
16. **Note:** The UFS\_UTRIACR register initialization may be executed at any time when the Run/Stop register (FS\_UTRLRSR) is not enabled or when no requests are outstanding.
17. Complete the UFS host controller configuration via *UIC* command interface if required.
18. Allocate and initialize UTP Task Management Request List.
19. Program the UTP Task Management Request List Base Address (UFS\_UTMRLBA) with a 32-bit address pointing to the starting address of the UTP Task Management Request List created at the step **15**.
20. Allocate and initialize UTP Transfer Request List.
21. Program the UTP Transfer Request List Base Address (UFS\_UTRLBA) with a 32-bit address pointing to the starting address of the UTP Transfer Request List created at the step **17**.
22. Enable the UTP Task Management Request List by setting the UTP Task Management Request List Run Stop register (UFS\_UTMRLRSR) to 1h. This operation allows the UFS host controller to begin accepting UTP Task Management Request via the UTP Task Management Request Door Bell mechanism (UFS\_UTMRLDBR).
23. Enable the UTP Transfer Request List by setting the UTP Transfer Request List Run Stop register (UFS\_UTRLRSR) to 1h. This operation allows the UFS host controller to begin accepting UTP Transfer Request via the UTP Transfer Request Door Bell mechanism (UFS\_UTRLDBR).
24. At this point, the UFS host controller is up and running.



ufs-009

**Figure 12-355. UFS Host Controller Reset Waveform**

#### 12.3.7.5.1.3 HCE Bit

When the Host Controller Enable (UFS\_HCE) bit transitions from 0h to 1h, the UFS host controller will go through an initialization sequence that is outlined as follows:

- Warm hardware reset of the UFS host controller
- *DME\_RESET* (Warm)
- *DME\_ENABLE*
- *LINK\_STARTUP*

As a part of the Link Startup procedure, the UFS interface will configure the CPort Attributes of the local and eventually the peer UniPro. Refer to [Section 12.3.7.5.1.1.2, UniPro CPort Connection Management](#).

### 12.3.7.5.2 UFS Host Controller Programming

#### 12.3.7.5.2.1 UFS Software Model

This section provides details on the transactions, protocol elements, and data structures which need to be used to control the UFS host controller, MIPI UniPro stack, and external device. The transactions are built by the system hosts and sent to the UFS controller via the UTP Transfer Request List, the UTP Task Management Request List, and the UTP Command Descriptors.

While some transactions are already prepared as UPIU packets by the host software, for some other transactions (mainly the transfer of the raw data), the UPIU packets need to be constructed first. The construction takes place on the fly in the UFS Controller as listed in [Table 12-401](#).

**Table 12-401. UFS Host Controller Commands**

Command	Format	Action
UTP Command	UTP Transfer Request List	Send to device
DME Command	Register Access	Access UniPro stack
Task Management Command	UTP Task Management Request List	Send to device

To send commands to the UFS host controller, UniPro stack, and UFS device, the system host needs to perform a series of setup and configurations steps. The following list gives a brief summary on the necessary sequence:

- Find an empty slot in the UTP Transfer Request List (UTP Task Management Request List) by reading the *UFS\_UTRDLDBR* (*UFS\_UTMRLDBR*) register.
- Program the required transactions in the UTP Transfer Request List (UTP Task Management Request List) and depending on the transfer setup a UTP Command Descriptor.
- Then setup the appropriate bit in the *UFS\_UTRDLDBR* (*UFS\_UTMRLDBR*) register.

For a detailed description, refer to [Section 12.3.7.5.2.2, UFS Theory of Operation](#).

The UFS host controller starts processing the UTP Transfer Request List and UTP Task Management Request List.

Once the UFS host controller has finished processing a command, it updates the status field in the UTP Transfer Request List (UTP Task Management Request List) or UTP Command Descriptor respectively.

The UFS Transport Protocol (UTP) commands are commands based on UPIU packets. To set up the communication, both the transaction lists as well as the command descriptors need to be programmed.

#### 12.3.7.5.2.1.1 UFS Layers

The UFS interface is based on master/slave architecture. Direct device-to-device communication is not supported. The information flow is packet based and the packets are sent in a request/response method where the Initiator sends a request to a Target and the Target must always respond. There is only one UFS host controller on the physical interface and most requests are typically initiated by the UFS host controller. The requests initiated by UFS device could be an interrupt or any other request for service actions.

The communication between the UFS host controller and the device is layer oriented. The UFS specification defines the following layers:

- UFS communication is layered communication architecture. The information is packetized at different layer with the appropriate header.
- UFS InterConnect Layer (UIC) - the UIC Layer is based on MIPI M-PHY and the MIPI UniPro stack:





segments does not end on a 32-bit boundary, then additional padding with 0h data will be added to round up the UPIU to the next 32-bit address boundary.

#### 12.3.7.5.2.1.2.2 UFS Protocol

As described earlier, the communication between the host system host and the external device is based on UPIU packets. These packets are used to define the following protocols:

- UTP: the SCSI Architecture Model (SAM) is used as general architectural model for the UFS SCSI. This model uses the SCSI command set and the SAM Task Management features to establish the communication between the system host and the UFS device. For the UFS implementation a reduced SCSI command set is used.
- Native UFS communication: native UFS commands are used to communicate with the UFS host controller and the MIPI UniPro stack outside of the SCSI protocol.

#### 12.3.7.5.2.1.3 UFS Host Data Structure

Most of the communication between host software and the UFS host controller or the UFS device respectively is via system memory tables, descriptors and buffers. These memory locations describe commands to be executed, the data transfer operations that are part of the commands, store status information coming back from the device and provide buffer space for data transactions.

#### Note

For additional information on host data structures and programming procedures, refer to *Universal Flash Storage Host Controller Interface (UFSHCI)*, JESD223C.

#### 12.3.7.5.2.2 UFS Theory Of Operation

As described in [Section 12.3.7.5.2.1.3, UFS Host Data Structure](#), the host software presents a list of commands to the UFS host controller. Independent from the system host, this list is then being processed by the UFS host controller which asserts an interrupt line in case a command finishes.

On order to post new commands to the UFS host controller, the system host needs to find empty slots in the command list. In case no empty slot is available, the host thread needs to wait until a command finishes and the driver software has made a new slot available.

Nevertheless, the UFS host controller continuously looks at the UFS\_UTRLDBR and UFS\_UTMRLDBR registers to determine if there are commands ready to be processed. The processing order depends on three factors:

- Time at which the bit(s) inside the UFS\_UTRLDBR and UFS\_UTMRLDBR registers have been asserted.
- Index inside the UFS\_UTRLDBR and UFS\_UTMRLDBR registers.
- Command type (UTP or Task Management).

The processing rules are as follows:

1. Task Management commands have higher priority than UTP commands.
2. Commands with lower time stamp have higher priority than commands with higher time stamps.
3. Within the same time stamp, commands with lower doorbell register index have higher priority than commands with a higher doorbell register index.

While there is no indicator about active commands, it is not possible to know in advance when a particular command is being processed, or how the command will remain in the command list respectively.

To add an additional level of uncertainty, any UFS device is allowed to accept more than one command and is allowed to process those commands out-of-order. As a consequence, commands placed in the command list need to be robust enough to get arbitrarily re-ordered. In cases where this may cause problems, only one command at a time may be placed into the command list.

[Section 12.3.7.5.2.2.1, Building a UTP Transfer Request](#) provide more detail on the low-level driver to explain how the communication and data transfer is handled between system host, the UFS host controller, and UFS device.



### 12.3.7.5.2.2.1 Building A UTP Transfer Request

Typically a command for the UFS host controller consists of a UTP Transfer Request List entry and a command table element. Thus, the host software needs to start by finding an empty slot inside the UTP Transfer Request List first.

Here are the basic steps for setting up the command list:

- Find an empty command list slot by reading the UFS\_UTRLDBR register.
- In case there are no empty slots, wait and try again.
- In case there is at least one empty slot, start setting up the UTP Transfer Request Descriptor:
  - Write CT: define command type.
  - Write DD: define data direction.
  - Write I: define whether or not an interrupt has to be generated.
  - Initialize OCS with the value Fh.
- Allocate space inside the system memory for an UTP Command Descriptor.
- Initialize Command UPIU field of the UTP Command Descriptor:
  - Write TTY: transaction type, here 1h.
  - Write FL: specific flags for this command.
  - Write LUN: the SCSI address this command is directed to.
  - Write TT: task tag, a unique number to identify the task.
  - Write CST: command set type, 0h for SCSI commands.
  - Write TEL: total EHS Length, always 0h.
  - Write DSL: data segment length, always 0h.
  - Write EDTL: expected data transfer length. This field indicates the amount of data to be transferred between the Initiator and Target (read or write). Data is stored in system memory.
  - Write CDB: write the SCSI command into the following 16 bytes.
- Initialize Response UPIU field of the UTP Command Descriptor with zero.
- Initialize PRD field of the UTP Command Descriptor with the required data scatter/gather buffer information.
- Now the UTP Command Descriptor has been set up and the UTP Transfer Request List Descriptor needs to be completed:
  - Write UCDBA: base address of the UTP Command Descriptor.
  - Write UCDBAU: upper 32 bit of the UCDBA, always 0h.
  - Program field RUO with the offset (from the starting address of UCD) of Response UPIU within UCD.
  - Program field RUL with the length of Response UPIU.
  - Program field PRDTO with the offset (from the starting address of UCD) of the PRD table within UCD if required.
  - Program field PRDTL with the length of the PRD table if required.
- Repeat this procedure for all commands which need to be set up.
- Check the UFS\_UTRLRSR register and make sure it is read 1h before continuing.
- Set UTP Transfer Request Interrupt Aggregation Control Register (UFS\_UTRIACR) enable bit to 1h to enable the interrupt.
- Set Counter and Timer Reset (UFS\_UTRIACR[16] CTR) bit to 1h to reset the counter and timer associated with the interrupt.
- Program field Interrupt aggregation counter threshold (UFS\_UTRIACR[12-8] IACTH) with the number of command completions that are required to generate an interrupt.
- Program field Interrupt aggregation timeout value (UFS\_UTRIACR[7-0] IATOVAL) with the maximum time allowed between a response arrival to the UFS host controller and the generation of an interrupt.
- Set the UFS\_UTRLDBR register to ring the doorbell register to indicate to the UFS host controller that one or more transfer requests are ready to be sent to the attached device. The host software shall only write a 1h to the bit position that corresponding to the new command. All other bit positions within the UFS\_UTRLDBR register should be written with a 0h, which indicates no change to their current values.

Now the UFS host controller will take over and process the command list without further software interaction.

#### 12.3.7.5.2.2.2 Processing UTP Task Management Request Completion

The host software processes the interrupt generated by the UFS host controller for command completion. In the interrupt service routine, the host software checks the UFS\_IS register to determine if there is an interrupt pending. If the UFS host controller has an interrupt pending:

- The host software determines the cause of the interrupt by reading the UFS\_IS register. If the UFS\_IS[9] UTMRCs bit is set this indicates that a UTP Task Management Request has completed.
- The host software clears appropriate bits in the UFS\_IS register corresponding to the cause of the interrupt.
- The host software reads the UFS\_UTMRLDBR register, and compares the current value to the list of commands previously issued by the host software that are still outstanding. The host software completes with success any outstanding command whose corresponding bit has been cleared in the respective register. The UFS\_UTMRLDBR register is a volatile. The software should only use its value to determine commands that have completed, not to determine which commands have previously been issued.
- If there were errors, noted in the UFS\_IS register, the host software performs error recovery actions.

#### 12.3.7.5.2.2.3 Building UTP Task Management Request

Typically a command for the UFS host controller consists of a UTP Transfer Request List entry and a command table element. Thus, the host software needs to start by finding an empty slot inside the UTP Task Management Request List first.

Basic steps for setting up the Task Management Request:

- Find an empty command list slot by reading the UFS\_UTMRLDBR register.
- In case there are no empty slots, wait and try again.
- In case there is at least one empty slot, start setting up the UTP Task Management Request Descriptor:
  - Write I: define whether or not an interrupt has to be generated.
  - Initialize OCS with the value Fh.
- Allocate space inside the system memory for an UTP Command Descriptor.
- Repeat the above procedure for all task management commands that need to be set up.
- Check UTP Task Management Request List Run Stop register (UFS\_UTMRLRSR) and make sure it is read 1h before continuing.
- Set UTP Transfer Request Interrupt Aggregation Control register (UFS\_UTRIACR) enable bit to 1h to enable the interrupt.
- Set the UFS\_IE[9] UTMRCe bit to 1h to enable the interrupt.
- Set UTP Task Management Request List Doorbell register (UFS\_UTMRLDBR) to ring the doorbell register to indicate to the UFS host controller that one or more transfer requests are ready to be sent to the attached device. The host software shall only write a 1h to the bit position that corresponding to the new command. All other bit positions within the UFS\_UTMRLDBR register should be written with a 0h, which indicates no change to their current values.

Now the UFS host controller will take over and process the command list without further software interaction.

#### 12.3.7.5.2.2.4 Processing UTP Transfer Request Completion

Control will be given back to the software by assertion of the Transfer Request Completion Status interrupt (UFS\_IS[0] UTRCS). The UFS\_IS[0] UTRCS bit is set when at least one of the following 4 conditions is met:

- The UTRD.I bit is set (Interrupt Command).
- The counter, after incrementing, reaches the value configured in the UFS\_UTRIACR[12-8] IACTH bit field.
- The IA (Interrupt Aggregation) timer reaches the value configured in the UFS\_UTRIACR[7-0] IATOVAL bit field (this event may occur at any time, not necessarily coupled with request completion).
- The Overall Command Status (OCS) of the completed command is unequal to "SUCCESS".

An interrupt is generated by the write operation if the completion interrupt is not masked (disabled) by the UFS\_IE[0] UTRCE bit.

The host software processes the interrupt generated by the UFS host controller for command completion. In the interrupt service routine, the host software checks the UFS\_IS register to determine if there is an interrupt pending. If the UFS host controller has an interrupt pending:

1. If there were errors, noted in the UFS\_IS register, the host software performs error recovery actions.
2. In the case of the Transfer Request Completion Status interrupt (UFS\_IS[0] UTRCS), the host software clears the interrupt and then may use one of two methods to determine which UTP transfer requests (TRs) have completed:
  - a. Read the UFS\_UTRLDBR register, and compare the current value to the list of commands previously issued by the host software that are still outstanding. For any transfer request which is outstanding, a value of 0h in bit *i* (where *i* is the UTRL slot through which the TR is issued) of the UFS\_UTRLDBR register means that the TR has completed. The UFS\_UTRLDBR register is a volatile. The software should only use its value to determine commands that have completed, not to determine which commands have previously been issued.
  - b. Read the UTP Transfer Request List Completion Notification register (UFS\_UTRLCNR). For any TR, a value of 1h in bit *i* (where *i* is the UTRL slot through which the TR is issued) of the UFS\_UTRLCNR register means that the TR has completed.
3. For every TR *i* whose completion is detected, the software repeats the following steps:
  - a. Processes the request completion as required by higher OS layers (for example: file system).
  - b. Clears bit *i* of the UFS\_UTRLCNR register, by writing 1h to it.
  - c. Marks slot *i* as available for reuse (software only).
4. After processing all previously detected TRs, the software may reset and restart Interrupt Aggregation mechanism by writing 80010000h to the UFS\_UTRIACR register.
5. The software determines if new TRs have completed since step 2, by repeating one of the two methods described in step 2. If new TRs have completed, the software repeats the sequence from step 3.

#### 12.3.7.5.2.2.5 UFS Host Processing

The host processing of the commands initiated by the system host is done independent of the system host and software. The exact steps the UFS host controller will perform may vary, but the baseline steps for a read/write operation are given below:

- The UFS host controller will read the selected command list entry with one 4-beat burst.
- Depending on the command type, further actions are taken. In case the command is read/write:
  - The UFS host controller will fetch the first part of the UPIU (till the data segment) in 4-beat bursts. This will bring the header information and the SCSI command into the controller.
  - The UFS host controller will read the first PRD entry with a 2-beat burst.
  - Now the UFS host controller will set up its internal data structure and variables.
  - To start the device communication, the UFS host controller will send the already fetched UPIU packet to the device.
- Now the device will initiate the data transfer phase by sending Ready-To-Transfer (RTT) UPIUs to the UFS host controller.
  - The device either sends a RTT UPIU (in case of a write) or directly a Data In UPIU (in case of a read). The size of the Data In UPIU is not predefined.
  - The UFS host controller either sends back the requested data with a Data Out UPIU, or takes the data and stores it in an internal buffer.

The above transactions between the UFS host controller and the device are executed until all data is transferred or an error occurred.

In parallel to the transactions between the UFS host controller and device, the data is transferred between the UFS host controller and system memory.

- On receiving the Response UPIU, the UFS host controller updates the Transfer Response field of the UTP Command Descriptor with the received UPIU.
- Once a command has been completed, depending on the Interrupt Enable register (UFS\_IE), an interrupt may be asserted to signal the system host that the transfer is done.

#### 12.3.7.5.2.2.6 UFS Response Management And Command Completion

Although not a command, the management of command completion, response management, and error handling can be considered to be related to the command interface.

There are essentially three steps to perform when a response UPIU arrives:

- Analyzing the packet.
- Storing of the sense data which might be part of the response packet.
- Modification of the Response UPIU field in the related UPIU command table entry.

As a last step and depending on the command setup, the UFS host controller will set the interrupt flag.

### 12.3.7.5.3 UFS PHY Programming

This section details specific steps on how to program the M-PHY.

The M-PHY registers can be used to configure and read the status of the M-PHY. These M-PHY registers are part of the common UFS host controller address space . For detailed description of the UFS and M-PHY registers, refer to *UFS0\_P2A\_WRAP\_CFG\_VBP\_UFSHCI Registers*.

UFS and M-PHY address spaces:

- UFS address space: 04E8 0000h – 04E8 1103h
- M-PHY address space: 04E8 1104h – 04E8 1193h

The M-PHY configuration and status bits are connected directly either to the top-level ports or to the M-PHY register bank or to both exclusively.

The UFS\_MPHY\_MMIO\_A[0] MMIO\_A configuration bit is being used to choose if the dual-connection M-PHY configuration bits are connected to the registers or to IOs. After the reset, the UFS\_MPHY\_MMIO\_A register value is 0h, so the M-PHY configuration bits are connected to the IOs by default. These bits are listed in [Table 12-402](#).

**Table 12-402. The Dual Connection M-PHY Configuration Bits**

UFS_MPHYCFG_VCONTROL		UFS_MPHYCFG_MISC		UFS_MPHY_M MIO_A = 1	UFS_MPHY_M MIO_A = 0
Offset	1188h	Offset	1184h		
Bits	Name	Bits	Name		
16	VCONTROL_LA_SA_SEL	29	CMN_MPX_EN_MMIO	Registers	I/Os
11-10	VCONTROL_DEEMP_SEL	28-26	CMN_MPX_SEL_MMIO		
9-0	VCONTROL	25	TX0_TEST_15_MMIO		
		24	TX1_TEST_15_MMIO		
		17	REFCLK_NOGATED		
		16-15	REFCLK_FREQ_SEL		
		6-0	DEBUG_SEL		

#### Note

To enlarge the M-PHY HS-Gear mode initialization time, set M-PHY configuration for the digital part register (UFS\_MPHYCFG\_XCFGD1) to 01000000h.

The UFS\_MPHY\_MMIO\_A register will be stuck at zero in test mode in order to prevent toggling any of the dependent M-PHY input ports during scan procedure.

### 12.3.7.5.4 UFS Hibernate Timings Considerations

Software should take care of the following hibernate timings specifics:

1. An additional 22us needs to be added to the default TActivate time after establishing a link. This can be achieved by adjusting the host's *PA\_Granularity* and *PA\_TActivate* attributes in the Unipro layer as follows:

```
If (PA_Granularity == 6)
    PA_Granularity == 1;
```

```

        PA_TActivate = PA_TActivate*100 + 22;
else if (PA_Granularity == 5)
    PA_Granularity == 1;
    PA_TActivate = PA_TActivate*32 + 22;
else if (PA_Granularity == 4)
    PA_Granularity == 1;
    PA_TActivate = PA_TActivate*16 + 22;
else if (PA_Granularity == 3)
    PA_Granularity == 1;
    PA_TActivate = PA_TActivate*8 + 22;
else if (PA_Granularity == 2)
    PA_Granularity == 1;
    PA_TActivate = PA_TActivate*4 + 22;
else if (PA_Granularity == 1)
    PA_Granularity == 1;
    PA_TActivate = PA_TActivate + 22;

```

2. The TActivate time at hibernate exit on the peer side needs to be increased, by changing the *RX\_Advanced\_Min\_ActivateTime\_Capability* attribute default value from 0x04 to 0x0A in the M-PHY layer on host side.
3. The connection of differential termination resistance needs to be enabled, by setting to '1' the *Termination\_Force\_Enable* attribute in the M-PHY layer on host side.

## 12.4 Industrial and Control Interfaces

This section describes the industrial and control interfaces in the device.

### 12.4.1 Enhanced Capture (ECAP) Module

This section describes the Enhanced Capture (ECAP) module in the device.

#### 12.4.1.1 ECAP Overview

The Enhanced Capture (ECAP) module can be used for:

- Sample rate measurements of audio inputs
- Speed measurements of rotating machinery (for example, toothed sprockets sensed via Hall sensors)
- Elapsed time measurements between position sensor pulses
- Period and duty cycle measurements of pulse train signals
- Decoding current or voltage amplitude derived from duty cycle encoded current/voltage sensors

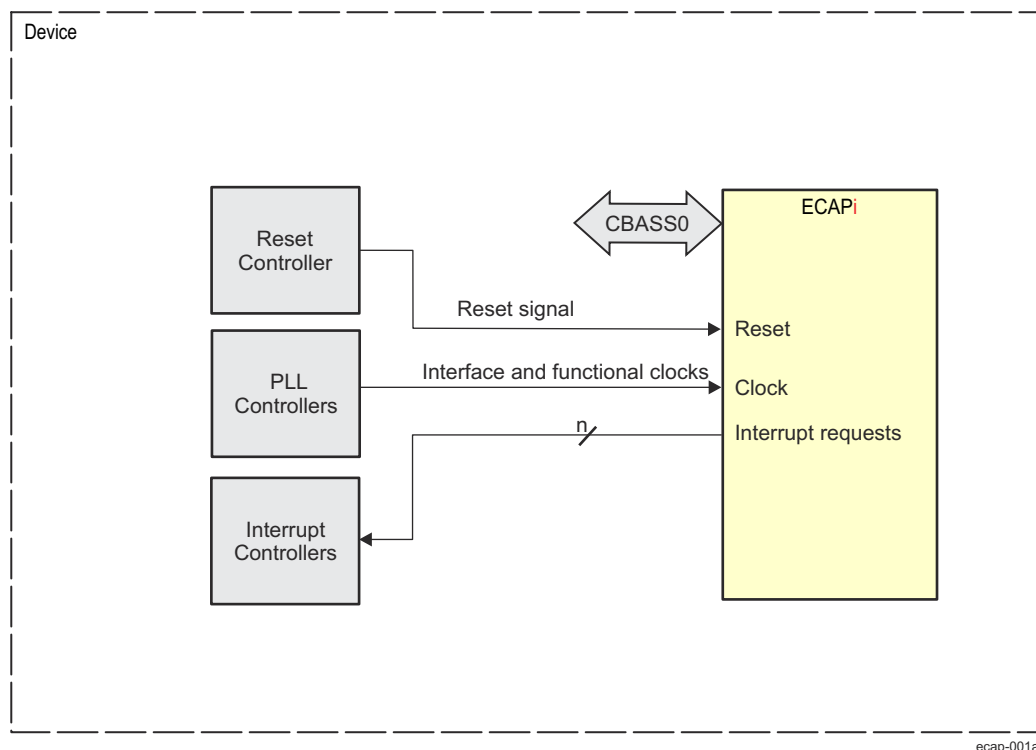
The device has three instances of the ECAP module.

Table 12-403 shows the ECAP allocation across device domains.

**Table 12-403. ECAP Allocation Across Device Domains**

Instance	Domain		
	WKUP	MCU	MAIN
ECAP0	-	-	✓
ECAP1	-	-	✓
ECAP2	-	-	✓

Figure 12-357 shows the ECAP modules overview.



A.  $i = 0 \text{ to } 2$

**Figure 12-357. ECAP Overview**

#### **12.4.1.1.1 ECAP Features**

The ECAP module includes the following features:

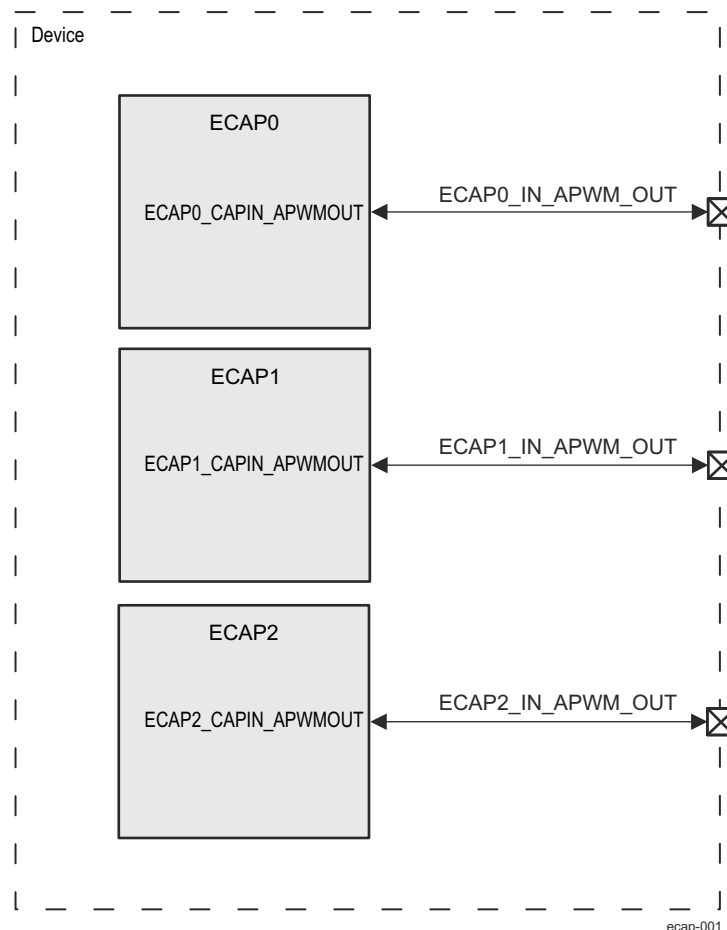
- 32-bit time base counter
- 4 × 32 bits event time-stamp capture registers (ECAP\_CAP1 through ECAP\_CAP4)
- 4-stage sequencer (Mod4 counter), synchronized to external events (ECAPx pin edges)
- Independent edge polarity (rising / falling edge) selection for all 4 sequenced time-stamp capture events
- Input capture signal pre-scaling (from 1 to 16)
- One-shot compare register (2 bits) to freeze captures after 1 to 4 time-stamp events
- Continuous mode capture of time-stamps in a four-deep circular buffer
- Interrupt capabilities on any of the 4 capture events
- Absolute time-stamp capture
- Difference (Delta) mode time-stamp capture
- All above resources dedicated to a single input pin
- When not used in capture mode, the ECAP module can be configured as a single channel PWM output

## 12.4.1.2 ECAP Environment

### 12.4.1.2.1 ECAP I/O Interface

This section describes the ECAP external connections (environment).

Figure 12-358 shows the ECAP interface signals.



**Figure 12-358. ECAP External Interface I/Os**

Table 12-404 describes the ECAP I/O signals.

**Table 12-404. ECAP I/O Signals**

Module Pin	Device Level Signal	I/O <sup>(1)</sup>	Description	Module Pin Reset Value <sup>(2)</sup>
<b>ECAP0</b>				
ECAP0_CAPIN_APWMOUT	ECAP0_IN_APWM_OUT	I/O	ECAP0 Capture input / PWM output	HiZ
<b>ECAP1</b>				
ECAP1_CAPIN_APWMOUT	ECAP1_IN_APWM_OUT	I/O	ECAP1 Capture input / PWM output	HiZ
<b>ECAP2</b>				
ECAP2_CAPIN_APWMOUT	ECAP2_IN_APWM_OUT	I/O	ECAP2 Capture input / PWM output	HiZ

(1) I = Input; O = Output

(2) HiZ = High Impedance



---

**Note**

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

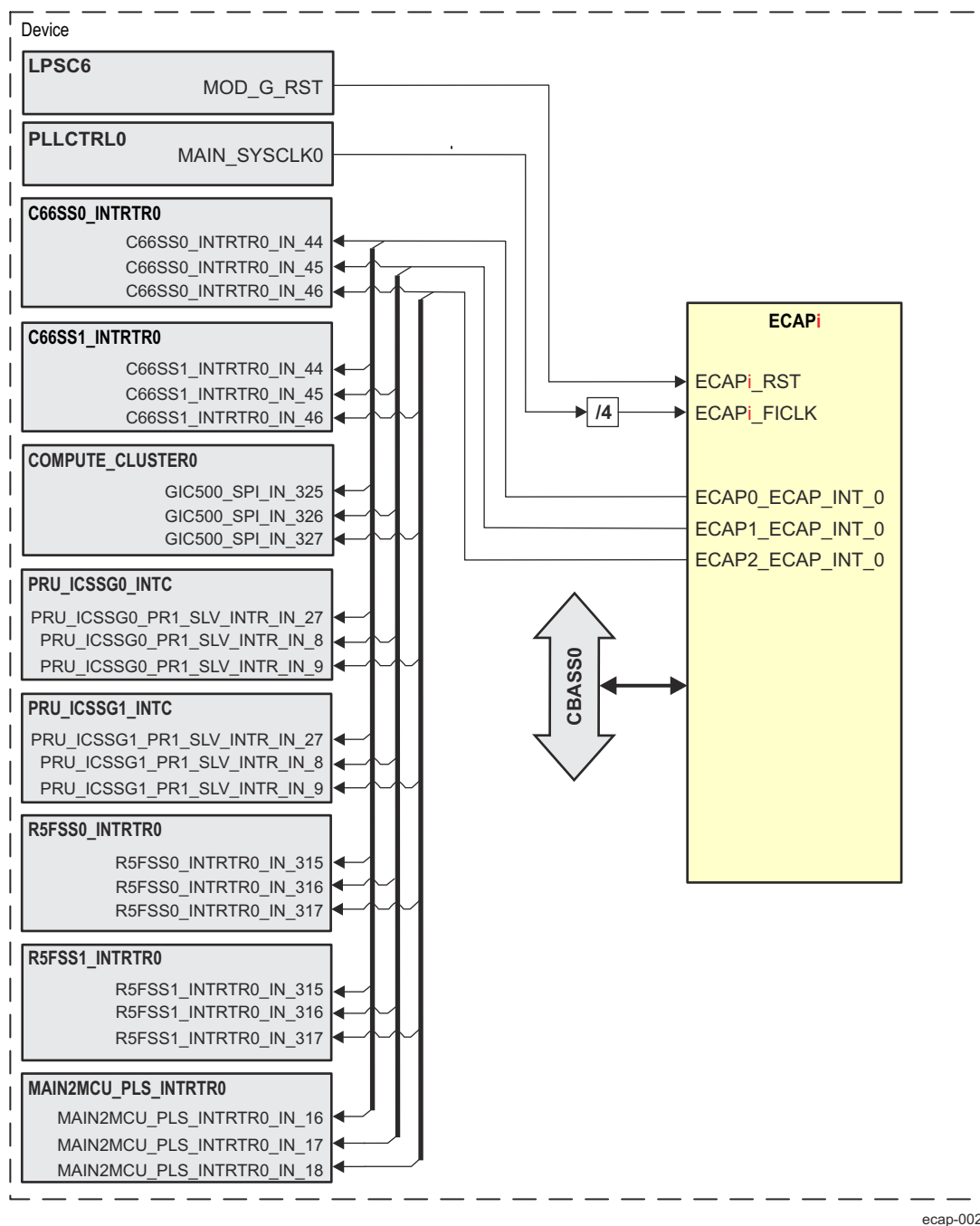
---

### 12.4.1.3 ECAP Integration

This section describes module integration in the device, including information about clocks, resets, and hardware requests.

Figure 12-359 shows the ECAP integration.

There are three instances of the Enhanced Capture (ECAP) module integrated in the device.



A.  $i = 0 \text{ to } 2$

**Figure 12-359. ECAP Integration**

Table 12-405 through Table 12-407 summarize the integration of ECAP0, ECAP1 and ECAP2 modules in device MAIN domain.

**Table 12-405. ECAP Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
ECAP0	PSC0	PD0	LPSC6	CBASS0
ECAP1	PSC0	PD0	LPSC6	CBASS0
ECAP2	PSC0	PD0	LPSC6	CBASS0

**Table 12-406. ECAP Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
ECAP0	ECAP0_FICLK	MAIN_SYSCCLK0/4	PLLCTRL0	ECAP0 functional and interface clock
ECAP1	ECAP1_FICLK	MAIN_SYSCCLK0/4	PLLCTRL0	ECAP1 functional and interface clock
ECAP2	ECAP2_FICLK	MAIN_SYSCCLK0/4	PLLCTRL0	ECAP2 functional and interface clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
ECAP0	ECAP0_RST	MOD_G_RST	LPSC6	Module Reset
ECAP1	ECAP1_RST	MOD_G_RST	LPSC6	Module Reset
ECAP2	ECAP2_RST	MOD_G_RST	LPSC6	Module Reset

**Table 12-407. ECAP Hardware Requests**

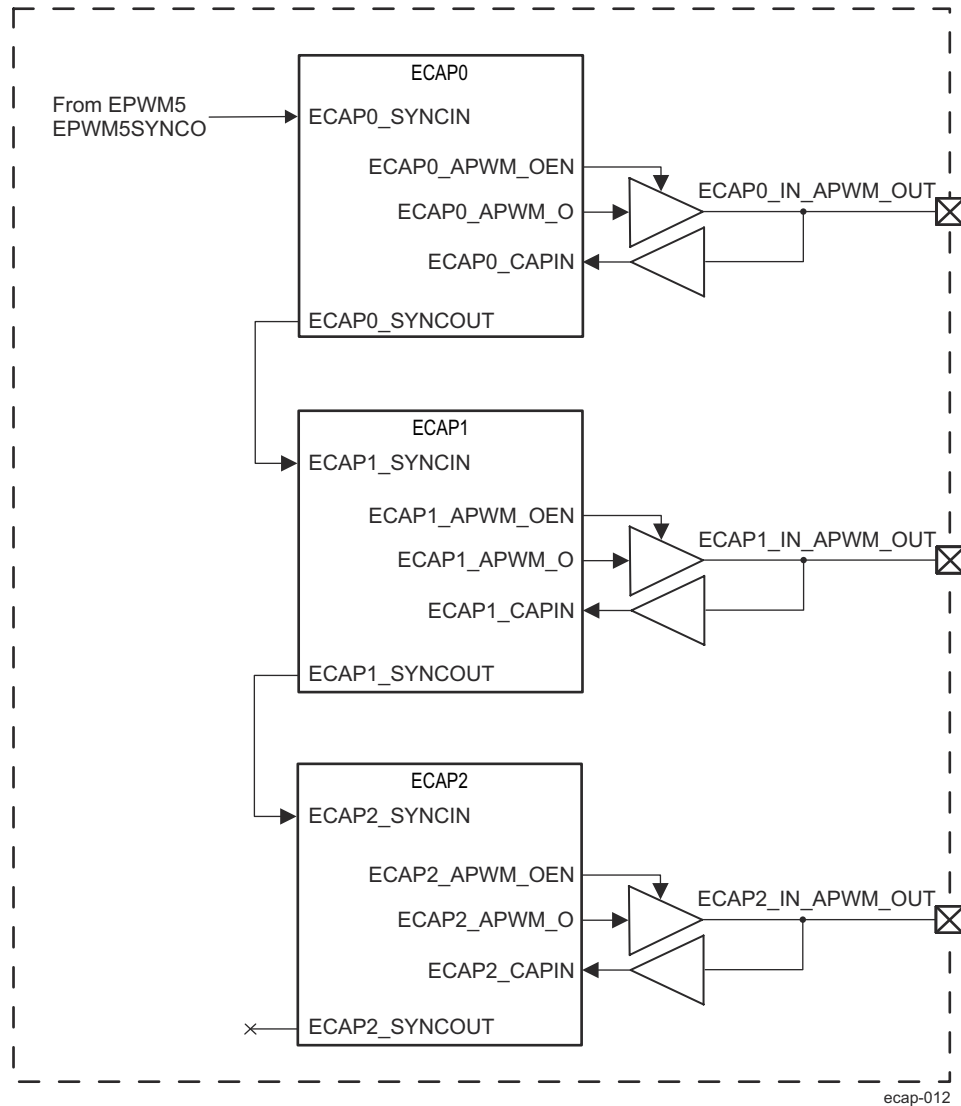
Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
ECAP0	ECAP0_ECAP_INT_0	C66SS0_INTRTR0_IN_44	C66SS0_INTRTR0	ECAP0 interrupt	Pulse
		C66SS1_INTRTR0_IN_44	C66SS1_INTRTR0		
		GIC500_SPI_IN_325	COMPUTE_CLUSTER0		
		PRU_ICSSG0_PR1_SLV_INTR_IN_27	PRU_ICSSG0_INTC		
		PRU_ICSSG1_PR1_SLV_INTR_IN_27	PRU_ICSSG1_INTC		
		R5FSS0_INTRTR0_IN_315	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_315	R5FSS1_INTRTR0		
		MAIN2MCU_PLS_INTRTR0_IN_16	MAIN2MCU_PLS_INTRTR0		
ECAP1	ECAP1_ECAP_INT_0	C66SS0_INTRTR0_IN_45	C66SS0_INTRTR0	ECAP1 interrupt	Pulse
		C66SS1_INTRTR0_IN_45	C66SS1_INTRTR0		
		GIC500_SPI_IN_326	COMPUTE_CLUSTER0		
		PRU_ICSSG0_PR1_SLV_INTR_IN_8	PRU_ICSSG0_INTC		
		PRU_ICSSG1_PR1_SLV_INTR_IN_8	PRU_ICSSG1_INTC		
		R5FSS0_INTRTR0_IN_316	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_316	R5FSS1_INTRTR0		
		MAIN2MCU_PLS_INTRTR0_IN_17	MAIN2MCU_PLS_INTRTR0		
ECAP2	ECAP2_ECAP_INT_0	C66SS0_INTRTR0_IN_46	C66SS0_INTRTR0	ECAP2 interrupt	Pulse
		C66SS1_INTRTR0_IN_46	C66SS1_INTRTR0		
		GIC500_SPI_IN_327	COMPUTE_CLUSTER0		
		PRU_ICSSG0_PR1_SLV_INTR_IN_9	PRU_ICSSG0_INTC		
		PRU_ICSSG1_PR1_SLV_INTR_IN_9	PRU_ICSSG1_INTC		
		R5FSS0_INTRTR0_IN_317	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_317	R5FSS1_INTRTR0		

**Table 12-407. ECAP Hardware Requests (continued)**

MAIN2MCU_PLS_INTRTR0_IN_18	MAIN2MCU_PLS_INTRTR0
----------------------------	----------------------

#### 12.4.1.3.1 Daisy-Chain Connectivity between ECAP Modules

The ECAP modules provide input synchronisation signals to allow them to be synchronized to other modules or events. On the device, these signals are connected in a daisy-chain fashion for the ECAP0 to ECAP2 module as shown in [Figure 12-360](#). The ECAP capture events come from external input pins.



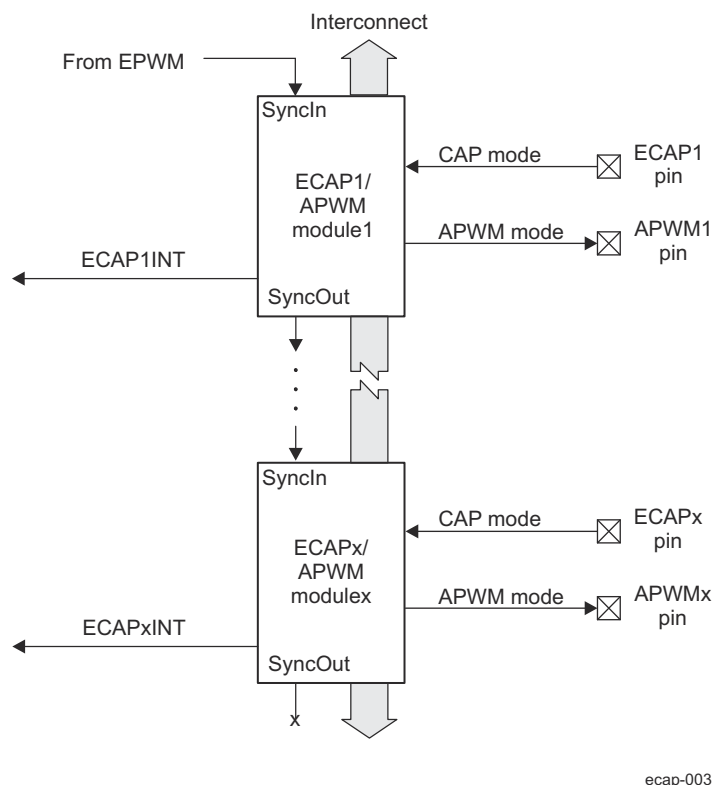
**Figure 12-360. ECAP Daisy-Chain Connectivity**

### 12.4.1.4 ECAP Functional Description

The ECAP module represents one complete capture channel, which has the following independent key resources:

- Dedicated input capture pin
- 32 events selection mapping capability to the input capture pin
- 32-bit time base counter
- 4 × 32-bit time-stamp capture registers (ECAP\_CAP1 through ECAP\_CAP4)
- 4-stage sequencer (Mod4 counter) that is synchronized to external events, ECAP pin rising/falling edges.
- Independent edge polarity (rising/falling edge) selection for all 4 events
- Input capture signal prescaling (from 2-62)
- One-shot compare register (2 bits) to freeze captures after 1 to 4 time-stamp events
- Control for continuous time-stamp captures using a 4-deep circular buffer (ECAP\_CAP1 through ECAP\_CAP4) scheme
- Interrupt capabilities on any of the 4 capture events

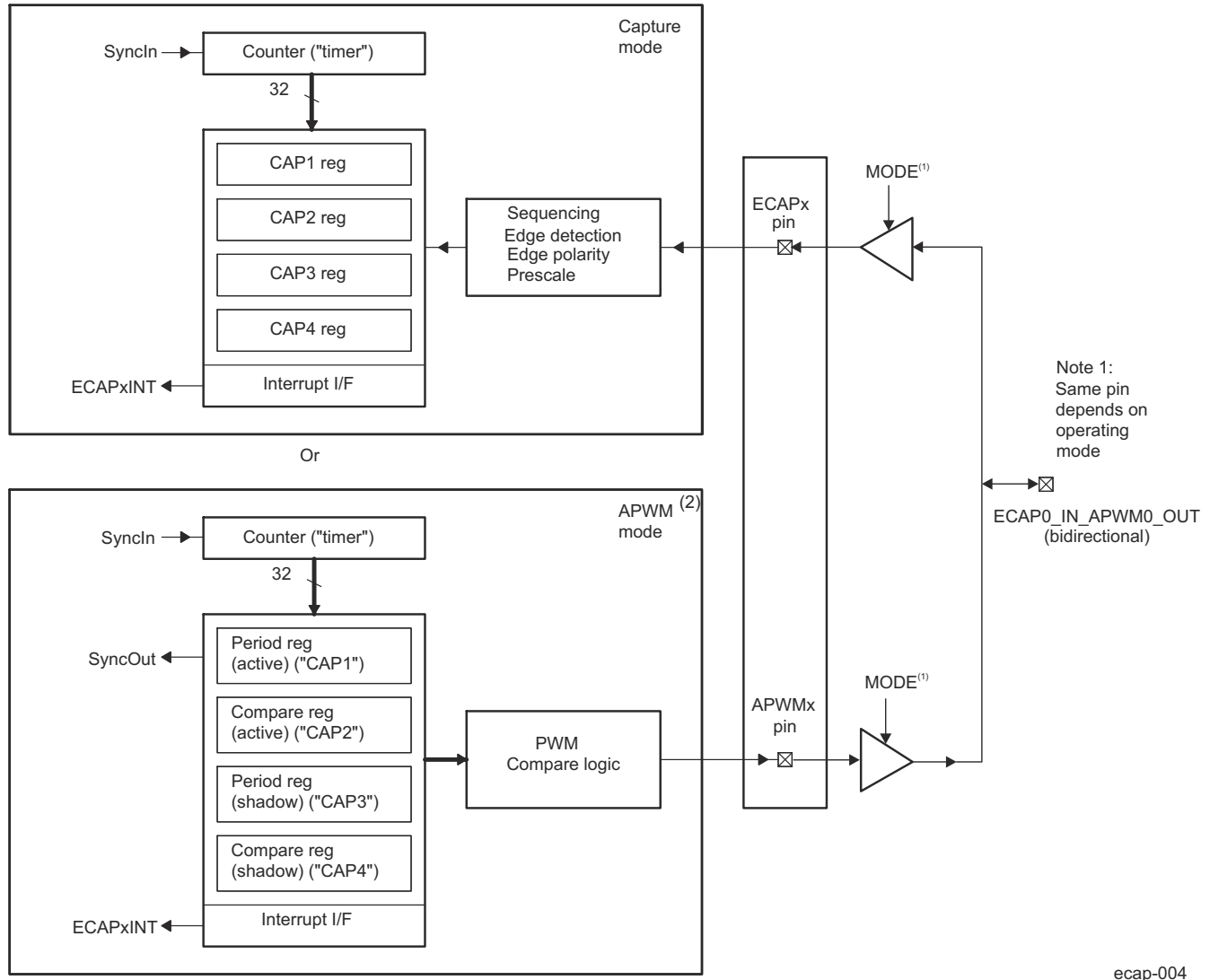
Multiple identical ECAP modules can be contained in a system as shown in [Figure 12-361](#). For actual number of the ECAP modules integrated in the device, refer to the *ECAP Integration*. The letter x within a signal or module name is used to indicate a generic ECAP instance on a device. For example, output interrupt request, ECAP1INT belongs to ECAP1, ECAP2INT belongs to ECAP2, and so forth.



**Figure 12-361. Multiple ECAP Modules**

#### 12.4.1.4.1 Capture and APWM Operating Modes

The ECAP module resources can be used to implement a single-channel PWM generator (with 32-bit capabilities) when it is not being used for input captures. The counter operates in count-up mode, providing a time-base for asymmetrical pulse width modulation (PWM) waveforms. The ECAP\_CAP1 and ECAP\_CAP2 registers become the active period and compare registers, respectively, while the ECAP\_CAP3 and ECAP\_CAP4 registers become the period and capture shadow registers, respectively. [Figure 12-362](#) is a high-level view of both the capture and auxiliary pulse-width modulator (APWM) modes of operation.



ecap-004

- A single pin is shared between CAP and APWM functions. In capture mode, it is an input. In APWM mode, it is an output.
- In APWM mode, writing any value to the ECAP\_CAP1/ECAP\_CAP2 active registers also writes the same value to the corresponding shadow registers (ECAP\_CAP3/ECAP\_CAP4). This emulates immediate mode. Writing to the shadow registers (ECAP\_CAP3/ECAP\_CAP4) invokes the shadow mode.

**Figure 12-362. Capture and APWM Modes of Operation**

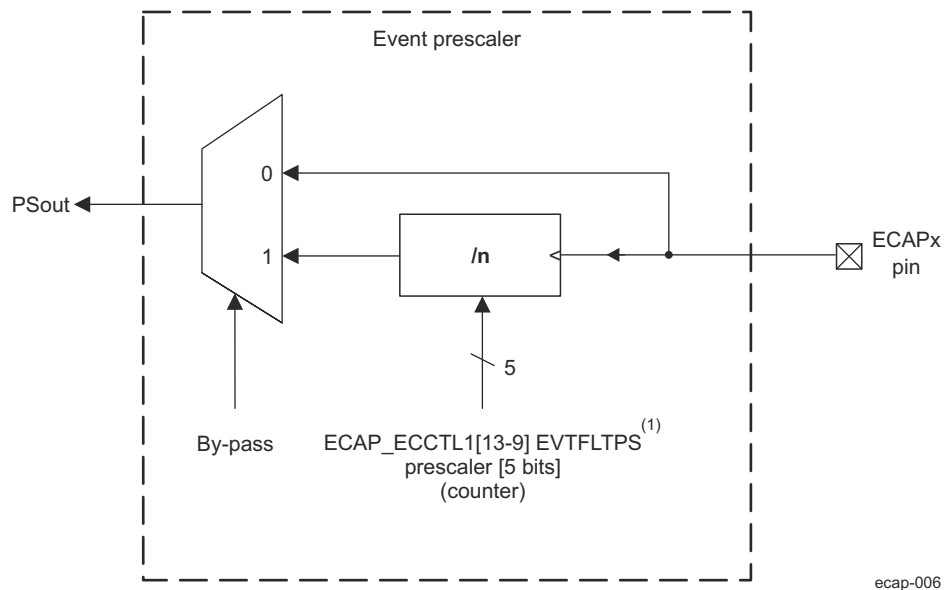
Figure 12-363 shows the various components that implement the capture function.



### Figure 12-363. Capture Function Diagram

An input capture signal (pulse train) can be prescaled by  $N = 2-62$  (in multiples of 2) or can bypass the prescaler. This is useful when very high frequency signals are used as inputs. [Figure 12-364](#) shows a functional diagram and [Figure 12-365](#) shows the operation of the prescale function.

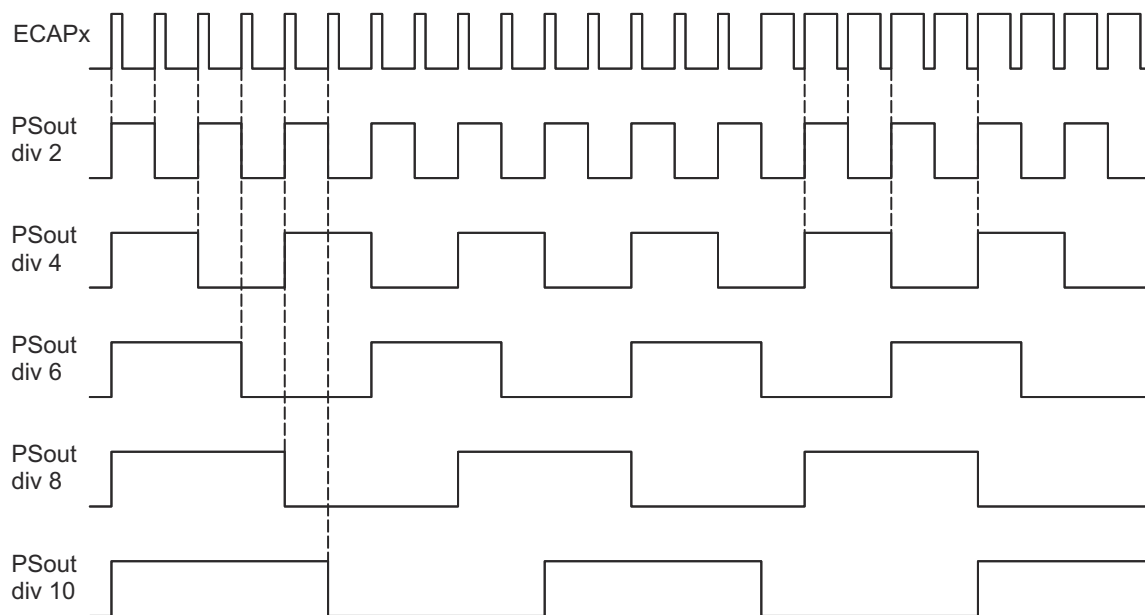




ecap-006

- A. When a prescale value of 1 is chosen (ECAP\_ECCTL[13-9] EVTFLTPS = 0b0000) the input capture signal by-passes the prescale logic completely.

**Figure 12-364. Event Prescale Control**



ecap-007

**Figure 12-365. Prescale Function Waveforms**

#### 12.4.1.4.1.1.2 ECAP Edge Polarity Select and Qualifier

- Four independent edge polarity (rising edge/falling edge) selection multiplexers are used, one for each capture event.
- Each edge (up to 4) is event qualified by the Mod4 sequencer.
- The edge event is gated to its respective CAP $n$  register by the Mod4 counter. The CAP $n$  register is loaded on the falling edge.

#### 12.4.1.4.1.1.3 ECAP Continuous/One-Shot Control

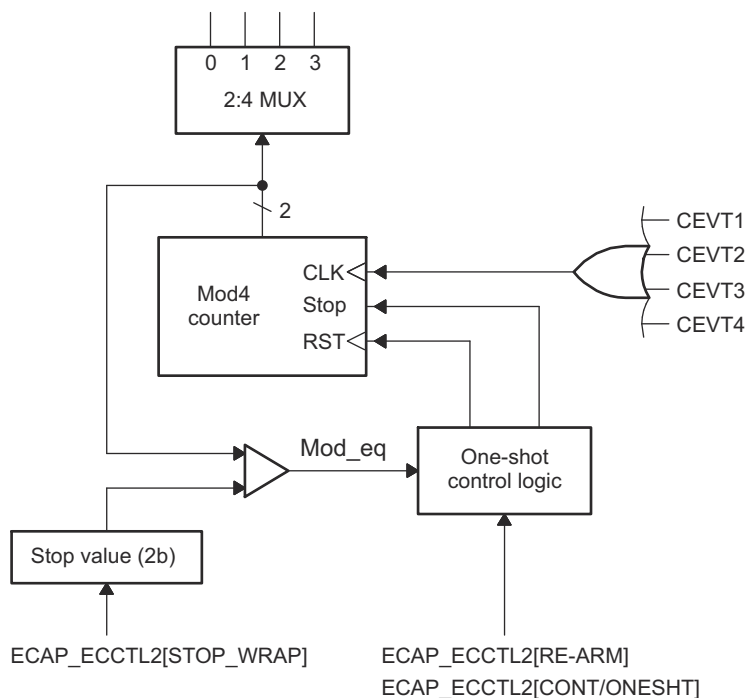
- The Mod4 (2-bit) counter is incremented via edge qualified events CEVT1 through CEVT4 (see the ECAP\_ECINT\_EN\_FLG register).
- The Mod4 counter continues counting (0->1->2->3->0) and wraps around unless stopped.
- A 2-bit stop register is used to compare the Mod4 counter output; when equal the Mod4 counter stops and inhibits further loads of the ECAP\_CAP1 through ECAP\_CAP4 registers. This occurs during one-shot operation.

The continuous/one-shot block (Figure 12-366) controls the start/stop and reset (zero) functions of the Mod4 counter via a mono-shot type of action that can be triggered by the stop-value comparator and re-armed via software control.

Once armed, the ECAP module waits for 1-4 (defined by stop-value) capture events before freezing both the Mod4 counter and contents of the ECAP\_CAP1 through ECAP\_CAP4 registers (time-stamps).

Re-arming prepares the ECAP module for another capture sequence. Also re-arming clears (to zero) the Mod4 counter and permits loading of the ECAP\_CAP1 through ECAP\_CAP4 registers again, providing the ECAP\_ECCTL[8] CAPLDEN bit is set.

In continuous mode, the Mod4 counter continues to run (0->1->2->3->0), the one-shot action is ignored, and capture values continue to be written to the ECAP\_CAP1 through ECAP\_CAP4 registers in a circular buffer sequence.



ecap-008

**Figure 12-366. ECAP Continuous/One-shot Block Diagram**



#### 12.4.1.4.1.1.5 CAP1-CAP4 Registers

These 32-bit registers are fed by the 32-bit counter timer bus, CTR[0-31] and are loaded (capture a time-stamp) when their respective LD inputs are strobed.

Loading of the capture registers can be inhibited via the control ECAP\_ECCTL[8] CAPLDEN bit. During one-shot operation, this bit is cleared (loading is inhibited) automatically when a stop condition occurs, StopValue = Mod4.

The ECAP\_CAP1 and ECAP\_CAP2 registers become the active period and compare registers, respectively, in APWM mode.

The ECAP\_CAP3 and ECAP\_CAP4 registers become the respective shadow registers (APRD and ACMP) for the ECAP\_CAP1 and ECAP\_CAP2 registers during APWM operation.

#### 12.4.1.4.1.1.6 ECAP Interrupt Control

An interrupt can be generated on capture events CEVT1 through CEVT4, CNTOVF (see the ECAP\_ECINT\_EN\_FLG[21] CNTOVF\_FLG bit) or APWM events (TSCNT = PRD, TSCNT = CMP). See [Figure 12-368](#).

A counter overflow event (FFFF FFFFh->0000 0000h) is also provided as an interrupt source (CNTOVF).

The capture events are edge and sequencer qualified (that is, ordered in time) by the polarity select and Mod4 gating, respectively.

One of these events can be selected as the interrupt source (from the ECAP<sub>n</sub> module) going to the interrupt controller.

Seven interrupt events (CEVT1, CEVT2, CEVT3, CEVT4, CNTOVF, TSCNT = PRD, TSCNT = CMP) can be generated. The interrupt enable register (ECAP\_ECINT\_EN\_FLG) is used to enable/disable individual interrupt event sources. The interrupt flag register (ECAP\_ECINT\_EN\_FLG) indicates if any interrupt event has been latched and contains the global interrupt flag ECAP\_ECINT\_EN\_FLG[16] INT\_FLG bit. An interrupt pulse is generated to the interrupt controller only if any of the interrupt events are enabled, the flag bit is 1h, and the INT\_FLG flag bit is 0h. The interrupt service routine must clear the global interrupt flag bit and the serviced event via the interrupt clear register (ECAP\_ECINT\_CLR\_FRC) before any other interrupt pulses are generated. The interrupt force register (ECAP\_ECINT\_CLR\_FRC) can force an interrupt event. This is useful for test purposes.

#### 12.4.1.4.1.1.7 ECAP Shadow Load and Lockout Control

In capture mode, this logic inhibits (locks out) any shadow loading of the ECAP\_CAP1 or ECAP\_CAP2 registers from APRD and ACMP registers, respectively.

In APWM mode, shadow loading is active and two choices are permitted:

- Immediate - APRD or ACMP are transferred to the ECAP\_CAP1 or ECAP\_CAP2 register immediately upon writing a new value.
- On period equal, CTR[31-0] = PRD[31-0]

#### Note

The CEVT1\_FLG, CEVT2\_FLG, CEVT3\_FLG, CEVT4\_FLG flags are only active in capture mode (ECAP\_ECCTL[25] CAP\_APWM == 0h). The TSCNT = PRD, TSCNT = CMP flags are only valid in APWM mode (ECAP\_ECCTL[25] CAP\_APWM == 1h). CNTOVF\_FLG flag is valid in both modes.

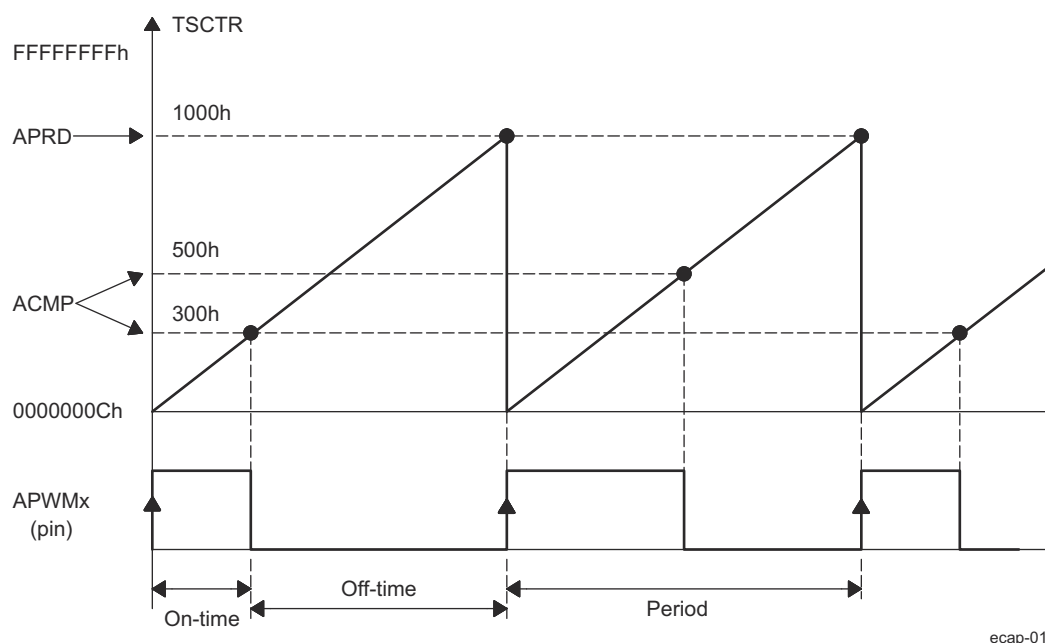


Copyright © 2024 Texas Instruments Incorporated

#### 12.4.1.4.1.2 ECAP APWM Mode Operation

Main operating highlights of the APWM section:

- The time-stamp counter bus is made available for comparison via 2 digital (32-bit) comparators.
- When the ECAP\_CAP1/ECAP\_CAP2 registers are not used in capture mode, their contents can be used as Period and Compare values in APWM mode.
- Double buffering is achieved via the shadow registers - APRD and ACMP (ECAP\_CAP3/ECAP\_CAP4). The shadow register contents are transferred over to ECAP\_CAP1/ECAP\_CAP2 registers either immediately upon a write, or on a TSCNT = PRD trigger.
- In APWM mode, writing to the ECAP\_CAP1/ECAP\_CAP2 active registers will also write the same value to the corresponding shadow registers (ECAP\_CAP3/ECAP\_CAP4). This emulates immediate mode. Writing to the shadow registers (ECAP\_CAP3/ECAP\_CAP4) will invoke the shadow mode.
- During initialization, software must write the PRD and CMP values to the active registers. This automatically copies the initial values into the shadow values. For subsequent compare updates, during run-time, software should use only shadow registers.



**Figure 12-369. PWM Waveform Details of ECAP APWM Mode Operation**

The behavior of APWM active-high mode (APWMPOL == 0) is:

CMP = 0x00000000, output low for duration of period (0% duty)

CMP = 0x00000001, output high 1 cycle

CMP = 0x00000002, output high 2 cycles

CMP = PERIOD, output high except for 1 cycle (<100% duty)

CMP = PERIOD+1, output high for complete period (100% duty)

CMP > PERIOD+1, output high for complete period

The behavior of APWM active-low mode (APWMPOL == 1) is:

CMP = 0x00000000, output high for duration of period (0% duty)

CMP = 0x00000001, output low 1 cycle

CMP = 0x00000002, output low 2 cycles

CMP = PERIOD, output low except for 1 cycle (<100% duty)

CMP = PERIOD+1, output low for complete period (100% duty)

CMP > PERIOD+1, output low for complete period

#### 12.4.1.4.2 Summary of ECAP Functional Registers

[Table 12-408](#) shows the ECAP module control and status register set. All 32-bit registers are aligned on even address boundaries and are organized in little-endian mode.

#### Note

In APWM mode, writing to the ECAP\_CAP1/ECAP\_CAP2 active registers also writes the same value to the corresponding shadow registers (ECAP\_CAP3/ECAP\_CAP4). This emulates immediate mode. Writing to the shadow registers (ECAP\_CAP3/ECAP\_CAP4) invokes the shadow mode.

**Table 12-408. ECAP Control and Status Functional Registers**

Offset	Register Name	Description	Size (×16)
0h	ECAP_TSCNT	Time-Stamp Counter Register	2
4h	ECAP_CNTPHS	Counter Phase Offset Value Register	2
8h	ECAP_CAP1	Capture 1 Register	2
Ch	ECAP_CAP2	Capture 2 Register	2
10h	ECAP_CAP3	Capture 3 Register	2
14h	ECAP_CAP4	Capture 4 Register	2
28h	ECAP_ECCTL	Capture Control Register	2
2Ch	ECAP_ECINT_EN_FLG	Capture Interrupt Enable and Flag Register	2
30h	ECAP_ECINT_CLR_FRC	Capture Interrupt Clear and Forcing Register	2
5Ch	ECAP_PID	Revision ID Register	2

#### Note

For more information on the ECAP registers, see *ECAP Registers*.

### 12.4.1.5 ECAP Use Cases

The following sections will provide applications examples and code snippets to show how to configure and operate the ECAP module. For clarity and ease of use, below are useful #defines which will help in the understanding of the examples.

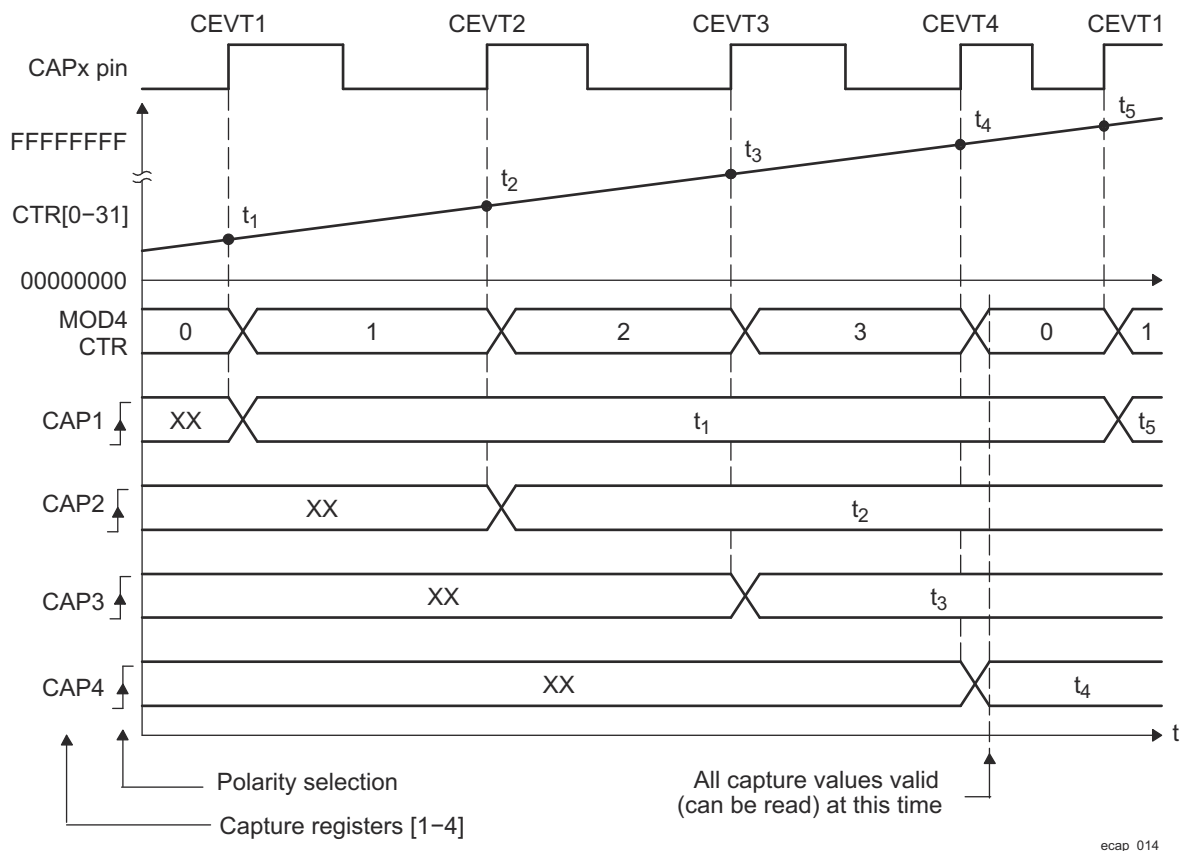
```
// ECCTL1 ( ECAP Control Reg 1)
//=====
// CAPXPOL bits
#define EC_RISING           0x0
#define EC_FALLING         0x1
// CTRRSTx bits
#define EC_ABS_MODE        0x0
#define EC_DELTA_MODE      0x1
// PRESCALE bits
#define EC_BYPASS          0x0
#define EC_DIV1            0x0
#define EC_DIV2            0x1
#define EC_DIV4            0x2
#define EC_DIV6            0x3
#define EC_DIV8            0x4
#define EC_DIV10           0x5
// ECCTL2 ( ECAP Control Reg 2)
//=====
// CONT/ONESHOT bit
#define EC_CONTINUOUS       0x0
#define EC_ONESHOT         0x1
// STOPVALUE bit
#define EC_EVENT1          0x0
#define EC_EVENT2          0x1
#define EC_EVENT3          0x2
#define EC_EVENT4          0x3
// RE-ARM bit
#define EC_ARM              0x1
// TSCTRSTOP bit
#define EC_FREEZE          0x0
#define EC_RUN              0x1
// SYNC0_SEL bit
#define EC_SYNCIN          0x0
#define EC_CTR_PRD         0x1
#define EC_SYNC0_DIS       0x2
// CAP/APWM mode bit
#define EC_CAP_MODE        0x0
#define EC_APWM_MODE       0x1
// APWMPOL bit
#define EC_ACTV_HI         0x0
#define EC_ACTV_LO         0x1
// Generic
#define EC_DISABLE          0x0
#define EC_ENABLE          0x1
#define EC_FORCE            0x1
```



#### 12.4.1.5.1 Absolute Time-Stamp Operation Rising Edge Trigger Example

Figure 12-370 shows an example of continuous capture operation (Mod4 counter wraps around). In this figure, TSCTR counts-up without resetting and capture events are qualified on the rising edge only, this gives period (and frequency) information.

On an event, the TSCTR contents (time-stamp) is first captured, then Mod4 counter is incremented to the next state. When the TSCTR reaches FFFF FFFFh (maximum value), it wraps around to 0000 0000h (not shown in Figure 12-370), if this occurs, the CNTOVF\_FLG (counter overflow) flag is set, and an interrupt (if enabled) occurs. Captured time-stamps are valid at the point indicated by the diagram, after the 4-th event, hence event CEVT4 can conveniently be used to trigger an interrupt and the CPU can read data from the CAP $n$  registers.



**Figure 12-370. Capture Sequence for Absolute Time-Stamp, Rising Edge Detect**

**Table 12-409. ECAP Initialization for CAP Mode Absolute Time, Rising Edge Trigger**

Register	Bit	Value
ECCTL1	CAP1POL	EC_RISING
ECCTL1	CAP2POL	EC_RISING
ECCTL1	CAP3POL	EC_RISING
ECCTL1	CAP4POL	EC_RISING
ECCTL1	CTRRST1	EC_ABS_MODE
ECCTL1	CTRRST2	EC_ABS_MODE
ECCTL1	CTRRST3	EC_ABS_MODE
ECCTL1	CTRRST4	EC_ABS_MODE
ECCTL1	CAPLDEN	EC_ENABLE
ECCTL1	PRESCALE	EC_DIV1
ECCTL2	CAP_APWM	EC_CAP_MODE
ECCTL2	CONT_ONESHT	EC_CONTINUOUS
ECCTL2	SYNCO_SEL	EC_SYNCO_DIS
ECCTL2	SYNCl_EN	EC_DISABLE
ECCTL2	TSCTRSTOP	EC_RUN

**Example 12-1. Code Snippet for CAP Mode Absolute Time, Rising Edge Trigger**

```
// Code snippet for CAP mode Absolute Time, Rising edge trigger
// Run Time ( e.g. CEVT4 triggered ISR call)
//=====
TSt1 = ECAPxRegs.CAP1;      // Fetch Time-Stamp captured at t1
TSt2 = ECAPxRegs.CAP2;      // Fetch Time-Stamp captured at t2
TSt3 = ECAPxRegs.CAP3;      // Fetch Time-Stamp captured at t3
TSt4 = ECAPxRegs.CAP4;      // Fetch Time-Stamp captured at t4
Period1 = TSt2-TSt1;         // Calculate 1st period
Period2 = TSt3-TSt2;         // Calculate 2nd period
Period3 = TSt4-TSt3;         // Calculate 3rd period
```

#### 12.4.1.5.2 Absolute Time-Stamp Operation Rising and Falling Edge Trigger Example

In Figure 12-371 the ECAP operating mode is almost the same as in the previous section except capture events are qualified as either rising or falling edge, this now gives both period and duty cycle information: Period1 =  $t_3 - t_1$ , Period2 =  $t_5 - t_3$ , ...etc. Duty Cycle1 (on-time %) =  $(t_2 - t_1) / \text{Period1} \times 100\%$ , etc. Duty Cycle1 (off-time %) =  $(t_3 - t_2) / \text{Period1} \times 100\%$ , etc.

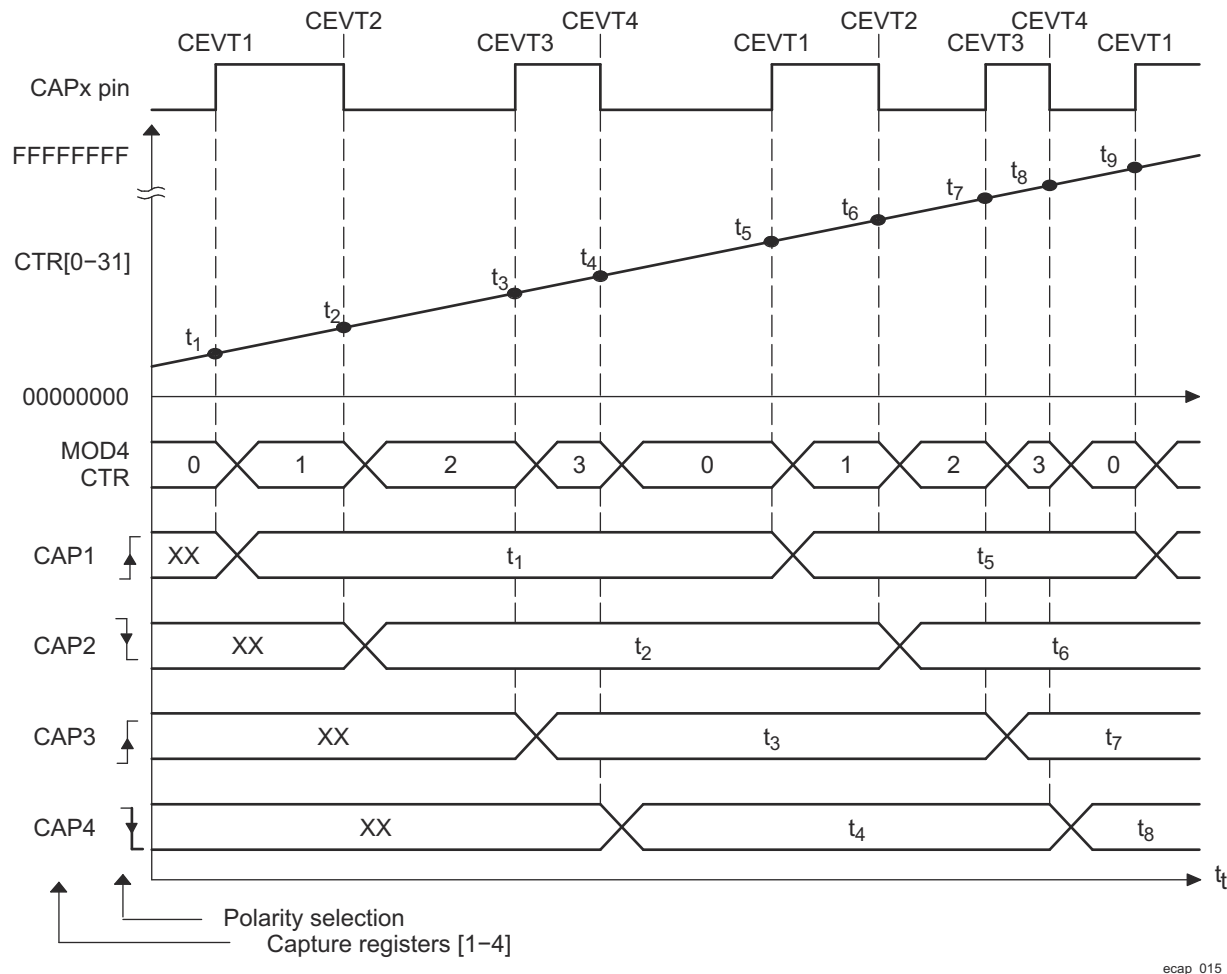


Figure 12-371. Capture Sequence for Absolute Time-Stamp, Rising and Falling Edge Detect

**Table 12-410. ECAP Initialization for CAP Mode Absolute Time, Rising and Falling Edge Trigger**

Register	Bit	Value
ECCTL1	CAP1POL	EC_RISING
ECCTL1	CAP2POL	EC_FALLING
ECCTL1	CAP3POL	EC_RISING
ECCTL1	CAP4POL	EC_FALLING
ECCTL1	CTRRST1	EC_ABS_MODE
ECCTL1	CTRRST2	EC_ABS_MODE
ECCTL1	CTRRST3	EC_ABS_MODE
ECCTL1	CTRRST4	EC_ABS_MODE
ECCTL1	CAPLDEN	EC_ENABLE
ECCTL1	PRESCALE	EC_DIV1
ECCTL2	CAP_APWM	EC_CAP_MODE
ECCTL2	CONT_ONESHT	EC_CONTINUOUS
ECCTL2	SYNCO_SEL	EC_SYNCO_DIS
ECCTL2	SYNCl_EN	EC_DISABLE
ECCTL2	TSCTRSTOP	EC_RUN

**Example 12-2. Code Snippet for CAP Mode Absolute Time, Rising and Falling Edge Trigger**

```
// Code snippet for CAP mode Absolute Time, Rising & Falling edge triggers
// Run Time ( e.g. CEVT4 triggered ISR call)
//=====
TSt1 = ECAPxRegs.CAP1;    // Fetch Time-Stamp captured at t1
TSt2 = ECAPxRegs.CAP2;    // Fetch Time-Stamp captured at t2
TSt3 = ECAPxRegs.CAP3;    // Fetch Time-Stamp captured at t3
TSt4 = ECAPxRegs.CAP4;    // Fetch Time-Stamp captured at t4
Period1 = TSt3-TSt1;      // Calculate 1st period
DutyOnTime1 = TSt2-TSt1;   // Calculate On time
DutyOffTime1 = TSt3-TSt2;  // Calculate Off time
```

#### 12.4.1.5.3 Time Difference (Delta) Operation Rising Edge Trigger Example

Figure 12-372 shows how the ECAP module can be used to collect Delta timing data from pulse train waveforms. Here Continuous Capture mode (TSCTR counts-up without resetting, and Mod4 counter wraps around) is used. In Delta-time mode, TSCTR is Reset back to Zero on every valid event. Here Capture events are qualified as Rising edge only. On an event, TSCTR contents (time-stamp) is captured first, and then TSCTR is reset to Zero. The Mod4 counter then increments to the next state. If TSCTR reaches FFFF FFFFh (maximum value), before the next event, it wraps around to 0000 0000h and continues, a CNTOVF\_FLG (counter overflow) Flag is set, and an Interrupt (if enabled) occurs. The advantage of Delta-time Mode is that the CAP<sub>n</sub> contents directly give timing data without the need for CPU calculations: Period1 = T<sub>1</sub>, Period2 = T<sub>2</sub>,...etc. As shown in Figure 12-372, the CEVT1 event is a good trigger point to read the timing data, T<sub>1</sub>, T<sub>2</sub>, T<sub>3</sub>, T<sub>4</sub> are all valid here.

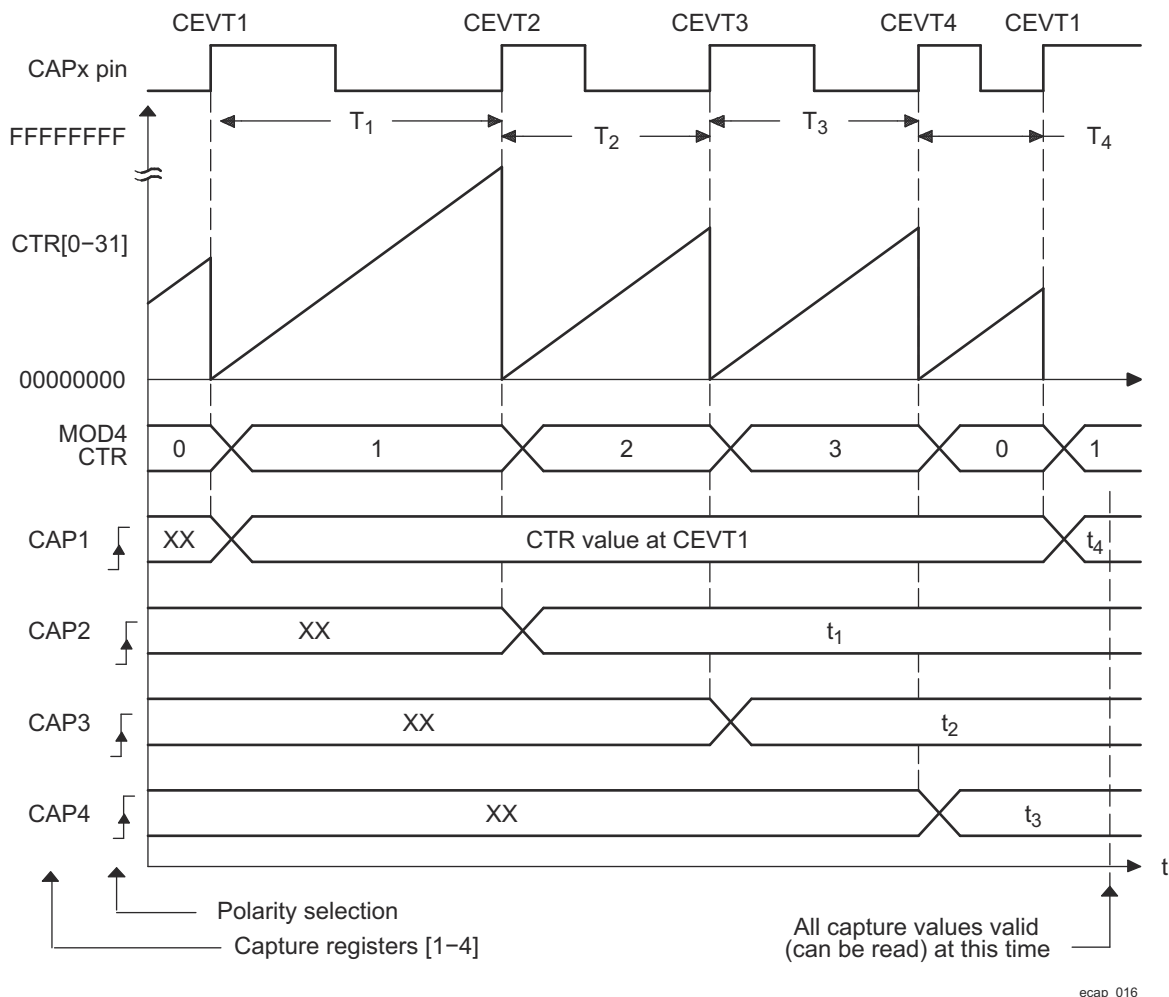


Figure 12-372. Capture Sequence for Delta Mode Time-Stamp, Rising Edge Detect

**Table 12-411. ECAP Initialization for CAP Mode Delta Time, Rising Edge Trigger**

Register	Bit	Value
ECCTL1	CAP1POL	EC_RISING
ECCTL1	CAP2POL	EC_RISING
ECCTL1	CAP3POL	EC_RISING
ECCTL1	CAP4POL	EC_RISING
ECCTL1	CTRRST1	EC_DELTA_MODE
ECCTL1	CTRRST2	EC_DELTA_MODE
ECCTL1	CTRRST3	EC_DELTA_MODE
ECCTL1	CTRRST4	EC_DELTA_MODE
ECCTL1	CAPLDEN	EC_ENABLE
ECCTL1	PRESCALE	EC_DIV1
ECCTL2	CAP_APWM	EC_CAP_MODE
ECCTL2	CONT_ONESHT	EC_CONTINUOUS
ECCTL2	SYNCO_SEL	EC_SYNCO_DIS
ECCTL2	SYNCl_EN	EC_DISABLE
ECCTL2	TSCTRSTOP	EC_RUN

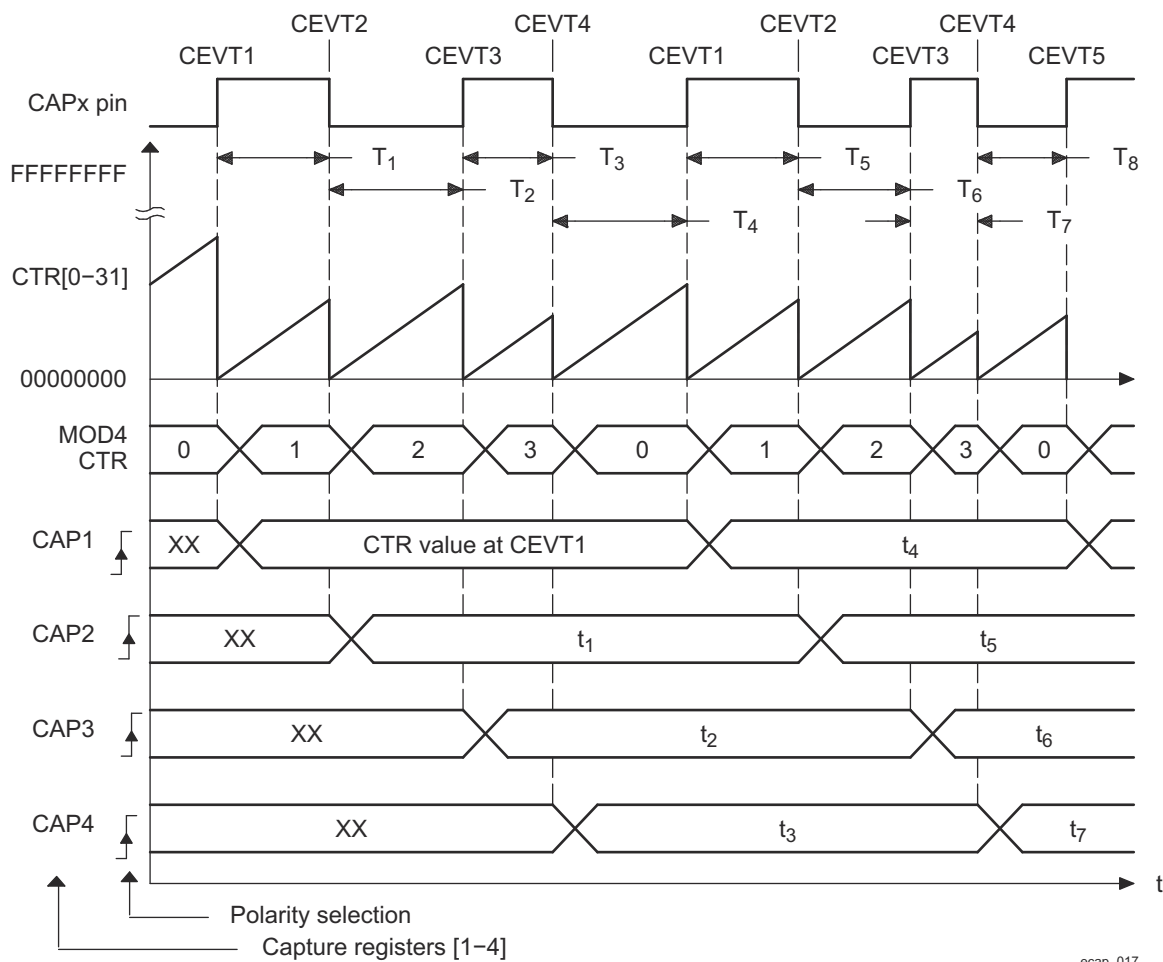
**Example 12-3. Code Snippet for CAP Mode Delta Time, Rising Edge Trigger**

```
// Code snippet for CAP mode Delta Time, Rising edge trigger
// Run Time ( e.g. CEVT1 triggered ISR call)
//=====
// Note: here Time-stamp directly represents the Period value.
Period4 = ECAPxRegs.CAP1;    // Fetch Time-Stamp captured at T1
Period1 = ECAPxRegs.CAP2;    // Fetch Time-Stamp captured at T2
Period2 = ECAPxRegs.CAP3;    // Fetch Time-Stamp captured at T3
Period3 = ECAPxRegs.CAP4;    // Fetch Time-Stamp captured at T4
```

#### 12.4.1.5.4 Time Difference (Delta) Operation Rising and Falling Edge Trigger Example

In Figure 12-373 the ECAP operating mode is almost the same as in previous section except Capture events are qualified as either Rising or Falling edge, this now gives both Period and Duty cycle information: Period1 =  $T_1 + T_2$ , Period2 =  $T_3 + T_4$ , ...etc Duty Cycle1 (on-time %) =  $T_1 / \text{Period1} \times 100\%$ , etc Duty Cycle1 (off-time %) =  $T_2 / \text{Period1} \times 100\%$ , etc.

During initialization, you must write to the active registers for both period and compare. This will then automatically copy the init values into the shadow values. For subsequent compare updates, that is, during run-time, only the shadow registers must be used.



**Figure 12-373. Capture Sequence for Delta Mode Time-Stamp, Rising and Falling Edge Detect**

**Table 12-412. ECAP Initialization for CAP Mode Delta Time, Rising and Falling Edge Triggers**

Register	Bit	Value
ECCTL1	CAP1POL	EC_RISING
ECCTL1	CAP2POL	EC_FALLING
ECCTL1	CAP3POL	EC_RISING
ECCTL1	CAP4POL	EC_FALLING
ECCTL1	CTRRST1	EC_DELTA_MODE
ECCTL1	CTRRST2	EC_DELTA_MODE
ECCTL1	CTRRST3	EC_DELTA_MODE
ECCTL1	CTRRST4	EC_DELTA_MODE
ECCTL1	CAPLDEN	EC_ENABLE
ECCTL1	PRESCALE	EC_DIV1
ECCTL2	CAP_APWM	EC_CAP_MODE
ECCTL2	CONT_ONESHT	EC_CONTINUOUS
ECCTL2	SYNCO_SEL	EC_SYNCO_DIS
ECCTL2	SYNCl_EN	EC_DISABLE
ECCTL2	TSCTRSTOP	EC_RUN

**Example 12-4. Code Snippet for CAP Mode Delta Time, Rising and Falling Edge Triggers**

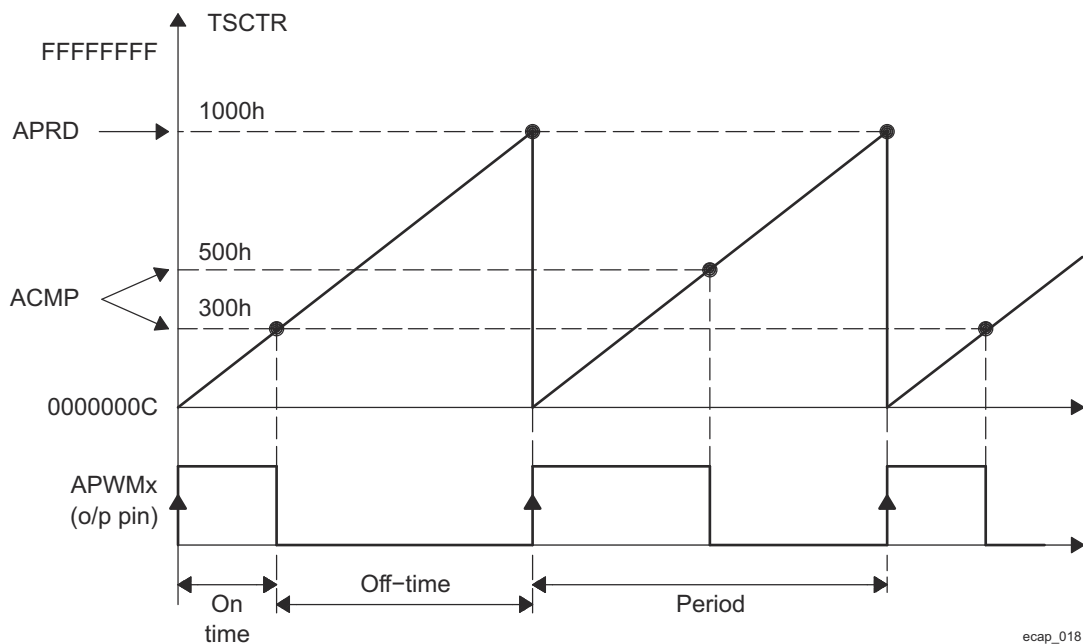
```
// Code snippet for CAP mode Delta Time, Rising and Falling edge triggers
// Run Time ( e.g. CEVT1 triggered ISR call)
//=====
// Note: here Time-stamp directly represents the Duty cycle values.
DutyOnTime1 = ECAPxRegs.CAP2;    // Fetch Time-Stamp captured at T2
DutyOffTime1 = ECAPxRegs.CAP3;    // Fetch Time-Stamp captured at T3
DutyOnTime2 = ECAPxRegs.CAP4;    // Fetch Time-Stamp captured at T4
DutyOffTime2 = ECAPxRegs.CAP1;    // Fetch Time-Stamp captured at T1
Period1 = DutyOnTime1 + DutyOffTime1;
Period2 = DutyOnTime2 + DutyOffTime2;
```



### 12.4.1.5.5 Application of the APWM Mode

#### 12.4.1.5.5.1 Simple PWM Generation (Independent Channel/s) Example

In this example, the ECAP module is configured to operate as a PWM generator. Here a very simple single channel PWM waveform is generated from output pin APWM $n$ . The PWM polarity is active high, which means that the compare value (CAP2 reg is now a compare register) represents the on-time (high level) of the period. Alternatively, if the APWMPOL bit is configured for active low, then the compare value represents the off-time.



**Figure 12-374. PWM Waveform Details of APWM Mode Operation**

**Table 12-413. ECAP Initialization for APWM Mode**

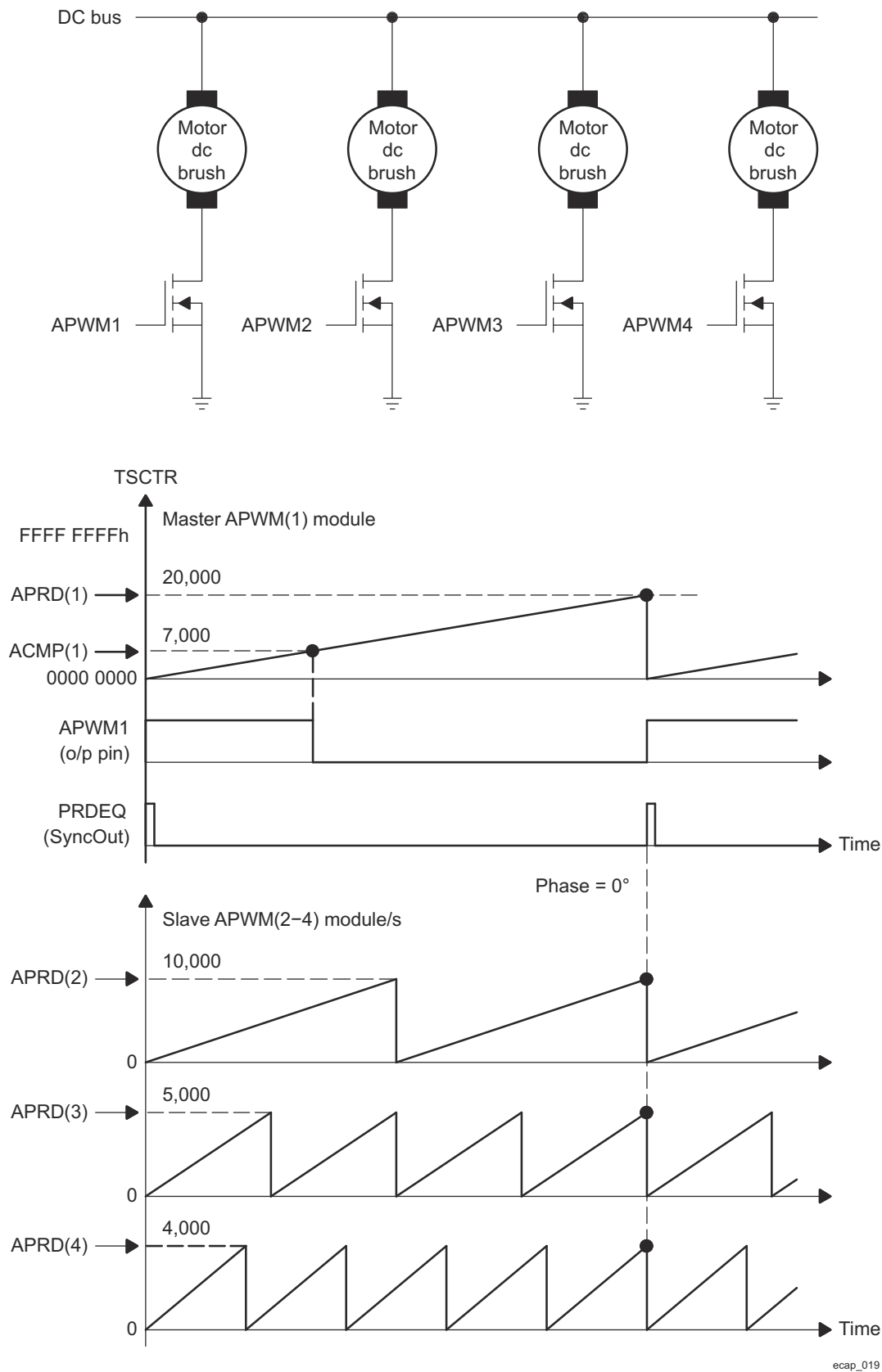
Register	Bit	Value
CAP1	CAP1	0x1000
CTRPHS	CTRPHS	0x0
ECCTL2	CAP_APWM	EC_APWM_MODE
ECCTL2	APWMPOL	EC_ACTV_HI
ECCTL2	SYNCL_EN	EC_DISABLE
ECCTL2	SYNCO_SEL	EC_SYNCO_DIS
ECCTL2	TSCTRSTOP	EC_RUN

**Example 12-5. Code Snippet for APWM Mode**

```
// Code snippet for APWM mode Example 1
// Run Time (Instant 1, e.g. ISR call)
//=====
ECAPxRegs.CAP2 = 0x300;      // Set Duty cycle i.e. compare value
// Run Time (Instant 2, e.g. another ISR call)
//=====
ECAPxRegs.CAP2 = 0x500;      // Set Duty cycle i.e. compare value
```

**12.4.1.5.5.2 Multichannel PWM Generation with Synchronization Example**

Figure 12-375 takes advantage of the synchronization feature between the ECAP modules. Here 4 independent PWM channels are required with different frequencies, but at integer multiples of each other to avoid "beat" frequencies. Hence one ECAP module is configured as the Master and the remaining 3 are Slaves all receiving their synch pulse (CTR = PRD) from the master. Note the Master is chosen to have the lower frequency ( $F1 = 1/20,000$ ) requirement. Here Slave2 Freq =  $2 \times F1$ , Slave3 Freq =  $4 \times F1$  and Slave4 Freq =  $5 \times F1$ . Note here values are in decimal notation. Also, only the APWM1 output waveform is shown.



**Figure 12-375. Multichannel PWM Example Using 4 ECAP Modules**

**Table 12-414. ECAP1 Initialization for Multichannel PWM Generation with Synchronization**

Register	Bit	Value
CAP1	CAP1	20000
CTRPHS	CTRPHS	0
ECCTL2	CAP_APWM	EC_APWM_MODE
ECCTL2	APWMPOL	EC_ACTV_HI
ECCTL2	SYNCl_EN	EC_DISABLE
ECCTL2	SYNCO_SEL	EC_CTR_PRD
ECCTL2	TSCTRSTOP	EC_RUN

**Table 12-415. ECAP2 Initialization for Multichannel PWM Generation with Synchronization**

Register	Bit	Value
CAP1	CAP1	10000
CTRPHS	CTRPHS	0
ECCTL2	CAP_APWM	EC_APWM_MODE
ECCTL2	APWMPOL	EC_ACTV_HI
ECCTL2	SYNCl_EN	EC_ENABLE
ECCTL2	SYNCO_SEL	EC_SYNCl
ECCTL2	TSCTRSTOP	EC_RUN

**Table 12-416. ECAP3 Initialization for Multichannel PWM Generation with Synchronization**

Register	Bit	Value
CAP1	CAP1	5000
CTRPHS	CTRPHS	0
ECCTL2	CAP_APWM	EC_APWM_MODE
ECCTL2	APWMPOL	EC_ACTV_HI
ECCTL2	SYNCl_EN	EC_ENABLE
ECCTL2	SYNCO_SEL	EC_SYNCl
ECCTL2	TSCTRSTOP	EC_RUN

**Table 12-417. ECAP4 Initialization for Multichannel PWM Generation with Synchronization**

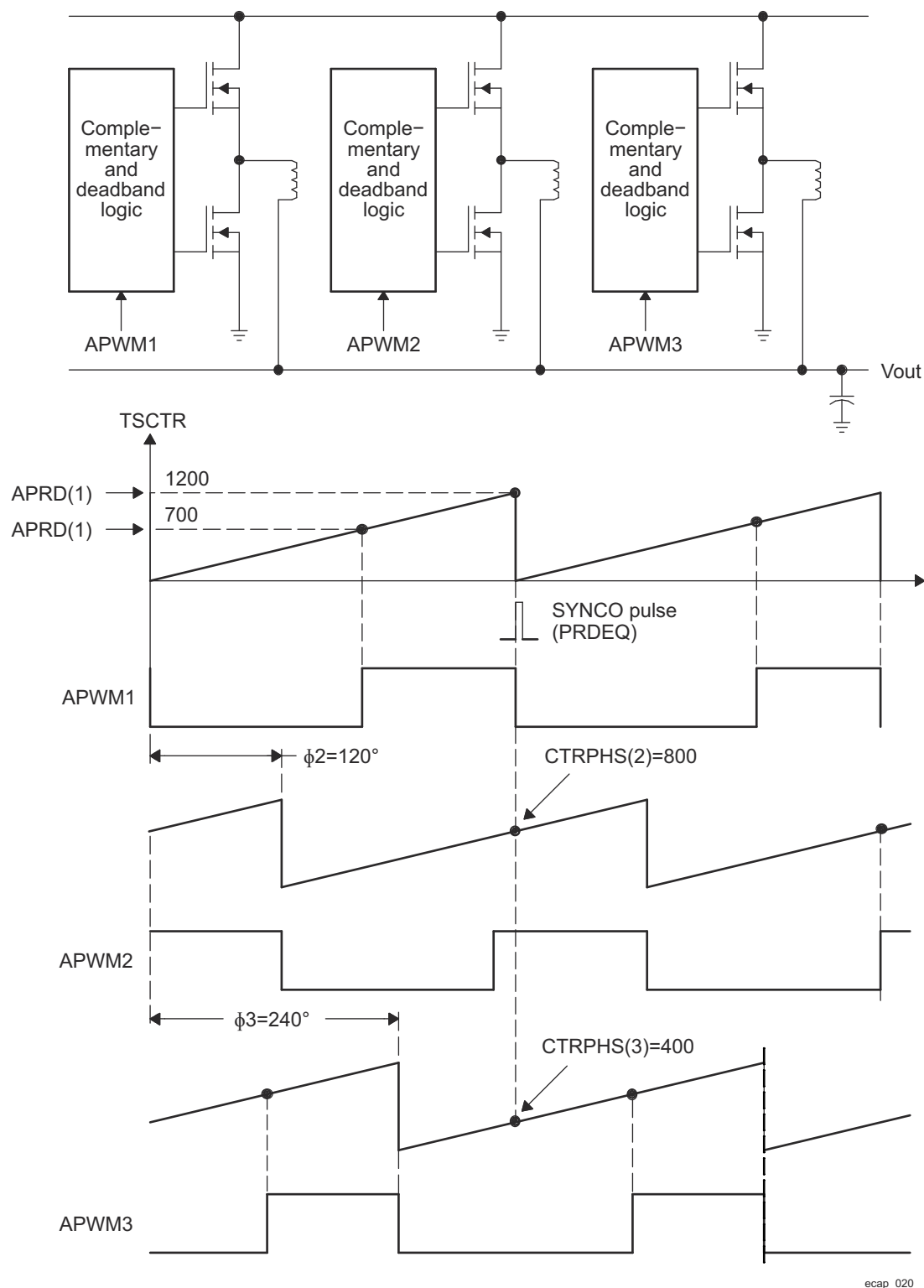
Register	Bit	Value
CAP1	CAP1	4000
CTRPHS	CTRPHS	0
ECCTL2	CAP_APWM	EC_APWM_MODE
ECCTL2	APWMPOL	EC_ACTV_HI
ECCTL2	SYNCl_EN	EC_ENABLE
ECCTL2	SYNCO_SEL	EC_SYNCO_DIS
ECCTL2	TSCTRSTOP	EC_RUN

**Example 12-6. Code Snippet for Multichannel PWM Generation with Synchronization**

```
// Code snippet for APWM mode Example 2
// Run Time (Note: Example execution of one run-time instant)
//=====
ECAP1Regs.CAP2 = 7000;    // Set Duty cycle i.e., compare value = 7000
ECAP2Regs.CAP2 = 2000;    // Set Duty cycle i.e., compare value = 2000
ECAP3Regs.CAP2 = 550;     // Set Duty cycle i.e., compare value = 550
ECAP4Regs.CAP2 = 6500;    // Set Duty cycle i.e., compare value = 6500
```

#### 12.4.1.5.5.3 Multichannel PWM Generation with Phase Control Example

In [Figure 12-376](#), the Phase control feature of the APWM mode is used to control a 3 phase Interleaved DC/DC converter topology. This topology requires each phase to be off-set by  $120^\circ$  from each other. Hence if "Leg" 1 (controlled by APWM1) is the reference Leg (or phase), that is,  $0^\circ$ , then Leg 2 need  $120^\circ$  off-set and Leg 3 needs  $240^\circ$  off-set. The waveforms in [Figure 12-376](#) show the timing relationship between each of the phases (Legs). Note ECAP1 module is the Master and issues a SyncOut pulse to the slaves (modules 2, 3) whenever  $TSCTR = \text{Period value}$ .



**Figure 12-376. Multiphase (channel) Interleaved PWM Example Using 3 ECAP Modules**

**Table 12-418. ECAP1 Initialization for Multichannel PWM Generation with Phase Control**

Register	Bit	Value
CAP1	CAP1	1200
CTRPHS	CTRPHS	0
ECCTL2	CAP_APWM	EC_APWM_MODE
ECCTL2	APWMPOL	EC_ACTV_HI
ECCTL2	SYNCl_EN	EC_DISABLE
ECCTL2	SYNCO_SEL	EC_CTR_PRD
ECCTL2	TSCTRSTOP	EC_RUN

**Table 12-419. ECAP2 Initialization for Multichannel PWM Generation with Phase Control**

Register	Bit	Value
CAP1	CAP1	1200
CTRPHS	CTRPHS	800
ECCTL2	CAP_APWM	EC_APWM_MODE
ECCTL2	APWMPOL	EC_ACTV_HI
ECCTL2	SYNCl_EN	EC_ENABLE
ECCTL2	SYNCO_SEL	EC_SYNCl
ECCTL2	TSCTRSTOP	EC_RUN

**Table 12-420. ECAP3 Initialization for Multichannel PWM Generation with Phase Control**

Register	Bit	Value
CAP1	CAP1	1200
CTRPHS	CTRPHS	400
ECCTL2	CAP_APWM	EC_APWM_MODE
ECCTL2	APWMPOL	EC_ACTV_HI
ECCTL2	SYNCl_EN	EC_ENABLE
ECCTL2	SYNCO_SEL	EC_SYNCO_DIS
ECCTL2	TSCTRSTOP	EC_RUN

**Example 12-7. Code Snippet for Multichannel PWM Generation with Phase Control**

```
// Code snippet for APWM mode Example 3
// Run Time (Note: Example execution of one run-time instant)
//=====
// All phases are set to the same duty cycle
ECAP1Regs.CAP2 = 700;    // Set Duty cycle i.e. compare value = 700
ECAP2Regs.CAP2 = 700;    // Set Duty cycle i.e. compare value = 700
ECAP3Regs.CAP2 = 700;    // Set Duty cycle i.e. compare value = 700
```

**12.4.2 Enhanced Pulse Width Modulation (EPWM) Module**

This chapter describes the Enhanced PWM (EPWM) module in the device.

**12.4.2.1 EPWM Overview**

An effective PWM peripheral must be able to generate complex pulse width waveforms with minimal CPU overhead or intervention. It needs to be highly programmable and very flexible while being easy to understand and use. The EPWM unit described here addresses these requirements by allocating all needed timing and control resources on a per PWM channel basis. Cross coupling or sharing of resources has been avoided; instead, the EPWM is built up from smaller single channel modules with separate resources and that can operate together as required to form a system. This modular approach results in an orthogonal architecture and provides a more transparent view of the peripheral structure, helping users to understand its operation quickly.

In the further description the letter x within a signal or module name is used to indicate a generic EPWM instance on a device. For example, output signals EPWMxA and EPWMxB refer to the output signals from the EPWMx instance. Thus, EPWM1A and EPWM1B belong to EPWM1, EPWM2A and EPWM2B belong to EPWM2, and so forth.

The EPWM module represents one complete PWM channel composed of two PWM outputs: EPWMxA and EPWMxB. A given EPWM module functionality can be extended with the so called **High-Resolution Pulse Width Modulator**. Refer to the [Section 12.4.2.3](#), to determine which EPWM instances include the HRPWM feature. The HRPWM functionalities are described in [Section 12.4.2.4.9](#).

As also described in [Section 12.4.2.3.2](#), the EPWM modules are chained together via a clock synchronization scheme that allows them to operate as a single system when required. Additionally, the EPWM integration allows this synchronization scheme to be extended to the capture peripheral modules (ECAP). The number of modules is device-dependent and based on target application needs. Modules can also operate stand-alone.

The device has six instances of the EPWM modules.

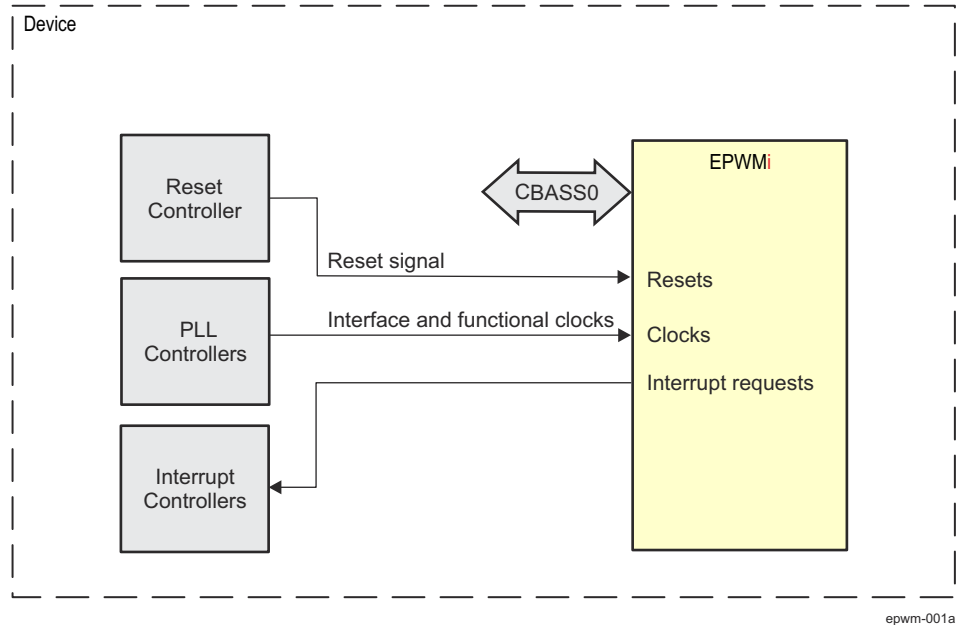
[Table 12-421](#) shows the EPWM allocation across device domains.

**Table 12-421. EPWM Allocation Across Device Domains**

Instance	Domain		
	WKUP	MCU	MAIN
EPWM0	-	-	✓
EPWM1	-	-	✓
EPWM2	-	-	✓
EPWM3	-	-	✓
EPWM4	-	-	✓
EPWM5	-	-	✓

[Figure 12-377](#) shows the EPWM module overview.





A.  $i = 0 \text{ to } 5$

**Figure 12-377. EPWM Overview**

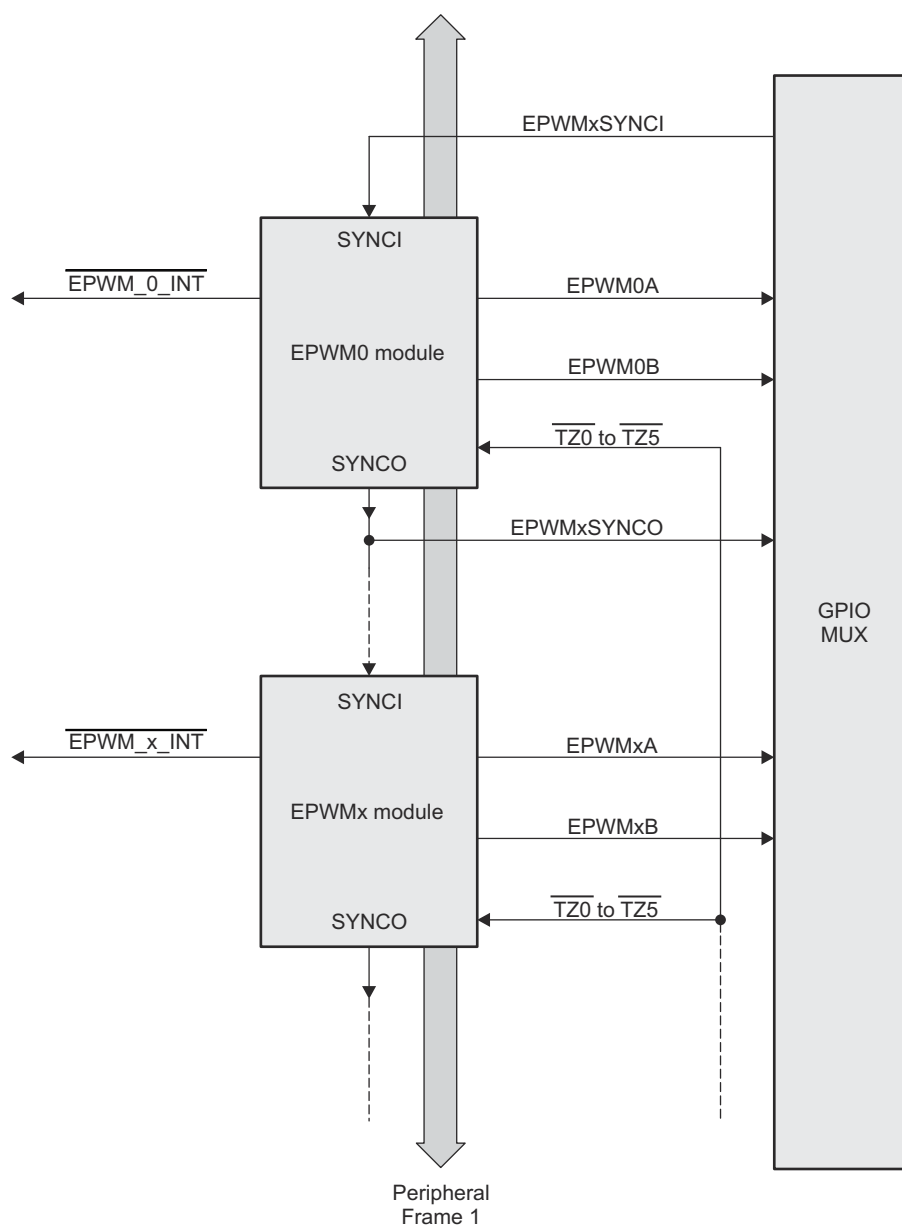
#### 12.4.2.1.1 EPWM Features

Each EPWM module supports the following features:

- Dedicated 16-bit time-base counter with period and frequency control
- Two PWM outputs (EPWMxA and EPWMxB) that can be used in the following configurations:
  - Two independent PWM outputs with single-edge operation
  - Two independent PWM outputs with dual-edge symmetric operation
  - One independent PWM output with dual-edge asymmetric operation
- Asynchronous override control of PWM signals through software
- Programmable phase-control support for lag or lead operation relative to other EPWM modules
- Hardware-locked (synchronized) phase relationship on a cycle-by-cycle basis
- Dead-band generation with independent rising and falling edge delay control
- Programmable trip zone allocation of both cycle-by-cycle trip and one-shot trip on fault conditions
- A trip condition can force either high, low, or high-impedance state logic levels at PWM outputs
- Allows events to trigger both CPU interrupts and ADC start of conversions
- Programmable event prescaling minimizes CPU overhead on interrupts
- PWM chopping by high-frequency carrier signal, useful for pulse transformer gate drives
- High-resolution module with programmable delay line:
  - Programmable on a per PWM period basis
  - Can be inserted either on the rising edge or falling edge of the PWM pulse or both or not at all.

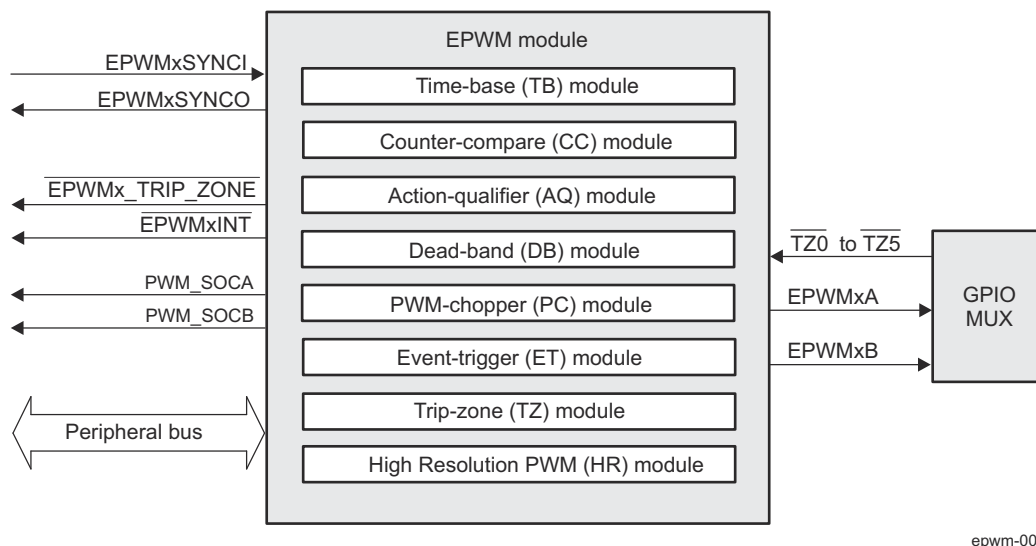
Each EPWM module is connected to the input/output signals shown in [Figure 12-378](#). The signals are described in detail in subsequent sections.

The order in which the EPWM modules are connected may differ from what is shown in [Figure 12-378](#). See *Daisy-Chain Connectivity between EPWM Modules* for the actual synchronization scheme implemented in the device. Each EPWM module consists of eight submodules and is connected within a system via the signals shown in [Figure 12-379](#).



epwm-001

**Figure 12-378. Multiple EPWM Modules**

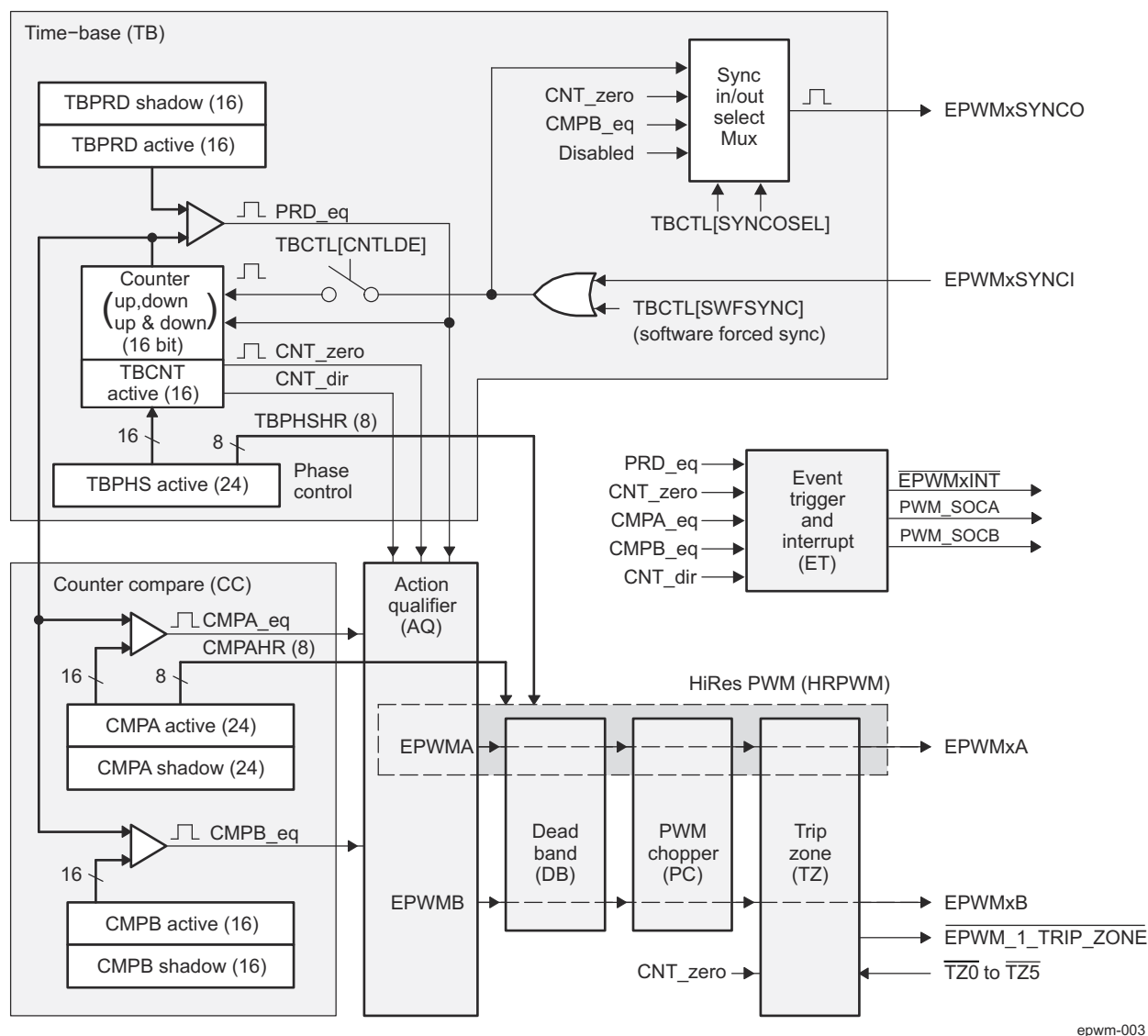


**Figure 12-379. Submodules and Signal Connections for an EPWM Module**

Figure 12-380 shows more internal details of a single EPWM module. The main signals used by the EPWM module are:

- **PWM output signals (EPWMxA and EPWMxB).** The PWM output signals are made available external to the device through the GPIO peripheral described in the system control and interrupts guide for the device.
- **Trip-zone signals (TZ0 to TZ5).** These input signals alert the EPWM module of an external fault condition. Each module on a device can be configured to either use or ignore any of the trip-zone signals. The trip-zone signal can be configured as an asynchronous input through the GPIO peripheral.
- **Time-base synchronization input (EPWMxSYNCl) and output (EPWMxSYNCO) signals.** The synchronization signals daisy chain the EPWM modules together. Each module can be configured to either use or ignore its synchronization input. The clock synchronization input and output signal are brought out to pins for EPWM0 (EPWM module 0) and EPWM3. The EPWM5 synchronization output (EPWM5SYNCO) is also connected to the input SYNCIN of the Enhanced Capture Module (ECAP0).
- **ADC start-of-conversion signals (EPWMxSOCA and EPWMxSOCB).** Each EPWM module has two ADC start of conversion signals (one for each sequencer). Any EPWM module can trigger a start of conversion for either sequencer. Which event triggers the start of conversion is configured in the Event-Trigger submodule of the EPWM.
- **Peripheral Bus.** The peripheral bus is 32-bits wide and allows both 16-bit and 32-bit writes to the EPWM register file.

Figure 12-380 also shows the key internal submodule interconnect signals. Each submodule is described in Section 12.4.2.4.



**Figure 12-380. EPWM Submodules and Critical Internal Signal Interconnects**

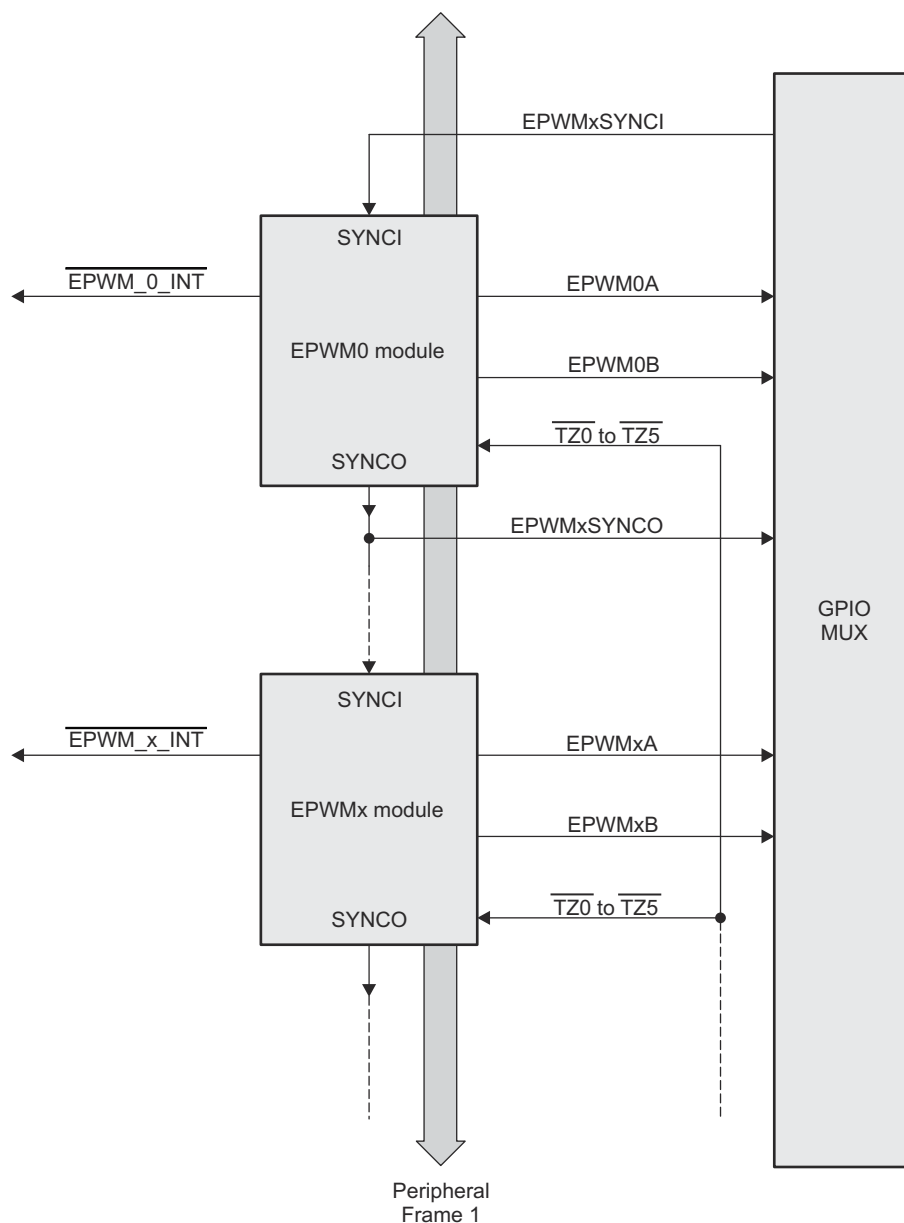
#### 12.4.2.1.2 EPWM Not Supported Features

- EPWM digital comparator modules are not supported

#### 12.4.2.1.3

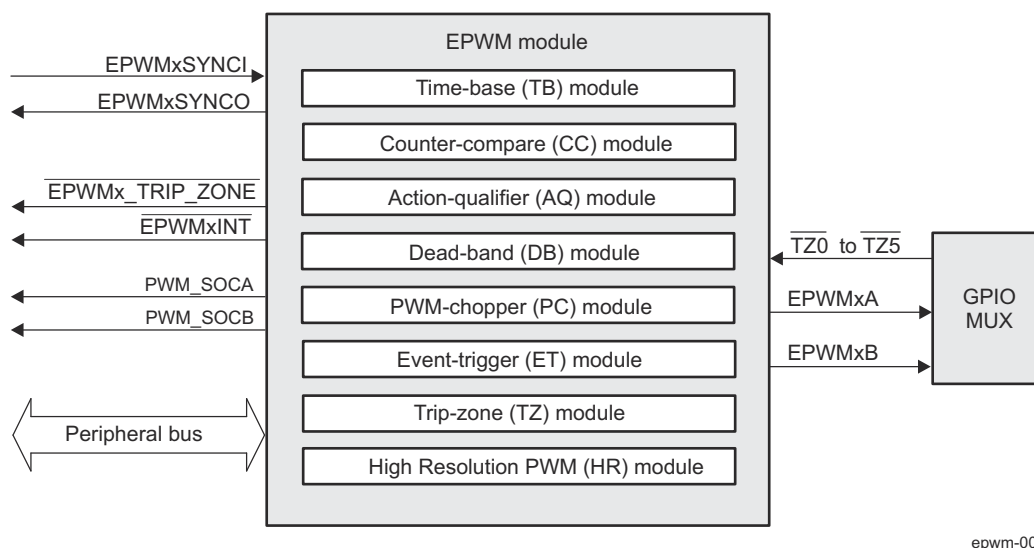
Each EPWM module is connected to the input/output signals shown in [Figure 12-381](#). The signals are described in detail in subsequent sections.

The order in which the EPWM modules are connected may differ from what is shown in [Figure 12-381](#). See [Section 12.4.2.3.2](#) for the actual synchronization scheme implemented in the device. Each EPWM module consists of eight submodules and is connected within a system via the signals shown in [Figure 12-382](#).



### Figure 12-381. Multiple EPWM Modules

epwm-001



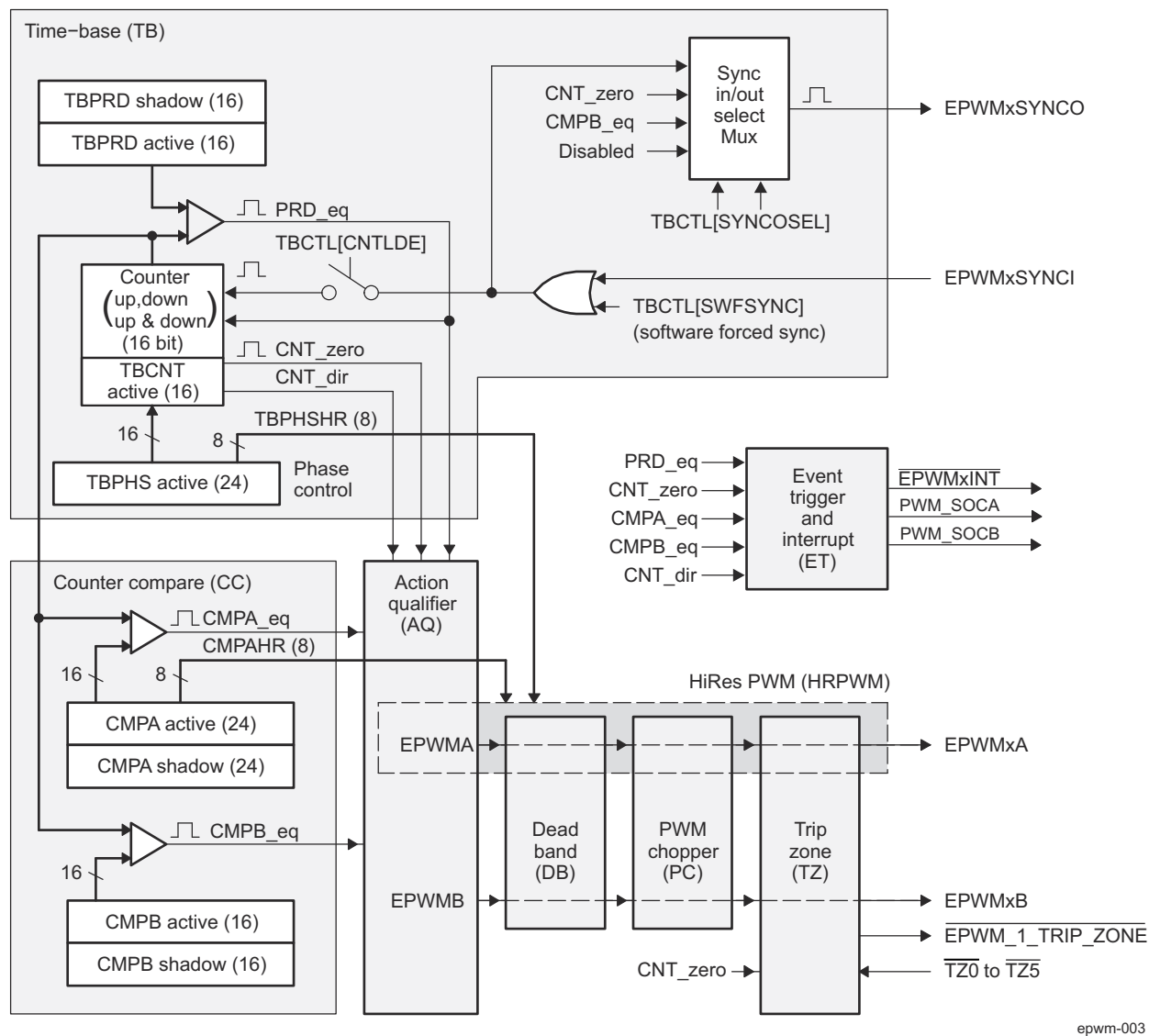
epwm-002

**Figure 12-382. Submodules and Signal Connections for an EPWM Module**

Figure 12-383 shows more internal details of a single EPWM module. The main signals used by the EPWM module are:

- **PWM output signals (EPWMxA and EPWMxB).** The PWM output signals are made available external to the device through the GPIO peripheral described in the system control and interrupts guide for the device.
- **Trip-zone signals (TZ0 to TZ5).** These input signals alert the EPWM module of an external fault condition. Each module on a device can be configured to either use or ignore any of the trip-zone signals. The trip-zone signal can be configured as an asynchronous input through the GPIO peripheral.
- **Time-base synchronization input (EPWMxSYNCl) and output (EPWMxSYNCO) signals.** The synchronization signals daisy chain the EPWM modules together. Each module can be configured to either use or ignore its synchronization input. The clock synchronization input and output signal are brought out to pins for EPWM0 (EPWM module 0) and EPWM3. The EPWM5 synchronization output (EPWM5SYNCO) is also connected to the input SYNCIN of the Enhanced Capture Module (ECAP0).
- **ADC start-of-conversion signals (EPWMxSOCA and EPWMxSOCB).** Each EPWM module has two ADC start of conversion signals (one for each sequencer). Any EPWM module can trigger a start of conversion for either sequencer. Which event triggers the start of conversion is configured in the Event-Trigger submodule of the EPWM.
- **Peripheral Bus.** The peripheral bus is 32-bits wide and allows both 16-bit and 32-bit writes to the EPWM register file.

Figure 12-383 also shows the key internal submodule interconnect signals. Each submodule is described in Section 12.4.2.4.



epwm-003

**Figure 12-383. EPWM Submodules and Critical Internal Signal Interconnects**

### 12.4.2.2 EPWM Environment

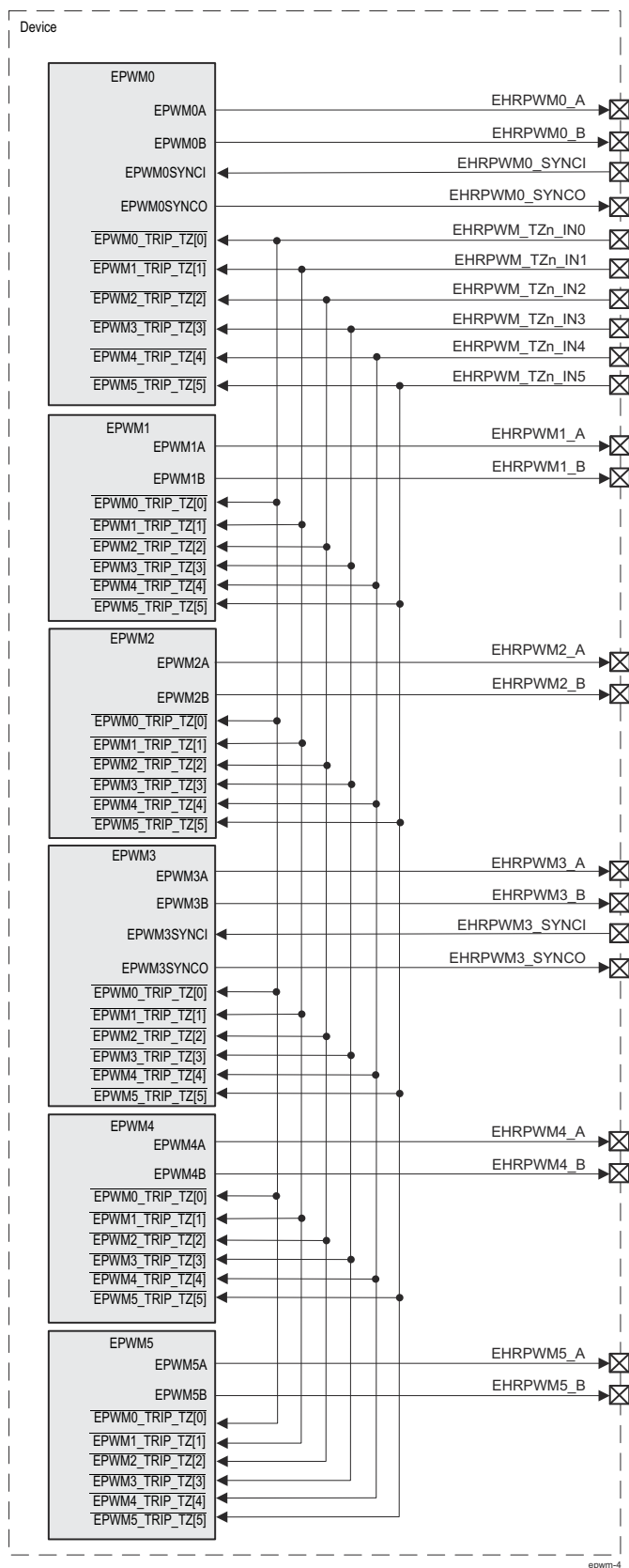
EPWMi (where i = 0 to 5) module is hereinafter referred to as EPWM module.

This section describes the EPWM external connections (environment).

#### 12.4.2.2.1 EPWM I/O Interface

[Figure 12-384](#) shows the EPWM interface signals.





**Figure 12-384. EPWM External Interface I/Os**

Table 12-422 describes the EPWM I/O signals.

**Table 12-422. EPWM Subsystems I/O Signals**

Module Pin	Device Level Signal	I/O <sup>(1)</sup>	Description	Module Pin Reset Value <sup>(2)</sup>
<b>EPWM0</b>				
EPWM0A	EHRPWM0_A	O	EPWM0 output A	0
EPWM0B	EHRPWM0_B	O	EPWM0 output B	0
EPWM0SYNCl	EHRPWM0_SYNCl	I	EPWM0 Sync input	HiZ
EPWM0SYNCO	EHRPWM0_SYNCO	O	EPWM0 Sync output	0
EPWM0_TRIP_TZ[0]	EHRPWM_TZn_IN0	I	EPWM0 TripZone input	HiZ
EPWM1_TRIP_TZ[1]	EHRPWM_TZn_IN1	I	EPWM1 TripZone input	HiZ
EPWM2_TRIP_TZ[2]	EHRPWM_TZn_IN2	I	EPWM2 TripZone input	HiZ
EPWM3_TRIP_TZ[3]	EHRPWM_TZn_IN3	I	EPWM3 TripZone input	HiZ
EPWM4_TRIP_TZ[4]	EHRPWM_TZn_IN4	I	EPWM4 TripZone input	HiZ
EPWM5_TRIP_TZ[5]	EHRPWM_TZn_IN5	I	EPWM5 TripZone input	HiZ
<b>EPWM1</b>				
EPWM1A	EHRPWM1_A	O	EPWM1 output A	0
EPWM1B	EHRPWM1_B	O	EPWM1 output B	0
EPWM0_TRIP_TZ[0]	EHRPWM_TZn_IN0	I	EPWM0 TripZone input	HiZ
EPWM1_TRIP_TZ[1]	EHRPWM_TZn_IN1	I	EPWM1 TripZone input	HiZ
EPWM2_TRIP_TZ[2]	EHRPWM_TZn_IN2	I	EPWM2 TripZone input	HiZ
EPWM3_TRIP_TZ[3]	EHRPWM_TZn_IN3	I	EPWM3 TripZone input	HiZ
EPWM4_TRIP_TZ[4]	EHRPWM_TZn_IN4	I	EPWM4 TripZone input	HiZ
EPWM5_TRIP_TZ[5]	EHRPWM_TZn_IN5	I	EPWM5 TripZone input	HiZ
<b>EPWM2</b>				
EPWM2A	EHRPWM2_A	O	EPWM2 output A	0
EPWM2B	EHRPWM2_B	O	EPWM2 output B	0
EPWM0_TRIP_TZ[0]	EHRPWM_TZn_IN0	I	EPWM0 TripZone input	HiZ
EPWM1_TRIP_TZ[1]	EHRPWM_TZn_IN1	I	EPWM1 TripZone input	HiZ
EPWM2_TRIP_TZ[2]	EHRPWM_TZn_IN2	I	EPWM2 TripZone input	HiZ
EPWM3_TRIP_TZ[3]	EHRPWM_TZn_IN3	I	EPWM3 TripZone input	HiZ
EPWM4_TRIP_TZ[4]	EHRPWM_TZn_IN4	I	EPWM4 TripZone input	HiZ
EPWM5_TRIP_TZ[5]	EHRPWM_TZn_IN5	I	EPWM5 TripZone input	HiZ
<b>EPWM3</b>				
EPWM3A	EHRPWM3_A	O	EPWM3 output A	0
EPWM3B	EHRPWM3_B	O	EPWM3 output B	0
EPWM3SYNCl	EHRPWM3_SYNCl	I	EPWM3 Sync input	HiZ
EPWM3SYNCO	EHRPWM3_SYNCO	O	EPWM3 Sync output	0
EPWM0_TRIP_TZ[0]	EHRPWM_TZn_IN0	I	EPWM0 TripZone input	HiZ
EPWM1_TRIP_TZ[1]	EHRPWM_TZn_IN1	I	EPWM1 TripZone input	HiZ
EPWM2_TRIP_TZ[2]	EHRPWM_TZn_IN2	I	EPWM2 TripZone input	HiZ
EPWM3_TRIP_TZ[3]	EHRPWM_TZn_IN3	I	EPWM3 TripZone input	HiZ
EPWM4_TRIP_TZ[4]	EHRPWM_TZn_IN4	I	EPWM4 TripZone input	HiZ
EPWM5_TRIP_TZ[5]	EHRPWM_TZn_IN5	I	EPWM5 TripZone input	HiZ
<b>EPWM4</b>				
EPWM4A	EHRPWM4_A	O	EPWM4 output A	0

**Table 12-422. EPWM Subsystems I/O Signals (continued)**

Module Pin	Device Level Signal	I/O <sup>(1)</sup>	Description	Module Pin Reset Value <sup>(2)</sup>
EPWM4B	EHRPWM4_B	O	EPWM4 output B	0
EPWM0_TRIP_TZ[0]	EHRPWM_TZn_IN0	I	EPWM0 TripZone input	HiZ
EPWM1_TRIP_TZ[1]	EHRPWM_TZn_IN1	I	EPWM1 TripZone input	HiZ
EPWM2_TRIP_TZ[2]	EHRPWM_TZn_IN2	I	EPWM2 TripZone input	HiZ
EPWM3_TRIP_TZ[3]	EHRPWM_TZn_IN3	I	EPWM3 TripZone input	HiZ
EPWM4_TRIP_TZ[4]	EHRPWM_TZn_IN4	I	EPWM4 TripZone input	HiZ
EPWM5_TRIP_TZ[5]	EHRPWM_TZn_IN5	I	EPWM5 TripZone input	HiZ
<b>EPWM5</b>				
EPWM5A	EHRPWM5_A	O	EPWM5 output A	0
EPWM5B	EHRPWM5_B	O	EPWM5 output B	0
EPWM0_TRIP_TZ[0]	EHRPWM_TZn_IN0	I	EPWM0 TripZone input	HiZ
EPWM1_TRIP_TZ[1]	EHRPWM_TZn_IN1	I	EPWM1 TripZone input	HiZ
EPWM2_TRIP_TZ[2]	EHRPWM_TZn_IN2	I	EPWM2 TripZone input	HiZ
EPWM3_TRIP_TZ[3]	EHRPWM_TZn_IN3	I	EPWM3 TripZone input	HiZ
EPWM4_TRIP_TZ[4]	EHRPWM_TZn_IN4	I	EPWM4 TripZone input	HiZ
EPWM5_TRIP_TZ[5]	EHRPWM_TZn_IN5	I	EPWM5 TripZone input	HiZ
<b>EPWM Start of Conversion</b>				
PWM_SOC_A	EHRPWM_SOC_A	O	EPWM start of conversion output A	OZ
PWM_SOC_B	EHRPWM_SOC_B	O	EPWM start of conversion output B	OZ

(1) I = Input; O = Output

(2) HiZ = High Impedance

### Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables Pin Attributes and Pin Multiplexing in the device-specific Datasheet.

### 12.4.2.3 EPWM Integration

This section describes the EPWM module integration in the device, including information about clocks, resets, and hardware requests.

[Figure 12-385](#) shows the EPWM integration.

There are 6 instances of the EPWM integrated in the device. Each of the Enhanced Pulse Width Modulator (EPWM) includes an Enhanced High Resolution Modulator (eHRPWM).

The high-resolution functionality is implemented only on the EPWMxA output. EPWMxB output has conventional PWM capabilities.

At system level the EPWM0 through EPWM5 integration features are listed below:

- A 32-bit slave configuration port on the CBASS0 interconnect
- A single functional clock from PLLCTRL0 shared between all 5 EPWM modules
- 2 hardware events per EPWM, that is a total of 12 events. From these events each EPWM:
  - generates 2 interrupts to the device COMPUTE\_CLUSTER0, PRU\_ICSSG0\_INTC, PRU\_ICSSG1\_INTC and MAIN2MCU\_INTRTR\_PLS
- A synchronization input/output daisy chain-like connection exists between the EPWM0 through EPWM5. For more details, refer to [Section 12.4.2.3.2, Daisy-Chain Connectivity between EPWM Modules](#).

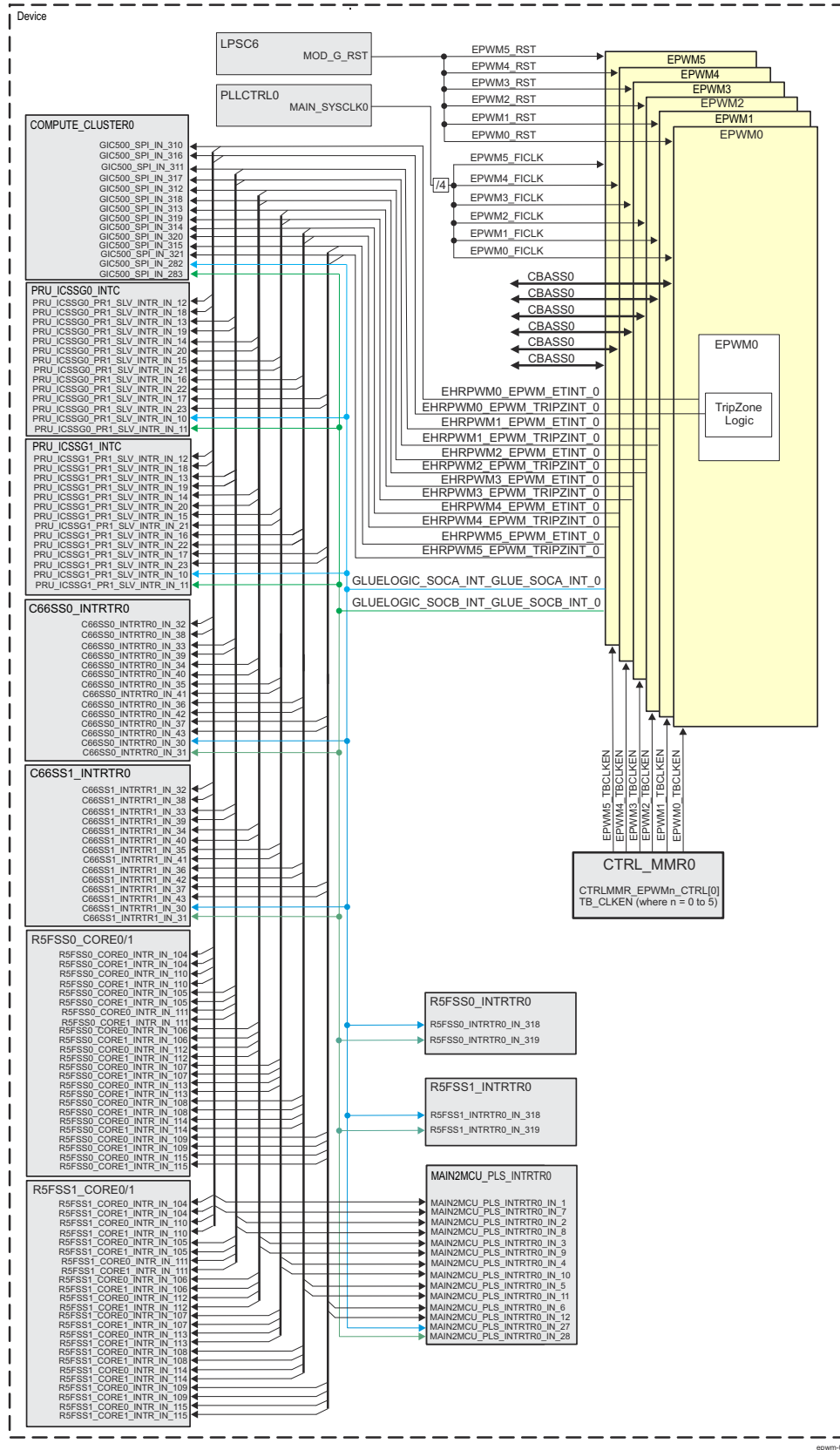


Figure 12-385. EPWM Integration

Table 12-423 through Table 12-425 summarize the integration of the module in the device.

**Table 12-423. EPWM Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
EPWM0	PSC0	PD0	LPSC6	CBASS0
EPWM1	PSC0	PD0	LPSC6	CBASS0
EPWM2	PSC0	PD0	LPSC6	CBASS0
EPWM3	PSC0	PD0	LPSC6	CBASS0
EPWM4	PSC0	PD0	LPSC6	CBASS0
EPWM5	PSC0	PD0	LPSC6	CBASS0

**Table 12-424. EPWM Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
EPWM0	EPWM0_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	EPWM0 functional and interface clock
EPWM1	EPWM1_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	EPWM1 functional and interface clock
EPWM2	EPWM2_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	EPWM2 functional and interface clock
EPWM3	EPWM3_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	EPWM3 functional and interface clock
EPWM4	EPWM4_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	EPWM4 functional and interface clock
EPWM5	EPWM5_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	EPWM5 functional and interface clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
EPWM0	EPWM0_RST	MOD_G_RST	LPSC6	Module Reset
EPWM1	EPWM1_RST	MOD_G_RST	LPSC6	Module Reset
EPWM2	EPWM2_RST	MOD_G_RST	LPSC6	Module Reset
EPWM3	EPWM3_RST	MOD_G_RST	LPSC6	Module Reset
EPWM4	EPWM4_RST	MOD_G_RST	LPSC6	Module Reset
EPWM5	EPWM5_RST	MOD_G_RST	LPSC6	Module Reset

**Table 12-425. EPWM Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
EPWM0	EHRPWM0_EPWM_ETINT_0	GIC500_SPI_IN_310	COMPUTE_CLUSTER0	EPWM0 interrupt	Pulse
		PRU_ICSSG0_PR1_SLV_INT_R_IN_12	PRU_ICSSG0_INTC		
		PRU_ICSSG1_PR1_SLV_INT_R_IN_12	PRU_ICSSG1_INTC		
		C66SS0_INTRTR0_IN_32	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_32	C66SS1_INTRTR0		
		R5FSS0_CORE0_INTR_IN_104	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_104	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_104	R5FSS1_CORE0		

**Table 12-425. EPWM Hardware Requests (continued)**

EHRPWM0_EPWM_TRIPZINT_0	R5FSS1_CORE1_INTR_IN_1		R5FSS1_CORE1_04	EPWM0 Tripzone interrupt	Pulse
	MAIN2MCU_PLS_INTRTR0_I_N_1		MAIN2MCU_PLS_INTRTR0		
	GIC500_SPI_IN_316		COMPUTE_CLUSTER0		
	PRU_ICSSG0_PR1_SLV_INT_R_IN_18		PRU_ICSSG0_INTC_R_IN_18		
	PRU_ICSSG1_PR1_SLV_INT_R_IN_18		PRU_ICSSG1_INTC_R_IN_18		
	C66SS0_INTRTR0_IN_38		C66SS0_INTRTR0		
	C66SS1_INTRTR0_IN_38		C66SS1_INTRTR0		
	R5FSS0_CORE0_INTR_IN_1_10		R5FSS0_CORE0_10		
	R5FSS0_CORE1_INTR_IN_1_10		R5FSS0_CORE1_10		
	R5FSS1_CORE0_INTR_IN_1_10		R5FSS1_CORE0_10		
EPWM1 EHRPWM1_EPWM_ETINT_0	R5FSS1_CORE1_INTR_IN_1_10		R5FSS1_CORE1_10	EPWM1 interrupt	Pulse
	MAIN2MCU_PLS_INTRTR0_I_N_7		MAIN2MCU_PLS_INTRTR0		
	GIC500_SPI_IN_311		COMPUTE_CLUSTER0		
	PRU_ICSSG0_PR1_SLV_INT_R_IN_13		PRU_ICSSG0_INTC_R_IN_13		
	PRU_ICSSG1_PR1_SLV_INT_R_IN_13		PRU_ICSSG1_INTC_R_IN_13		
	C66SS0_INTRTR0_IN_33		C66SS0_INTRTR0		
	C66SS1_INTRTR0_IN_33		C66SS1_INTRTR0		
	R5FSS0_CORE0_INTR_IN_1_05		R5FSS0_CORE0_05		
	R5FSS0_CORE1_INTR_IN_1_05		R5FSS0_CORE1_05		
	R5FSS1_CORE0_INTR_IN_1_05		R5FSS1_CORE0_05		
EHRPWM1_EPWM_TRIPZINT_0	R5FSS1_CORE1_INTR_IN_1_05		R5FSS1_CORE1_05	EPWM1 Tripzone interrupt	Pulse
	MAIN2MCU_PLS_INTRTR0_I_N_2		MAIN2MCU_PLS_INTRTR0		
	GIC500_SPI_IN_317		COMPUTE_CLUSTER0		
	PRU_ICSSG0_PR1_SLV_INT_R_IN_19		PRU_ICSSG0_INTC_R_IN_19		
	PRU_ICSSG1_PR1_SLV_INT_R_IN_19		PRU_ICSSG1_INTC_R_IN_19		
	C66SS0_INTRTR0_IN_39		C66SS0_INTRTR0		
	C66SS1_INTRTR0_IN_39		C66SS1_INTRTR0		
	R5FSS0_CORE0_INTR_IN_1_11		R5FSS0_CORE0_11		
	R5FSS0_CORE1_INTR_IN_1_11		R5FSS0_CORE1_11		

**Table 12-425. EPWM Hardware Requests (continued)**

EPWM2	EHRPWM2_EPWM_ETINT_0	R5FSS1_CORE0_INTR_IN_1	R5FSS1_CORE0	EPWM2 interrupt	Pulse
		11			
		R5FSS1_CORE1_INTR_IN_1	R5FSS1_CORE1		
		11			
		MAIN2MCU_PLS_INTRTR0_I	MAIN2MCU_PLS_IN		
		N_8	TRTR0		
		GIC500_SPI_IN_312	COMPUTE_CLUSTER0		
		PRU_ICSSG0_PR1_SLV_INT	PRU_ICSSG0_INTC		
		R_IN_14			
		PRU_ICSSG1_PR1_SLV_INT	PRU_ICSSG1_INTC		
		R_IN_14			
		C66SS0_INTRTR0_IN_34	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_34	C66SS1_INTRTR0		
EPWM2	EHRPWM2_EPWM_TRIPZINT_0	R5FSS0_CORE0_INTR_IN_1	R5FSS0_CORE0	EPWM2 Tripzone interrupt	Pulse
		06			
		R5FSS0_CORE1_INTR_IN_1	R5FSS0_CORE1		
		06			
		R5FSS1_CORE0_INTR_IN_1	R5FSS1_CORE0		
		06			
		R5FSS1_CORE1_INTR_IN_1	R5FSS1_CORE1		
		06			
		MAIN2MCU_PLS_INTRTR0_I	MAIN2MCU_PLS_IN		
		N_3	TRTR0		
		GIC500_SPI_IN_318	COMPUTE_CLUSTER0		
		PRU_ICSSG0_PR1_SLV_INT	PRU_ICSSG0_INTC		
		R_IN_20			
		PRU_ICSSG1_PR1_SLV_INT	PRU_ICSSG1_INTC		
		R_IN_20			
EPWM3	EHRPWM3_EPWM_ETINT_0	C66SS0_INTRTR0_IN_40	C66SS0_INTRTR0	EPWM3 interrupt	Pulse
		C66SS1_INTRTR0_IN_40	C66SS1_INTRTR0		
		R5FSS0_CORE0_INTR_IN_1	R5FSS0_CORE0		
		12			
		R5FSS0_CORE1_INTR_IN_1	R5FSS0_CORE1		
		12			
		R5FSS1_CORE0_INTR_IN_1	R5FSS1_CORE0		
		12			
		R5FSS1_CORE1_INTR_IN_1	R5FSS1_CORE1		
		12			
		MAIN2MCU_PLS_INTRTR0_I	MAIN2MCU_PLS_IN		
		N_9	TRTR0		
		GIC500_SPI_IN_313	COMPUTE_CLUSTER0		
		PRU_ICSSG0_PR1_SLV_INT	PRU_ICSSG0_INTC		
		R_IN_15			
		PRU_ICSSG1_PR1_SLV_INT	PRU_ICSSG1_INTC		
		R_IN_15			
EPWM3	EHRPWM3_EPWM_ETINT_0	C66SS0_INTRTR0_IN_35	C66SS0_INTRTR0	EPWM3 interrupt	Pulse
		C66SS1_INTRTR0_IN_35	C66SS1_INTRTR0		
		R5FSS0_CORE0_INTR_IN_1	R5FSS0_CORE0		
		07			



**Table 12-425. EPWM Hardware Requests (continued)**

		R5FSS0_CORE1_INTR_IN_1	R5FSS0_CORE1 07	EPWM3 Tripzone interrupt	Pulse
		R5FSS1_CORE0_INTR_IN_1	R5FSS1_CORE0 07		
		R5FSS1_CORE1_INTR_IN_1	R5FSS1_CORE1 07		
		MAIN2MCU_PLS_INTRTR0_I N_4	MAIN2MCU_PLS_IN TRTR0		
EHRPWM3_EPWM_TRIPZINT_0	GIC500_SPI_IN_319	COMPUTE_CLUSTER R0			
		PRU_ICSSG0_PR1_SLV_INT R_IN_21	PRU_ICSSG0_INT C		
		PRU_ICSSG1_PR1_SLV_INT R_IN_21	PRU_ICSSG1_INT C		
		C66SS0_INTRTR0_IN_41	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_41	C66SS1_INTRTR0		
		R5FSS0_CORE0_INTR_IN_1	R5FSS0_CORE0 13		
		R5FSS0_CORE1_INTR_IN_1	R5FSS0_CORE1 13	EPWM4 interrupt	Pulse
		R5FSS1_CORE0_INTR_IN_1	R5FSS1_CORE0 13		
		R5FSS1_CORE1_INTR_IN_1	R5FSS1_CORE1 13		
		MAIN2MCU_PLS_INTRTR0_I N_10	MAIN2MCU_PLS_IN TRTR0		
EPWM4	EHRPWM4_EPWM_ETINT_0	GIC500_SPI_IN_314	COMPUTE_CLUSTER R0		
		PRU_ICSSG0_PR1_SLV_INT R_IN_16	PRU_ICSSG0_INT C		
		PRU_ICSSG1_PR1_SLV_INT R_IN_16	PRU_ICSSG1_INT C		
		C66SS0_INTRTR0_IN_36	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_36	C66SS1_INTRTR0		
		R5FSS0_CORE0_INTR_IN_1	R5FSS0_CORE0 08		
		R5FSS0_CORE1_INTR_IN_1	R5FSS0_CORE1 08	EPWM4 Tripzone interrupt	Pulse
		R5FSS1_CORE0_INTR_IN_1	R5FSS1_CORE0 08		
		R5FSS1_CORE1_INTR_IN_1	R5FSS1_CORE1 08		
		MAIN2MCU_PLS_INTRTR0_I N_5	MAIN2MCU_PLS_IN TRTR0		
EHRPWM4_EPWM_TRIPZINT_0	GIC500_SPI_IN_320	COMPUTE_CLUSTER R0			
		PRU_ICSSG0_PR1_SLV_INT R_IN_22	PRU_ICSSG0_INT C		
		PRU_ICSSG1_PR1_SLV_INT R_IN_22	PRU_ICSSG1_INT C		
		C66SS0_INTRTR0_IN_42	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_42	C66SS1_INTRTR0		

**Table 12-425. EPWM Hardware Requests (continued)**

		R5FSS0_CORE0_INTR_IN_1 14	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_1 14	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_1 14	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_1 14	R5FSS1_CORE1		
		MAIN2MCU_PLS_INTRTR0_I N_11	MAIN2MCU_PLS_IN TRTR0		
EPWM5	EHRPWM5_EPWM_ETINT_0	GIC500_SPI_IN_315	COMPUTE_CLUSTER R0	EPWM5 interrupt	Pulse
		PRU_ICSSG0_PR1_SLV_INT R_IN_17	PRU_ICSSG0_INTC		
		PRU_ICSSG1_PR1_SLV_INT R_IN_17	PRU_ICSSG1_INTC		
		C66SS0_INTRTR0_IN_37	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_37	C66SS1_INTRTR0		
		R5FSS0_CORE0_INTR_IN_1 09	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_1 09	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_1 09	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_1 09	R5FSS1_CORE1		
		MAIN2MCU_PLS_INTRTR0_I N_6	MAIN2MCU_PLS_IN TRTR0		
	EHRPWM5_EPWM_TRIPZINT_0	GIC500_SPI_IN_321	COMPUTE_CLUSTER R0	EPWM5 Tripzone interrupt	Pulse
		PRU_ICSSG0_PR1_SLV_INT R_IN_23	PRU_ICSSG0_INTC		
		PRU_ICSSG1_PR1_SLV_INT R_IN_23	PRU_ICSSG1_INTC		
		C66SS0_INTRTR0_IN_43	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_43	C66SS1_INTRTR0		
		R5FSS0_CORE0_INTR_IN_1 15	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_1 15	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_1 15	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_1 15	R5FSS1_CORE1		
		MAIN2MCU_PLS_INTRTR0_I N_12	MAIN2MCU_PLS_IN TRTR0		
EPWMx (where x = 0 to 5)	GLUELOGIC_SOCA_INT_GLUE_S OCA_INT_0	GIC500_SPI_IN_282	COMPUTE_CLUSTER R0	EPWM Start of Conversion A event	Pulse
		PRU_ICSSG0_PR1_SLV_INT R_IN_10	PRU_ICSSG0_INTC		
		PRU_ICSSG1_PR1_SLV_INT R_IN_10	PRU_ICSSG1_INTC		
		C66SS0_INTRTR0_IN_30	C66SS0_INTRTR0		

**Table 12-425. EPWM Hardware Requests (continued)**

	C66SS1_INTRTR0_IN_30	C66SS1_INTRTR0		
	R5FSS0_INTRTR0_IN_318	R5FSS0_INTRTR0		
	R5FSS1_INTRTR0_IN_318	R5FSS1_INTRTR0		
	R5FSS1_INTRTR0_IN_BIT0_143	R5FSS1_INTRTR0		
	MAIN2MCU_PLS_INTRTR0_IN_27	MAIN2MCU_PLS_INTRTR0		
GLUELOGIC_SOCB_INT_GLUE_S OCB_INT_0	GIC500_SPI_IN_283	COMPUTE_CLUSTER0	EPWM Start of Conversion B event	Pulse
	PRU_ICSSG0_PR1_SLV_INT R_IN_11	PRU_ICSSG0_INTC		
	PRU_ICSSG1_PR1_SLV_INT R_IN_11	PRU_ICSSG1_INTC		
	C66SS0_INTRTR0_IN_31	C66SS0_INTRTR0		
	C66SS1_INTRTR0_IN_31	C66SS1_INTRTR0		
	R5FSS0_INTRTR0_IN_319	R5FSS0_INTRTR0		
	R5FSS1_INTRTR0_IN_319	R5FSS1_INTRTR0		
	R5FSS1_INTRTR0_IN_BIT0_144	R5FSS1_INTRTR0		
	MAIN2MCU_PLS_INTRTR0_IN_28	MAIN2MCU_PLS_INTRTR0		

#### 12.4.2.3.1 Device Specific EPWM Features

A High-Resolution PWM (HRPWM) modulator is added to each of the device EPWM modules.

---

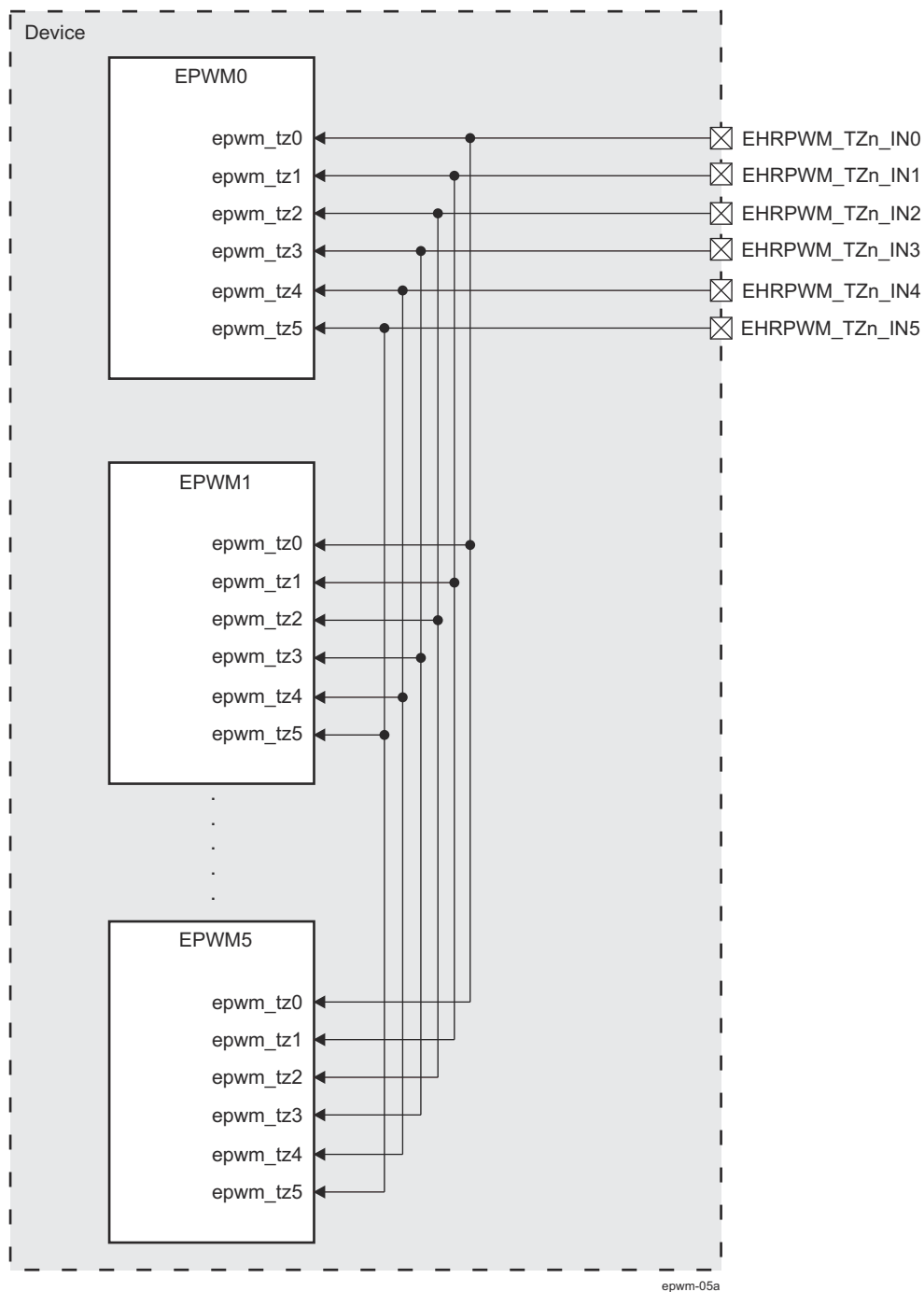
##### Note

The HRPWM option applies only to the EPWMxA output channel. The EPWMxB channel has conventional capabilities. For more details on HRPWM features of EPWM modules refer to the [Section 12.4.2.4.9](#).

---

The EPWM module interface has some restrictions in functionality as listed below:

- **For EPWM comparators — only the outputs (ehrpwmxA and ehripwmxB, where x = 0 to 5) are available at chip level**
- **Each EPWM supports 6 tripzone event inputs and each of those inputs from all six of the EPWMs are tied to a common tripzone\_input pin so that any EPWM can be triggered by any of the six tripzone inputs as shown in [Figure 12-386](#).**



**Figure 12-386. EPWM Tripzone Connectivity Detail**

The EPWM functional interface signals which are NOT available to user are summarized in [Table 12-426](#).

**Table 12-426. Device Limitations for the EPWM Functional Interfaces**

Module Interface	Signal	Description	Comment
<b>EPWM_n</b> (where n = 0 to 5 for the device)	EPWM_COMP_EPWMDCMAH	EPWM Comparator A input (HIGH)	Not available at chip boundary (can not be used)
	EPWM_COMP_EPWMDCMAL	EPWM Comparator A input (LOW)	
	EPWM_COMP_EPWMDCMBH	EPWM Comparator B input (HIGH)	
	EPWM_COMP_EPWMDCMBL	EPWM Comparator B input (LOW)	
	EPWM_TRIP_TZ_O[5:0]	EPWM Tripzone outputs	
	EPWM_TRIP_TZ_OEN[5:0]	EPWM Tripzone outputs enable	

### Note

Only the SYNCI input and SYNCO output signals of EPWM0 and EPWM3 modules are available at chip-level. Respective names of the signals are: EHRPWM0\_SYNCI and EHRPWM0\_SYNCO (for EPWM0) and EHRPWM3\_SYNCI and EHRPWM3\_SYNCO (for EPWM3). For more information, see [Section 12.4.2.2, EPWM Environment](#).

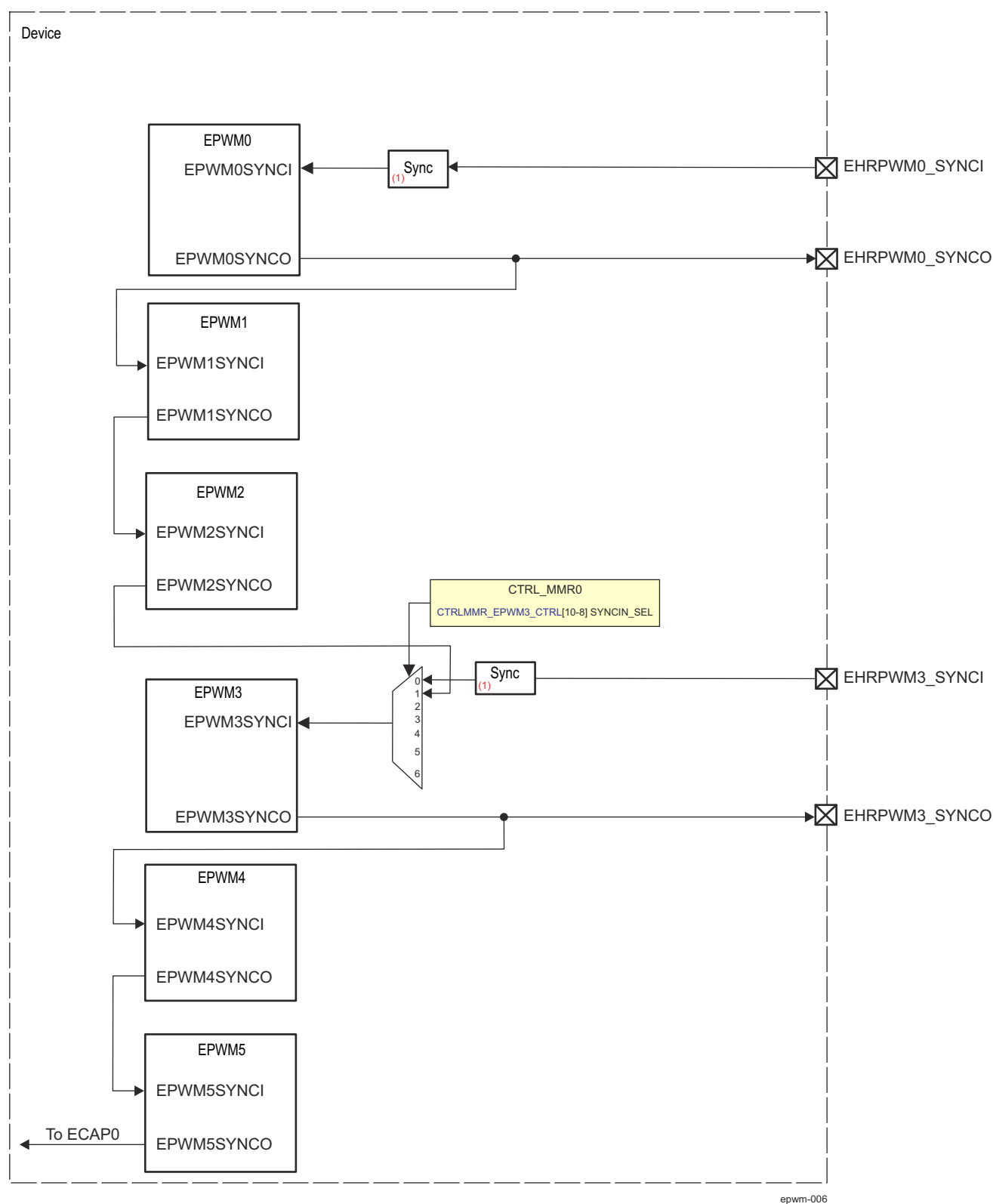
#### 12.4.2.3.2 Daisy-Chain Connectivity between EPWM Modules

For synchronization to other modules or events EPWMs are provided with synchronization signals. In the device, these signals are connected in a daisy-chain fashion as shown in [Figure 12-387](#). The EPWM0 input synchronization signal is terminated at a device pad (EHRPWM0\_SYNCI signal), such that device external sync events can be directly applied only to the EPWM0 submodule. Additionally, the EPWM0 input synchronization scheme can be extended through PRU\_ICSSG Host interrupts (ICSSG\_n\_HOST\_INT6, where n = 0 to 1), routed through device PRU\_ICSSG modules. The EPWM3 through EPWM5 modules can be synchronized either from a separate external signal on the EHRPWM3\_SYNCI pin or from EPWM2 output synchronization signal. Additionally, the EPWM3 input synchronization scheme can be extended through PRU\_ICSSG Host interrupts (ICSSG\_n\_HOST\_INT7, where n = 0 to 1). The EPWM3 input synchronization signal is selected using the corresponding SYNCIN\_SEL bit field of the CTRLMMR\_EPWM3\_CTRL register in the CTRL\_MMR0 module. A synchronization output is available from the EPWM0 on the EHRPWM0\_SYNCO output pin and from EPWM3 on the EHRPWM3\_SYNCO pin. PWM0\_SYNC\_OUT and PWM3\_SYNC\_OUT output events are internally routed to the Interrupt Controllers (INTC) for each of the PRU\_ICSSG modules in this device.

#### Note

The EPWM0 EPWM0SYNCI signal (device EHRPWM0\_SYNCI input signal) triggers the event of the EPWM0 Phase Register being loaded into the Counter register (TBPHS -> TBCNT). This event is synchronous to the EPWM0 **time-base clock** (TBCLK).

The EPWM0 EPWM0SYNCO (device EHRPWM0\_SYNCO output signal) is implicitly synchronous to the time-base clock, as this signal has a programmable source of event (in the EPWM\_TBCTL[5-4] SYNCOSSEL bit field) triggered synchronously to the EPWM0 TBCLK.



**Figure 12-387. Daisy-Chain Connectivity between EPWM Modules**



---

**Note**

The EPWM0SYNCl and EPWM3SYNCl inputs of EPWM0 and EPWM3 modules can be connected to one of several external event sources. These sources must be synchronized to the EPWM clock domain to prevent propagation of metastable state. A synchronizer (signified with (1) Sync of the figure above) has been implemented in the path to resolve this potential issue. However, the synchronizer inserts a delay which may need to be accounted in some applications of the EPWM module.

---

#### 12.4.2.3.3 ADC start of conversion signals (PWM\_SOCA and PWM\_SOCB)

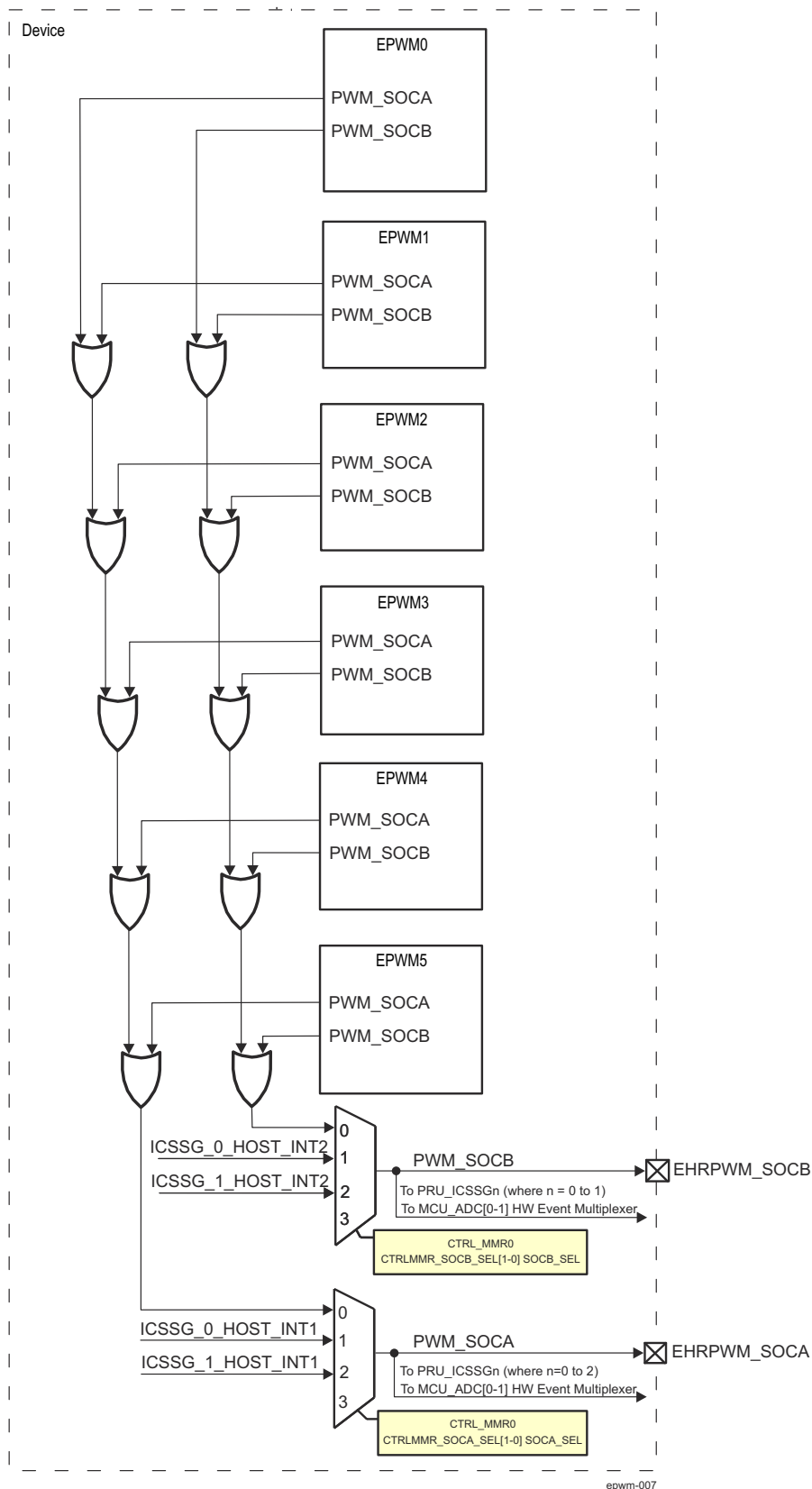
Each EPWM module provides start of conversion events (PWM\_SOCA and PWM\_SOCB), that can be used to start conversion of external ADC module. In this device all PWM\_SOCA and all PWM\_SOCB events are ORed together and hence any of the 6 EPWMs can initiate an ADC start of conversion. These events are then multiplexed with an PRU\_ICSSG[0-1] host interrupts (ICSSG\_x\_HOST\_INTn, where x = 0 to 1 and n = 1 or 2), allowing either PRU\_ICSSG[0-1] or EPWM0 through EPWM5 modules to trigger the PWM\_SOCA/PWM\_SOCB event pins (EHRPWM\_SOCA and EHRPWM\_SOCB). The PWM\_SOCA and PWM\_SOCB event output signals are also routed back to the PRU\_ICSSGn (where n = 0 to 1) modules as interrupts as shown in [Figure 12-388](#)

---

#### Note

ICSSG\_x\_HOST\_INTn (where x = 0 to 1 and n = 1 or 2) interrupts are mapped to Host-3 and Host-4 of the PRU\_ICSSG Interrupt Controller.

---



**Figure 12-388. Interconnectivity of ADC start of conversion**

#### 12.4.2.3.4 EPWM Modules Time Base Clock Gating

Each EPWM module has an EPWMTBCLKEN module input used to individually **enable / disable its EPWM time-base clock**. The EPWM time-base clock enable input comes from the EPWMn\_CTRL (where n = 0 to 5) register in CTRL\_MMR0 module, as follows:

- EPWM0 -> CTRLMMR\_EPWM0\_CTRL[0] TB\_CLKEN bit
- EPWM1 -> CTRLMMR\_EPWM1\_CTRL[0] TB\_CLKEN bit
- EPWM2 -> CTRLMMR\_EPWM2\_CTRL[0] TB\_CLKEN bit
- EPWM3 -> CTRLMMR\_EPWM3\_CTRL[0] TB\_CLKEN bit
- EPWM4 -> CTRLMMR\_EPWM4\_CTRL[0] TB\_CLKEN bit
- EPWM5 -> CTRLMMR\_EPWM5\_CTRL[0] TB\_CLKEN bit.

This individual TBCLKEN control can be used to align the EPWM time base clock. TB\_CLKEN bit set to 0h, holds the TBCLK generation counter in its reset state. When TB\_CLKEN is set to 1h, then the TBCLK generation counter is allowed to count.

#### 12.4.2.4 EPWM Functional Description

8 submodules are included in each EPWM peripheral. Each of these submodules performs specific tasks that can be configured by software.

##### 12.4.2.4.1 EPWM Submodule Features

[Table 12-427](#) lists the eight key submodules together with a list of their main configuration parameters.

**Table 12-427. Submodule Configuration Parameters**

Submodule	Configuration Parameter or Option
Time-base (TB)	<ul style="list-style-type: none"> <li>Scale the time-base clock (TBCLK) relative to the system clock (FICLK).</li> <li>Configure the PWM time-base counter (TBCNT) frequency or period.</li> <li>Time-base counter mode selection: <ul style="list-style-type: none"> <li>count-up mode: used for asymmetric PWM</li> <li>count-down mode: used for asymmetric PWM</li> <li>count-up-and-down mode: used for symmetric PWM</li> </ul> </li> <li>Configure the time-base phase relative to another EPWM module.</li> <li>Synchronize the time-base counter between modules through hardware or software.</li> <li>Configure the direction (up or down) of the time-base counter after a synchronization event.</li> <li>Configure how the time-base counter will behave when the device is halted by an emulator.</li> <li>Specify the source for the synchronization output of the EPWM module: <ul style="list-style-type: none"> <li>Synchronization input signal</li> <li>Time-base counter equal to zero</li> <li>Time-base counter equal to counter-compare B (CMPB)</li> <li>No output synchronization signal generated</li> </ul> </li> </ul>
Counter-compare (CC)	<ul style="list-style-type: none"> <li>Specify the PWM duty cycle for output EPWMxA and/or output EPWMxB</li> <li>Specify the time at which switching events occur on the EPWMxA or EPWMxB output</li> </ul>
Action-qualifier (AQ)	<ul style="list-style-type: none"> <li>Specify the type of action taken when a time-base or counter-compare submodule event occurs: <ul style="list-style-type: none"> <li>No action taken</li> <li>Output EPWMxA and/or EPWMxB switched high</li> <li>Output EPWMxA and/or EPWMxB switched low</li> <li>Output EPWMxA and/or EPWMxB toggled</li> </ul> </li> <li>Force the PWM output state through software control</li> <li>Configure and control the PWM dead-band through software</li> </ul>
Dead-band (DB)	<ul style="list-style-type: none"> <li>Control of traditional complementary dead-band relationship between upper and lower switches</li> <li>Specify the output rising-edge-delay value</li> <li>Specify the output falling-edge delay value</li> <li>Bypass the dead-band module entirely. In this case the PWM waveform is passed through without modification</li> </ul>
PWM-chopper (PC)	<ul style="list-style-type: none"> <li>Create a chopping (carrier) frequency</li> <li>Pulse width of the first pulse in the chopped pulse train</li> <li>Duty cycle of the second and subsequent pulses</li> <li>Bypass the PWM-chopper module entirely. In this case the PWM waveform is passed through without modification.</li> </ul>

**Table 12-427. Submodule Configuration Parameters (continued)**

Submodule	Configuration Parameter or Option
Trip-zone (TZ)	<ul style="list-style-type: none"> <li>Configure the EPWM module to react to one, all, or none of the trip-zone pins</li> <li>Specify the tripping action taken when a fault occurs: <ul style="list-style-type: none"> <li>Force EPWMxA and/or EPWMxB high</li> <li>Force EPWMxA and/or EPWMxB low</li> <li>Force EPWMxA and/or EPWMxB to a high-impedance state</li> <li>Configure EPWMxA and/or EPWMxB to ignore any trip condition</li> </ul> </li> <li>Configure how often the EPWM will react to the trip-zone pin: <ul style="list-style-type: none"> <li>One-shot</li> <li>Cycle-by-cycle</li> </ul> </li> <li>Enable the trip-zone to initiate an interrupt</li> <li>Bypass the trip-zone module entirely</li> </ul>
Event-trigger (ET)	<ul style="list-style-type: none"> <li>Enable the EPWM events that will trigger an interrupt.</li> <li>Specify the rate at which events cause triggers (every occurrence or every second or third occurrence)</li> <li>Poll, set, or clear event flags</li> </ul>
High-Resolution PWM (HRPWM)	<ul style="list-style-type: none"> <li>Enable extended time resolution capabilities</li> <li>Configure finer time granularity control or edge positioning</li> </ul>

Some examples on various EPWM module configurations are shown below. These examples use the constant definitions shown in [Example 12-8](#).

## Constant Definitions Used in the EPWM Code Examples

```
// TBCTL (Time-Base Control)
// =====
// TBCNT MODE bits
#define TB_COUNT_UP      0x0
#define TB_COUNT_DOWN    0x1
#define TB_COUNT_UPDOWN  0x2
#define TB_FREEZE        0x3
// PHSEN bit
#define TB_DISABLE       0x0
#define TB_ENABLE        0x1
// PRDLD bit
#define TB_SHADOW        0x0
#define TB_IMMEDIATE     0x1
// SYNCSEL bits
#define TB_SYNC_IN       0x0
#define TB_CTR_ZERO      0x1
#define TB_CTR_CMPB      0x2
#define TB_SYNC_DISABLE  0x3
// HSPCLKDIV and CLKDIV bits
#define TB_DIV1          0x0
#define TB_DIV2          0x1
#define TB_DIV4          0x2
// PHSDIR bit
#define TB_DOWN          0x0
#define TB_UP            0x1

// CMPCTL (Compare Control)
// =====
```

```
// LOADAMODE and LOADBMODE bits
#define CC_CTR_ZERO 0x0
#define CC_CTR_PRD 0x1
#define CC_CTR_ZERO_PRD 0x2
#define CC_LD_DISABLE 0x3
// SHDWAMODE and SHDWBMODE bits
#define CC_SHADOW 0x0
#define CC_IMMEDIATE 0x1
// AQCTLA and AQCTLB (Action-qualifier Control)
// =====
// ZRO, PRD, CAU, CAD, CBU, CBD bits
#define AQ_NO_ACTION 0x0
#define AQ_CLEAR 0x1
#define AQ_SET 0x2
#define AQ_TOGGLE 0x3
// DBCTL (Dead-Band Control)
// =====
// MODE bits
#define DB_DISABLE 0x0
#define DBA_ENABLE 0x1
#define DBB_ENABLE 0x2
#define DB_FULL_ENABLE 0x3
// POLSEL bits
#define DB_ACTV_HI 0x0
#define DB_ACTV_LO 0x1
#define DB_ACTV_HIC 0x2
#define DB_ACTV_LO 0x3
// PCCTL (chopper control)
// =====
// CHPEN bit
#define CHP_ENABLE 0x0
#define CHP_DISABLE 0x1
// CHPFREQ bits
#define CHP_DIV1 0x0
#define CHP_DIV2 0x1
#define CHP_DIV3 0x2
#define CHP_DIV4 0x3
#define CHP_DIV5 0x4
#define CHP_DIV6 0x5
#define CHP_DIV7 0x6
#define CHP_DIV8 0x7
// CHPDUTY bits
#define CHP1_8TH 0x0
#define CHP2_8TH 0x1
#define CHP3_8TH 0x2
#define CHP4_8TH 0x3
#define CHP5_8TH 0x4
#define CHP6_8TH 0x5
#define CHP7_8TH 0x6
// TZSEL (Trip-zone Select)
// =====
// CBCn and OSHn bits
#define TZ_ENABLE 0x0
#define TZ_DISABLE 0x1
// TZCTL (Trip-zone Control)
// =====
// TZA and TZB bits
#define TZ_HIZ 0x0
#define TZ_FORCE_HI 0x1
#define TZ_FORCE_LO 0x2
#define TZ_DISABLE 0x3
// ETSEL (Event-trigger Select)
// =====
// INTSEL, SOCASEL, SOCBSEL bits
#define ET_CTR_ZERO 0x1
#define ET_CTR_PRD 0x2
#define ET_CTRU_CMPA 0x4
#define ET_CTRD_CMPA 0x5
#define ET_CTRU_CMPB 0x6
#define ET_CTRD_CMPB 0x7
// ETPS (Event-trigger Prescale)
// =====
// INTPRD, SOCAPRD, SOCBPRD bits
#define ET_DISABLE 0x0
```

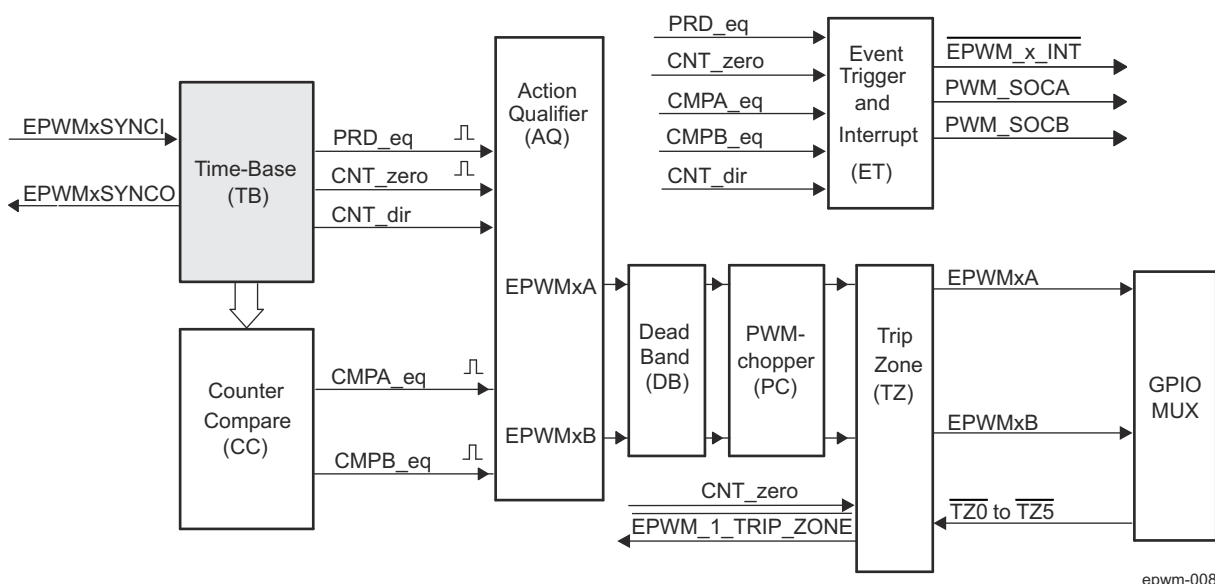
```
#define ET_1ST      0x1
#define ET_2ND      0x2
#define ET_3RD      0x3
```

#### 12.4.2.4.2 EPWM Time-Base (TB) Submodule

This section describes the Time-Base (TB) submodule in the PWM module.

##### 12.4.2.4.2.1 Overview

Each EPWM module has its own time-base submodule that determines all of the timing events. Built-in synchronization logic allows the time-base of multiple EPWM modules (EPWMx) to work together as a single system. [Figure 12-389](#) illustrates the time-base module's place within the EPWM.



**Figure 12-389. EPWM Time-Base Submodule**

The TB module generates the period (or frequency) of the PWM output waveforms and consists of a 16-bit counter that can be configured for up, down or up and down counting. The time base for the counter is a pre-scaled version of the system clock (FICLK/n) which can be programmed for a pre-scale of 1, that is same as system clock.

TB module features:

- Specify the EPWMx time-base counter (TBCNT) frequency or period in the EPWM\_TBCNT register to control how often events occur.
- Manage time-base synchronization with other EPWMx modules.
- Maintain a phase relationship with other EPWMx modules.
- Set the time-base counter to count-up, count-down, or count-up-and-down mode.
- Generate the following events:
  - PRD\_eq (Time-base counter (EPWM\_TBCNT register) equal to the specified period in EPWM\_TBPRD register (that is TBCNT = TBPRD)).
  - CNT\_zero (Time-base counter equal to zero (TBCNT = 0000h)).
- Configure the rate of the time-base clock; a prescaled version of the CPU m clock (FICLK). This allows the time-base counter to increment/decrement at a slower rate.

##### 12.4.2.4.2.2 Controlling and Monitoring the EPWM Time-Base Submodule

[Table 12-428](#) lists the registers used to control and monitor the time-base submodule.

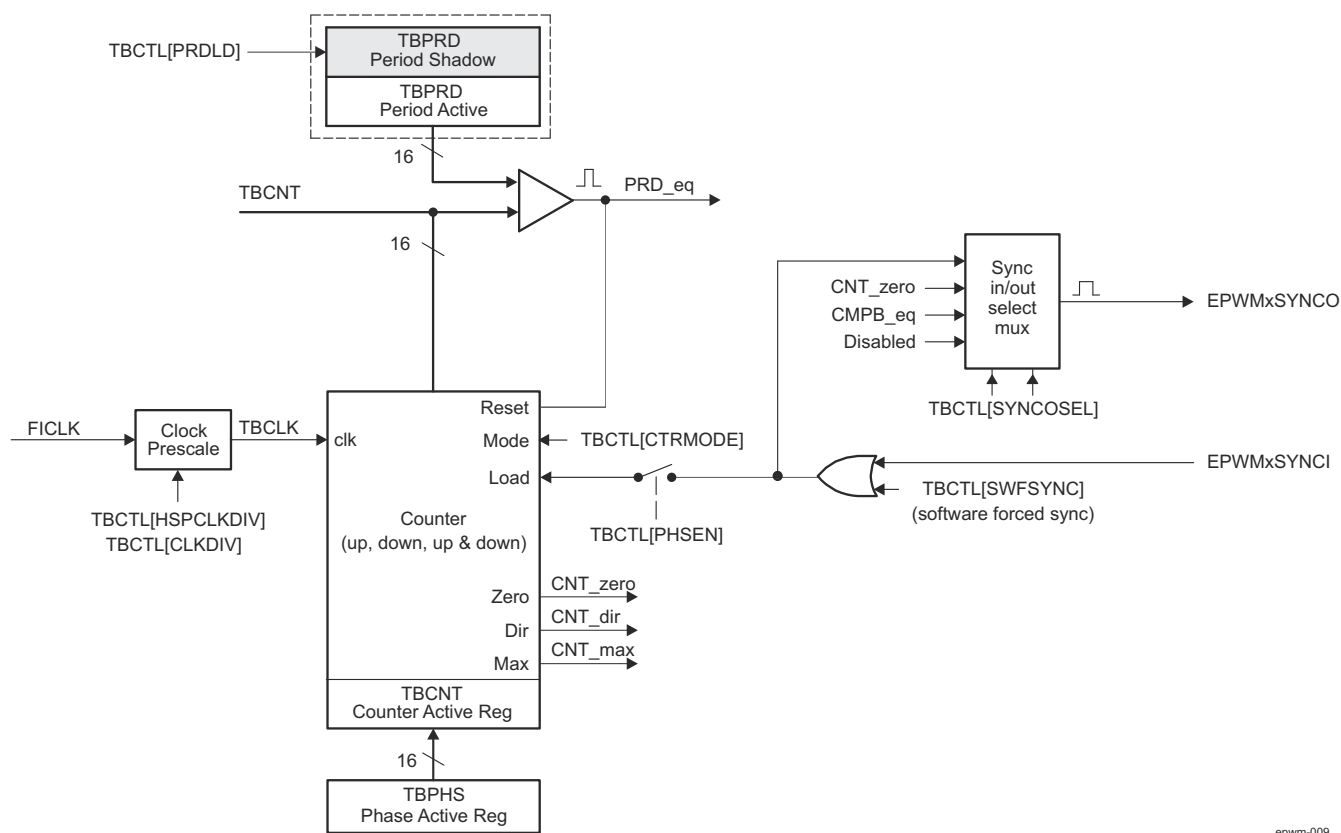


### Table 12-428. EPWM Time-Base Submodule Registers

Acronym	Register Description	Address Offset	Shadowed
EPWM_TBCTL	Time-Base Control Register	0h	No
EPWM_TBSTS	Time-Base Status Register	2h	No
HRPWM_TBPHSHR	HRPWM extension Phase Register <sup>(1)</sup>	4h	No
EPWM_TBPHS	Time-Base Phase Register	6h	No
EPWM_TBCNT	Time-Base Counter Register	8h	No
EPWM_TBPRD	Time-Base Period Register	Ah	Yes

- (1) This register is available only on EPWM instances that include the high-resolution extension (HRPWM). On EPWM modules that do not include the HRPWM, this location is reserved. See *EPWM Integration* to determine which EPWM instances include this feature.

Figure 12-390 shows the critical signals and registers of the time-base submodule. Table 12-429 provides descriptions of the key signals associated with the time-base submodule.



### Figure 12-390. EPWM Time-Base Submodule Signals and Registers

### Table 12-429. EPWM Time-Base Submodule Key Signals

Signal	Description
EPWMxSYNCl	Time-base synchronization input.  Input pulse used to synchronize the time-base counter with the counter of another EPWM module earlier in the synchronization chain. An EPWM peripheral can be configured to use or ignore this signal. For the first EPWM module (EPWM0) this signal comes from a device pin. For subsequent EPWM modules this signal is passed from another EPWM peripheral. For example, EPWM2SYNCl is generated by the EPWM1 peripheral and the EPWM3SYNCl is generated (optional through internal multiplexer) by EPWM2 or from a device pin. See <a href="#">Section 12.4.2.4.2.3.2</a> for information on the synchronization order of a particular device.

**Table 12-429. EPWM Time-Base Submodule Key Signals (continued)**

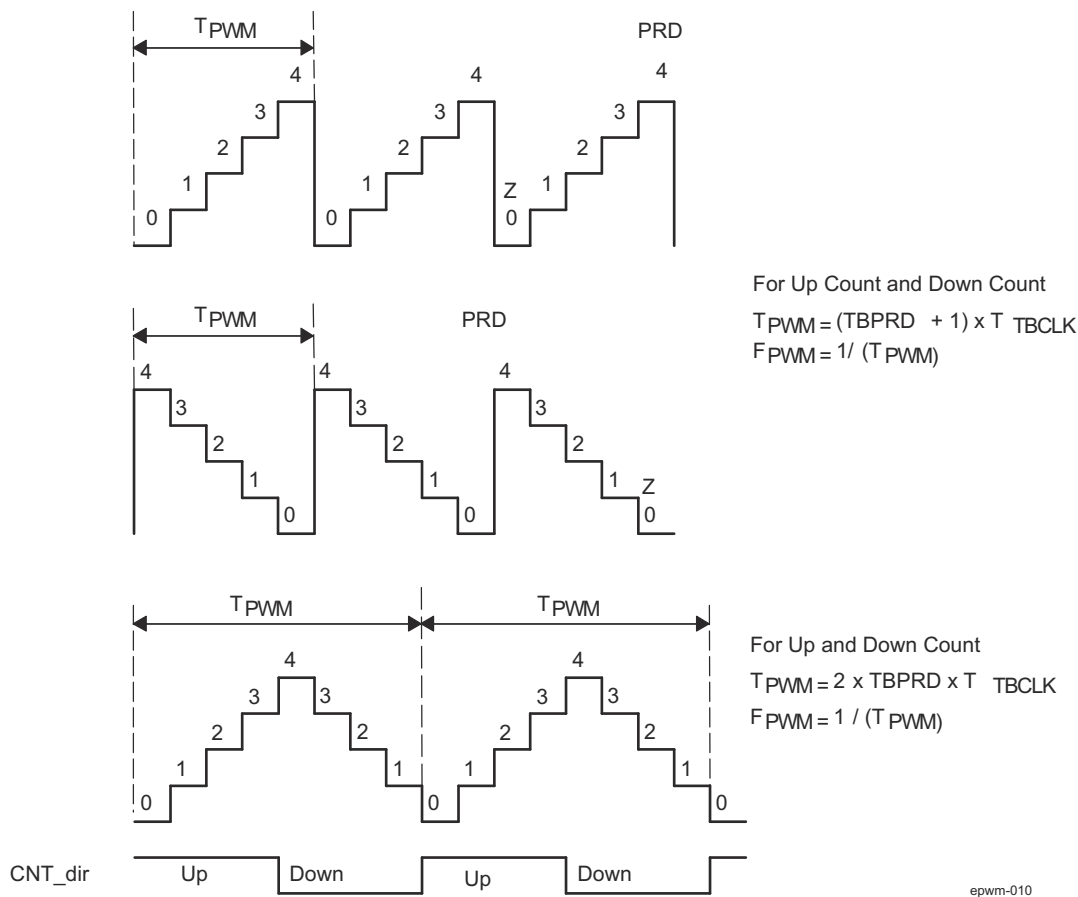
Signal	Description
EPWMxSYNCO	Time-base synchronization output.  This output pulse is used to synchronize the counter of an EPWM module later in the synchronization chain. The EPWM module generates this signal from one of three event sources: <ol style="list-style-type: none"> <li>1. EPWMxSYNCl (Synchronization input pulse)</li> <li>2. TBCNT = 0: The time-base counter (EPWM_TBCNT register) equal to zero (TBCNT = 0000h).</li> <li>3. TBCNT = CMPB: The time-base counter (EPWM_TBCNT register) equal to the counter-compare B register — EPWM_CMPB (that is bitfield TBCNT = bitfield CMPB).</li> </ol>
PRD_eq	Time-base counter equal to the specified period.  This signal is generated whenever the counter value is equal to the active period register value. That is when TBCNT = TBPRD.
CNT_zero	Time-base counter equal to zero.  This signal is generated whenever the counter value is zero. That is when TBCNT equals 0000h.
CMPB_eq	Time-base counter equal to active counter-compare B register (TBCNT = CMPB).  This event is generated by the counter-compare submodule and used by the synchronization out logic.
CNT_dir	Time-base counter direction.  Indicates the current direction of the EPWM's time-base counter. This signal is high when the counter is increasing and low when it is decreasing.
CNT_max	Time-base counter equal max value (TBCNT = FFFFh).  Generated event when the EPWM_TBCNT register value reaches its maximum value. This signal is only used only as a status bit.
TBCLK	Time-base clock.  This is a prescaled version of the system clock — FICLK and is used by all submodules within the EPWMn. This clock determines the rate at which time-base counter increments or decrements.

#### 12.4.2.4.2.3 Calculating PWM Period and Frequency

The frequency of PWM events is controlled by the time-base period register (EPWM\_TBPRD) and the mode of the time-base counter. [Figure 12-391](#) shows the period ( $T_{pwm}$ ) and frequency ( $F_{pwm}$ ) relationships for the up-count, down-count, and up-down-count time-base counter modes when the period is set to 4 (EPWM\_TBPRD[15:0] TBPRD = 0x4). The time increment for each step is defined by the time-base clock (TBCLK) which is a prescaled version of the system clock (FICLK).

The time-base counter has three modes of operation selected by the time-base control register (EPWM\_TBCTL):

- **Up-Down-Count Mode:** In up-down-count mode, the time-base counter starts from zero and increments until the period (TBPRD) value is reached. When the period value is reached, the time-base counter then decrements until it reaches zero. At this point the counter repeats the pattern and begins to increment.
- **Up-Count Mode:** In this mode, the time-base counter starts from zero and increments until it reaches the value in the period register (TBPRD). When the period value is reached, the time-base counter resets to zero and begins to increment once again.
- **Down-Count Mode:** In down-count mode, the time-base counter starts from the period (TBPRD) value and decrements until it reaches zero. When it reaches zero, the time-base counter is reset back to the period value and it repeats this pattern.



**Figure 12-391. EPWM Time-Base Frequency and Period**

#### 12.4.2.4.2.3.1 EPWM Time-Base Period Shadow Register

The time-base period register (EPWM\_TBPRD) has a shadow register. Shadowing allows the register update to be synchronized with the hardware. The following definitions are used to describe all shadow registers in the EPWM module:

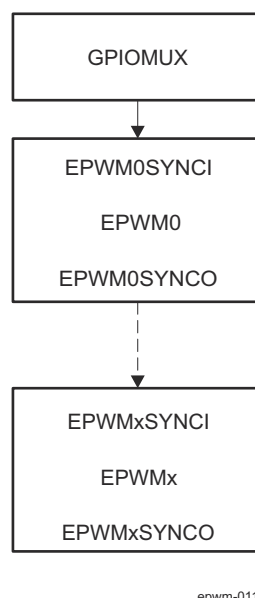
- **Active Register:** The active register controls the hardware and is responsible for actions that the hardware causes or invokes.
- **Shadow Register:** The shadow register buffers or provides a temporary holding location for the active register. It has no direct effect on any control hardware. At a strategic point in time the shadow register's content is transferred to the active register. This prevents corruption or spurious operation due to the register being asynchronously modified by software.

The memory address of the shadow period register is the same as the active register. Which register is written to or read from is determined by the EPWM\_TBCTL[3] PRDL bit. This bit enables and disables the EPWM\_TBPRD shadow register as follows:

- **Time-Base Period Shadow Mode:** The EPWM\_TBPRD shadow register is enabled when EPWM\_TBCTL[3] PRDL = 0h. Reads from and writes to the EPWM\_TBPRD register memory address go to the shadow register. The shadow register contents are transferred to the active register (EPWM\_TBPRD (Active) ← EPWM\_TBPRD (shadow)) when the time-base counter register (EPWM\_TBCNT) equals zero (TBCNT = 0000h). By default the EPWM\_TBPRD shadow register is enabled.
- **Time-Base Period Immediate Load Mode:** If immediate load mode is selected (EPWM\_TBCTL[3] PRDL = 1h), then a read from or a write to the TBPRD memory address goes directly to the active register.

#### 12.4.2.4.2.3.2 EPWM Time-Base Counter Synchronization

A time-base synchronization scheme connects all of the EPWM modules on a device. Each EPWM module has a synchronization input (EPWMxSYNCl) and a synchronization output (EPWMxSYNCO). The input synchronization for the first instance (EPWM0) comes from an external pin. For the device EPWM environment sync pin details refer to *EPWM Environment*. The possible synchronization connections for the remaining EPWM modules is shown in [Figure 12-392](#).



epwm-011

**Figure 12-392. EPWM Time-Base Counter Synchronization Scheme 1**

Each EPWM module can be configured to use or ignore the synchronization input. If the EPWM\_TBCTL[2] PHSEN bit is set, then the time-base counter (TBCNT) of the EPWM module (EPWM\_TBCNT register) will be automatically loaded with the phase register (EPWM\_TBPHS) contents when one of the following conditions occur:

- **EPWMxSYNCl: Synchronization Input Pulse:** The value of the phase register is loaded into the counter register when an input synchronization pulse is detected (EPWM\_TBPHS → EPWM\_TBCNT). This operation occurs on the next valid time-base clock (TBCLK) edge.
- **Software Forced Synchronization Pulse:** Writing a 1h to the EPWM\_TBCTL[6] SWFSYNC control bit invokes a software forced synchronization. This pulse is ORed with the synchronization input signal, and therefore has the same effect as a pulse on EPWMxSYNCl.

This feature enables the EPWM module to be automatically synchronized to the time base of another EPWM module. Lead or lag phase control can be added to the waveforms generated by different EPWM modules to synchronize them. In up-down-count mode, the EPWM\_TBCTL[13] PHSDIR bit configures the direction of the time-base counter immediately after a synchronization event. The new direction is independent of the direction prior to the synchronization event. The TBPHS bit is ignored in count-up or count-down modes. See [Figure 12-393](#) through [Figure 12-396](#) for examples.

Clearing the EPWM\_TBCTL[2] PHSEN bit configures the EPWM to ignore the synchronization input pulse. The synchronization pulse can still be allowed to flow-through to the EPWMxSYNCO and be used to synchronize other EPWM modules. In this way, a controller time-base (for example, EPWM1) and downstream modules (EPWM2–EPWMx) can be set and they may select to run in synchronization with the controller.

#### 12.4.2.4.2.4 Phase Locking the Time-Base Clocks of Multiple EPWM Modules

As already described in *EPWM Modules Time-Base Clock Gating*, TB\_CLKEN bit in the EPWMn\_CTRL (where n = 0 to 5) register of the device CTRL\_MMR0 can be used to individually control or globally synchronize the time-base clocks of all enabled EPWM modules on a device. When all TB\_CLKEN bits are set to 0b0, the

time-base clocks of all EPWMx (where x = 0 to 5) modules are stopped (default). When all TB\_CLKEN bits are simultaneously set in software to 0b1, all EPWMx modules time-base clocks are started with the rising edge of TBCLK aligned. For perfectly synchronized TBCLKs, the prescaler bits in the EPWM\_TBCTL register of each EPWM module must be set identically. The proper procedure for enabling the EPWM clocks is as follows:

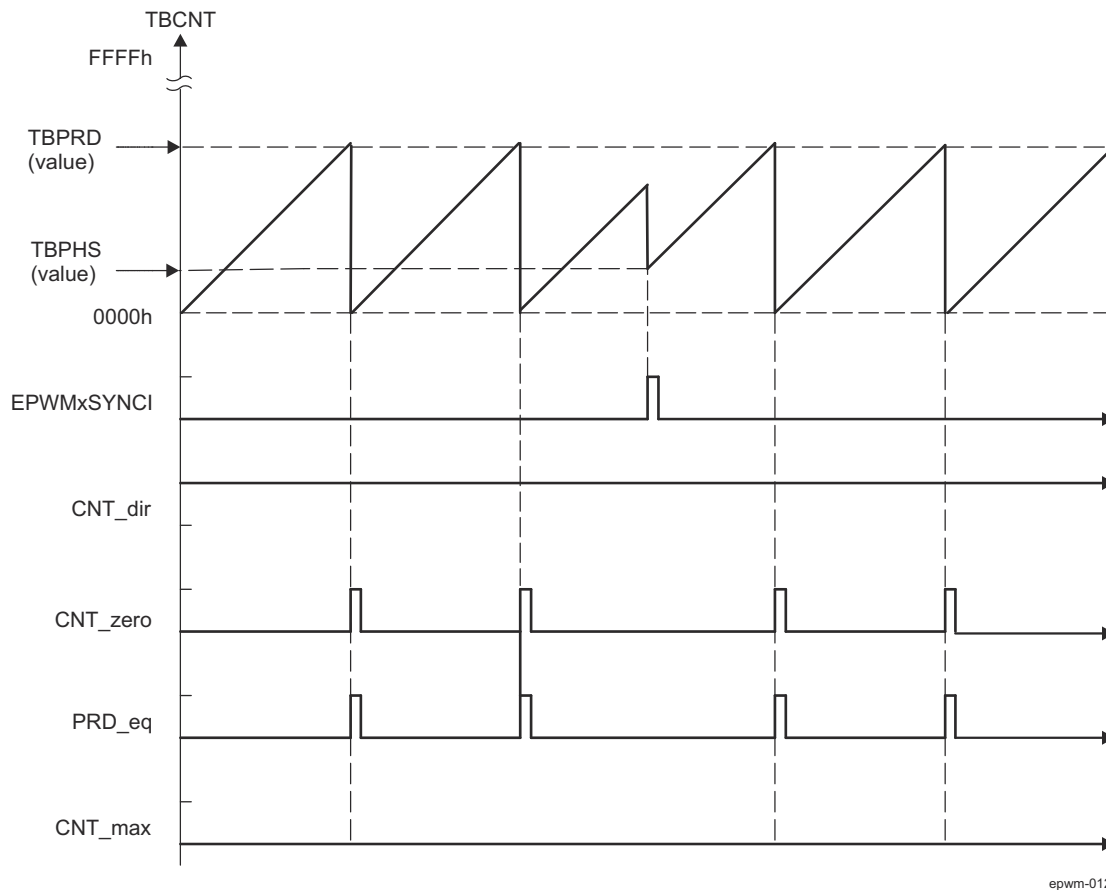
1. Enable the EPWM module clocks.
2. Set TB\_CLKEN = 0. This will stop the time-base clock within any enabled EPWMx module.
3. Configure the prescaler values and desired EPWM modes per each involved EPWMx.
4. Simultaneously set TB\_CLKEN bits to 0b1.

#### 12.4.2.4.2.5 EPWM Time-Base Counter Modes and Timing Waveforms

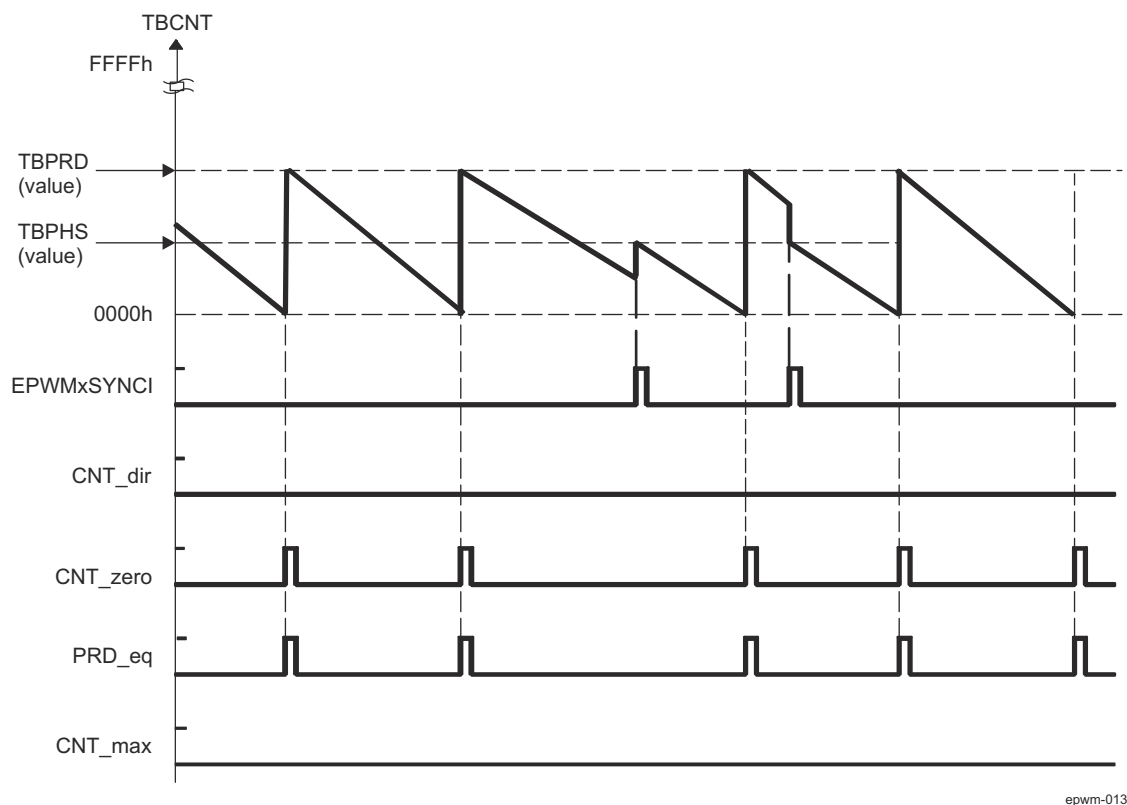
The time-base counter operates in one of four modes:

- Up-count mode which is asymmetrical.
- Down-count mode which is asymmetrical.
- Up-down-count which is symmetrical.
- Frozen where the time-base counter is held constant at the current value.

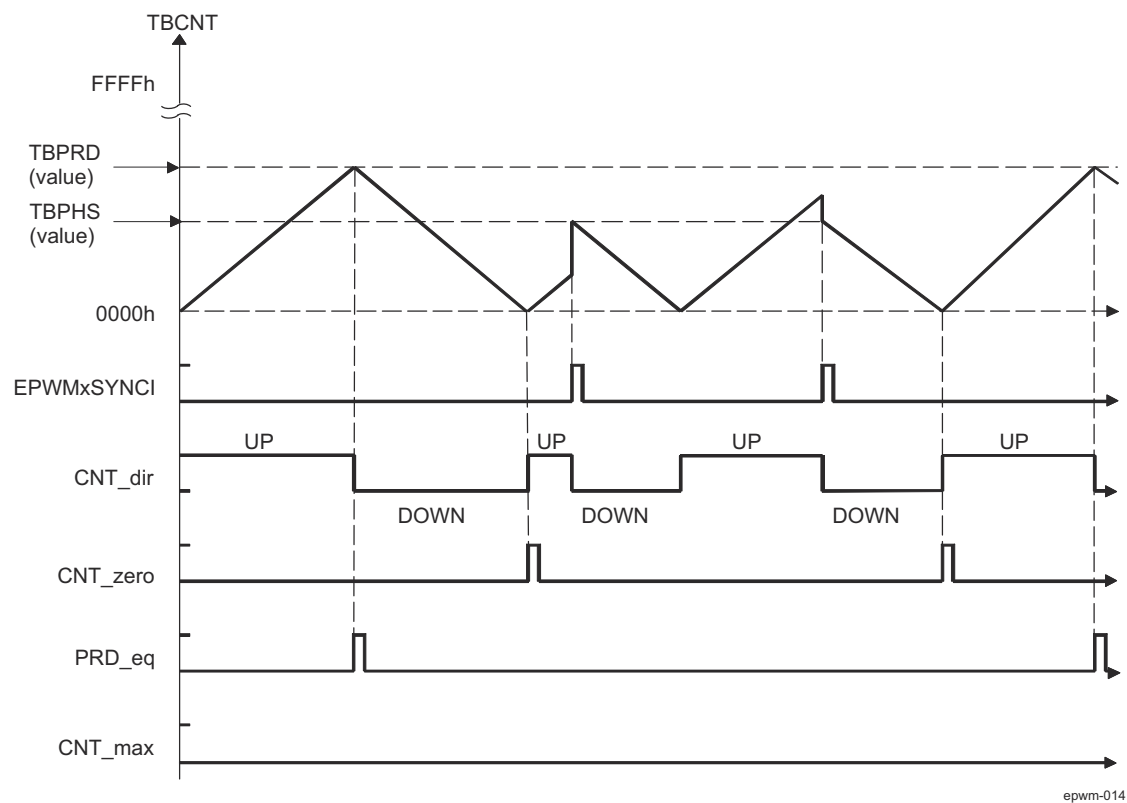
To illustrate the operation of the first three modes, [Figure 12-393](#) to [Figure 12-396](#) show when events are generated and how the time-base responds to an EPWMxSYNCl signal.



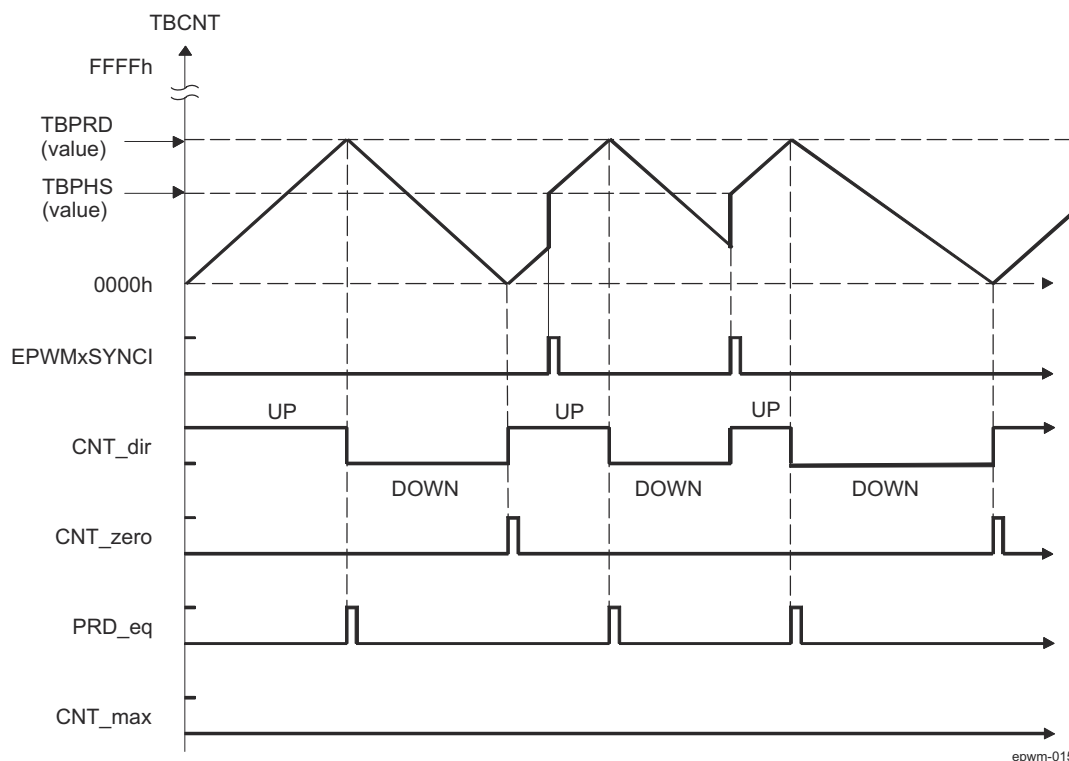
**Figure 12-393. EPWM Time-Base Up-Count Mode Waveforms**



**Figure 12-394. EPWM Time-Base Down-Count Mode Waveforms**



**Figure 12-395. EPWM Time-Base Up-Down-Count Waveforms, EPWM\_TBCTL[13] PHSDIR = 0 Count Down on Synchronization Event**



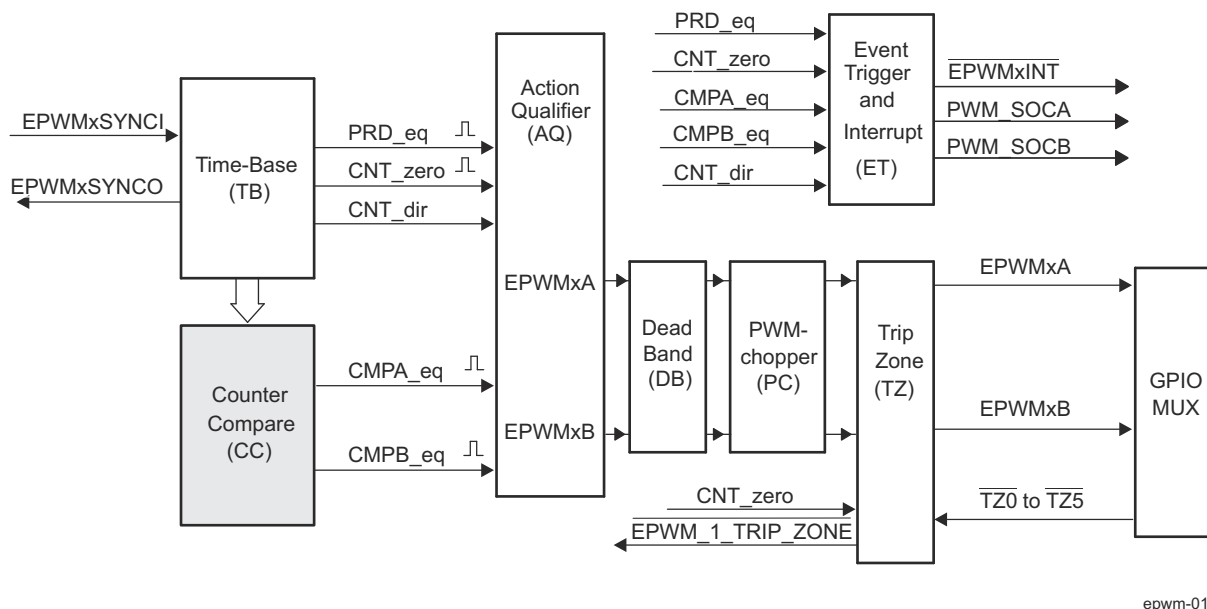
**Figure 12-396. EPWM Time-Base Up-Down Count Waveforms, EPWM\_TBCTL[13] PHSDIR = 1 Count Up on Synchronization Event**

### 12.4.2.4.3 EPWM Counter-Compare (CC) Submodule

This section describes the Counter-Compare (CC) submodule in the PWM module.

#### 12.4.2.4.3.1 Overview

Figure 12-397 illustrates the counter-compare submodule within the EPWM. Figure 12-398 shows the basic structure of the counter-compare submodule.



epwm-016

**Figure 12-397. EPWM Counter-Compare Submodule**

CC module features:

- Generates events based on programmable time stamps using the EPWM\_CMPA and EPWM\_CMPB registers
  - CMPA\_eq (Time-base counter equals counter-compare A register (TBCNT = CMPA)).
  - CMPB\_eq (Time-base counter equals counter-compare B register (TBCNT = CMPB)).
- Controls the PWM duty cycle if the action-qualifier submodule is configured appropriately
- Shadows new compare values to prevent corruption or glitches during the active PWM cycle.

The counter-compare submodule takes as input the time-base counter value. This value is continuously compared to the counter-compare A (EPWM\_CMPA) and counter-compare B (EPWM\_CMPB) registers. When the time-base counter is equal to one of the compare registers, the counter-compare unit generates an appropriate event.

#### 12.4.2.4.3.2 Controlling and Monitoring the EPWM Counter-Compare Submodule

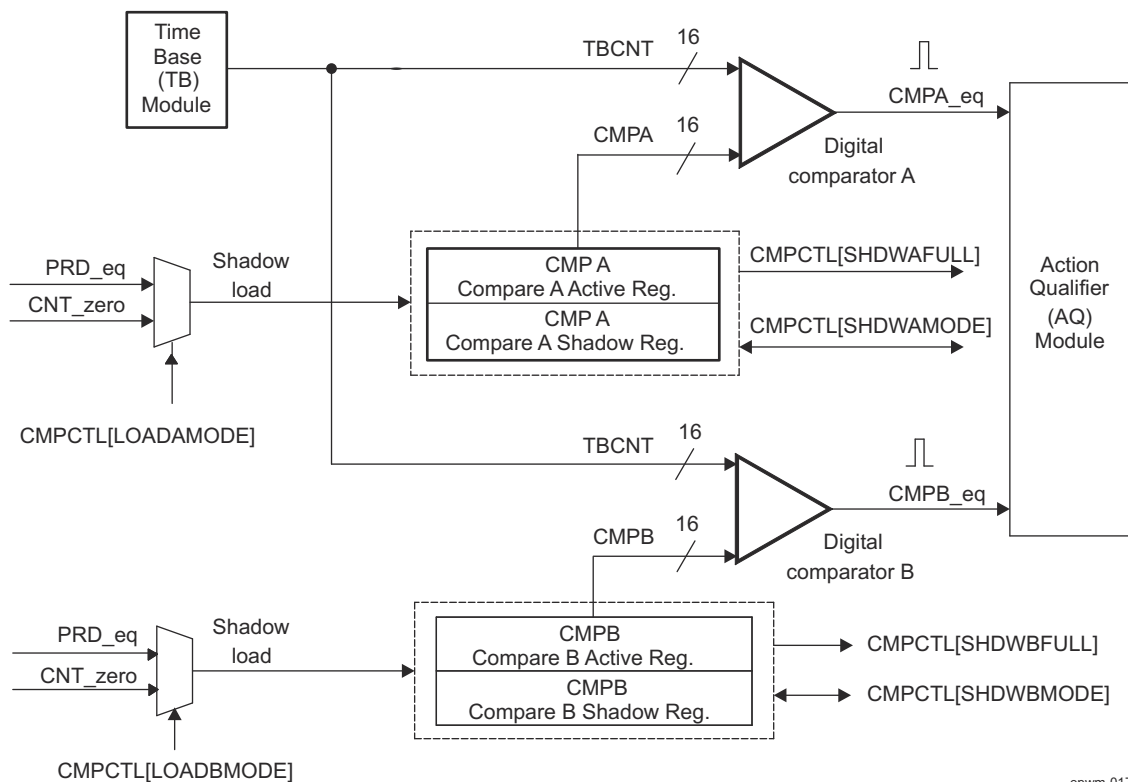
Table 12-430 lists the registers used to control and monitor the counter-compare submodule. Table 12-431 lists the key signals associated with the counter-compare submodule.

**Table 12-430. EPWM Counter-Compare Submodule Registers**

Acronym	Register Description	Address Offset	Shadowed
EPWM_CMPCTL	Counter-Compare Control Register.	Eh	No
HRPWM_CMPAHR	HRPWM Counter-Compare A Extension Register <sup>(1)</sup>	10h	Yes
EPWM_CMPA	Counter-Compare A Register	12h	Yes
EPWM_CMPB	Counter-Compare B Register	14h	Yes

(1) This register is available only on the EPWM modules with the high-resolution extension (HRPWM). On the EPWM modules that do not include the HRPWM, this location is reserved. See *EPWM Integration* to determine which EPWM instances include this feature.





epwm-017

**Figure 12-398. EPWM Counter-Compare Submodule Signals and Registers**

**Table 12-431. EPWM Counter-Compare Submodule Key Signals**

Signal	Description of Event	Register Bitfields Compared
CMPA_eq	Time-base counter equal to the active counter-compare A value	TBCNT = CMPA
CMPB_eq	Time-base counter equal to the active counter-compare B value	TBCNT = CMPB
PRD_eq	Time-base counter equal to the active period. Used to load active counter-compare A and B registers from the shadow register	TBCNT = TBPRD
CNT_zero	Time-base counter equal to zero. Used to load active counter-compare A and B registers from the shadow register	TBCNT = 0000h

### 12.4.2.4.3.3 Operational Highlights for the EPWM Counter-Compare Submodule

The counter-compare submodule is responsible for generating two independent compare events based on two compare registers:

1. CMPA: Time-base counter equal to counter-compare A register (EPWM\_TBCNT = EPWM\_CMPA).
2. CMPB: Time-base counter equal to counter-compare B register (EPWM\_TBCNT = EPWM\_CMPB).

For up-count or down-count mode, each event occurs only once per cycle. For up-down-count mode each event occurs twice per cycle, if the compare value is between 0000h and TBPRD; and occurs once per cycle, if the compare value is equal to 0000h or equal to TBPRD. These events are fed into the action-qualifier submodule where they are qualified by the counter direction and converted into actions if enabled. Refer to [Section 12.4.2.4.4](#) for more details.

The counter-compare EPWM\_CMPA and EPWM\_CMPB registers each have an associated shadow register. Shadowing provides a way to keep updates to the registers synchronized with the hardware. When shadowing is used, updates to the active registers only occurs at strategic points. This prevents corruption or spurious operation due to the register being asynchronously modified by software. The memory address of the active register and the shadow register is identical. Which register is written to or read from is determined by the EPWM\_CMPCTL[4] SHDWAMODE and EPWM\_CMPCTL[6] SHDWBMODE bits. These bits enable and disable the EPWM\_CMPA shadow register and EPWM\_CMPB shadow register respectively. The behavior of the two load modes is described below:

- **Shadow Load Mode:**

The shadow mode for the EPWM\_CMPA register is enabled by clearing the EPWM\_CMPCTL[4] SHDWAMODE bit and the shadow register for EPWM\_CMPB register is enabled by clearing the EPWM\_CMPCTL[6] SHDWBMODE bit. Shadow mode is enabled by default for both EPWM\_CMPA and EPWM\_CMPB register.

If the shadow register is enabled then the content of the shadow register is transferred to the active register on one of the following events:

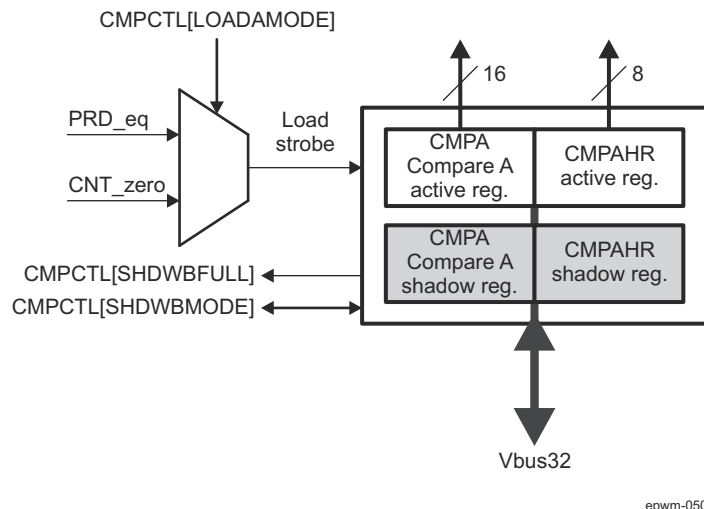
- PRD\_eq: Time-base counter equal to the period (TBCNT = TBPRD).
- CNT\_zero: Time-base counter equal to zero (TBCNT = 0000h)
- Both PRD\_eq and CNT\_zero events occurrence

Which of these three events will drive the counter compare module is specified by the EPWM\_CMPCTL[1-0] LOADAMODE and EPWM\_CMPCTL[3-2] LOADBMODE register bit fields. Only the active register contents are used by the counter-compare submodule to generate events to be sent to the action-qualifier.

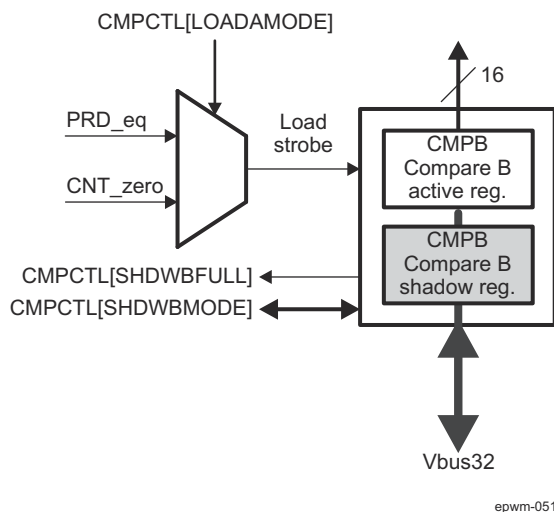
- **Immediate Load Mode:**

If immediate load mode is selected (EPWM\_CMPCTL[4] SHDWAMODE = 1h or EPWM\_CMPCTL[6] SHDWBMODE = 1h), then a read from or a write to the register will go directly to the active register.

[Figure 12-399](#) and [Figure 12-400](#) show Compare A and B Dual Shadow registers.



**Figure 12-399. Compare A Dual Shadow register**



**Figure 12-400. Compare B Dual Shadow register**

#### Note

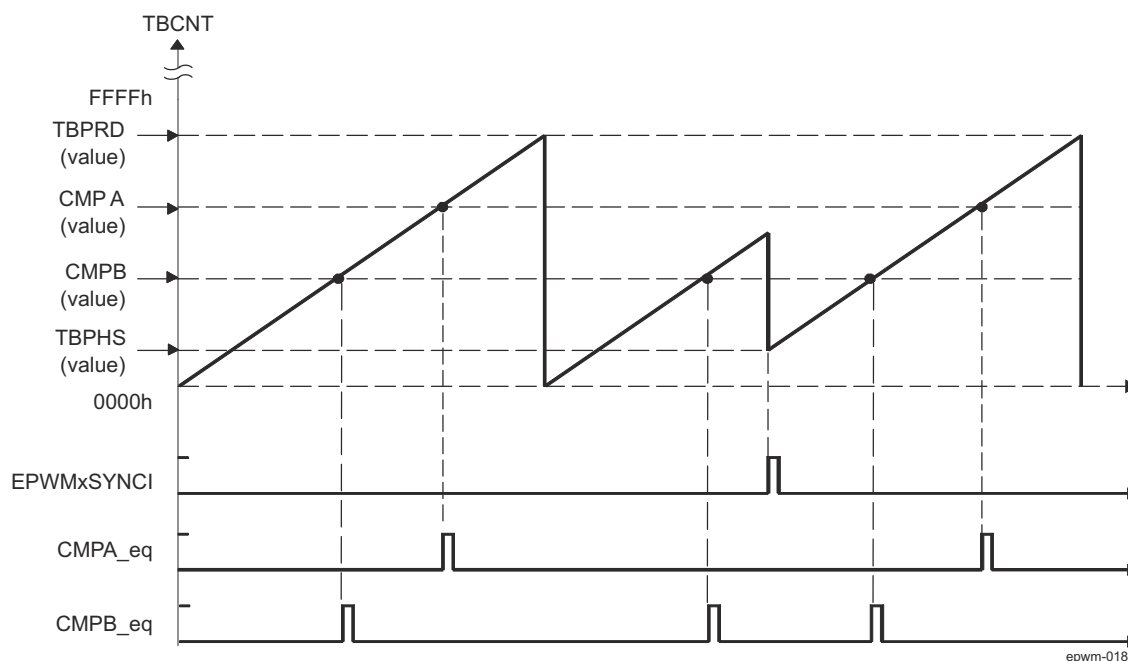
In HRPWM mode the user must perform a single 32-bit write access to both the HRPWM\_CMPAHR and EPWM\_CMPA registers. Two 16-bit accesses are not allowed.

#### 12.4.2.4.3.4 EPWM Counter-Compare Submodule Timing Waveforms

As described in [Section 12.4.2.4.2](#), the Time Base (TB) module can be configured to operate in 3 distinct count modes:

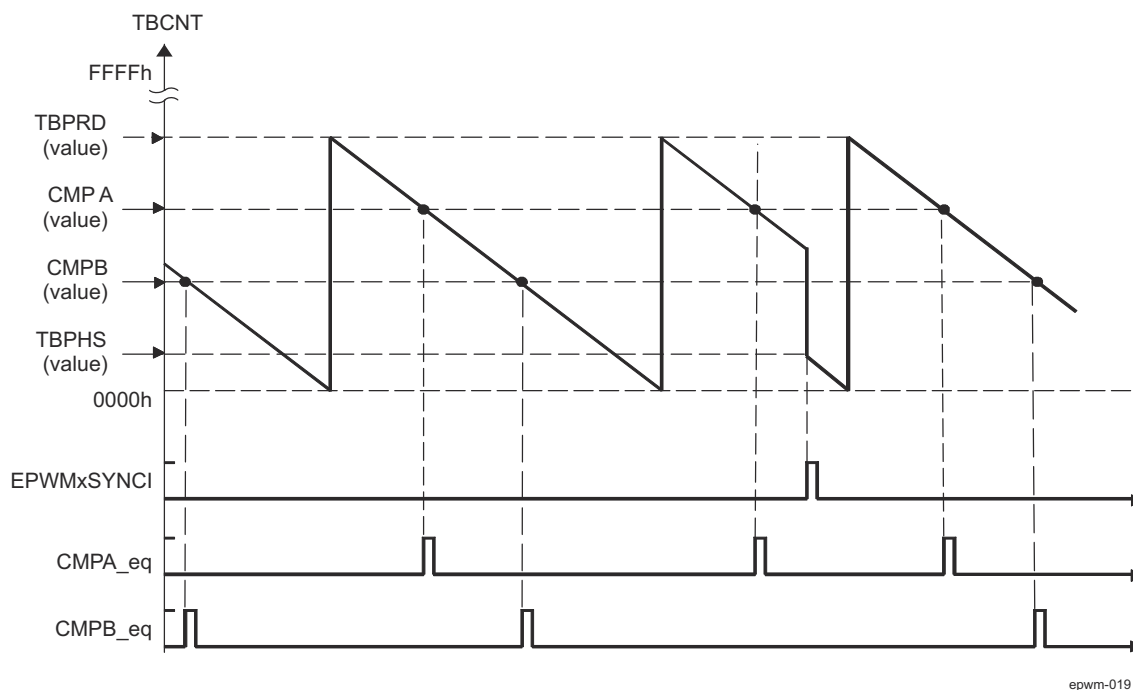
- Up-count mode: used to generate an asymmetrical PWM waveform.
- Down-count mode: used to generate an asymmetrical PWM waveform.
- Up-down-count mode: used to generate a symmetrical PWM waveform.

The timing diagrams in [Figure 12-401](#) to [Figure 12-404](#) show how CMPA and CMPB events are generated in each of the 3 count modes and how the EPWMxSYNCI signal interacts.

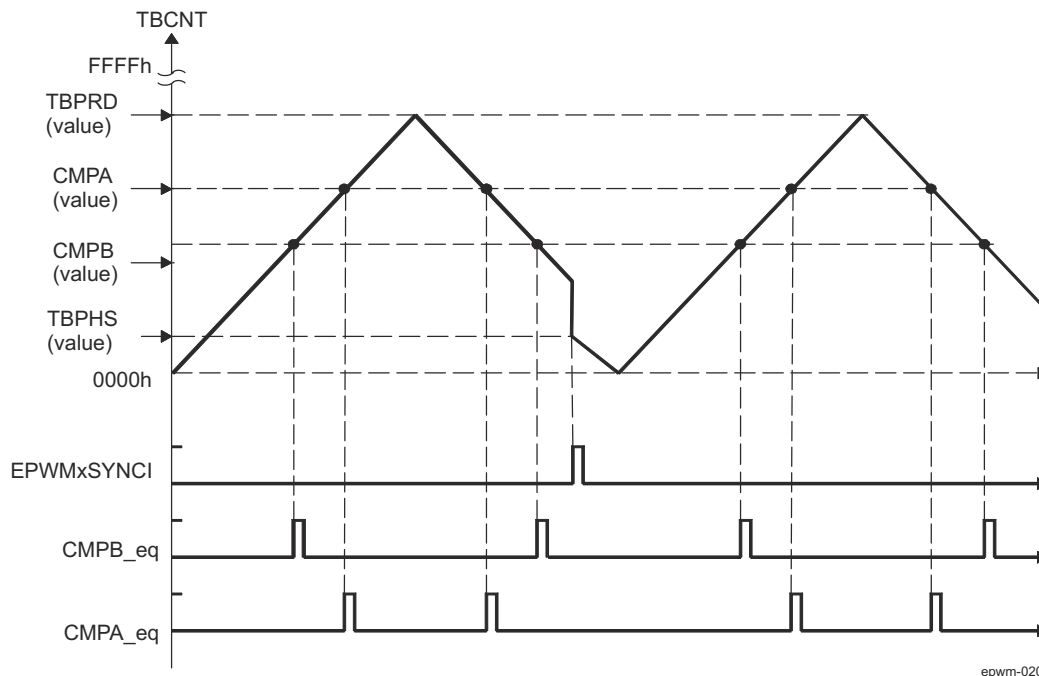


An EPWMxSYNCl external synchronization event can cause a discontinuity in the TBCNT count sequence. This can lead to a compare event being skipped. This skipping is considered normal operation and must be taken into account.

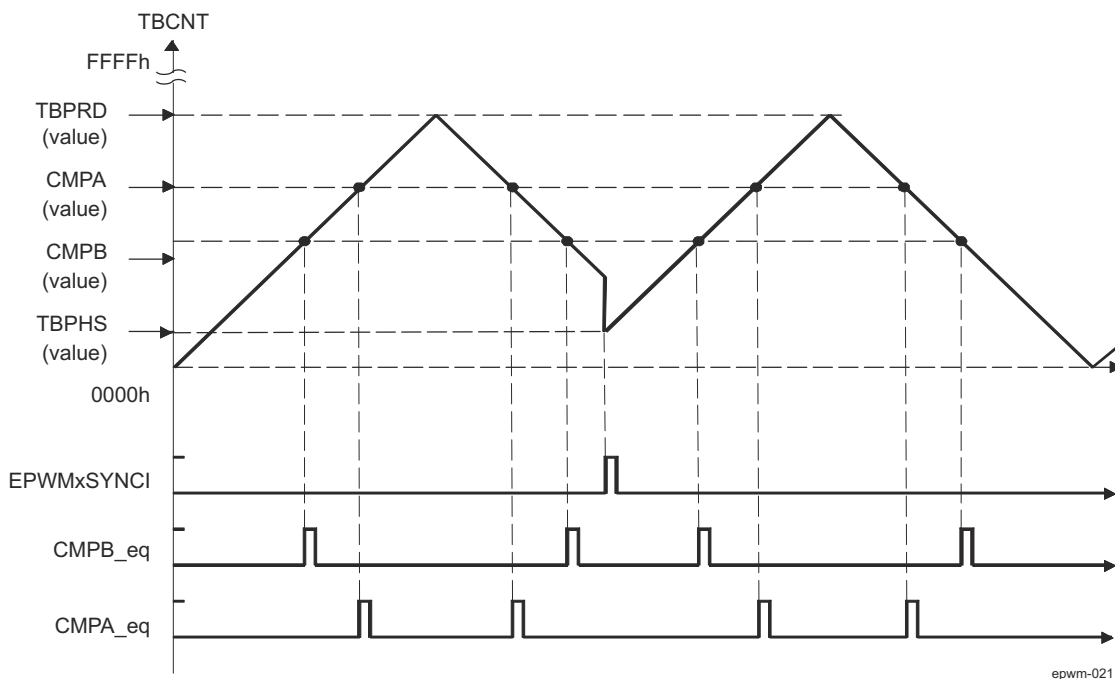
**Figure 12-401. EPWM Counter-Compare Event Waveforms in Up-Count Mode**



**Figure 12-402. EPWM Counter-Compare Events in Down-Count Mode**



**Figure 12-403. EPWM Counter-Compare Events in Up-Down-Count Mode, EPWM\_TBCTL[13] PHSDIR = 0  
Count Down on Synchronization Event**



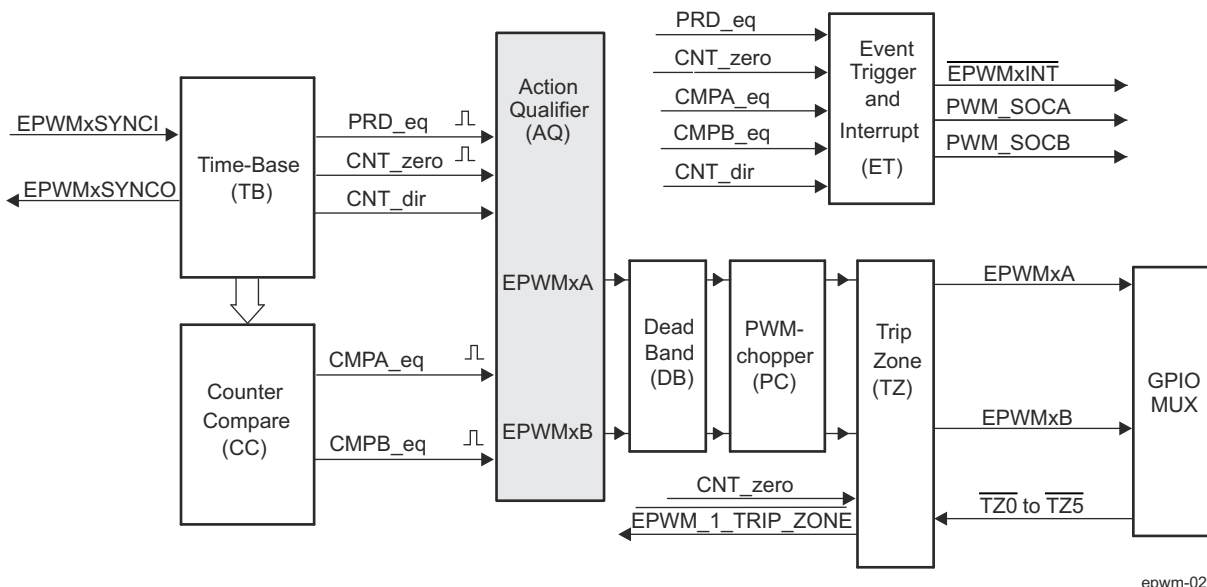
**Figure 12-404. EPWM Counter-Compare Events in Up-Down-Count Mode, EPWM\_TBCTL[13] PHSDIR = 1  
Count Up on Synchronization Event**

#### 12.4.2.4.4 EPWM Action-Qualifier (AQ) Submodule

This section describes the Action-Qualifier (AQ) submodule in the PWM module.

##### 12.4.2.4.4.1 Overview

Figure 12-405 shows the action-qualifier (AQ) submodule in the EPWM system. This submodule has the most important role in waveform construction and PWM generation. It decides which events are converted into various action types, thereby producing the required switched waveforms at the EPWMxA and EPWMxB outputs.



epwm-022

**Figure 12-405. EPWM Action-Qualifier Submodule**

AQ module features:

- Qualifying and generating actions (set, clear, toggle) based on the following events:
  - PRD\_eq**: Time-base counter equal to the period (TBCNT = TBPRD)
  - CNT\_zero**: Time-base counter equal to zero (TBCNT = 0000h)
  - CMPA\_eq**: Time-base counter equal to the counter-compare A register (TBCNT = CMPA)
  - CMPB\_eq**: Time-base counter equal to the counter-compare B register (TBCNT = CMPB)
- Managing priority when these events occur concurrently
- Providing independent control of events when the time-base counter is increasing and when it is decreasing

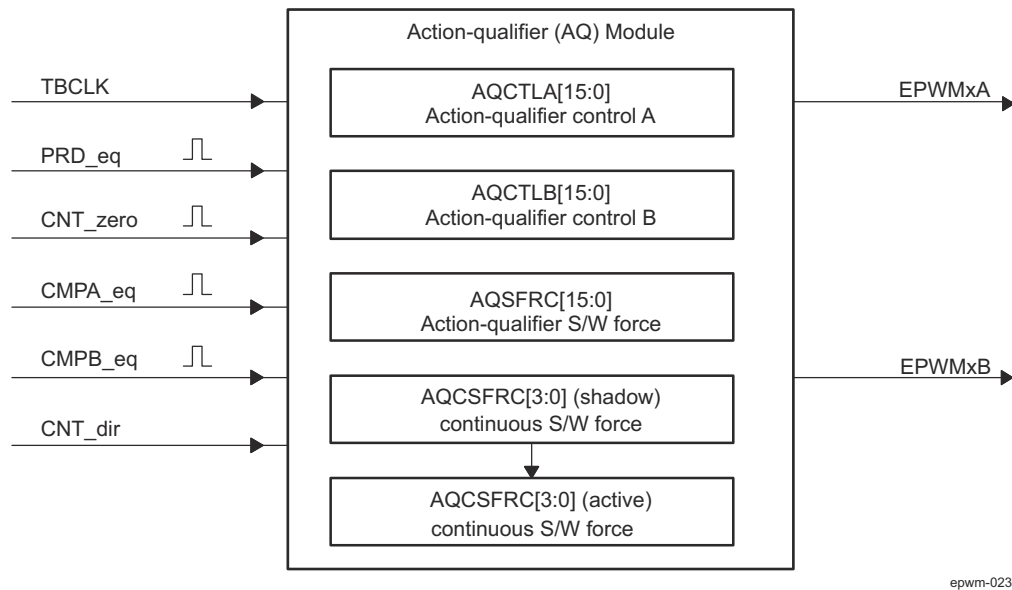
##### 12.4.2.4.4.2 Controlling and Monitoring the EPWM Action-Qualifier Submodule

Table 12-432 lists the registers used to control and monitor the action-qualifier submodule.

**Table 12-432. EPWM Action-Qualifier Submodule Registers**

Acronym	Register Description	Address Offset	Shadowed
EPWM_AQCTLA	Action-Qualifier Control Register For Output A (EPWMxA)	16h	No
EPWM_AQCTLB	Action-Qualifier Control Register For Output B (EPWMxB)	18h	No
EPWM_AQSFRC	Action-Qualifier Software Force Register	1Ah	No
EPWM_AQCSFRC	Action-Qualifier Continuous Software Force	1Ch	Yes

The action-qualifier submodule is based on event-driven logic. It can be thought of as a programmable cross switch with events at the input and actions at the output, all of which are software controlled via the set of registers as shown in Figure 12-406. The possible input events are summarized again in Table 12-433.



**Figure 12-406. EPWM Action-Qualifier Submodule Inputs and Outputs**

**Table 12-433. EPWM Action-Qualifier Submodule Possible Input Events**

Signal	Description	Register Bitfield Compared
PRD_eq	Time-base counter equal to the period value	TBCNT = TBPRD
CNT_zero	Time-base counter equal to zero	TBCNT = 0000h
CMPA_eq	Time-base counter equal to the counter-compare A	TBCNT = CMPA
CMPB_eq	Time-base counter equal to the counter-compare B	TBCNT = CMPB
Software forced event	Asynchronous event initiated by software	

The software forced action is a useful asynchronous event. This control is handled by EPWM\_AQSFRC and EPWM\_AQCSFRC registers.

The action-qualifier submodule controls how the two outputs EPWMxA and EPWMxB behave when a particular event occurs. The event inputs to the action-qualifier submodule are further qualified by the counter direction (up or down). This allows for independent action on outputs on both the count-up and count-down phases.

The possible actions imposed on outputs EPWMxA and EPWMxB are:

- **Set High:** Set output EPWMxA or EPWMxB to a high level.
- **Clear Low:** Set output EPWMxA or EPWMxB to a low level.
- **Toggle:** If EPWMxA or EPWMxB is currently pulled high, then pull the output low. If EPWMxA or EPWMxB is currently pulled low, then pull the output high.
- **Do Nothing:** Keep outputs EPWMxA and EPWMxB at same level as currently set. Although the "Do Nothing" option prevents an event from causing an action on the EPWMxA and EPWMxB outputs, this event can still trigger interrupts. See the Event\_Trigger (ET) submodule description in [Section 12.4.2.4.8](#) for details.

Actions are specified independently for either output (EPWMxA or EPWMxB). Any or all events can be configured to generate actions on a given output. All qualifier actions are configured via the control registers found at the end of this section.

For clarity, the drawings in this chapter use a set of symbolic actions. These symbols are summarized in [Figure 12-407](#). Each symbol represents an action as a marker in time. Some actions are fixed in time (zero and period) while the CMPA and CMPB actions are moveable and their time positions are programmed via the counter-compare A and B registers, respectively. To turn off or disable an action, use the "Do Nothing option"; it is the default at reset.

S/W force	TB Counter equals:				Actions
	Zero	Comp A	Comp B	Period	
<div>SW ×</div>	<div>Z ×</div>	<div>CA ×</div>	<div>CB ×</div>	<div>P ×</div>	Do Nothing
<div>SW ↓</div>	<div>Z ↓</div>	<div>CA ↓</div>	<div>CB ↓</div>	<div>P ↓</div>	Clear Low
<div>SW ↑</div>	<div>Z ↑</div>	<div>CA ↑</div>	<div>CB ↑</div>	<div>P ↑</div>	Set High
<div>SW T</div>	<div>Z T</div>	<div>CA T</div>	<div>CB T</div>	<div>P T</div>	Toggle

epwm-024

**Figure 12-407. Possible Action-Qualifier Actions for EPWMxA and EPWMxB Outputs**



#### 12.4.2.4.4.3 EPWM Action-Qualifier Event Priority

It is possible for the EPWM action qualifier to receive more than one event at the same time. In this case events are assigned a priority by the hardware. The general rule is: events occurring later in time have a higher priority and software forced events always have the highest priority. The event priority levels for up-down-count mode are shown in [Table 12-434](#). A priority level 1 is the highest priority and level 7 is the lowest. The priority changes slightly depending on the direction of TBCNT.

**Table 12-434. EPWM Action-Qualifier Event Priority for Up-Down-Count Mode**

Priority Level	Event if TBCNT is Incrementing TBCNT = 0 up to TBCNT = TBPRD	Event if TBCNT is Decrementing TBCNT = TBPRD down to TBCNT = 1
1 (Highest)	Software forced event	Software forced event
2	Counter equals CMPB on up-count (CBU)	Counter equals CMPB on down-count (CBD)
3	Counter equals CMPA on up-count (CAU)	Counter equals CMPA on down-count (CAD)
4	Counter equals zero	Counter equals period (TBPRD in EPWM_TBPRD active register)
5	Counter equals CMPB on down-count (CBD) <sup>(1)</sup>	Counter equals CMPB on up-count (CBU) <sup>(1)</sup>
6 (Lowest)	Counter equals CMPA on down-count (CAD) <sup>(1)</sup>	Counter equals CMPA on up-count (CBU) <sup>(1)</sup>

(1) To maintain symmetry for up-down-count mode, both up-events (CAU/CBU) and down-events (CAD/CBD) can be generated for TBPRD. Otherwise, up-events can occur only when the counter is incrementing and down-events can occur only when the counter is decrementing.

[Table 12-435](#) shows the action-qualifier priority for up-count mode. In this case, the counter direction is always defined as up and thus down-count events will never be taken.

**Table 12-435. EPWM Action-Qualifier Event Priority for Up-Count Mode**

Priority Level	Event
1 (Highest)	Software forced event
2	Counter equal to period (TBPRD)
3	Counter equal to CMPB on up-count (CBU)
4	Counter equal to CMPA on up-count (CAU)
5 (Lowest)	Counter equal to Zero

[Table 12-436](#) shows the action-qualifier priority for down-count mode. In this case, the counter direction is always defined as down and thus up-count events will never be taken.

**Table 12-436. EPWM Action-Qualifier Event Priority for Down-Count Mode**

Priority Level	Event
1 (Highest)	Software forced event
2	Counter equal to Zero
3	Counter equal to CMPB on down-count (CBD)
4	Counter equal to CMPA on down-count (CAD)
5 (Lowest)	Counter equal to period (TBPRD)

It is possible to set the compare value greater than the period. In this case the action will take place as shown in [Table 12-437](#).

**Table 12-437. Behavior if CMPA/CMPB is Greater than the Period**

Counter Mode	Compare on Up-Count Event CAU/CBU	Compare on Down-Count Event CAU/CBU
Up-Count Mode	If CMPA/CMPB $\leq$ TBPRD period, then the event occurs on a compare match (TBCNT = CMPA or CMPB). If CMPA/CMPB $>$ TBPRD, then the event will not occur.	Never occurs.
Down-Count Mode	Never occurs.	If CMPA/CMPB $<$ TBPRD, the event will occur on a compare match (TBCNT = CMPA or CMPB). If CMPA/CMPB $\geq$ TBPRD, the event will occur on a period match (TBCNT = TBPRD).
Up-Down-Count Mode	If CMPA/CMPB $<$ TBPRD and the counter is incrementing, the event occurs on a compare match (TBCNT = CMPA or CMPB). If CMPA/CMPB is $\geq$ TBPRD, the event will occur on a period match (TBCNT = TBPRD).	If CMPA/CMPB $<$ TBPRD and the counter is decrementing, the event occurs on a compare match (TBCNT = CMPA or CMPB). If CMPA/CMPB $\geq$ TBPRD, the event occurs on a period match (TBCNT = TBPRD).

#### 12.4.2.4.4.4 Waveforms for Common EPWM Configurations

##### Note

The waveforms in this chapter show the EPWMs behavior for a static compare register value. In a running system, the active compare registers (EPWM\_CMPA and EPWM\_CMPB) are typically updated from their respective shadow registers once every period. The user specifies when the update will take place — either when the time-base counter reaches zero or when the time-base counter reaches period. There are some cases when the action based on the new value can be delayed by one period or the action based on the old value can take effect for an extra period. Some PWM configurations avoid this situation. These include, but are not limited to, the following:

##### Use up-down-count mode to generate a symmetric PWM:

- If EPWM\_CMPA / EPWM\_CMPB is loaded on zero, then use EPWM\_CMPA / EPWM\_CMPB values greater than or equal to 1.
- If EPWM\_CMPA / EPWM\_CMPB is loaded on period, then use EPWM\_CMPA / EPWM\_CMPB values less than or equal to TBPRD - 1.

This means there will always be a pulse of at least one TBCLK cycle in a PWM period which, when very short, tend to be ignored by the system.

##### Use up-down-count mode to generate an asymmetric PWM:

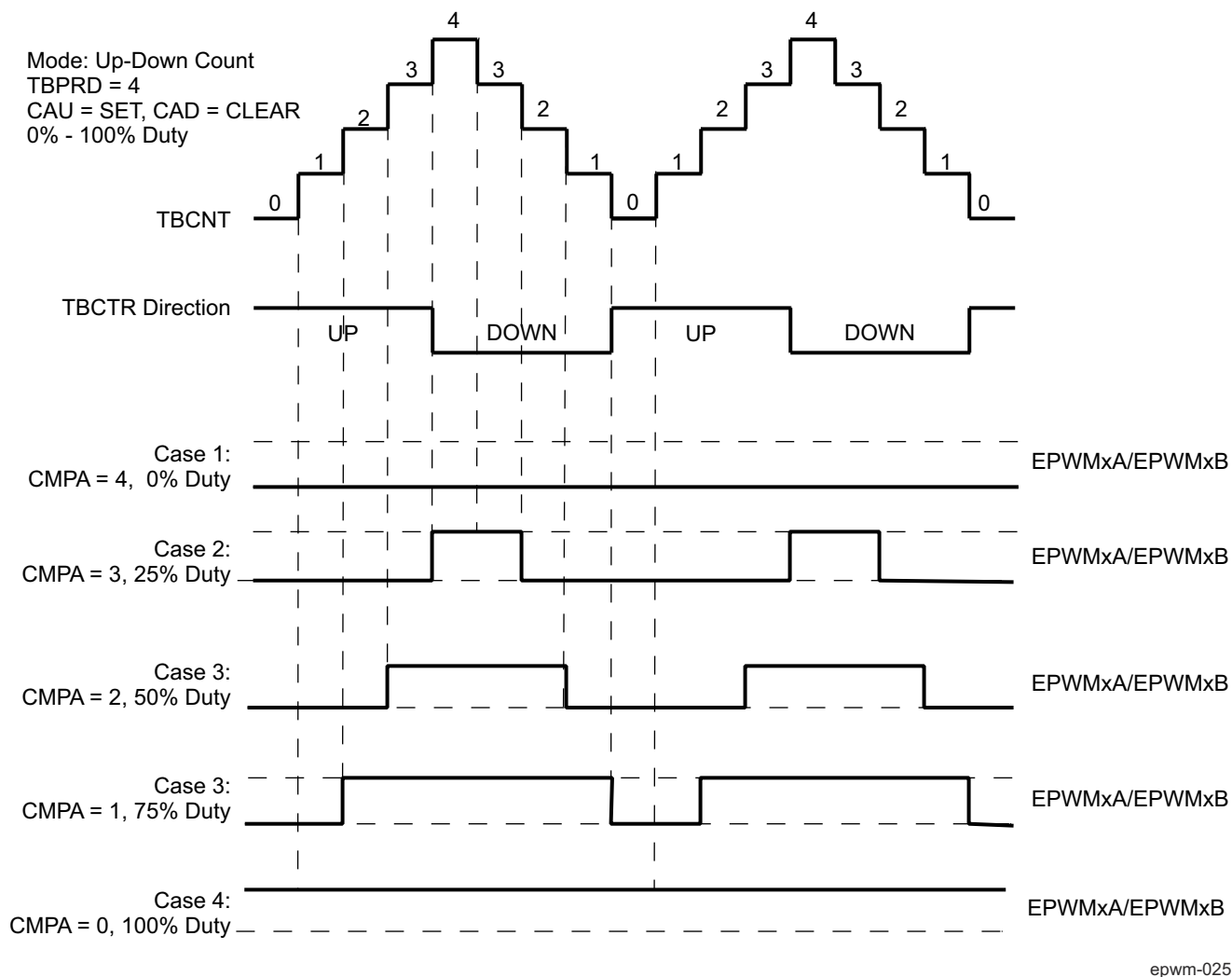
- To achieve 50-0% asymmetric PWM use the following configuration: Load EPWM\_CMPA / EPWM\_CMPB on period and use the period action to clear the PWM and a compare-up action to set the PWM. Modulate the compare value from 0 to TBPRD to achieve 50-0% PWM duty.

##### When using up-count mode to generate an asymmetric PWM:

- To achieve 0-100% asymmetric PWM use the following configuration: Load EPWM\_CMPA / EPWM\_CMPB on TBPRD. Use the Zero action to set the PWM and a compare-up action to clear the PWM. Modulate the compare value from 0 to TBPRD+1 to achieve 0-100% PWM duty.

Figure 12-408 shows how a symmetric PWM waveform can be generated using the up-down-count mode of the TBCNT. In this mode 0-100% DC modulation is achieved by using equal compare matches on the up count and down count portions of the waveform. In the example shown, CMPA is used to make the comparison. When the counter is incrementing the CMPA match will pull the PWM output high. Likewise, when the counter is decrementing the compare match will pull the PWM signal low. When  $CMPA = 0$ , the PWM signal is low for the entire period giving the 0% duty waveform. When  $EPWM\_CMPA = EPWM\_TBPRD$ , the PWM signal is high achieving 100% duty.

When using this configuration in practice, if  $CMPA/CMPB$  is loaded on zero, then use  $CMPA/CMPB$  values greater than or equal to 1. If  $CMPA/CMPB$  is loaded on period, then use  $CMPA/CMPB$  values less than or equal to  $TBPRD - 1$ . This means there will always be a pulse of at least one TBCLK cycle in a PWM period which, when very short, tend to be ignored by the system.



**Figure 12-408. EPWM Up-Down-Count Mode Symmetrical Waveform**

- TBPRD, CMPA, and CMPB refer to the value written in their respective registers (EPWM\_TBPRD, EPWM\_CMPA, and EPWM\_CMPB). The active register, not the shadow register, is used by the hardware.
- CMPx, refers to either CMPA or CMPB.
- EPWMxA and EPWMxB refer to the output signals from EPWMx
- Up-Down means Count-up-and-down mode, Up means up-count mode and Dwn means down-count mode
- Sym = Symmetric, Asym = Asymmetric

- Figure 12-409. Up, Single Edge Asymmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Active High**

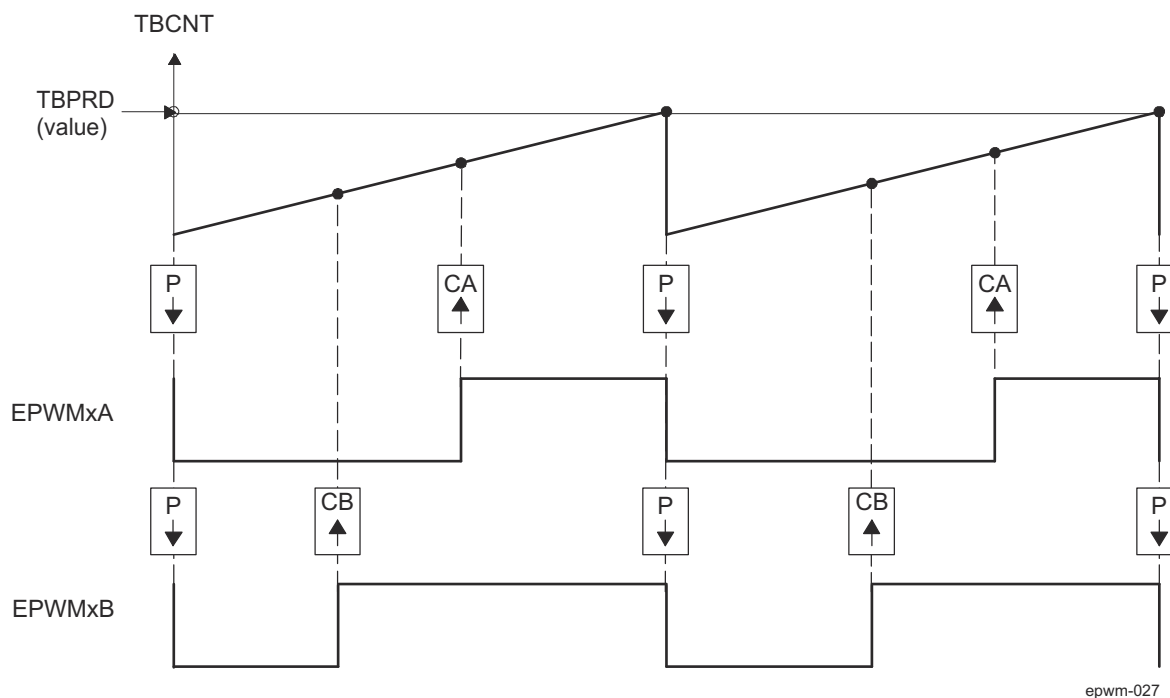
**Table 12-438. EPWMx Initialization for Figure 12-409**

Register	Bitfield	Value	Comments
EPWM_TBPRD	TBPRD	600 (258h)	Period = 601 TBCLK counts
EPWM_TBPHS	TBPHS	0	Clear Phase Register to 0
EPWM_TBCNT	TBCNT	0	Clear TB counter
EPWM_TBCTL	CTRMODE	TB_UP	Phase loading disabled
	PHSEN	TB_DISABLE	
	PRDLD	TB_SHADOW	
	SYNCOSEL	TB_SYNC_DISABLE	
	HSPCLKDIV	TB_DIV1	
	CLKDIV	TB_DIV1	
EPWM_CMPA	CMPA	350 (15Eh)	Compare A = 350 TBCLK counts
EPWM_CMPB	CMPB	200 (C8h)	Compare B = 200 TBCLK counts
EPWM_CMPCTL	SHDWAMODE	CC_SHADOW	Load on TBCNT = 0
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	
	LOADBMODE	CC_CTR_ZERO	
EPWM_AQCTLA	ZRO	AQ_SET	
	CAU	AQ_CLEAR	
EPWM_AQCTLB	ZRO	AQ_SET	
	CBU	AQ_CLEAR	

**Table 12-439. EPWMx Run Time Changes for Figure 12-409**

Register	Bitfield	Value	Comments
EPWM_CMPA	CMPA	Duty1A	Adjust duty for output EPWM1A
EPWM_CMPB	CMPB	Duty1B	Adjust duty for output EPWM1B

Table 12-440 and Table 12-441 contains initialization and runtime register configurations for the waveforms in Figure 12-410.



- PWM period =  $(TBPRD + 1) \times T_{TBCLK}$
- Duty modulation for EPWMxA is set by CMPA, and is active low (that is, the low time duty is proportional to CMPA).
- Duty modulation for EPWMxB is set by CMPB and is active low (that is, the low time duty is proportional to CMPB).
- The Do Nothing actions ( X ) are shown for completeness here, but will not be shown on subsequent diagrams.
- Actions at zero and period, although appearing to occur concurrently, are actually separated by one TBCLK period. TBCNT wraps from period to 0000h.

**Figure 12-410. Up, Single Edge Asymmetric Waveform With Independent Modulation on EPWMxA and EPWMxB — Active Low**

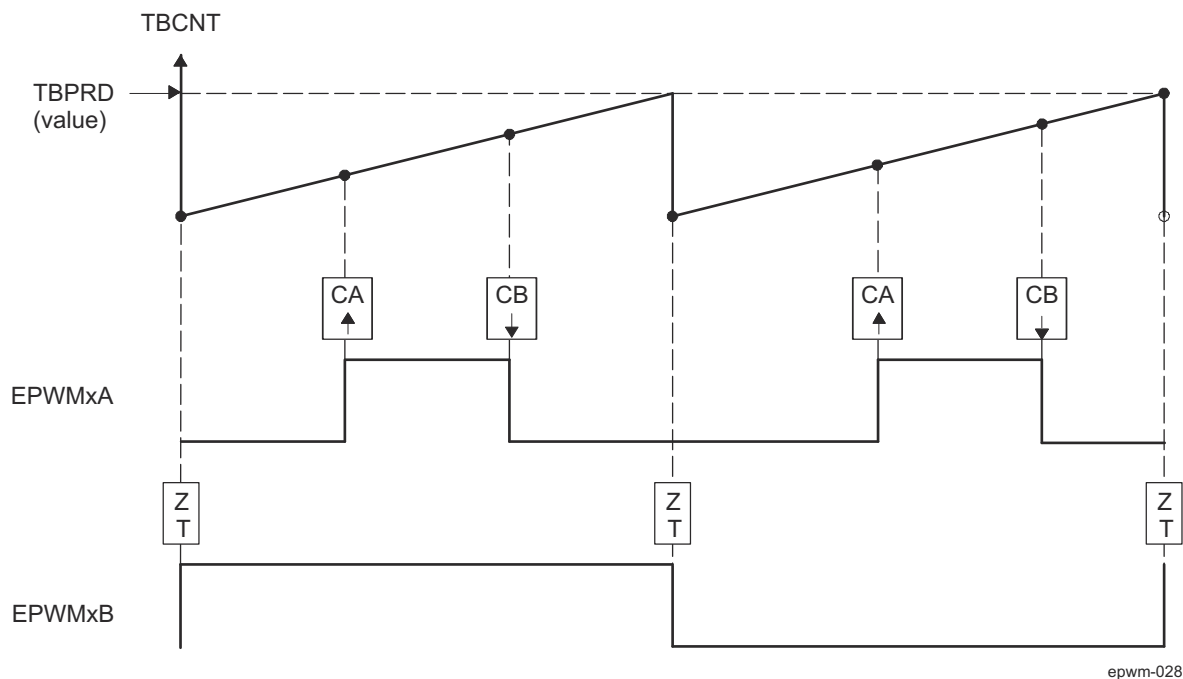
**Table 12-440. EPWMx Initialization for Figure 12-410**

Register	Bitfield	Value	Comments
EPWM_TBPRD	TBPRD	600 (258h)	Period = 601 TBCLK counts
EPWM_TBPHS	TBPHS	0	Clear Phase Register to 0
EPWM_TBCNT	TBCNT	0	Clear TB counter
EPWM_TBCTL	CTRMODE	TB_UP	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDLD	TB_SHADOW	
	SYNCOSEL	TB_SYNC_DISABLE	
	HSPCLKDIV	TB_DIV1	TBCLK = FICLK
	CLKDIV	TB_DIV1	
EPWM_CMPA	CMPA	350 (15Eh)	Compare A = 350 TBCLK counts
EPWM_CMPB	CMPB	200 (C8h)	Compare B = 200 TBCLK counts
EPWM_CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on TBCNT = 0
	LOADBMODE	CC_CTR_ZERO	Load on TBCNT = 0
EPWM_AQCTLA	PRD	AQ_CLEAR	
	CAU	AQ_SET	
EPWM_AQCTLB	PRD	AQ_CLEAR	
	CBU	AQ_SET	

**Table 12-441. EPWMx Run Time Changes for Figure 12-410**

Register	Bit	Value	Comments
EPWM_CMPA	CMPA	Duty1A	Adjust duty for output EPWM1A
EPWM_CMPB	CMPB	Duty1B	Adjust duty for output EPWM1B

Table 12-442 and Table 12-443 contains initialization and runtime register configurations for the waveforms Figure 12-411. Use the code in *Constant Definitions Used in the EPWM Code Examples* to define the headers.



- A.  $\text{PWM frequency} = 1 / ((\text{TBPRD} + 1) \times T_{\text{TBCLK}})$
- B. Pulse can be placed anywhere within the PWM cycle (0000h - TBPRD)
- C. High time duty proportional to (CMPB - CMPA)
- D. EPWMxB can be used to generate a 50% duty square wave with frequency =  $1/2 \times ((\text{TBPRD} + 1) \times \text{TBCLK})$

**Figure 12-411. Up-Count, Pulse Placement Asymmetric Waveform With Independent Modulation on EPWMxA**



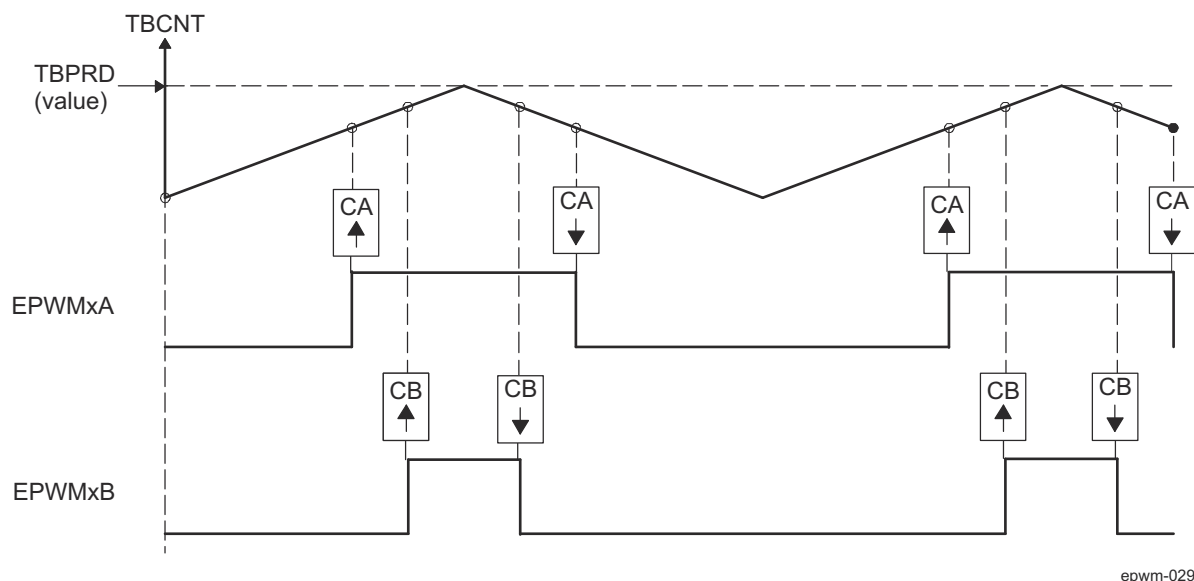
**Table 12-442. EPWMx Initialization for Figure 12-411**

Register	Bitfield	Value	Comments
EPWM_TBPRD	TBPRD	600 (258h)	Period = 601 TBCLK counts
EPWM_TBPHS	TBPHS	0	Clear Phase Register to 0
EPWM_TBCNT	TBCNT	0	Clear TB counter
EPWM_TBCTL	CTRMODE	TB_UP	Phase loading disabled
	PHSEN	TB_DISABLE	
	PRDLD	TB_SHADOW	
	SYNCOSEL	TB_SYNC_DISABLE	
	HSPCLKDIV	TB_DIV1	
	CLKDIV	TB_DIV1	
EPWM_CMPA	CMPA	200 (C8h)	Compare A = 200 TBCLK counts
EPWM_CMPB	CMPB	400 (190h)	Compare B = 400 TBCLK counts
EPWM_CMPCTL	SHDWAMODE	CC_SHADOW	Load on TBCNT = 0
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	
	LOADBMODE	CC_CTR_ZERO	
EPWM_AQCTLA	CAU	AQ_SET	
	CBU	AQ_CLEAR	
EPWM_AQCTLB	ZRO	AQ_TOGGLE	

**Table 12-443. EPWMx Run Time Changes for Figure 12-411**

Register	Bitfield	Value	Comments
EPWM_CMPA	CMPA	EdgePosA	Adjust duty for output EPWM1A
EPWM_CMPB	CMPB	EdgePosB	

Table 12-444 and Table 12-445 contains initialization and runtime register configurations for the waveforms in Figure 12-412. Use the code in *Constant Definitions Used in the EPWM Code Examples* to define the headers.



- A.  $\text{PWM period} = 2 \times \text{TBPRD} \times T_{\text{TBCLK}}$
- B. Duty modulation for EPWMxA is set by CMPA, and is active low (that is, the low time duty is proportional to CMPA).
- C. Duty modulation for EPWMxB is set by CMPB and is active low (that is, the low time duty is proportional to CMPB).
- D. Outputs EPWMxA and EPWMxB can drive independent power switches.

**Figure 12-412. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Active Low**

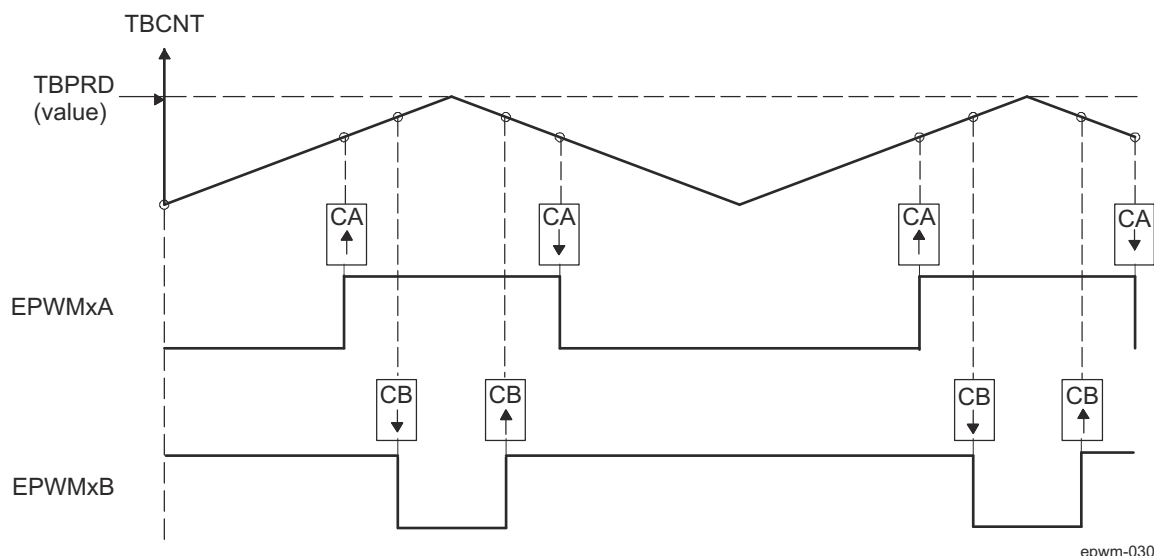
**Table 12-444. EPWMx Initialization for Figure 12-412**

Register	Bitfield	Value	Comments
EPWM_TBPRD	TBPRD	600 (258h)	Period = 601 TBCLK counts
EPWM_TBPHS	TBPHS	0	Clear Phase Register to 0
EPWM_TBCNT	TBCNT	0	Clear TB counter
EPWM_TBCTL	CTRMODE	TB_UPDOWN	Phase loading disabled
	PHSEN	TB_DISABLE	
	PRDLD	TB_SHADOW	
	SYNCOSEL	TB_SYNC_DISABLE	
	HSPCLKDIV	TB_DIV1	
	CLKDIV	TB_DIV1	
EPWM_CMPA	CMPA	400 (190h)	Compare A = 400 TBCLK counts
EPWM_CMPB	CMPB	500 (1F4h)	Compare B = 500 TBCLK counts
EPWM_CMPCTL	SHDWAMODE	CC_SHADOW	Load on TBCNT = 0
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	
	LOADBMODE	CC_CTR_ZERO	
EPWM_AQCTLA	CAU	AQ_SET	
	CAD	AQ_CLEAR	
EPWM_AQCTLB	CBU	AQ_SET	
	CBD	AQ_CLEAR	

**Table 12-445. EPWMx Run Time Changes for Figure 12-412**

Register	Bitfield	Value	Comments
EPWM_CMPA	CMPA	Duty1A	Adjust duty for output EPWM1A
EPWM_CMPB	CMPB	Duty1B	Adjust duty for output EPWM1B

Table 12-446 and Table 12-447 contains initialization and runtime register configurations for the waveforms in Figure 12-413. Use the code in *Constant Definitions Used in the EPWM Code Examples* to define the headers.



- A.  $\text{PWM period} = 2 \times \text{TBPRD} \times T_{\text{TBCLK}}$
- B. Duty modulation for EPWMxA is set by CMPA, and is active low, that is, low time duty proportional to CMPA.
- C. Duty modulation for EPWMxB is set by CMPB and is active high, that is, high time duty proportional to CMPB.
- D. Outputs EPWMx can drive upper/lower (complementary) power switches.
- E. Dead-band =  $\text{CMPB} - \text{CMPA}$  (fully programmable edge placement by software). Note the dead-band module is also available if the more classical edge delay method is required.

**Figure 12-413. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Complementary**

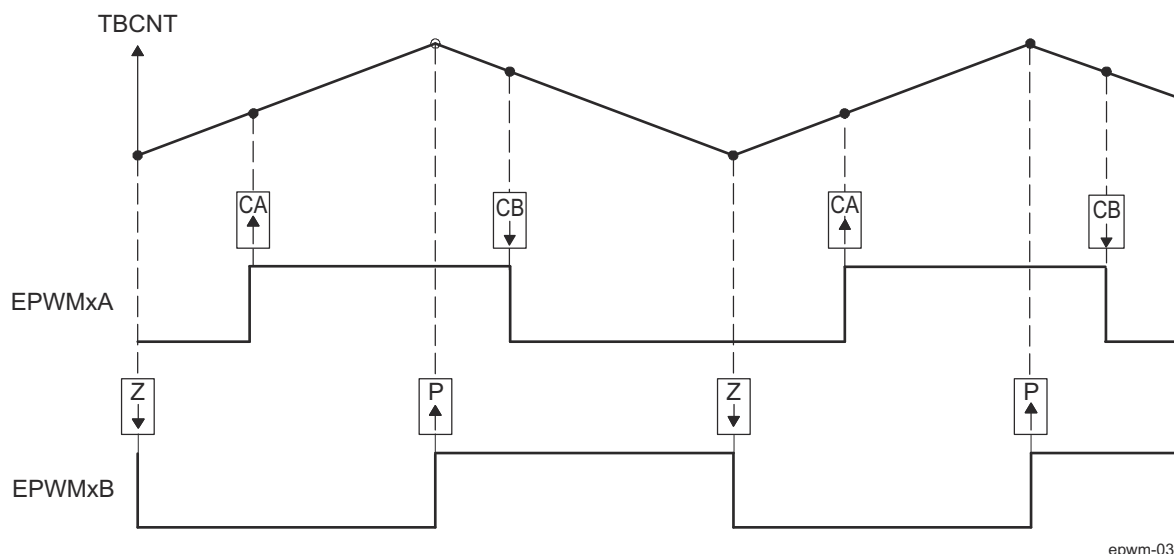
**Table 12-446. EPWMx Initialization for Figure 12-413**

Register	Bitfield	Value	Comments
EPWM_TBPRD	TBPRD	600 (258h)	Period = 601 TBCLK counts
EPWM_TBPHS	TBPHS	0	Clear Phase Register to 0
EPWM_TBCNT	TBCNT	0	Clear TB counter
EPWM_TBCTL	CTRMODE	TB_UPDOWN	Phase loading disabled
	PHSEN	TB_DISABLE	
	PRDLD	TB_SHADOW	
	SYNCOSEL	TB_SYNC_DISABLE	
	HSPCLKDIV	TB_DIV1	
	CLKDIV	TB_DIV1	
EPWM_CMPA	CMPA	350 (15Eh)	Compare A = 350 TBCLK counts
EPWM_CMPB	CMPB	400 (190h)	Compare B = 400 TBCLK counts
EPWM_CMPCTL	SHDWAMODE	CC_SHADOW	Load on TBCNT = 0
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	
	LOADBMODE	CC_CTR_ZERO	
EPWM_AQCTLA	CAU	AQ_SET	
	CAD	AQ_CLEAR	
EPWM_AQCTLB	CBU	AQ_CLEAR	
	CBD	AQ_SET	

**Table 12-447. EPWMx Run Time Changes for Figure 12-413**

Register	Bitfield	Value	Comments
EPWM_CMPA	CMPA	Duty1A	Adjust duty for output EPWM1A
EPWM_CMPB	CMPB	Duty1B	Adjust duty for output EPWM1B

Table 12-448 and Table 12-449 contains initialization and runtime register configurations for the waveforms in Figure 12-414. Use the code in *Constant Definitions Used in the EPWM Code Examples* to define the headers.



- A.  $PWM\ period = 2 \times TBPRD \times TBCLK$
- B. Rising edge and falling edge can be asymmetrically positioned within a PWM cycle. This allows for pulse placement techniques.
- C. Duty modulation for EPWMxA is set by CMPA and CMPB.
- D. Low time duty for EPWMxA is proportional to  $(CMPA + CMPB)$ .
- E. To change this example to active high, CMPA and CMPB actions need to be inverted (that is, Set ! Clear and Clear Set).
- F. Duty modulation for EPWMxB is fixed at 50% (utilizes spare action resources for EPWMxB).

**Figure 12-414. Up-Down-Count, Dual Edge Asymmetric Waveform, With Independent Modulation on EPWMxA — Active Low**

**Table 12-448. EPWMx Initialization for [Figure 12-414](#)**

Register	Bitfield	Value	Comments
EPWM_TBPRD	TBPRD	600 (258h)	Period = 601 TBCLK counts
EPWM_TBPHS	TBPHS	0	Clear Phase Register to 0
EPWM_TBCNT	TBCNT	0	Clear TB counter
EPWM_TBCTL	CTRMODE	TB_UPDOWN	Phase loading disabled
	PHSEN	TB_DISABLE	
	PRDLD	TB_SHADOW	
	SYNCOSEL	TB_SYNC_DISABLE	
	HSPCLKDIV	TB_DIV1	
	CLKDIV	TB_DIV1	
EPWM_CMPA	CMPA	250 (FAh)	Compare A = 250 TBCLK counts
EPWM_CMPB	CMPB	450 (1C2h)	Compare B = 450 TBCLK counts
EPWM_CMPCTL	SHDWAMODE	CC_SHADOW	Load on TBCNT = 0
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	
	LOADBMODE	CC_CTR_ZERO	
EPWM_AQCTLA	CAU	AQ_SET	
	CBD	AQ_CLEAR	
EPWM_AQCTLB	ZRO	AQ_CLEAR	
	PRD	AQ_SET	

**Table 12-449. EPWMx Run Time Changes for [Figure 12-414](#)**

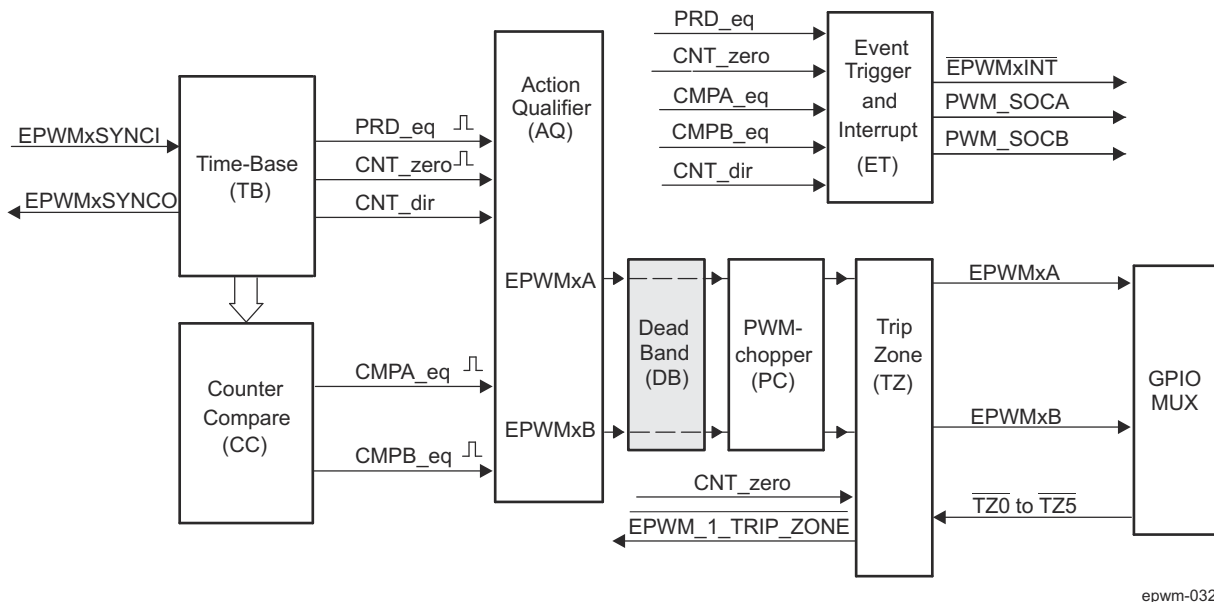
Register	Bitfield	Value	Comments
EPWM_CMPA	CMPA	EdgePosA	Adjust duty for output EPWM1A
EPWM_CMPB	CMPB	EdgePosB	Adjust duty for output EPWM1B

#### 12.4.2.4.5 EPWM Dead-Band Generator (DB) Submodule

This section describes the Dead-Band Generator (DB) submodule in the PWM module.

##### 12.4.2.4.5.1 Overview

Figure 12-415 illustrates the dead-band generator submodule within the EPWM module.



**Figure 12-415. Dead-Band Generator Submodule**

The "Action-qualifier (AQ) Module" section discussed how it is possible to generate the required dead-band by having full control over edge placement using both the CMPA and CMPB resources of the EPWM module. However, if the more classical edge delay-based dead-band with polarity control is required, then the dead-band generator submodule should be used.

The key functions of the dead-band generator submodule are:

- Generating appropriate signal pairs (EPWMxA and EPWMxB) with dead-band relationship from a single EPWMxA input
- Programming signal pairs for:
  - Active high (AH)
  - Active low (AL)
  - Active high complementary (AHC)
  - Active low complementary (ALC)
- Adding programmable delay to rising edges (RED)
- Adding programmable delay to falling edges (FED)
- Can be totally bypassed from the signal path (note dotted lines in Figure 12-415)

##### 12.4.2.4.5.2 Controlling and Monitoring the EPWM Dead-Band Submodule

The dead-band generator submodule operation is controlled and monitored via the following registers:

**Table 12-450. Dead-Band Generator Submodule Registers**

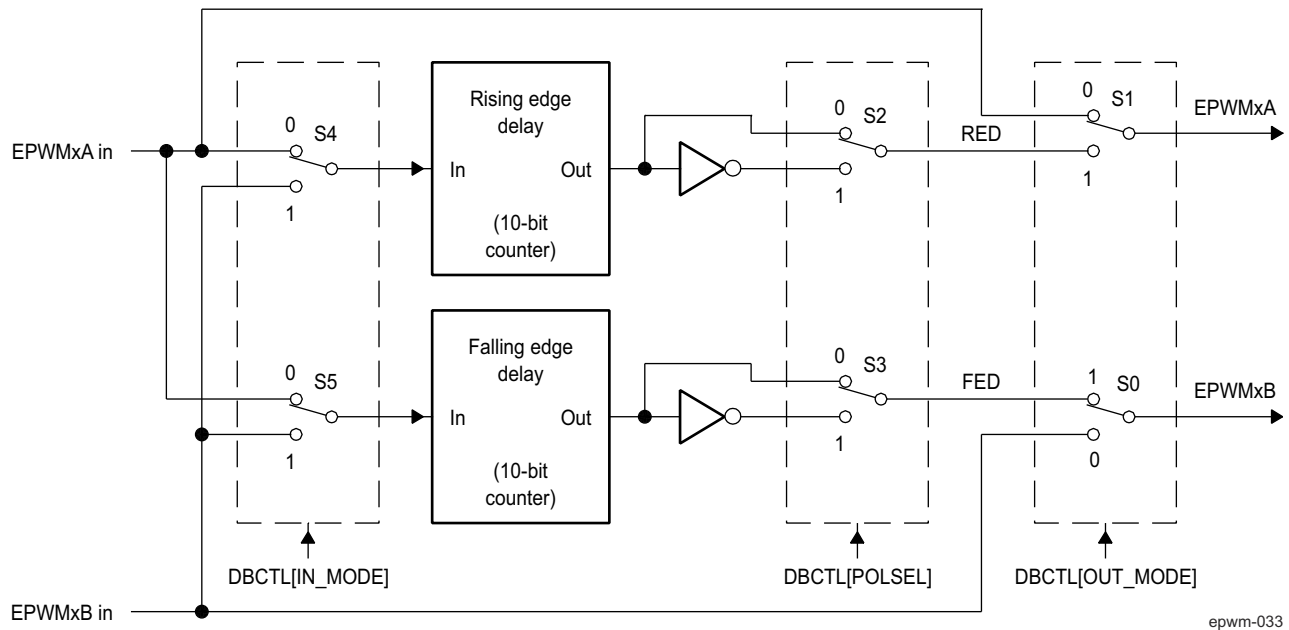
Acronym	Register Description	Address Offset	Shadowed
EPWM_DBCTL	Dead-Band Control Register	1Eh	No
EPWM_DBRED	Dead-Band Rising Edge Delay Count Register	20h	No
EPWM_DBFED	Dead-Band Falling Edge Delay Count Register	22h	No



#### 12.4.2.4.5.3 Operational Highlights for the EPWM Dead-Band Generator Submodule

The dead-band submodule has two groups of independent selection options as shown in Figure 12-416.

- **Input Source Selection:** The input signals to the dead-band module are the EPWMxA and EPWMxB output signals from the action-qualifier. In this section they will be referred to as EPWMxA In and EPWMxB In. Using the EPWM\_DBCTL[5-4] IN\_MODE control bits, the signal source for each delay, falling-edge or rising-edge, can be selected:
  - EPWMxA In is the source for both falling-edge and rising-edge delay. This is the default mode.
  - EPWMxA In is the source for falling-edge delay, EPWMxB In is the source for rising-edge delay.
  - EPWMxA In is the source for rising edge delay, EPWMxB In is the source for falling-edge delay.
  - EPWMxB In is the source for both falling-edge and rising-edge delay.
- **Output Mode Control:** The output mode is configured by way of the EPWM\_DBCTL[1-0] OUT\_MODE bit fields. These bits determine if the falling-edge delay, rising-edge delay, neither, or both are applied to the input signals.
- **Polarity Control:** The polarity control (EPWM\_DBCTL[3-2] POLSEL) allows to be specified whether the rising-edge delayed signal and/or the falling-edge delayed signal is to be inverted before being sent out of the dead-band submodule.



**Figure 12-416. Configuration Options for the EPWM Dead-Band Generator Submodule**

Although all combinations are supported, not all are typical usage modes. [Table 12-451](#) lists some classical dead-band configurations. These modes assume that the EPWM\_DBCTL[5-4] IN\_MODE is configured such that EPWMxA In is the source for both falling-edge and rising-edge delay. Enhanced, or non-traditional modes can be achieved by changing the input signal source. The modes shown in [Table 12-451](#) fall into the following categories:

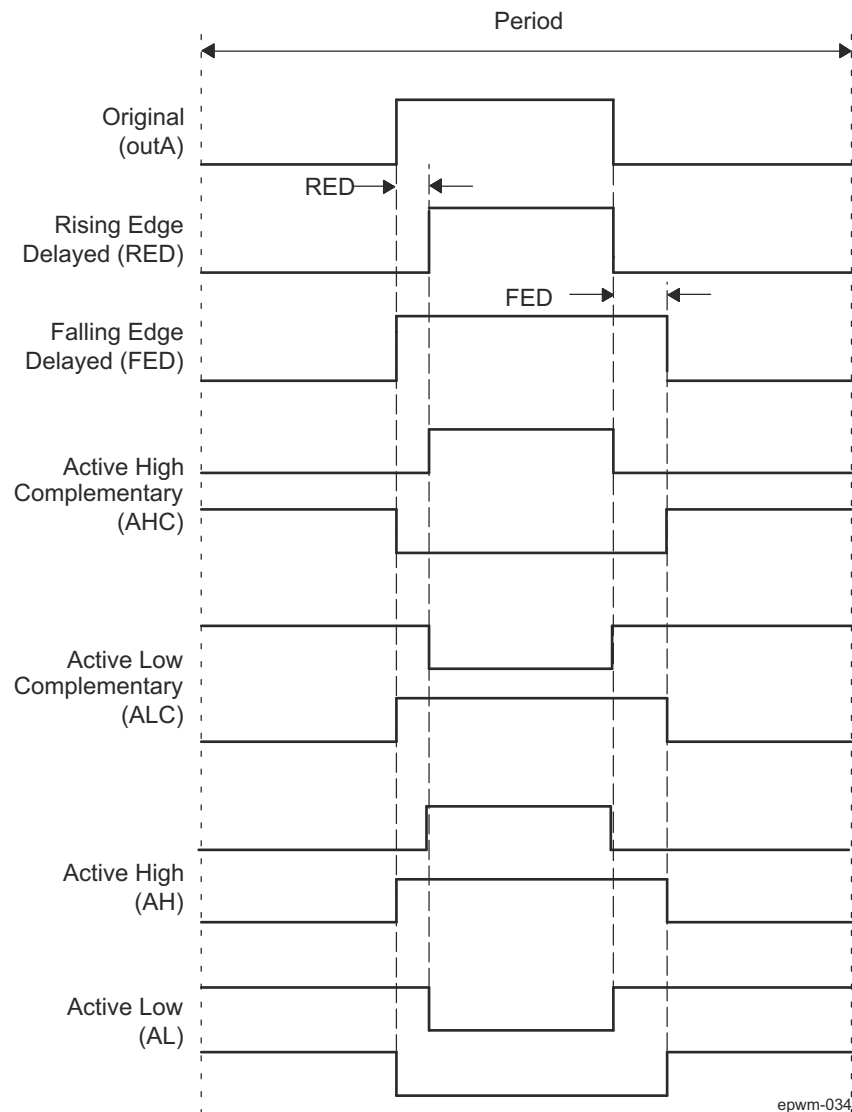
- **Mode 1: Bypass both falling-edge delay (FED) and rising-edge delay (RED)** Allows to be fully disabled the dead-band submodule from the PWM signal path.
- **Mode 2-5: Classical Dead-Band Polarity Settings** These represent typical polarity configurations that should address all the active high/low modes required by available industry power switch gate drivers. The waveforms for these typical cases are shown in [Figure 12-417](#). Note that to generate equivalent waveforms to [Figure 12-417](#), configure the action-qualifier submodule to generate the signal as shown for EPWMxA.
- **Mode 6: Bypass rising-edge-delay and Mode 7: Bypass falling-edge-delay** Finally the last two entries in [Table 12-451](#) show combinations where either the falling-edge-delay (FED) or rising-edge-delay (RED) blocks are bypassed.

**Table 12-451. Classical Dead-Band Operating Modes**

Mode	Mode Description <sup>(1)</sup>	EPWM_DBCTL[3-2] POLSEL		EPWM_DBCTL[1-0] OUT_MODE	
		S3	S2	S1	S0
1	EPWMxA and EPWMxB Passed Through (No Delay)	x	x	0	0
2	Active High Complementary (AHC)	1	0	1	1
3	Active Low Complementary (ALC)	0	1	1	1
4	Active High (AH)	0	0	1	1
5	Active Low (AL)	1	1	1	1
6	EPWMxA Out = EPWMxA In (No Delay) EPWMxB Out = EPWMxA In with Falling Edge Delay	0 or 1	0 or 1	0	1
7	EPWMxA Out = EPWMxA In with Rising Edge Delay EPWMxB Out = EPWMxB In with No Delay	0 or 1	0 or 1	1	0

(1) These are classical dead-band modes and assume that EPWM\_DBCTL[5-4] IN\_MODE = 0b00. That is, EPWMxA in is the source for both the falling-edge and rising-edge delays. Enhanced, non-traditional modes can be achieved by changing the IN\_MODE configuration.

[Figure 12-417](#) shows waveforms for typical cases where 0% < duty < 100%.



**Figure 12-417. Dead-Band Waveforms for Typical Cases (0% < Duty < 100%)**

The dead-band submodule supports independent values for rising-edge (RED) and falling-edge (FED) delays. The amount of delay is programmed using the EPWM\_DBRED and EPWM\_DBFED registers. These are 10-bit registers and their value represents the number of time-base clock, TBCLK, periods a signal edge is delayed by. For example, the formulas to calculate falling-edge-delay and rising-edge-delay are:

$$FED = EPWM\_DBFED \times T_{TBCLK}$$

$$RED = EPWM\_DBRED \times T_{TBCLK}$$

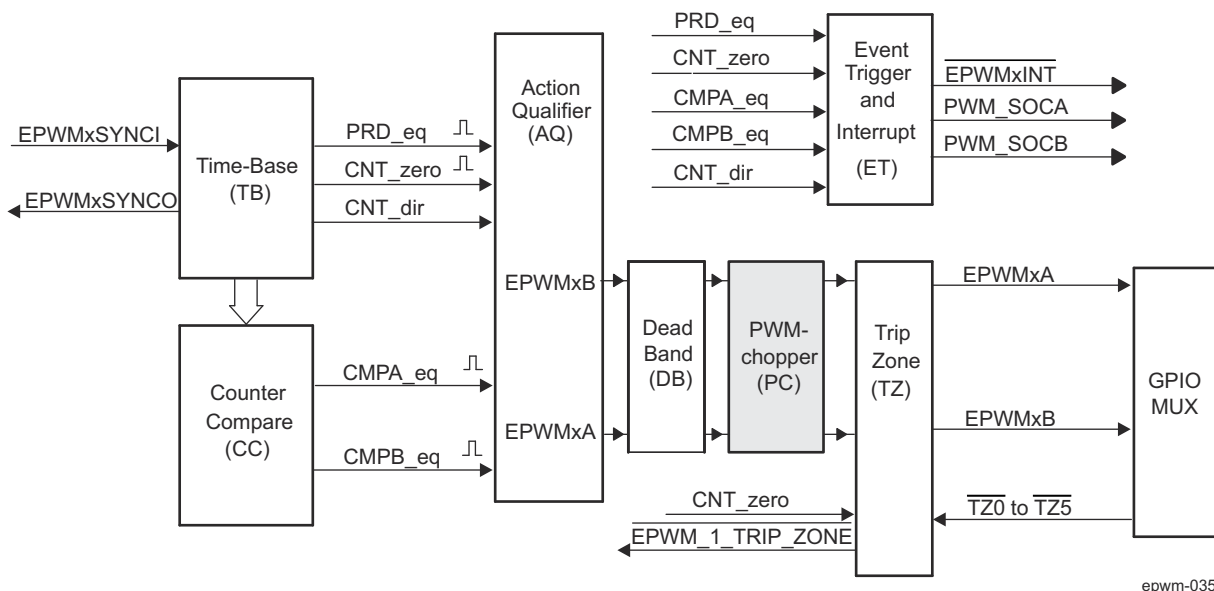
Where  $T_{TBCLK}$  is the period of TBCLK, the prescaled version of FICLK.

#### 12.4.2.4.6 EPWM-Chopper (PC) Submodule

This section describes the PWM-Chopper (PC) submodule in the PWM module.

##### 12.4.2.4.6.1 Overview

Figure 12-418 illustrates the PWM-chopper (PC) submodule within the EPWM module. The PWM-chopper submodule allows a high-frequency carrier signal to modulate the PWM waveform generated by the action-qualifier and dead-band submodules. This capability is important if a pulse transformer-based gate drivers to control the power switching elements is needed.



**Figure 12-418. PWM-Chopper Submodule**

PC module key features:

- Programmable chopping (carrier) frequency
- Programmable pulse width of first pulse
- Programmable duty cycle of second and subsequent pulses
- Can be fully bypassed if not required

##### 12.4.2.4.6.2

PC module key features:

- Programmable chopping (carrier) frequency
- Programmable pulse width of first pulse
- Programmable duty cycle of second and subsequent pulses
- Can be fully bypassed if not required

##### 12.4.2.4.6.3 Controlling the EPWM-Chopper Submodule

The EPWM-chopper submodule operation is controlled via the register in [Table 12-452](#).

**Table 12-452. EPWM-Chopper Submodule Registers**

Acronym	Register Description	Address Offset	Shadowed
EPWM_PCCTL	PWM-chopper Control Register	3Ch	No

#### 12.4.2.4.6.4 Operational Highlights for the EPWM-Chopper Submodule

Figure 12-419 shows the operational details of the EPWM-chopper submodule. The carrier clock is derived from FICLK. Its frequency and duty cycle are controlled via the CHPFREQ and CHPDUTY bits in the EPWM\_PCCTL register. The one-shot block is a feature that provides a high energy first pulse to ensure hard and fast power switch turn on, while the subsequent pulses sustain pulses, ensuring the power switch remains on. The one-shot width is programmed via the OSHTWTH bits. The PWM-chopper submodule can be fully disabled (bypassed) via the CHPEN bit.

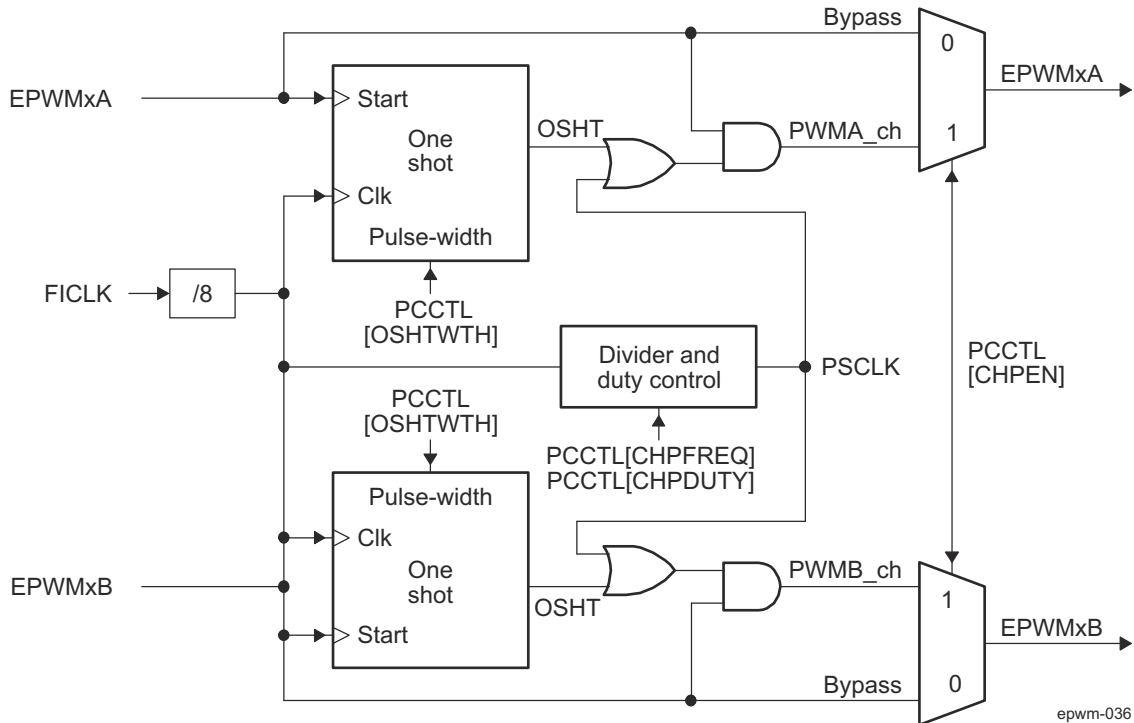
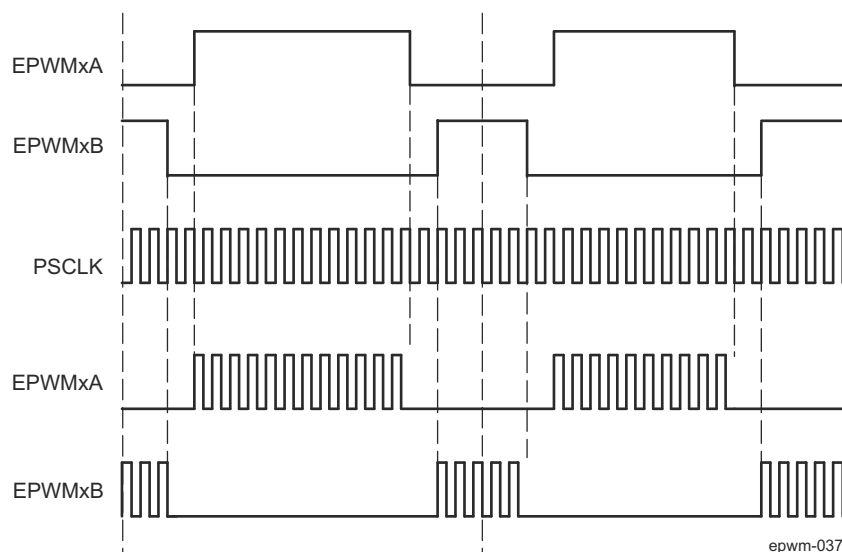


Figure 12-419. PWM-Chopper Submodule Signals and Registers

#### 12.4.2.4.6.5 EPWM-Chopper Waveforms

Figure 12-420 shows simplified waveforms of the chopping action only; one-shot and duty-cycle control are not shown. Details of the one-shot and duty-cycle control are discussed in the following sections.



**Figure 12-420. Simple EPWM-Chopper Submodule Waveforms Showing Chopping Action Only**

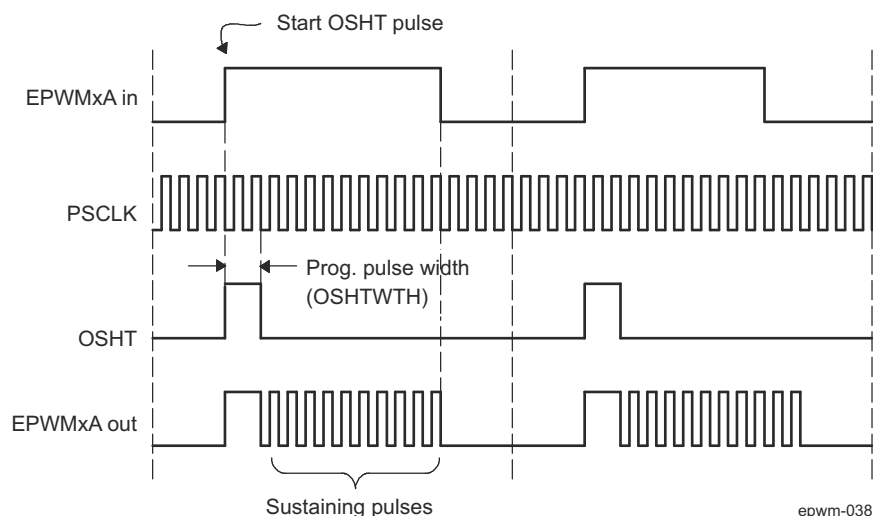
#### 12.4.2.4.6.5.1 EPWM-Chopper One-Shot Pulse

The width of the first pulse can be programmed to any of 16 possible pulse width values. The width or period of the first pulse is given by:

$$T_{1stpulse} = T_{FICLK} \times 8 \times OSHTWTH$$

Where  $T_{FICLK}$  is the period of the system clock (FICLK) and OSHTWTH is the four control bits (value from 1 to 16)

Figure 12-421 shows the first and subsequent sustaining pulses.

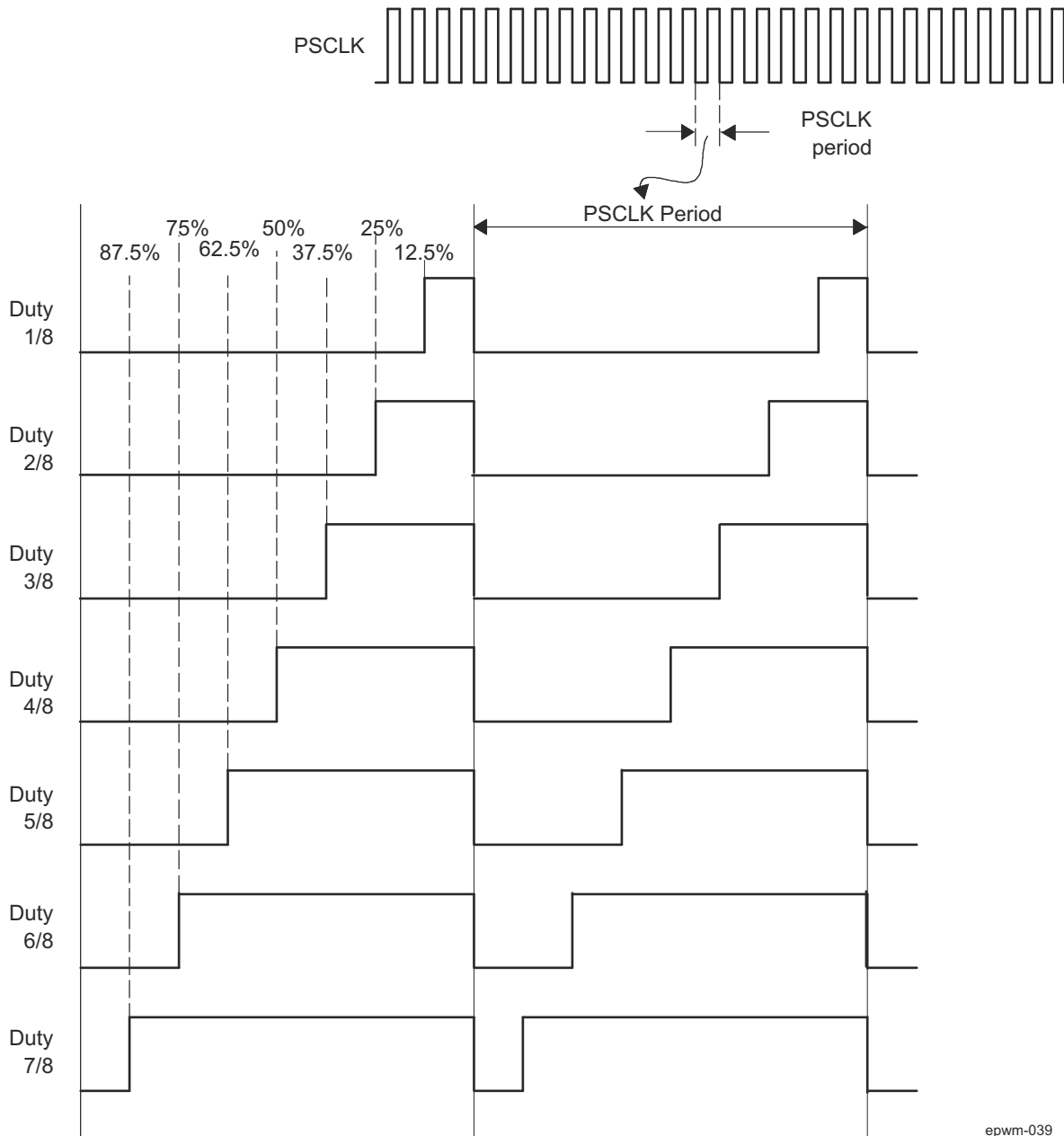


**Figure 12-421. EPWM-Chopper Submodule Waveforms Showing the First Pulse and Subsequent Sustaining Pulses**

#### 12.4.2.4.6.5.2 EPWM-Chopper Duty Cycle Control

Pulse transformer-based gate drive designs need to comprehend the magnetic properties or characteristics of the transformer and associated circuitry. Saturation is one such consideration. To assist the gate drive designer, the duty cycles of the second and subsequent pulses have been made programmable. These sustaining pulses ensure the correct drive strength and polarity is maintained on the power switch gate during the on period, and hence a programmable duty cycle allows a design to be tuned or optimized via software control.

Figure 12-422 shows the duty cycle control that is possible by programming the CHPDUTY bits. One of seven possible duty ratios can be selected ranging from 12.5 to 87.5%.



epwm-039

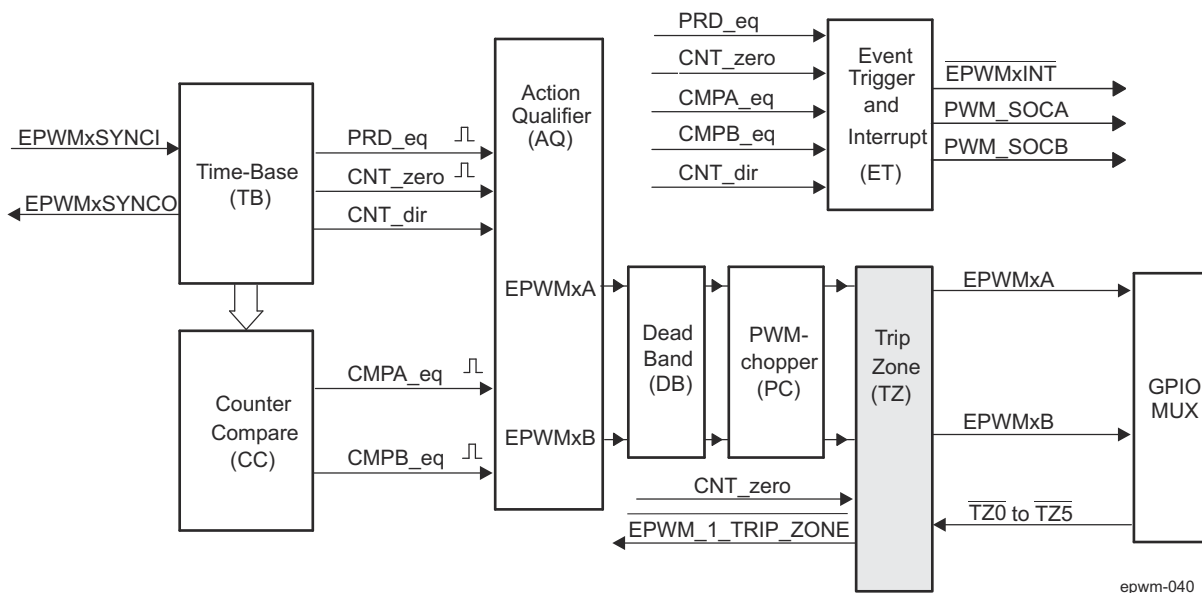
**Figure 12-422. EPWM-Chopper Submodule Waveforms Showing the Pulse Width (Duty Cycle) Control of Sustaining Pulses**

#### 12.4.2.4.7 EPWM Trip-Zone (TZ) Submodule

This section describes the Trip-Zone (TZ) submodule in the PWM module.

##### 12.4.2.4.7.1 Overview

Figure 12-423 shows how the trip-zone (TZ) submodule fits within the EPWM module. Each EPWM module is connected to six  $\overline{TZ}$  signals ( $\overline{TZ0}$  to  $\overline{TZ5}$ ) that are sourced from the GPIO MUX. These signals indicate external fault or trip conditions, and the EPWM outputs can be programmed to respond accordingly when faults occur. See *EPWM Integration* to determine the number of trip-zone pins available for the device.



**Figure 12-423. EPWM Trip-Zone Submodule**

The key functions of the trip-zone submodule are:

- Trip inputs  $\overline{TZ0}$  to  $\overline{TZ5}$  can be flexibly mapped to any EPWM module.
- Upon a fault condition, outputs EPWMxA and EPWMxB can be forced to one of the following:
  - High
  - Low
  - High-impedance
  - No action taken
- Support for one-shot trip (OSHT) for major short circuits or over-current conditions.
- Support for cycle-by-cycle tripping (CBC) for current limiting operation.
- Each trip-zone input pin can be allocated to either one-shot or cycle-by-cycle operation.
- Interrupt generation is possible on any trip-zone pin.
- Software-forced tripping is also supported.
- The trip-zone submodule can be fully bypassed if it is not required.

#### Note

For each EPWMx module, 6 tripzone input events are supported ( $\overline{EPWMx\_TRIP\_TZ[5:0]}$ ). Each tripzone input for all EPWMx modules (where x = 0 to 5) are tied to a common tripzone input pin, so that any EPWMx can be triggered by any of the six tripzone inputs.



#### 12.4.2.4.7.2 Controlling and Monitoring the EPWM Trip-Zone Submodule

The trip-zone submodule operation is controlled and monitored through the following registers:

**Table 12-453. EPWM Trip-Zone Submodule Registers**

Acronym	Register Description	Address Offset	Shadowed
EPWM_TZSEL	Trip-Zone Select Register <sup>(1)</sup>	24h	No
EPWM_TZCTL	Trip-Zone Control Register <sup>(1)</sup>	28h	No
EPWM_TZEINT	Trip-Zone Enable Interrupt Register <sup>(1)</sup>	2Ah	No
EPWM_TZFLG	Trip-Zone Flag Register <sup>(1)</sup>	2Ch	No
EPWM_TZCLR	Trip-Zone Clear Register <sup>(1)</sup>	2Eh	No
EPWM_TZFRC	Trip-Zone Force Register <sup>(1)</sup>	30h	No

(1) All trip-zone registers are EALLOW protected and can be modified only after executing the EALLOW instruction.

#### 12.4.2.4.7.3 Operational Highlights for the EPWM Trip-Zone Submodule

The trip-zone signals at pin  $\overline{TZ0}$  to  $\overline{TZ5}$  is an active-low input signal. When the pin goes low, it indicates that a trip event has occurred. Each EPWM module can be individually configured to ignore or use each of the trip-zone pins. Which trip-zone pins are used by a particular EPWM module is determined by the EPWM\_TZSEL register for that specific EPWM module. The trip-zone signal may or may not be synchronized to the system clock (FICLK). A minimum of 1 FICLK low pulse on the  $\overline{TZ0}$  to  $\overline{TZ5}$  inputs is sufficient to trigger a fault condition in the EPWM module. The asynchronous trip makes sure that if clocks are missing for any reason, the outputs can still be tripped by a valid event present on the  $\overline{TZ0}$  to  $\overline{TZ5}$  inputs.

The  $\overline{TZ0}$  to  $\overline{TZ5}$  inputs can be individually configured to provide either a cycle-by-cycle or one-shot trip event for a EPWM module. The configuration is determined by the EPWM\_TZSEL[5-0] CBCK and EPWM\_TZSEL[13-8] OSHTk bit field (where k = 0 to 5 corresponds to the trip pin), respectively.

- **Cycle-by-Cycle (CBC):** When a cycle-by-cycle trip event occurs, the action specified in the EPWM\_TZCTL register is carried out immediately on the EPWMxA and/or EPWMxB output. Table 12-454 lists the possible actions. In addition, the cycle-by-cycle trip event flag (EPWM\_TZFLG[1] CBC) is set and a EPWMxTZINT interrupt is generated if it is enabled in the EPWM\_TZEINT register.

The specified condition on the pins is automatically cleared when the EPWM time-base counter reaches zero (EPWM\_TBCNT[15-0] TBCNT = 0000h) if the trip event is no longer present. Therefore, in this mode, the trip event is cleared or reset every PWM cycle. The EPWM\_TZFLG[1] CBC flag bit will remain set until it is manually cleared by writing to the EPWM\_TZCLR[1] CBC bit. If the cycle-by-cycle trip event is still present when the EPWM\_TZFLG[1] CBC bit is cleared, then it will again be immediately set.

- **One-Shot (OSHT):** When a one-shot trip event occurs, the action specified in the EPWM\_TZCTL register is carried out immediately on the EPWMxA and/or EPWMxB output. Table 12-454 lists the possible actions. In addition, the one-shot trip event flag (EPWM\_TZFLG[2] OST) is set and a EPWMxTZINT interrupt is generated if it is enabled in the EPWM\_TZEINT register. The one-shot trip condition must be cleared manually by writing to the EPWM\_TZCLR[2] OST bit.

The action taken when a trip event occurs can be configured individually for each of the EPWM output pins by way of the EPWM\_TZCTL[1-0] TZA and EPWM\_TZCTL[3-2] TZB register bit field. One of four possible actions, shown in Table 12-454, can be taken on a trip event.

**Table 12-454. Possible Actions On an EPWM Trip Event**

EPWM_TZCTL[1-0] TZA and/or EPWM_TZCTL[3-2] TZB	EPWMxA and/or EPWMxB	Comment
0	High-Impedance	Tripped
1h	Force to High State	Tripped
2h	Force to Low State	Tripped
3h	No Change	Do Nothing. No change is made to the output.

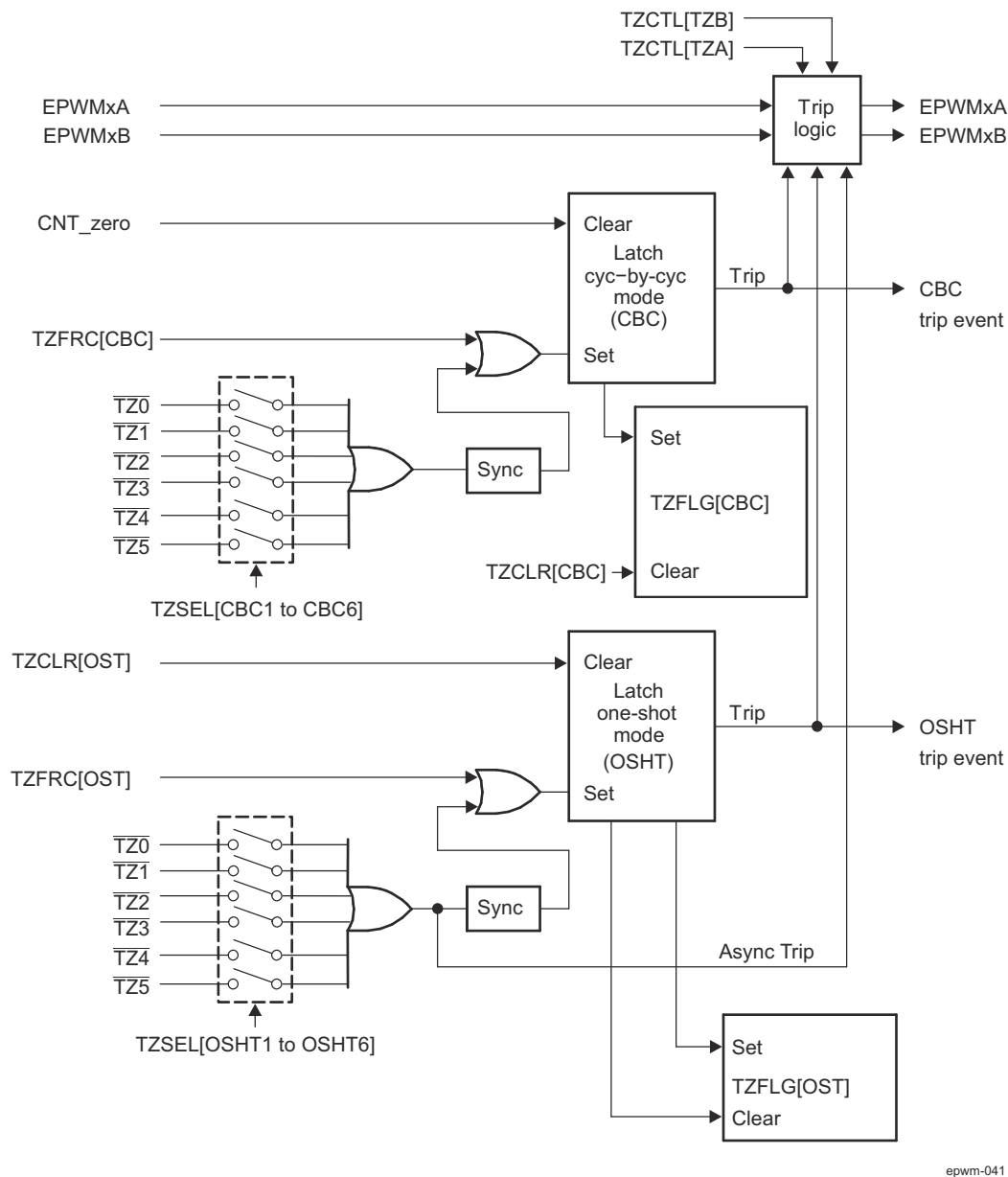
### Example of EPWM Trip-Zone Configurations

A one-shot trip event on  $\overline{TZ0}$  pulls both EPWM1A, EPWM1B low and also forces EPWM2A and EPWM2B high.

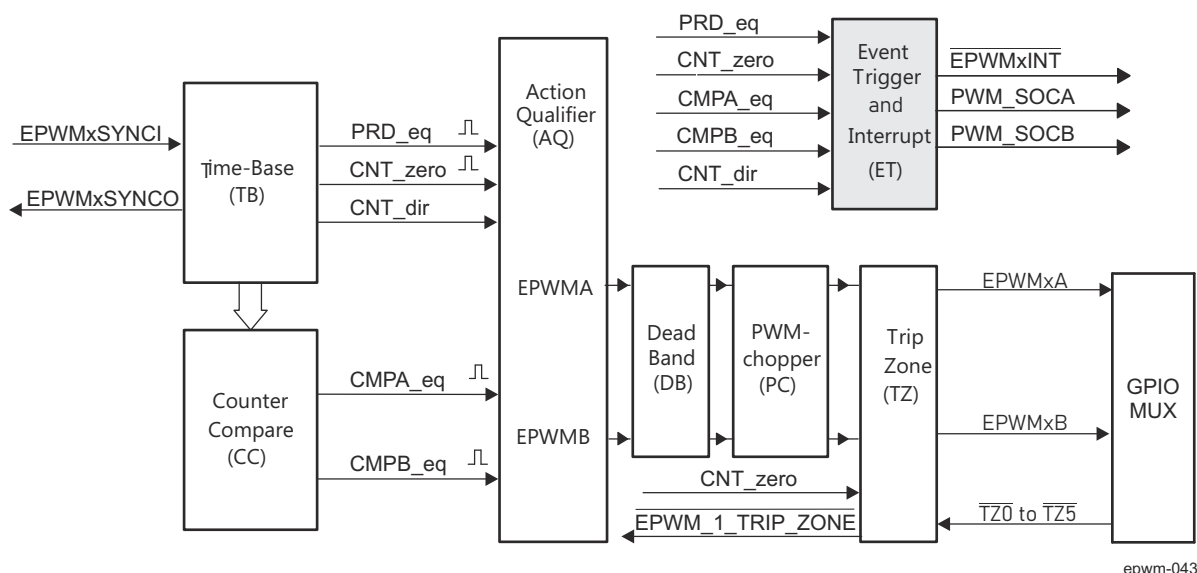
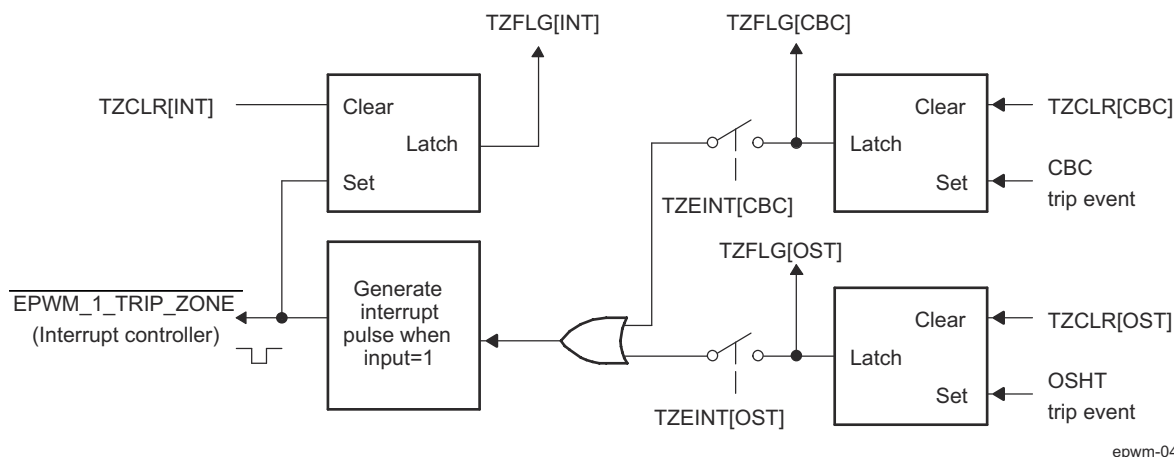
- Configure the EPWM1 registers as follows:
  - EPWM\_TZSEL[8] OSHT1 = 1: enables  $\overline{TZ}$  as a one-shot event source for EPWM1
  - EPWM\_TZCTL[1-0] TZA = 2: EPWM1A will be forced low on a trip event
  - EPWM\_TZCTL[3-2] TZB = 2: EPWM1B will be forced low on a trip event
- Configure the EPWM2 registers as follows:
  - EPWM\_TZSEL[8] OSHT1 = 1: enables  $\overline{TZ}$  as a one-shot event source for EPWM2
  - EPWM\_TZCTL[1-0] TZA = 1: EPWM2A will be forced high on a trip event
  - EPWM\_TZCTL[3-2] TZB = 1: EPWM2B will be forced high on a trip event

#### 12.4.2.4.7.4 Generating EPWM Trip-Event Interrupts

Figure 12-424 and Figure 12-425 illustrate the trip-zone submodule control and interrupt logic, respectively.



**Figure 12-424. EPWM Trip-Zone Submodule Mode Control Logic**



#### 12.4.2.4.8.2 Controlling and Monitoring the EPWM Event-Trigger Submodule

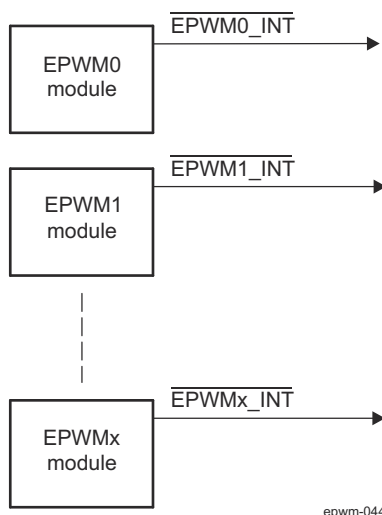
The key registers used to configure the event-trigger submodule are shown in [Table 12-455](#):

**Table 12-455. EPWM Event-Trigger Submodule Registers**

Acronym	Register Description	Address Offset	Shadowed
EPWM_ETSEL	Event-Trigger Selection Register	32h	No
EPWM_ETPS	Event-Trigger Prescale Register	34h	No
EPWM_ETFLG	Event-Trigger Flag Register	36h	No
EPWM_ETCLR	Event-Trigger Clear Register	38h	No
EPWM_ETFRC	Event-Trigger Force Register	3Ah	No

#### 12.4.2.4.8.3 Operational Overview of the EPWM Event-Trigger Submodule

Each EPWM module has one interrupt request line as shown in [Figure 12-427](#). Mapping interrupt lines to device host interrupt controllers is device specific and is covered in *EPWM Integration*.

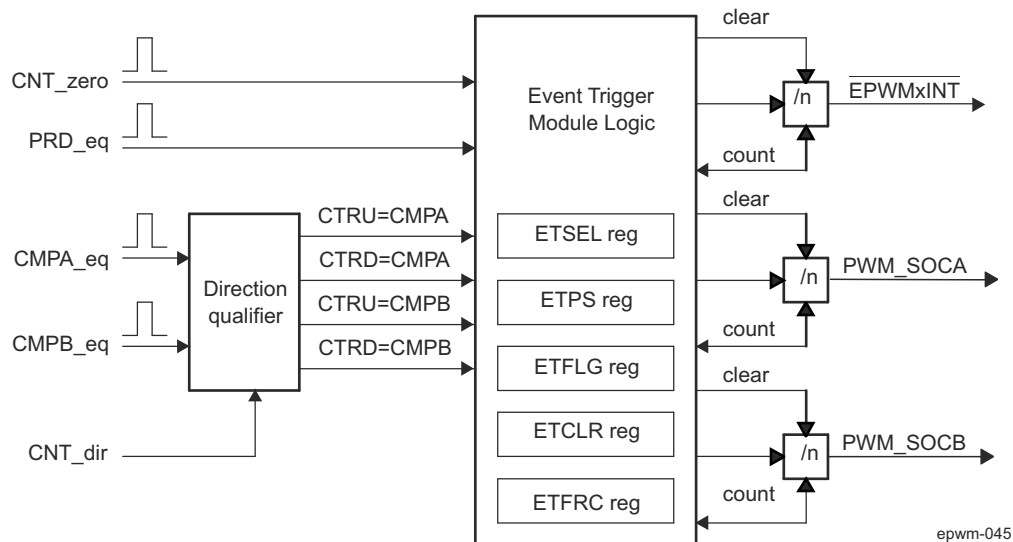


epwm-044

**Figure 12-427. EPWM Event-Trigger Submodule Inter-Connectivity to Interrupt Controller**

The event-trigger submodule monitors various event conditions (the left side inputs to event-trigger submodule shown in [Figure 12-428](#)) and can be configured to prescale these events before issuing an Interrupt request. The event-trigger prescaling logic can issue Interrupt requests at:

- Every event
- Every second event
- Every third event



epwm-045

**Figure 12-428. EPWM Event-Trigger Submodule Showing Event Inputs and Prescaled Outputs**

- **ETSEL** — This selects which of the possible events will trigger an interrupt.
- **ETPS** — This programs the event prescaling options previously mentioned.
- **ETFLG** — These are flag bits indicating status of the selected and prescaled events.
- **ETCLR** — These bits allow to clear the flag bits in the EPWM\_ETFLG register via software.
- **ETFRC** — These bits allow software forcing of an event. Useful for debugging or software intervention.

A more detailed look at how the various register bits interact with the Interrupt is shown in [Figure 12-429](#).

[Figure 12-429](#) shows the event-trigger's interrupt generation logic. The interrupt-period (EPWM\_ETPS[1-0] INTPRD) bits specify the number of events required to cause an interrupt pulse to be generated. The choices available are:

- Do not generate an interrupt
- Generate an interrupt on every event
- Generate an interrupt on every second event
- Generate an interrupt on every third event

An interrupt cannot be generated on every fourth or more events.

Which event can cause an interrupt is configured by the interrupt selection (EPWM\_ETSEL[2-0] INTSEL) bits. The event can be one of the following:

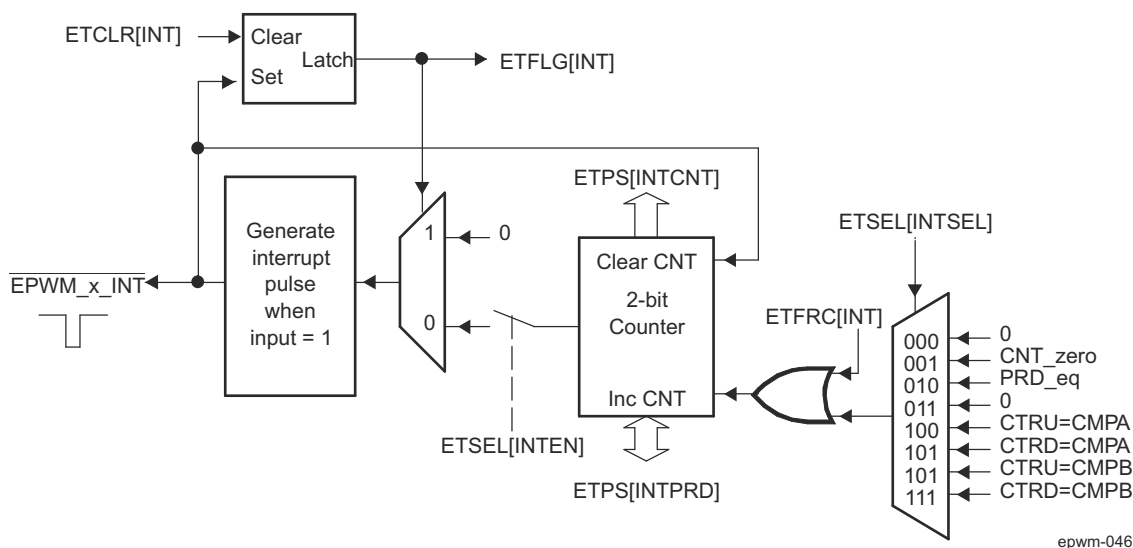
- Time-base counter equal to zero (EPWM\_TBCNT[15-0] TBCNT = 0000h).
- Time-base counter equal to period (EPWM\_TBCNT[15-0] TBCNT = EPWM\_TBPRD[15-0] TBPRD).
- Time-base counter equal to the compare A register (EPWM\_CMPA) when the timer is incrementing.
- Time-base counter equal to the compare A register (EPWM\_CMPA) when the timer is decrementing.
- Time-base counter equal to the compare B register (EPWM\_CMPB) when the timer is incrementing.
- Time-base counter equal to the compare B register (EPWM\_CMPB) when the timer is decrementing.

The number of events that have occurred can be read from the interrupt event counter (EPWM\_ETPS[3-2] INTCNT) register bits. That is, when the specified event occurs the EPWM\_ETPS[3-2] INTCNT bits are incremented until they reach the value specified by the EPWM\_ETPS[1-0] INTPRD bit field. When EPWM\_ETPS[3-2] INTCNT = EPWM\_ETPS[1-0] INTPRD the counter stops counting and its output is set. The counter is only cleared when an interrupt is sent to the interrupt controller.

When EPWM\_ETPS[3-2] INTCNT reaches EPWM\_ETPS[1-0] INTPRD, one of the following behaviors will occur:

- If interrupts are enabled, EPWM\_ETSEL[3] INTEN = 1h and the interrupt flag is clear, EPWM\_ETFLG[0] INT = 0h, then an interrupt pulse is generated and the interrupt flag is set, EPWM\_ETFLG[0] INT = 1h, and the event counter is cleared EPWM\_ETPS[3-2] INTCNT = 0h. The counter will begin counting events again.
- If interrupts are disabled, EPWM\_ETSEL[3] INTEN = 0, or the interrupt flag is set, EPWM\_ETFLG[0] INT = 1h, the counter stops counting events when it reaches the period value EPWM\_ETPS[3-2] INTCNT = EPWM\_ETPS[1-0] INTPRD.
- If interrupts are enabled, but the interrupt flag is already set, then the counter will hold its output high until the ENTFLG[0] INT flag is cleared. This allows for one interrupt to be pending while one is serviced.

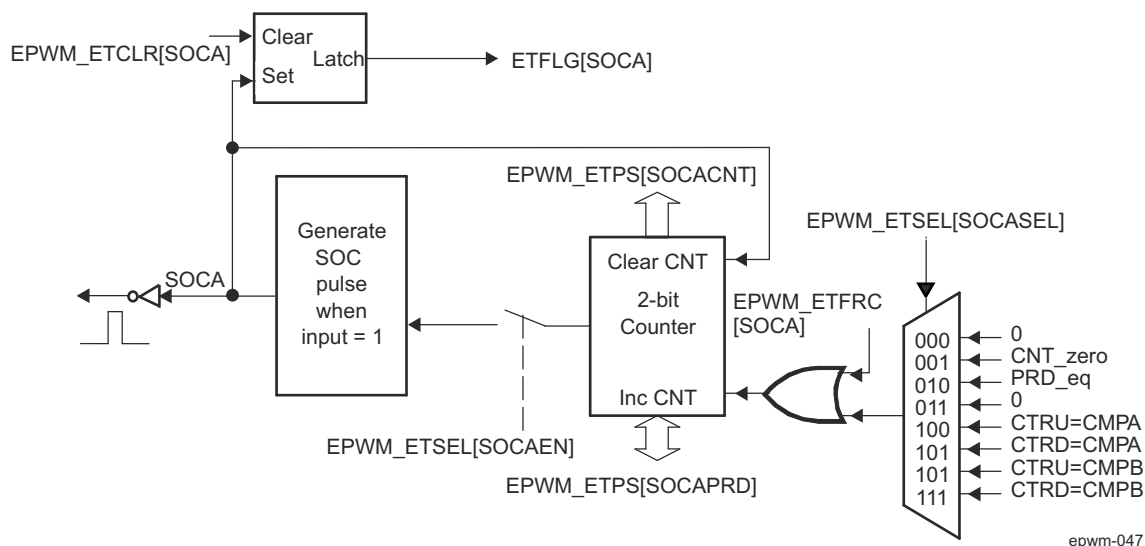
Writing to the INTPRD bits will automatically clear the counter INTCNT = 0 and the counter output will be reset (so no interrupts are generated). Writing a 1h to the EPWM ETFRC[0] INT bit will increment the event counter INTCNT. The counter will behave as described above when INTCNT = INTPRD. When INTPRD = 0h, the counter is disabled and hence no events will be detected and the EPWM ETFRC[0] INT bit is also ignored.



**Figure 12-429. EPWM Event-Trigger Interrupt Generator**

#### 12.4.2.4.8.4

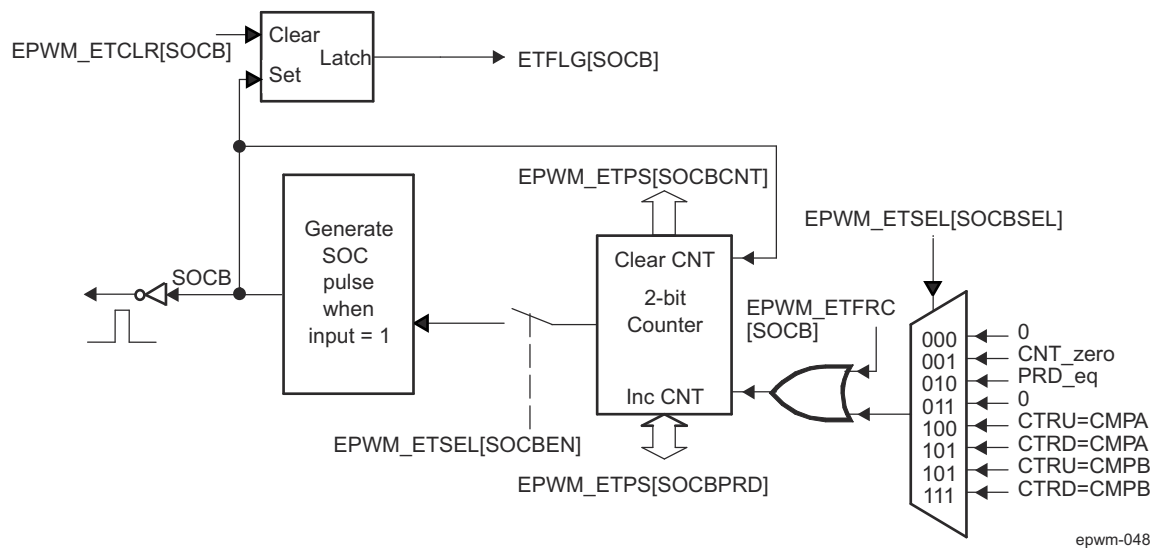
Figure 12-430 shows the operation of the event-trigger's start-of-conversion-A (SOCA) pulse generator. The EPWM\_ETPS[11-10] SOCA\_CNT counter and EPWM\_ETPS[9-8] SOCA\_PRD period values behave similarly to the interrupt generator except that the pulses are continuously generated. That is, the pulse flag EPWM\_ETFLG[2] SOCA is latched when a pulse is generated, but it does not stop further pulse generation. The enable/disable EPWM\_ETSEL[11] SOCAEN bit stops pulse generation, but input events can still be counted until the period value is reached as with the interrupt generation logic. The event that will trigger an SOCA and SOCB pulse can be configured separately in the EPWM\_ETSEL[10-8] SOCASEL and EPWM\_ETSEL[14-12] SOCBSEL bit field. The possible events are the same events that can be specified for the interrupt generation logic.



**Figure 12-430. EPWM Event-Trigger SOCA Pulse Generator**

Figure 12-431 shows the operation of the event-trigger's start-of-conversion-B (SOCB) pulse generator. The event-trigger's SOCB pulse generator operates the same way as the SOCA.





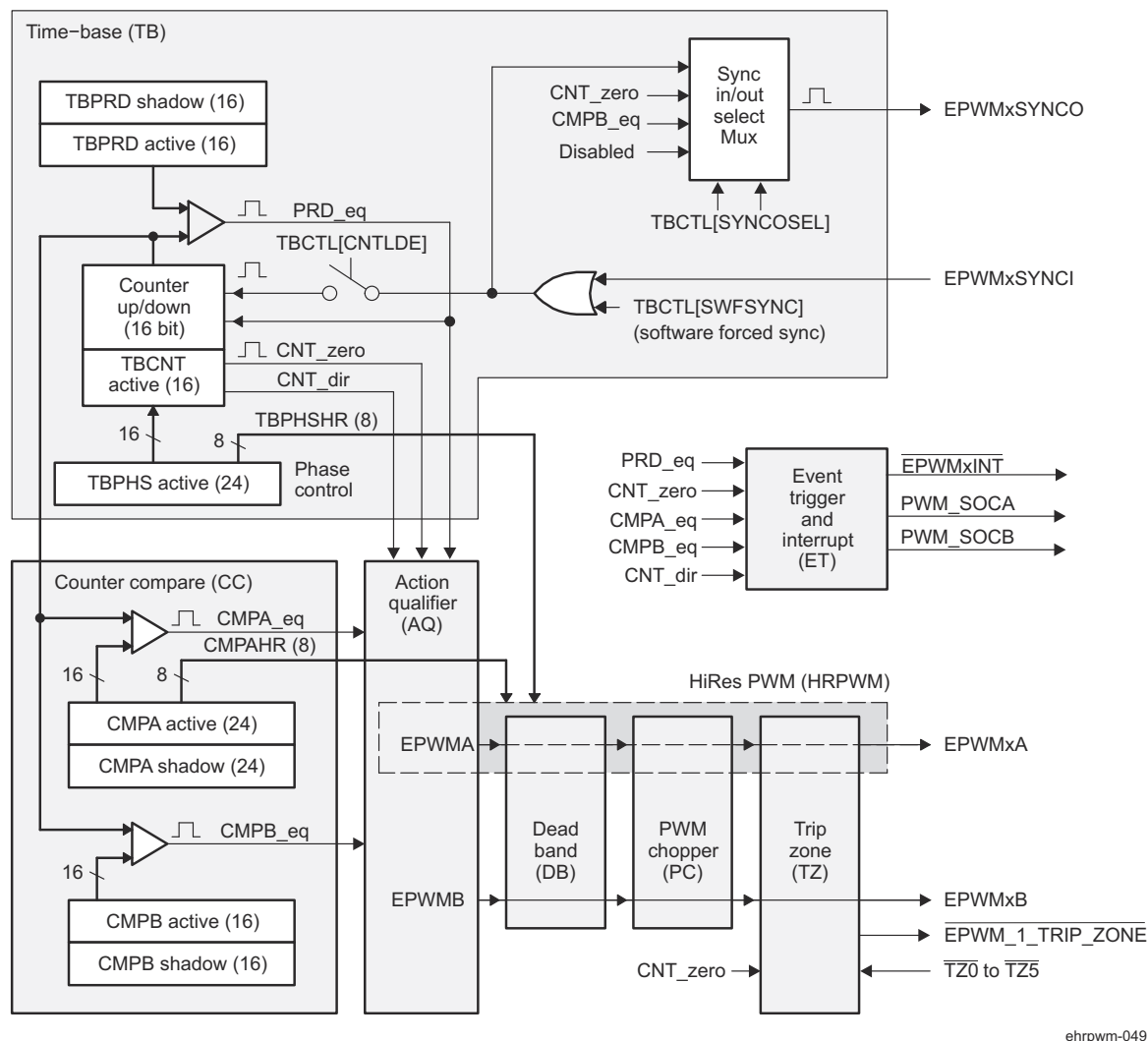
**Figure 12-431. EPWM Event-Trigger SOCB Pulse Generator**

#### 12.4.2.4.9 EPWM High Resolution (HRPWM) Submodule

This section describes the High-Resolution PWM (HRPWM) submodule in the PWM module.

##### 12.4.2.4.9.1 Overview

Figure 12-432 shows the high-resolution PWM (HRPWM) submodule in the EPWM system. Some devices include the high-resolution PWM submodule, see *EPWM Integration* to determine which EPWM instances include this feature.



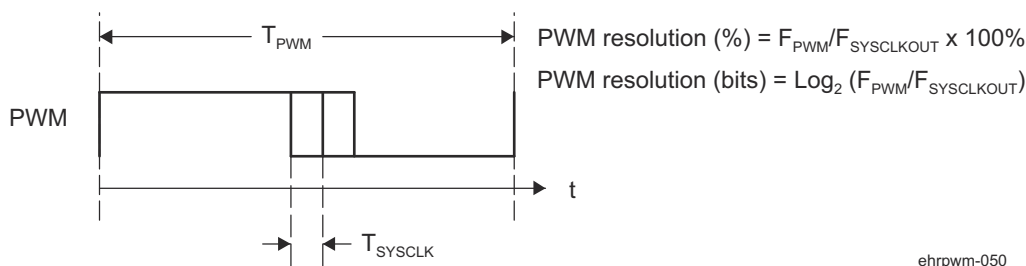
**Figure 12-432. HRPWM System Interface**

The high-resolution pulse-width modulator (HRPWM) extends the time resolution capabilities of the conventionally derived digital pulse-width modulator (PWM). HRPWM is typically used when PWM resolution falls below ~9-10 bits.

The key features of HRPWM are:

- Extended time resolution capability
- Used in both duty cycle and phase-shift control methods
- Finer time granularity control or edge positioning using extensions to the Compare A and Phase registers
- Implemented using the A signal path of PWM, that is, on the EPWMxA output. **EPWMxB output has conventional PWM capabilities**

The EPWM peripheral is used to perform a function that is mathematically equivalent to a digital-to-analog converter (DAC). As shown in [Figure 12-433](#), the effective resolution for conventionally generated PWM is a function of PWM frequency (or period) and system clock frequency.



**Figure 12-433. Resolution Calculations for Conventionally Generated PWM**

Use HRPWM when the required PWM operating frequency does not offer sufficient resolution in PWM mode. As an example of improved performance offered by HRPWM, [Table 12-456](#) shows resolution in bits for various PWM frequencies. [Table 12-456](#) values assume a MEP step size of 180 ps. See the device-specific Datasheet for typical and maximum performance specifications for the MEP.

**Table 12-456. Resolution for PWM and HRPWM**

PWM Frequency (kHz)	Regular Resolution (PWM)		High Resolution (HRPWM)	
	Bits	%	Bits	%
20	12.3	0.0	18.1	0.000
50	11.0	0.0	16.8	0.001
100	10.0	0.1	15.8	0.002
150	9.4	0.2	15.2	0.003
200	9.0	0.2	14.8	0.004
250	8.6	0.3	14.4	0.005
500	7.6	0.5	13.8	0.007
1000	6.6	1.0	12.4	0.018
1500	6.1	1.5	11.9	0.027
2000	5.6	2.0	11.4	0.036

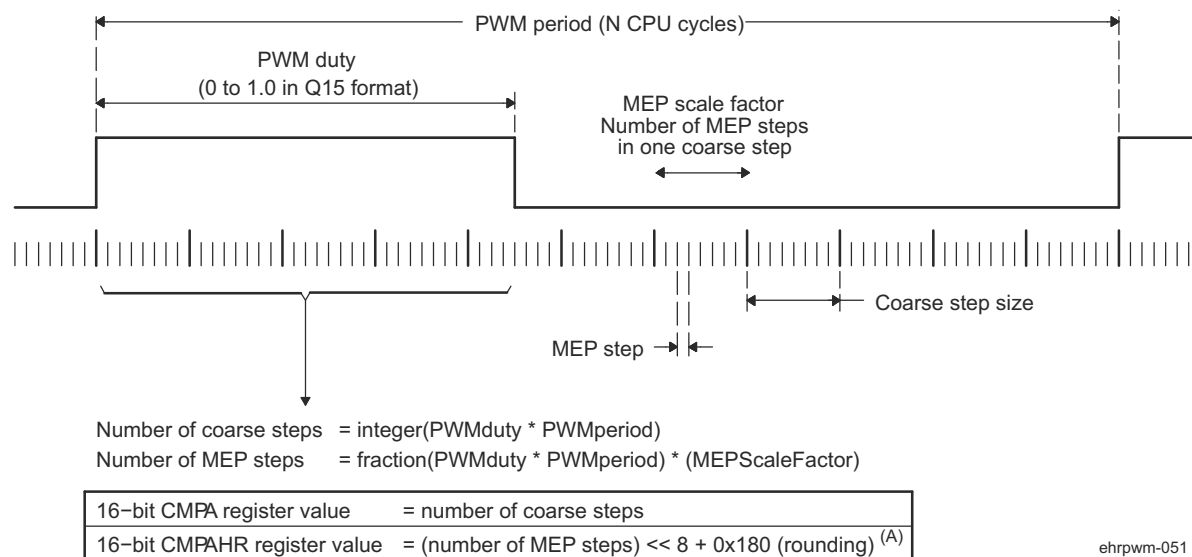
Although each application may differ, typical low-frequency PWM operation (below 250 kHz) may not require HRPWM. HRPWM capability is most useful for high-frequency PWM requirements of power conversion topologies such as:

- Single-phase buck, boost, and flyback
- Multi-phase buck, boost, and flyback
- Phase-shifted full bridge
- Direct modulation of D-Class power amplifiers

#### 12.4.2.4.9.2 Architecture of the High-Resolution PWM Submodule

The HRPWM is based on micro edge positioner (MEP) technology. MEP logic is capable of positioning an edge very finely by sub-dividing one coarse system clock of a conventional PWM generator. The time step accuracy is on the order of 150 ps. The HRPWM also has a self-check software diagnostics mode to check if the MEP logic is running optimally, under all operating conditions.

[Figure 12-434](#) shows the relationship between one coarse system clock and edge position in terms of MEP steps, which are controlled via an 8-bit field in the Compare A extension register (HRPWM\_CMPAHR).



A. For MEP range and rounding adjustment.

**Figure 12-434. Operating Logic Using MEP**

To generate an HRPWM waveform of a given frequency and polarity, the user needs to configure the TBM (Time-Base Module), CCM (Counter-Compare Module), and AQM (Action-Qualifier Module) registers. The HRPWM works together with the TBM, CCM, and AQM registers to extend edge resolution, and should be configured accordingly. Although many programming combinations are possible, only a few are needed and practical.

#### 12.4.2.4.9.3 Controlling and Monitoring the High-Resolution PWM Submodule

The MEP of the HRPWM is controlled by two extension registers, each 8-bits wide. These two HRPWM registers are concatenated with the 16-bit EPWM\_TBPHS and EPWM\_CMPA registers used to control PWM operation.

- HRPWM\_TBPHSHR — Time-Base Phase High-Resolution Register
- HRPWM\_CMPAHR — Counter-Compare A High-Resolution Register.

Table 12-457 lists the registers used to control and monitor the high-resolution PWM submodule.

**Table 12-457. HRPWM Submodule Registers**

Acronym	Register Description	Address Offset	Shadowed
HRPWM_TBPHSHR	Extension Register for HRPWM Phase	4h	No
HRPWM_CMPAHR	Extension Register for HRPWM Duty	10h	Yes
HRPWM_HRCTL	HRPWM Configuration Register	40h	No

#### 12.4.2.4.9.4 Configuring the High-Resolution PWM Submodule

Once the EPWM has been configured to provide conventional PWM of a given frequency and polarity, the HRPWM is configured by programming the HRPWM\_HRCTL register located at offset address 40h. This register provides configuration options for the following key operating modes:

- **Edge Mode:** The MEP can be programmed to provide precise position control on the rising edge (RE), falling edge (FE), or both edges (BE) at the same time. FE and RE are used for power topologies requiring duty cycle control, while BE is used for topologies requiring phase shifting, for example, phase shifted full bridge.
- **Control Mode:** The MEP is programmed to be controlled either from the HRPWM\_CMPAHR register (duty cycle control) or the HRPWM\_TBPHSHR register (phase control). RE or FE control mode should be used with the HRPWM\_CMPAHR register. BE control mode should be used with the HRPWM\_TBPHSHR register.
- **Shadow Mode:** This mode provides the same shadowing (double buffering) option as in regular PWM mode. This option is valid only when operating from the HRPWM\_CMPAHR register and should be chosen to be the

same as the regular load option for the CMPA register. If the HRPWM\_TBPHSHR register is used, then this option has no effect.

#### 12.4.2.4.9.5 Operational Highlights for the High-Resolution PWM Submodule

The MEP logic is capable of placing an edge in one of 255 (8 bits) discrete time steps, each of which has a time resolution on the order of 150 ps. The MEP works with the TBM (Time-Base Module) and CCM (Counter-Compare Module) registers to be certain that time steps are optimally applied and that edge placement accuracy is maintained over a wide range of PWM frequencies, system clock frequencies and other operating conditions. [Table 12-458](#) shows the typical range of operating frequencies supported by the HRPWM.

**Table 12-458. Relationship Between MEP Steps, PWM Frequency and Resolution**

System (MHz)	MEP Steps Per FICLK <sup>(1)</sup> (2) (3)	PWM Minimum (Hz) <sup>(4)</sup>	PWM Maximum (MHz)	Resolution at Maximum (Bits) <sup>(5)</sup>
50.0	111	763	2.50	11.1
60.0	93	916	3.00	10.9
70.0	79	1068	3.50	10.6
80.0	69	1221	4.00	10.4
90.0	62	1373	4.50	10.3
100.0	56	1526	5.00	10.1

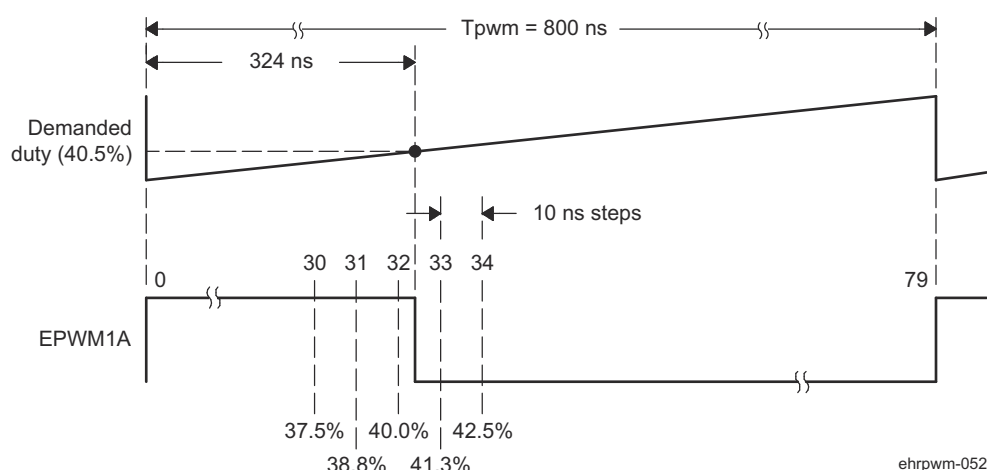
- (1) System frequency = FICLK, that is, CPU clock. TBCLK = FICLK.
- (2) Table data based on a MEP time resolution of 180 ps (this is an example value).
- (3) MEP steps applied =  $T_{FICLK}/180$  ps in this example.
- (4) PWM minimum frequency is based on a maximum period value, TBPRD = 65 535. PWM mode is asymmetrical up-count.
- (5) Resolution in bits is given for the maximum PWM frequency stated.

### 12.4.2.4.9.5.1 HRPWM Edge Positioning

In a typical power control loop (switch modes, digital motor control (DMC), uninterruptible power supply (UPS)), a digital controller (PID, 2pole/2zero, lag/lead, and so forth) issues a duty command, usually expressed in a per unit or percentage terms.

In the following example, assume that for a particular operating point, the demanded duty cycle is 0.405 or 40.5% on-time and the required converter PWM frequency is 1.25 MHz. In conventional PWM generation with a system clock of 100 MHz, the duty cycle choices are in the vicinity of 40.5%. Figure 12-435 shows that duty cycle of 40% (a compare value of 32 counts) is the closest to 40.5%. This is equivalent to an edge position of 320 ns instead of the desired 324 ns. This data is shown in Table 12-459.

Utilizing MEP allows to achieve an edge position much closer to the desired point of 324 ns. Table 12-459 shows that in addition to the CMPA value, 22 steps of the MEP (HRPWM\_CMPAHR register) will position the edge at 323.96 ns, resulting in almost zero error. In this example, it is assumed that the MEP has a step resolution of 180 ns.



**Figure 12-435. Required PWM Waveform for a Requested Duty = 40.5%**

**Table 12-459. CMPA vs Duty (left), and [CMPA:CMPAHR] vs Duty (right)**

CMPA (count) <sup>(1) (2) (3)</sup>	DUTY (%)	High Time (ns)	CMPA (count)	CMPAHR (count)	Duty (%)	High Time (ns)
28	35.0	280	32	18	40.405	323.24
29	36.3	290	32	19	40.428	323.42
30	37.5	300	32	20	40.450	323.60
31	38.8	310	32	21	40.473	323.78
32	40.0	320	32	22	40.495	323.96
33	41.3	330	32	23	40.518	324.14
34	42.5	340	32	24	40.540	324.32
			32	25	40.563	324.50
Required			32	26	40.585	324.68
32.40	40.5	324	32	27	40.608	324.86

(1) System clock, FICLK and TBCLK = 100 MHz, 10 ns

(2) For a PWM Period register value of 80 counts, PWM Period = 80 × 10 ns = 800 ns, PWM frequency = 1/800 ns = 1.25 MHz

(3) Assumed MEP step size for the above example = 180 ps

#### 12.4.2.4.9.5.2 HRPWM Scaling Considerations

The mechanics of how to position an edge precisely in time has been demonstrated using the resources of the standard (EPWM\_CMPA) and MEP (HRPWM\_CMPAHR) registers. In a practical application, however, it is necessary to seamlessly provide the CPU a mapping function from a per-unit (fractional) duty cycle to a final integer (non-fractional) representation that is written to the [CMPA:CMPAHR] register combination.

To do this, first examine the scaling or mapping steps involved. It is common in control software to express duty cycle in a per-unit or percentage basis. This has the advantage of performing all needed math calculations without concern for the final absolute duty cycle, expressed in clock counts or high time in ns. Furthermore, it makes the code more transportable across multiple converter types running different PWM frequencies.

To implement the mapping scheme, a two-step scaling procedure is required.

Assumptions for this example:

System clock, FICLK	=	10 ns (100 MHz)
PWM frequency	=	1.25 MHz (1/800 ns)
Required PWM duty cycle, <b>PWMDuty</b>	=	0.405 (40.5%)
PWM period in terms of coarse steps, <b>PWMperiod</b> (800 ns/10 ns)	=	80
Number of MEP steps per coarse step at 180 ps (10 ns/180 ps), <b>MEP_SF</b>	=	55
Value to keep CMPAHR within the range of 1-255 and fractional rounding constant (default value)	=	180h

#### Step 1: Percentage Integer Duty value conversion for EPWM\_CMPA register

EPWM_CMPA register value	=	$\text{int}(\text{PWMDuty} \times \text{PWMperiod})$ ; int means integer part
	=	$\text{int}(0.405 \times 80)$
	=	$\text{int}(32.4)$
EPWM_CMPA register value	=	32 (20h)

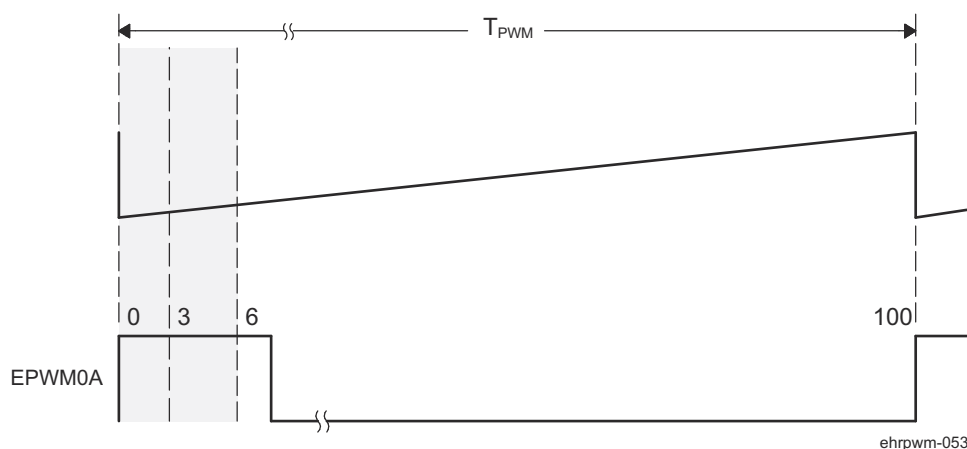
#### Step 2: Fractional value conversion for HRPWM\_CMPAHR register

HRPWM_CMPAHR register value	=	$(\text{frac}(\text{PWMDuty} \times \text{PWMperiod}) \times \text{MEP\_SF}) \ll 8) + 180\text{h}$ ; frac means fractional part
	=	$(\text{frac}(32.4) \times 55 \ll 8) + 180\text{h}$ ; Shift is to move the value as CMPAHR high byte
	=	$((0.4 \times 55) \ll 8) + 180\text{h}$
	=	$(22 \ll 8) + 180\text{h}$
	=	$22 \times 256 + 180\text{h}$ ; Shifting left by 8 is the same multiplying by 256.
	=	$5632 + 180\text{h}$
	=	$1600\text{h} + 180\text{h}$
HRPWM_CMPAHR value	=	1780h; HRPWM_CMPAHR value = 1700h, lower 8 bits will be ignored by hardware.

### 12.4.2.4.9.5.3 HRPWM Duty Cycle Range Limitation

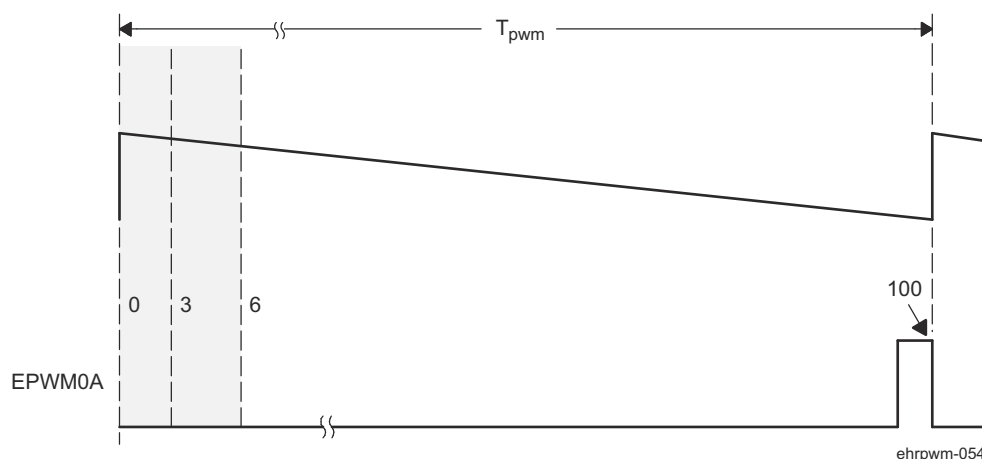
In high resolution mode, the MEP is not active for 100% of the PWM period. It becomes operational 3 FICLK cycles after the period starts.

Duty cycle range limitations are illustrated in Figure 12-436. This limitation imposes a lower duty cycle limit on the MEP. For example, precision edge control is not available all the way down to 0% duty cycle. Although for the first 3 or 6 cycles, the HRPWM capabilities are not available, regular PWM duty control is still fully operational down to 0% duty. In most applications this should not be an issue as the controller regulation point is usually not designed to be close to 0% duty cycle.



**Figure 12-436. Low % Duty Cycle Range Limitation Example When PWM Frequency = 1 MHz**

If the application demands HRPWM operation in the low percent duty cycle region, then the HRPWM can be configured to operate in count-down mode with the rising edge position (REP) controlled by the MEP. This is illustrated in Figure 12-437. In this case low percent duty limitation is no longer an issue.



**Figure 12-437. High % Duty Cycle Range Limitation Example when PWM Frequency = 1 MHz**

### 12.4.2.4.10 EPWM / HRPWM Functional Register Groups

The Table 12-460 lists the groups of EPWM and the high-resolution PWM module registers according to their functionalities.

**Table 12-460. EPWM / HRPWM Module Control and Status Registers Grouped by Submodule**

Register Name	Offset	Size (×16)	Shadow	Register Description
<b>Time-Base (TB) Submodule Registers</b>				
EPWM_TBCTL	0h	1	No	Time-Base Control Register



**Table 12-460. EPWM / HRPWM Module Control and Status Registers Grouped by Submodule (continued)**

Register Name	Offset	Size (×16)	Shadow	Register Description
EPWM_TBSTS	2h	1	No	Time-Base Status Register
EPWM_TBPHS	6h	1	No	Time-Base Phase Register
EPWM_TBCNT	8h	1	No	Time-Base Counter Register
EPWM_TBPRD	Ah	1	Yes	Time-Base Period Register
<b>Counter-Compare (CC) Submodule Registers</b>				
EPWM_CMPCTL	Eh	1	No	Counter-Compare Control Register
EPWM_CMPA	12h	1	Yes	Counter-Compare A Register
EPWM_CMPB	14h	1	Yes	Counter-Compare B Register
<b>Action-Qualifier (AQ) Submodule Registers</b>				
EPWM_AQCTLA	16h	1	No	Action-Qualifier Control Register for Output A (EPWMxA)
EPWM_AQCTLB	18h	1	No	Action-Qualifier Control Register for Output B (EPWMxB)
EPWM_AQSFRC	1Ah	1	No	Action-Qualifier Software Force Register
EPWM_AQCSFRC	1Ch	1	Yes	Action-Qualifier Continuous S/W Force Register Set
<b>Dead-Band (DB) Generator Submodule Registers</b>				
EPWM_DBCTL	1Eh	1	No	Dead-Band Generator Control Register
EPWM_DBRED	20h	1	No	Dead-Band Generator Rising Edge Delay Count Register
EPWM_DBFED	22h	1	No	Dead-Band Generator Falling Edge Delay Count Register
<b>Trip-Zone (TZ) Submodule Registers</b>				
EPWM_TZSEL	24h	1	No	Trip-Zone Select Register
EPWM_TZCTL	28h	1	No	Trip-Zone Control Register <sup>(1)</sup>
EPWM_TZEINT	2Ah	1	No	Trip-Zone Enable Interrupt Register <sup>(1)</sup>
EPWM_TZFLG	2Ch	1	No	Trip-Zone Flag Register <sup>(1)</sup>
EPWM_TZCLR	2Eh	1	No	Trip-Zone Clear Register <sup>(1)</sup>
EPWM_TZFRC	30h	1	No	Trip-Zone Force Register <sup>(1)</sup>
<b>Event-Trigger (ET) Submodule Registers</b>				
EPWM_ETSEL	32h	1	No	Event-Trigger Selection Register
EPWM_ETPS	34h	1	No	Event-Trigger Pre-Scale Register
EPWM_ETFLG	36h	1	No	Event-Trigger Flag Register
EPWM_ETCLR	38h	1	No	Event-Trigger Clear Register
EPWM_ETFRC	3Ah	1	No	Event-Trigger Force Register
<b>PWM-Chopper Submodule Registers</b>				
EPWM_PCCTL	3Ch	1	No	PWM-Chopper Control Register
<b>High-Resolution PWM (HRPWM) Submodule Registers</b>				
HRPWM_TBPHSHR	4h	1	No	Extension for HRPWM Phase Register
HRPWM_CMPAHR	10h	1	No	Extension for HRPWM Counter-Compare A Register
HRPWM_HRCTL	40h	1	No	HRPWM Control Register

(1) EALLOW protected registers.

#### 12.4.2.4.11 Proper EPWM Interrupt Initialization Procedure

When the EPWM peripheral clock is enabled it is possible that interrupt flags may be set due to spurious events as the EPWM registers not being properly initialized. The proper procedure for initializing the EPWM peripheral is:

1. Disable global interrupts (CPU INTM flag)

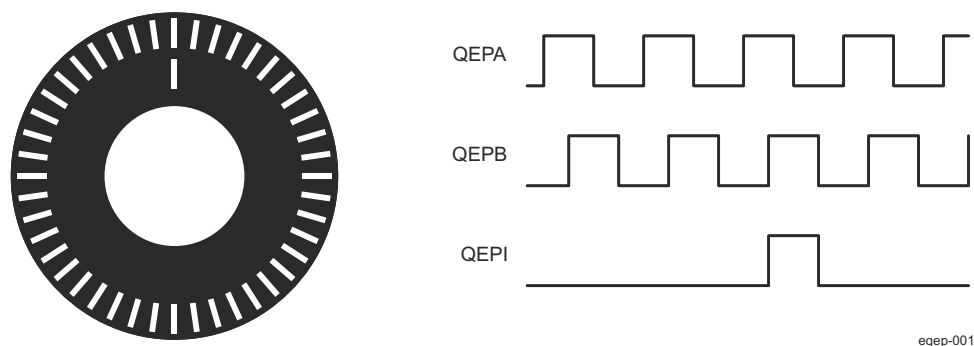
2. Disable EPWM interrupts
3. Initialize peripheral registers
4. Clear any spurious EPWM flags
5. Enable EPWM interrupts
6. Enable global interrupts

### 12.4.3 Enhanced Quadrature Encoder Pulse (EQEP) Module

This section describes the Enhanced Quadrature Encoder Pulse (EQEP) module in the device.

#### 12.4.3.1 EQEP Overview

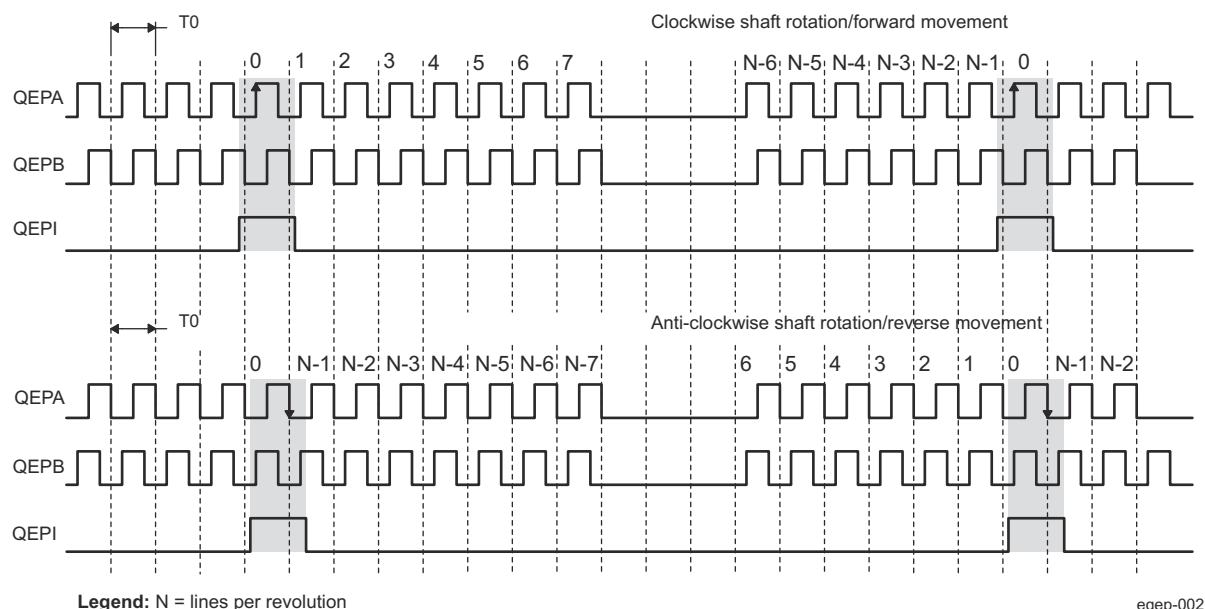
The Enhanced Quadrature Encoder Pulse (EQEP) peripheral is used for direct interface with a linear or rotary incremental encoder to get position, direction and speed information from a rotating machine for use in high performance motion and position control system. The disk of an incremental encoder is patterned with a single track of slots patterns, as shown in [Figure 12-438](#). These slots create an alternating pattern of dark and light lines. The disk count is defined as the number of dark/light line pairs that occur per revolution (lines per revolution). As a rule, a second track is added to generate a signal that occurs once per revolution (index signal: QEPI), which can be used to indicate an absolute position. Encoder manufacturers identify the index pulse using different terms such as index, marker, home position and zero reference.



**Figure 12-438. Optical Encoder Disk**

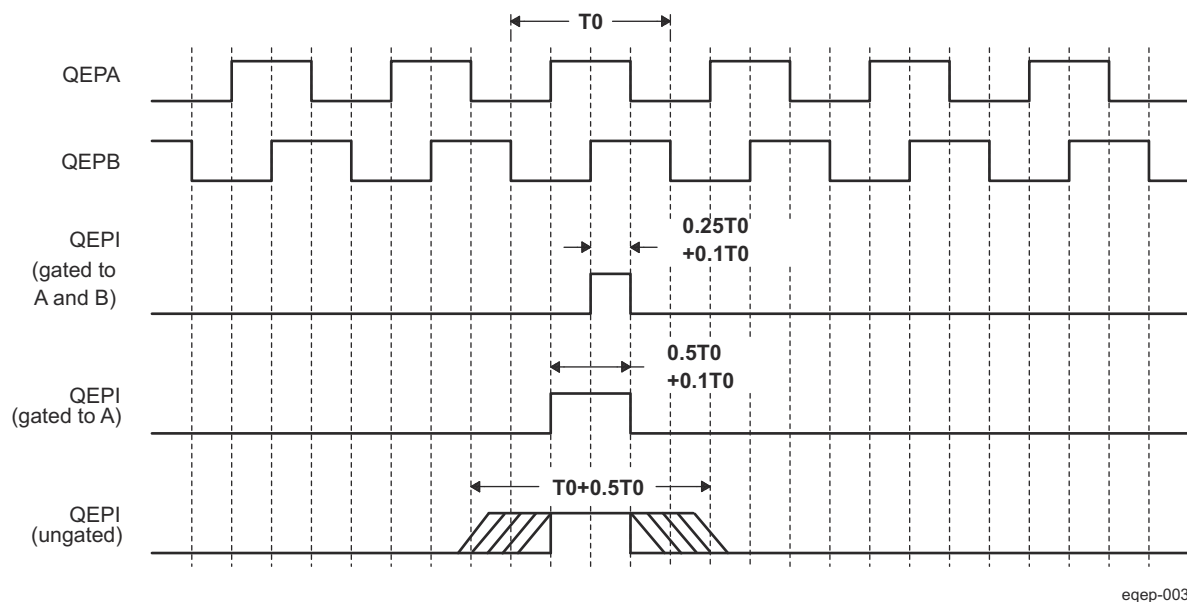
To derive direction information, the lines on the disk are read out by two different photo-elements that "look" at the disk pattern with a mechanical shift of 1/4 the pitch of a line pair between them. This shift is realized with a reticle or mask that restricts the view of the photo-element to the desired part of the disk lines. As the disk rotates, the two photo-elements generate signals that are shifted 90 degrees out of phase from each other. These are commonly called the quadrature QEPA and QEPB signals. The clockwise direction for most encoders is defined as the QEPA channel going positive before the QEPB channel and vice versa as shown in [Figure 12-439](#).

The encoder wheel typically makes one revolution for every revolution of the motor or the wheel may be at a geared rotation ratio with respect to the motor. Therefore, the frequency of the digital signal coming from the QEPA and QEPB outputs varies proportionally with the velocity of the motor. For example, a 2000-line encoder directly coupled to a motor running at 5000 revolutions per minute (rpm) results in a frequency of 166.6 kHz, so by measuring the frequency of either the QEPA or QEPB output, the processor can determine the velocity of the motor.



**Figure 12-439. QEP Encoder Output Signal for Forward/Reverse Movement**

Quadrature encoders from different manufacturers come with two forms of index pulse (gated index pulse or ungated index pulse) as shown in Figure 12-440. A nonstandard form of index pulse is ungated. In the ungated configuration, the index edges are not necessarily coincident with A and B signals. The gated index pulse is aligned to any of the four quadrature edges and width of the index pulse and can be equal to a quarter, half, or full period of the quadrature signal.



**Figure 12-440. Index Pulse Example**

Some typical applications for rotary encoders range from fax machines to motor controllers, and even robot localization. Rotary sensors are fundamental to the robotics and motion control systems found today. The optical shaft encoder can be used to improve a robot in various ways. The encoder can measure rotational distance traveled and speed, which can be used to monitor, for example, the angular position of a robot gripper arm or the speed of a robot.

**General Issues:** Estimating velocity from a digital position sensor is a cost-effective strategy in motor control. Two different first order approximations for velocity may be written as:

$$V(k) \approx \frac{x(k) - x(k-1)}{T} = \frac{\Delta X}{T} \quad \text{eqep-004} \quad (19)$$

$$V(k) \approx \frac{X}{t(k) - t(k-1)} = \frac{X}{\Delta T} \quad \text{eqep-005} \quad (20)$$

where

$v(k)$ : Velocity at time instant  $k$

$x(k)$ : Position at time instant  $k$

$x(k-1)$ : Position at time instant  $k - 1$

$T$ : Fixed unit time or inverse of velocity calculation rate

$\Delta X$ : Incremental position movement in unit time

$t(k)$ : Time instant " $k$ "

$t(k-1)$ : Time instant " $k - 1$ "

$X$ : Fixed unit position

$\Delta T$ : Incremental time elapsed for unit position movement.

[Equation 19](#) is the conventional approach to velocity estimation and it requires a time base to provide unit time event for velocity calculation. Unit time is the inverse of the velocity calculation rate.

The encoder count (position) is read once during each unit time event. The quantity  $[x(k) - x(k-1)]$  is formed by subtracting the previous reading from the current reading. Then the velocity estimate is computed by multiplying by the known constant  $1/T$  (where  $T$  is the constant time between unit time events and is known in advance).

Estimation based on [Equation 19](#) has an inherent accuracy limit directly related to the resolution of the position sensor and the unit time period  $T$ . For example, consider a 500-line per revolution quadrature encoder with a velocity calculation rate of 400 Hz. When used for position the quadrature encoder gives a four-fold increase in resolution, in this case, 2000 counts per revolution. The minimum rotation that can be detected is therefore 0.0005 revolutions, which gives a velocity resolution of 12 rpm when sampled at 400 Hz. While this resolution may be satisfactory at moderate or high speeds, for example, 1% error at 1200 rpm, it would clearly prove inadequate at low speeds. In fact, at speeds below 12 rpm, the speed estimate would erroneously be zero much of the time.

At low speed, [Equation 20](#) provides a more accurate approach. It requires a position sensor that outputs a fixed interval pulse train, such as the aforementioned quadrature encoder. The width of each pulse is defined by motor speed for a given sensor resolution. [Equation 20](#) can be used to calculate motor speed by measuring the elapsed time between successive quadrature pulse edges. However, this method suffers from the opposite limitation of [Equation 19](#). A combination of relatively large motor speeds and high sensor resolution makes the time interval  $\Delta T$  small, and thus more greatly influenced by the timer resolution. This can introduce considerable error into high-speed estimates.

For systems with a large speed range (that is, speed estimation is needed at both low and high speeds), one approach is to use [Equation 20](#) at low speed and have the software switch over to [Equation 19](#) when the motor speed rises above some specified threshold.

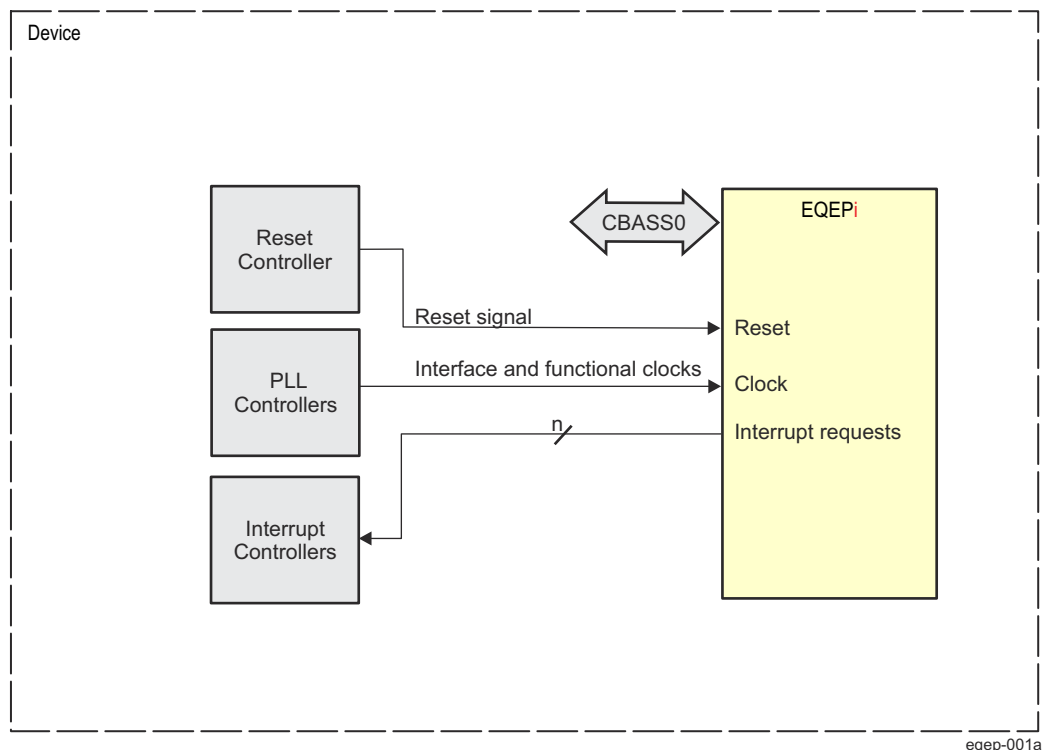
The device has three instances of the EQEP modules.

Table 12-461 shows the EQEP allocation across device domains.

**Table 12-461. EQEP Allocation Across Device Domains**

Instance	Domain		
	WKUP	MCU	MAIN
EQEP0	-	-	✓
EQEP1	-	-	✓
EQEP2	-	-	✓

Figure 12-441 shows the EQEP module overview.



A.  $i = 0 \text{ to } 2$

**Figure 12-441. EQEP Overview**

#### 12.4.3.1.1 EQEP Features

The EQEP module includes the following features:

- Input synchronization
- Three stage/six stage digital noise filter
- Quadrature decoder unit
- Position counter and control unit for position measurement
- Quadrature edge capture unit for low-speed measurement
- Unit time base for speed and frequency measurement
- Watchdog timer for detecting stalls
- EQEP inputs (A/B/INDEX and STROBE) are available at chip level
- EQEP phase error output is also available. The status of the phase error can be observed by software through the CTRLMMR\_EQEP\_STAT register in the CTRL\_MMR0 module.

#### 12.4.3.1.2 EQEP Not Supported Features

The following EQEP features are not supported:

- EQEP quadrature outputs (A and B) are not pinned out

For more information about EQEP module restrictions in functionality, see *Device Specific EQEP Features*.

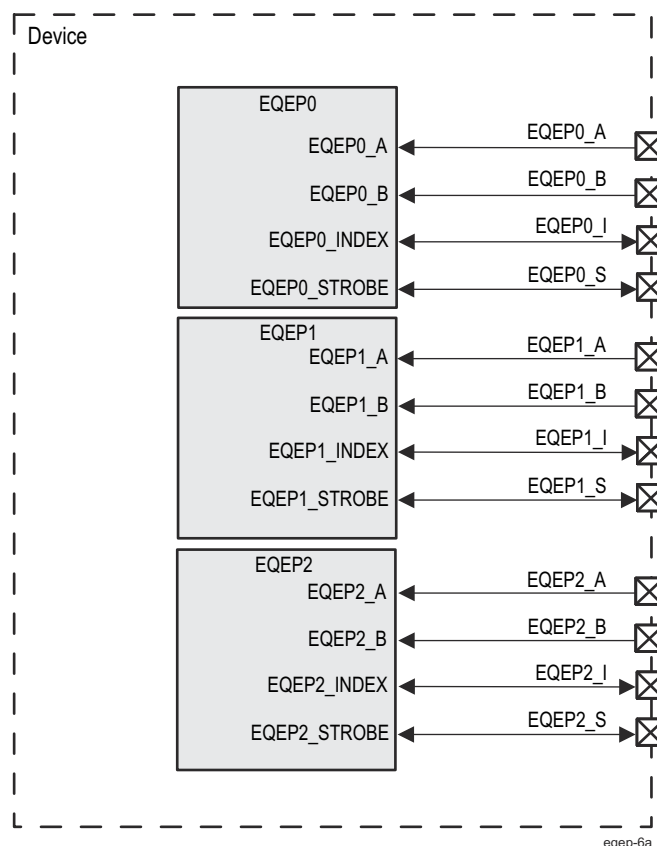
### 12.4.3.2 EQEP Environment

#### 12.4.3.2.1 EQEP I/O Interface

The EQEP<sub>i</sub> (where  $i = 0$  to 2) module is hereinafter referred to as EQEP module.

This section describes the EQEP external connections (environment).

Figure 12-442 shows the EQEP interface signals.



**Figure 12-442. EQEP External Interface I/Os**

Table 12-462 describes the EQEP I/O signals.

**Table 12-462. EQEP I/O Signals**

Module Pin	Device Level Signal Name	I/O Type <sup>(1)</sup>	Description	Module Pin Reset Value <sup>(2)</sup>
<b>EQEP0</b>				
EQEP0_A	EQEP0_A	I	EQEP0 Quadrature input A	HiZ
EQEP0_B	EQEP0_B	I	EQEP0 Quadrature input B	HiZ
EQEP0_INDEX	EQEP0_I	I/O <sup>(3)</sup>	EQEP0 Index input/output	HiZ
EQEP0_STROBE	EQEP0_S	I/O <sup>(3)</sup>	EQEP0 Strobe input/output	HiZ
<b>EQEP1</b>				
EQEP1_A	EQEP1_A	I	EQEP1 Quadrature input A	HiZ
EQEP1_B	EQEP1_B	I	EQEP1 Quadrature input B	HiZ
EQEP1_INDEX	EQEP1_I	I/O <sup>(3)</sup>	EQEP1 Index input/output	HiZ
EQEP1_STROBE	EQEP1_S	I/O <sup>(3)</sup>	EQEP1 Strobe input/output	HiZ
<b>EQEP2</b>				
EQEP2_A	EQEP2_A	I	EQEP2 Quadrature input A	HiZ



**Table 12-462. EQEP I/O Signals (continued)**

Module Pin	Device Level Signal Name	I/O Type <sup>(1)</sup>	Description	Module Pin Reset Value <sup>(2)</sup>
EQEP2_B	EQEP2_B	I	EQEP2 Quadrature input B	HiZ
EQEP2_INDEX	EQEP2_I	I/O <sup>(3)</sup>	EQEP2 Index input/output	HiZ
EQEP2_STROBE	EQEP2_S	I/O <sup>(3)</sup>	EQEP2 Strobe input/output	HiZ

(1) I = Input; O = Output

(2) HiZ = High Impedance

(3) Output functionality is not supported. Only inputs are pinned out. For more information see *Device Specific EQEP Features*.

### Note

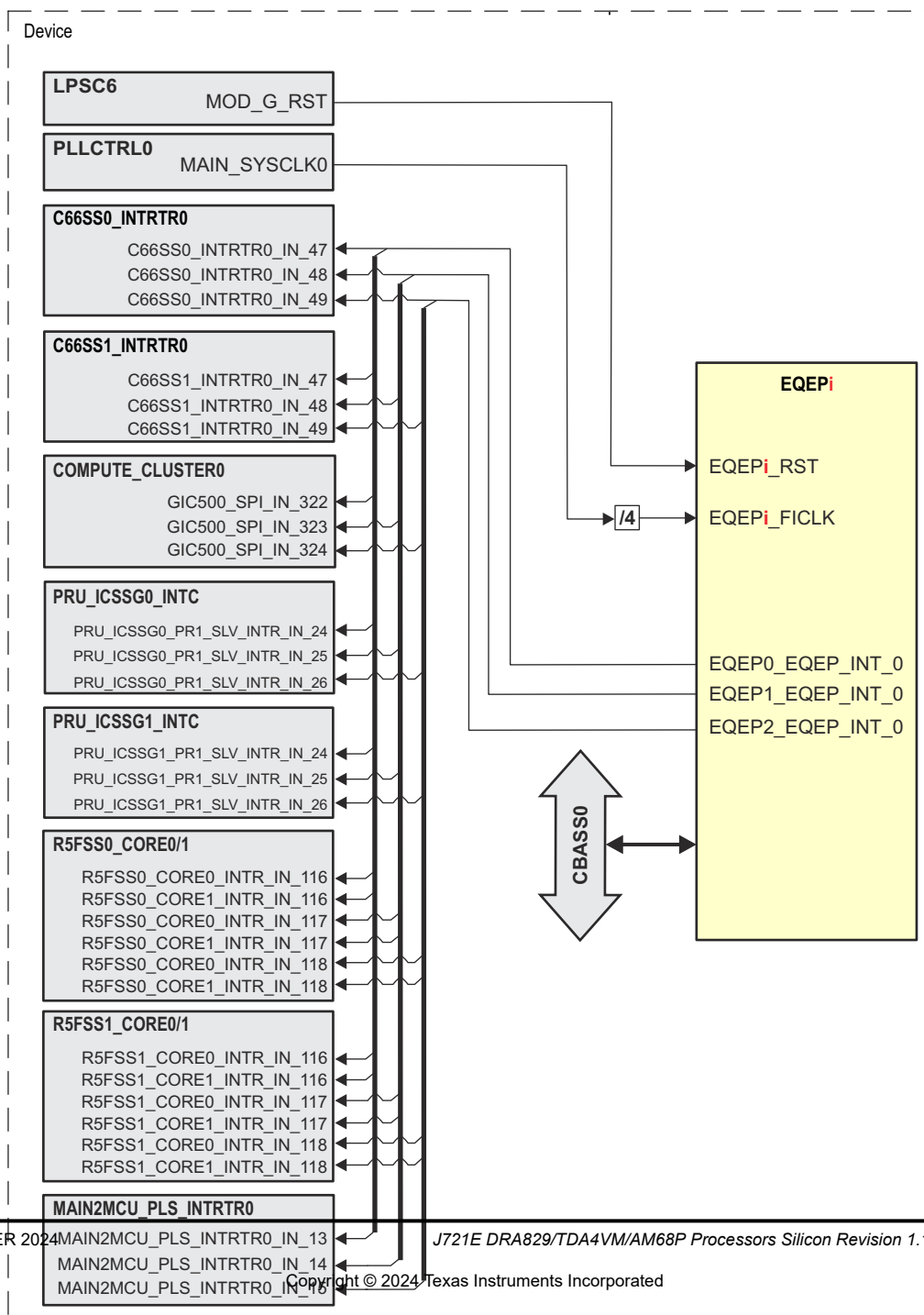
For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables Pin Attributes and Pin Multiplexing in the device-specific Datasheet.

### 12.4.3.3 EQEP Integration

This section describes module integration in the device, including information about clocks, resets, and hardware requests.

There are three EQEP modules integrated in the device MAIN domain - EQEP0, EQEP1 and EQEP2.

[Figure 12-443](#) shows the integration of EQEPi.



**Table 12-463. EQEP Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
EQEP0	PSC0	PD0	LPSC6	CBASS0
EQEP1	PSC0	PD0	LPSC6	CBASS0
EQEP2	PSC0	PD0	LPSC6	CBASS0

**Table 12-464. EQEP Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
EQEP0	EQEP0_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	EQEP0 functional and interface clock
EQEP1	EQEP1_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	EQEP1 functional and interface clock
EQEP2	EQEP2_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	EQEP2 functional and interface clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
EQEP0	EQEP0_RST	MOD_G_RST	LPSC6	Module Reset
EQEP1	EQEP1_RST	MOD_G_RST	LPSC6	Module Reset
EQEP2	EQEP2_RST	MOD_G_RST	LPSC6	Module Reset

**Table 12-465. EQEP Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
EQEP0	EQEP0_EQEP_INT_0	C66SS0_INTRTR0_IN_47	C66SS0_INTRTR0	EQEP0 interrupt	Pulse
		C66SS1_INTRTR0_IN_47	C66SS1_INTRTR0		
		GIC500_SPI_IN_322	COMPUTE_CLUSTER0		
		PRU_ICSSG0_PR1_SLV_INTR_IN_24	PRU_ICSSG0_INTC		
		PRU_ICSSG1_PR1_SLV_INTR_IN_24	PRU_ICSSG1_INTC		
		R5FSS0_CORE0_INTR_IN_116	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_116	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_116	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_116	R5FSS1_CORE1		
		MAIN2MCU_PLS_INTRTR0_IN_13	MAIN2MCU_PLS_INTRTR0		
EQEP1	EQEP1_EQEP_INT_0	C66SS0_INTRTR0_IN_48	C66SS0_INTRTR0	EQEP1 interrupt	Pulse
		C66SS1_INTRTR0_IN_48	C66SS1_INTRTR0		
		GIC500_SPI_IN_323	COMPUTE_CLUSTER0		

**Table 12-465. EQEP Hardware Requests (continued)**

EQEP2	EQEP2_EQEP_INT_0	PRU_ICSSG0_PR1_SLV_INTR_IN_25	PRU_ICSSG0_INTC	EQEP2 interrupt	Pulse
		PRU_ICSSG1_PR1_SLV_INTR_IN_25	PRU_ICSSG1_INTC		
		R5FSS0_CORE0_INTR_IN_117	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_117	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_117	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_117	R5FSS1_CORE1		
		MAIN2MCU_PLS_INTRTR0_IN_14	MAIN2MCU_PLS_INTRTR0		
		C66SS0_INTRTR0_IN_49	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_49	C66SS1_INTRTR0		
		GIC500_SPI_IN_324	COMPUTE_CLUSTER0		
		PRU_ICSSG0_PR1_SLV_INTR_IN_26	PRU_ICSSG0_INTC		
		PRU_ICSSG1_PR1_SLV_INTR_IN_26	PRU_ICSSG1_INTC		
		R5FSS0_CORE0_INTR_IN_118	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_118	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_118	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_118	R5FSS1_CORE1		
		MAIN2MCU_PLS_INTRTR0_IN_15	MAIN2MCU_PLS_INTRTR0		

### 12.4.3.3.1 Device Specific EQEP Features

The EQEP module interface has some restrictions in functionality. The EQEP restrictions are, as follows:

- **Only EQEP inputs (A / B / INDEX and STROBE) are available to user at chip level**
- **EQEP phase error output (EQEP\_ERR\_PHASE\_ERR) is also available**
  - The status of the phase error can be observed by software using the CTRLMMR\_EQEP\_STAT register in the CTRL\_MMR0 module.

The EQEP functional interface signals which are NOT available to user are summarized in [Table 12-466](#).

**Table 12-466. Device Limitations for the EQEP Functional Interfaces**

Module Interface	Signal	Description	Comment
<b>EQEPi</b> (where i = 0 to 2 for the device)	EQEP_EQEPA_O	EQEP quadrature A output	Not available at chip boundary (can not be used)
	EQEP_EQEPB_O	EQEP quadrature B output	
	EQEP_EQEPA_OEN	EQEP quadrature A output enable	
	EQEP_EQEPB_OEN	EQEP quadrature B output enable	

### 12.4.3.4 EQEP Functional Description

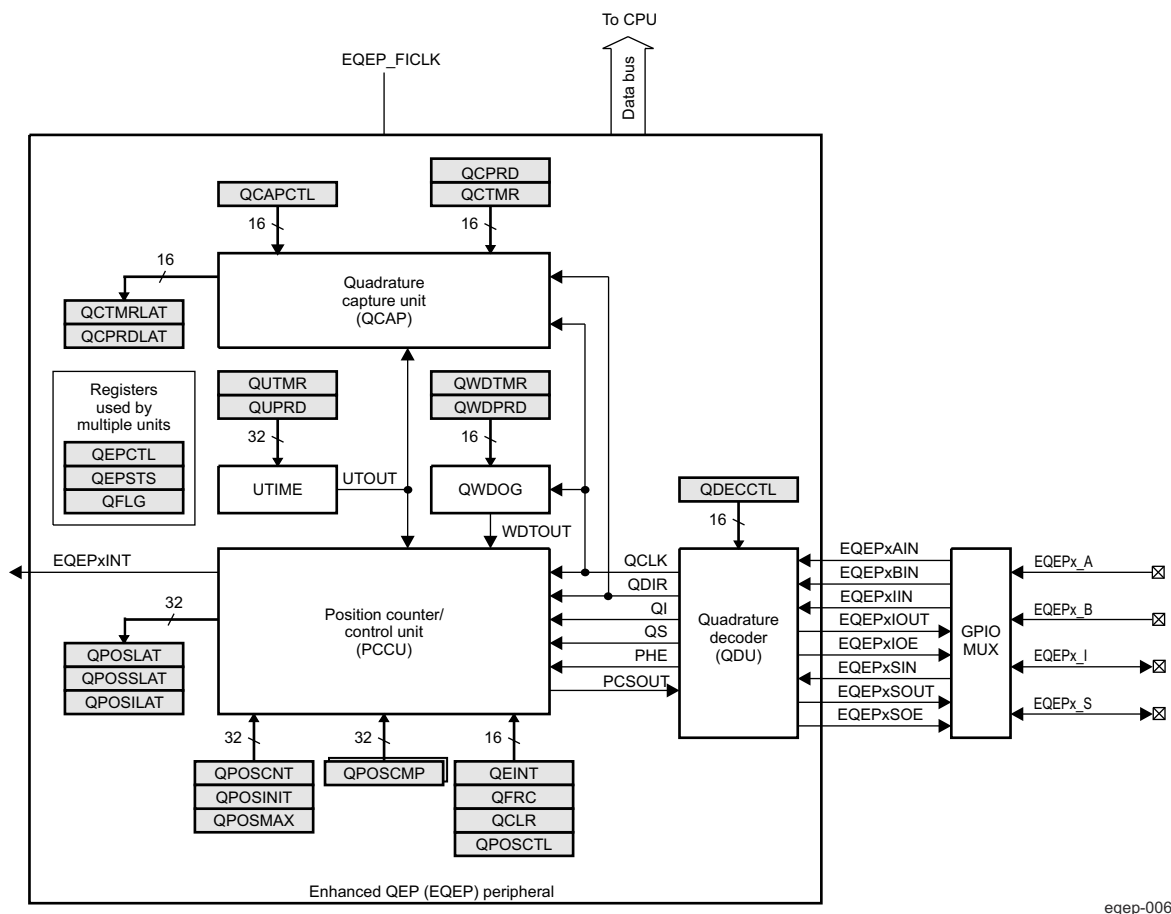
This section provides the EQEP functional description and corresponding functional details about EQEPx inputs.

#### Note

Multiple identical EQEP modules can be contained in a system. For actual number of EQEP modules integrated in the device, refer to *EQEP Integration*. The letter x within a signal or module name is used to indicate a generic EQEP instance on a device. For example, output interrupt request, EQEP0\_EQEP\_INT\_0 belongs to EQEP0, EQEP1\_EQEP\_INT\_0 belongs to EQEP1, and so forth.

The EQEP peripheral contains the following major functional units (as shown in Figure 12-444):

- Programmable input qualification for each pin (part of the GPIO MUX)
- Three stage/six stage digital noise filter
- Quadrature decoder unit (QDU)
- Position counter and control unit for position measurement (PCCU)
- Quadrature edge-capture unit for low-speed measurement (QCAP)
- Unit time base for speed/frequency measurement (UTIME)
- Watchdog timer for detecting stalls (QWDOG).



**Figure 12-444. Functional Block Diagram of the EQEP Peripheral**

#### 12.4.3.4.1 EQEP Inputs

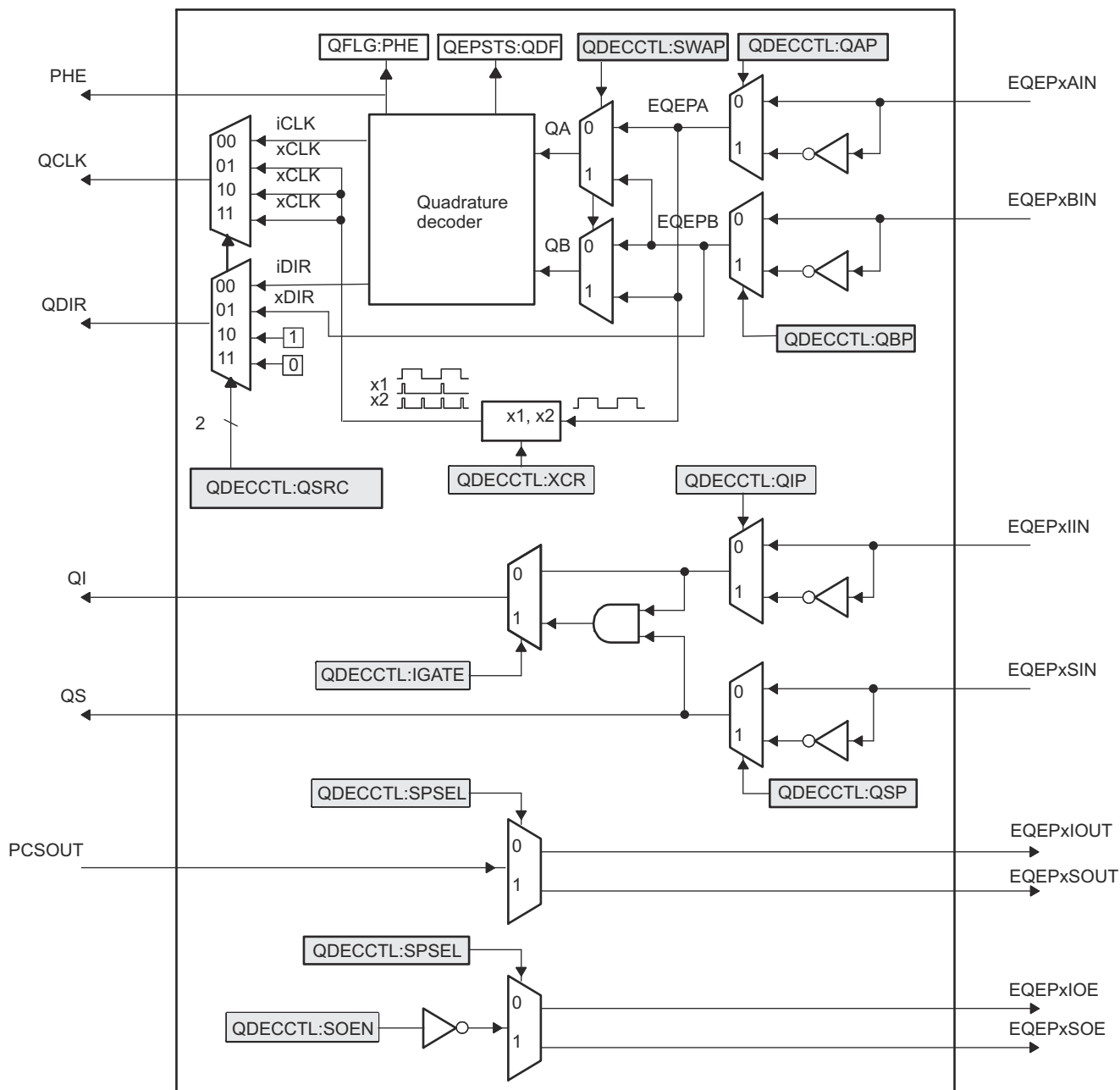
The EQEP inputs include two pins for quadrature-clock mode or direction-count mode, an index (or 0 marker), and a strobe input.

- EQEPx\_A and EQEPx\_B: These two pins can be used in quadrature-clock mode or direction-count mode.
  - Quadrature-clock mode: The EQEP encoders provide two square wave signals (A and B) 90 electrical degrees out of phase whose phase relationship is used to determine the direction of rotation of the input shaft and number of EQEP pulses from the index position to derive the relative position information. For forward or clockwise rotation, QEPA signal leads QEPB signal and vice versa. The quadrature decoder uses these two inputs to generate quadrature-clock and direction signals.
  - Direction-count mode: In direction-count mode, direction and clock signals are provided directly from the external source. Some position encoders have this type of output instead of quadrature output. The EQEPx\_A pin provides the clock input and the EQEPx\_B pin provides the direction input.
- EQEPx\_I: Index or Zero Marker: The EQEP encoder uses an index signal to assign an absolute start position from which position information is incrementally encoded using quadrature pulses. This pin is connected to the index output of the EQEP encoder to optionally reset the position counter for each revolution. This signal can be used to initialize or latch the position counter on the occurrence of a desired event on the index pin.
- EQEPx\_S: Strobe Input: This general-purpose strobe signal can initialize or latch the position counter on the occurrence of a desired event on the strobe pin. This signal is typically connected to a sensor or limit switch to notify that the motor has reached a defined position.



#### 12.4.3.4.2 EQEP Quadrature Decoder Unit (QDU)

Figure 12-445 shows a functional block diagram of the QDU.



eqep-007

**Figure 12-445. Functional Block Diagram of Decoder Unit**

#### 12.4.3.4.2.1 EQEP Position Counter Input Modes

Clock and direction input to position counter is selected using the QSRC bit in the EQEP decoder and control register (EQEP\_QDEC\_QEP\_CTL), based on interface input requirement as follows:

- Quadrature-count mode
- Direction-count mode
- UP-count mode
- DOWN-count mode.

##### 12.4.3.4.2.1.1 Quadrature Count Mode

The quadrature decoder generates the direction and clock to the position counter in quadrature count mode.

#### Direction Decoding

The direction decoding logic of the EQEP circuit determines which one of the sequences (EQEPA, EQEPB) is the leading sequence and accordingly updates the direction information in the QDF bit in the EQEP status register (EQEP\_QEP\_STS\_CT). [Table 12-467](#) and [Figure 12-446](#) show the direction decoding logic in truth table and state machine form. Both edges of the EQEPA and EQEPB signals are sensed to generate count pulses for the position counter. Therefore, the frequency of the clock generated by the EQEP logic is four times that of each input sequence. [Figure 12-447](#) shows the direction decoding and clock generation from the EQEP input signals.

#### Phase Error Flag

In normal operating conditions, quadrature inputs EQEPA and EQEPB will be 90 degrees out of phase. The phase error flag (QPEI\_FLG) is set in the EQEP\_QINT\_EN\_FLG register when edge transition is detected simultaneously on the EQEPA and EQEPB signals to optionally generate interrupts. State transitions marked by dashed lines in [Figure 12-446](#) are invalid transitions that generate a phase error.

#### Count Multiplication

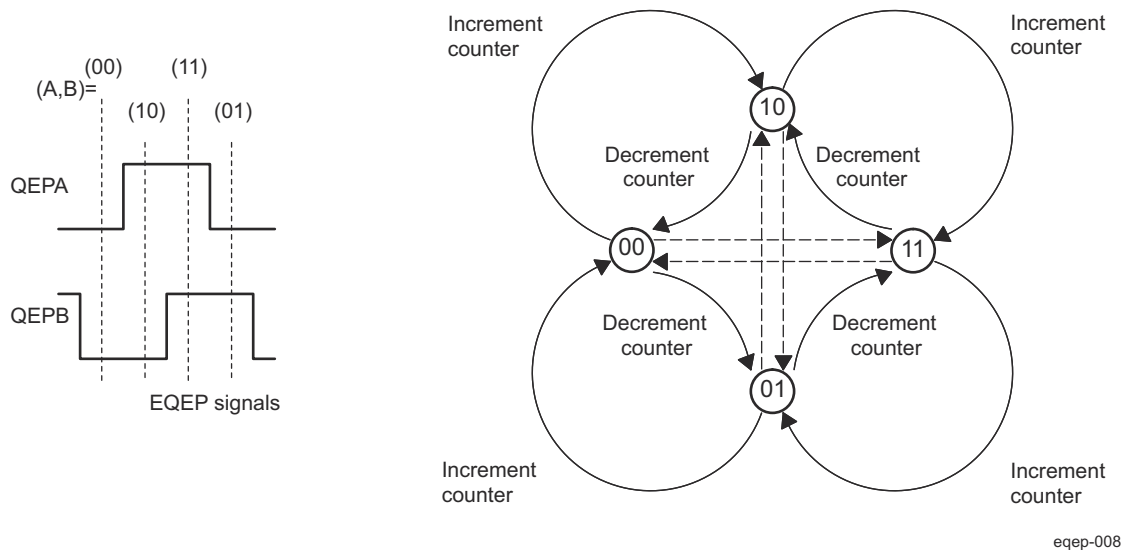
The EQEP position counter provides 4× times the resolution of an input clock by generating a quadrature-clock (QCLK) on the rising/falling edges of both EQEP input clocks (EQEPA and EQEPB) as shown in [Figure 12-447](#).

#### Reverse Count

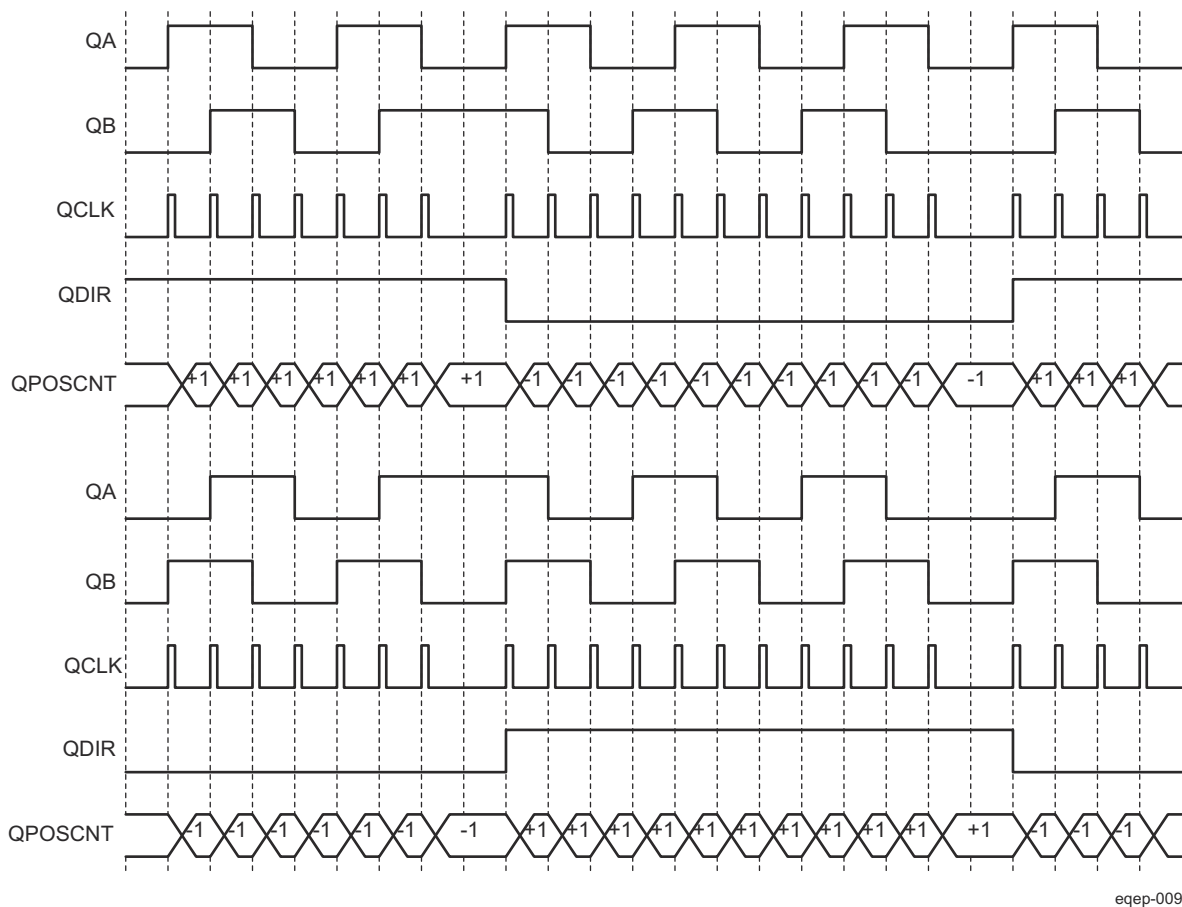
In normal quadrature count operation, EQEPA input is fed to the QA input of the quadrature decoder and the EQEPB input is fed to the QB input of the quadrature decoder. Reverse counting is enabled by setting the SWAP bit in the EQEP decoder and control register (EQEP\_QDEC\_QEP\_CTL). This will swap the input to the quadrature decoder thereby reversing the counting direction.

**Table 12-467. Quadrature Decoder Truth Table**

Previous Edge	Present Edge	QDIR	QPOSCNT
QA↑	QB↑	UP	Increment
	QB↓	DOWN	Decrement
	QA↓	TOGGLE	Increment or Decrement
QA↓	QB↓	UP	Increment
	QB↑	DOWN	Decrement
	QA↑	TOGGLE	Increment or Decrement
QB↑	QA↑	DOWN	Increment
	QA↓	UP	Decrement
	QB↓	TOGGLE	Increment or Decrement
QB↓	QA↓	DOWN	Increment
	QA↑	UP	Decrement
	QB↑	TOGGLE	Increment or Decrement



**Figure 12-446. Quadrature Decoder State Machine**



**Figure 12-447. Quadrature-clock and Direction Decoding**

#### 12.4.3.4.2.1.2 EQEP Direction-count Mode

Some position encoders provide direction and clock outputs, instead of quadrature outputs. In such cases, direction-count mode can be used. EQEPA input will provide the clock for position counter and the EQEPB input will have the direction information. The position counter is incremented on every rising edge of a EQEPA input when the direction input is high and decremented when the direction input is low.

#### 12.4.3.4.2.1.3 EQEP Up-Count Mode

The counter direction signal is hard-wired for up count and the position counter is used to measure the frequency of the EQEPA input. Setting of the XCR bit in the EQEP decoder and control register (EQEP\_QDEC\_QEP\_CTL) enables clock generation to the position counter on both edges of the QEPA input, thereby increasing the measurement resolution by 2× factor.

#### 12.4.3.4.2.1.4 EQEP Down-Count Mode

The counter direction signal is hardwired for a down count and the position counter is used to measure the frequency of the EQEPA input. Setting of the XCR bit in the EQEP decoder and control register (EQEP\_QDEC\_QEP\_CTL) enables clock generation to the position counter on both edges of a EQEPA input, thereby increasing the measurement resolution by 2× factor.

#### 12.4.3.4.2.2 EQEP Input Polarity Selection

Each EQEP input can be inverted using the in the EQEP decoder and control register (EQEP\_QDEC\_QEP\_CTL[8-5]) control bits. As an example, setting of the QIP bit in EQEP\_QDEC\_QEP\_CTL inverts the index input.

#### 12.4.3.4.2.3 EQEP Position-Compare Sync Output

The EQEP peripheral includes a position-compare unit that is used to generate the position-compare sync signal on compare match between the position counter register (EQEP\_QPOSCNT) and the position-compare register (EQEP\_QPOSCMP). This sync signal can be output using an index pin or strobe pin of the EQEP peripheral.

Setting the SOEN bit in the EQEP decoder and control register (EQEP\_QDEC\_QEP\_CTL) enables the position-compare sync output and the SPSEL bit in EQEP\_QDEC\_QEP\_CTL selects either an EQEP index pin or an EQEP strobe pin.

#### 12.4.3.4.3 EQEP Position Counter and Control Unit (PCCU)

The position counter and control unit provides two configuration registers (EQEP\_QDEC\_QEP\_CTL and EQEP\_QCAP\_QPOS\_CTL) for setting up position counter operational modes, position counter initialization/latch modes and position-compare logic for sync signal generation.

##### 12.4.3.4.3.1 EQEP Position Counter Operating Modes

Position counter data may be captured in different manners. In some systems, the position counter is accumulated continuously for multiple revolutions and the position counter value provides the position information with respect to the known reference. An example of this is the quadrature encoder mounted on the motor controlling the print head in the printer. Here the position counter is reset by moving the print head to the home position and then position counter provides absolute position information with respect to home position.

In other systems, the position counter is reset on every revolution using index pulse and position counter provides rotor angle with respect to index pulse position.

Position counter can be configured to operate in following four modes

- Position Counter Reset on Index Event
- Position Counter Reset on Maximum Position
- Position Counter Reset on the first Index Event
- Position Counter Reset on Unit Time Out Event (Frequency Measurement).

In all the above operating modes, position counter is reset to 0 on overflow and to QPOS MAX bifield value in EQEP\_QPOS MAX register on underflow. Overflow occurs when the position counter counts up after QPOS MAX value. Underflow occurs when position counter counts down after "0". Interrupt flag is set to indicate overflow/underflow in EQEP\_QINT\_EN\_FLG register.

#### 12.4.3.4.3.1.1 EQEP Position Counter Reset on Index Event (EQEP\_QDEC\_QEP\_CTL[29-28] PCRM] = 0b00)

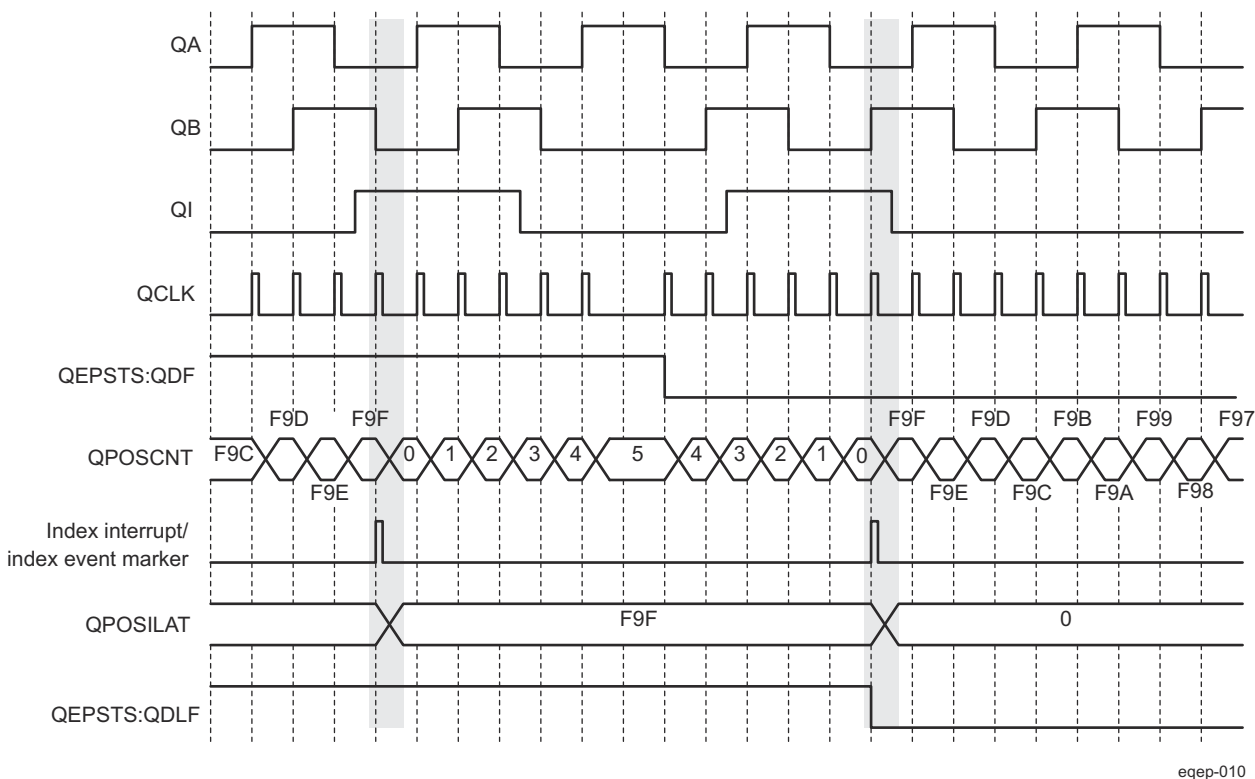
If the index event occurs during the forward movement, then position counter is reset to 0 on the next EQEP clock. If the index event occurs during the reverse movement, then the position counter is reset to the value in the EQEP\_QPOS MAX register on the next EQEP clock.

First index marker is defined as the quadrature edge following the first index edge. The EQEP peripheral records the occurrence of the first index marker (EQEP\_QEP\_STS\_CT[1] FIMF) and direction on the first index event marker (EQEP\_QEP\_STS\_CT[6] FIDF) in EQEP\_QEP\_STS\_CT registers, it also remembers the quadrature edge on the first index marker so that same relative quadrature transition is used for index event reset operation.

For example, if the first reset operation occurs on the falling edge of EQEPB during the forward direction, then all the subsequent reset must be aligned with the falling edge of EQEPB for the forward rotation and on the rising edge of EQEPB for the reverse rotation as shown in Figure 12-448.

The position-counter value is latched to the EQEP\_QPOSILAT register and direction information is recorded in the EQEP\_QEP\_STS\_CT[4] QDLF bit on every index event marker. The position-counter error flag (EQEP\_QEP\_STS\_CT[0] PCEF) and error interrupt flag (EQEP\_QINT\_EN\_FLG[17] PCEI\_FLG) are set if the latched value is not equal to 0 or QPOS MAX. The position-counter error flag (EQEP\_QEP\_STS\_CT[0] PCEF) is updated on every index event marker and an interrupt flag (EQEP\_QINT\_EN\_FLG[17] PCEI\_FLG) will be set on error that can be cleared only through software.

The index event latch configuration EQEP\_QDEC\_QEP\_CTL[21-20] IEL bits are ignored in this mode and position counter error flag/interrupt flag are generated only in index event reset mode.



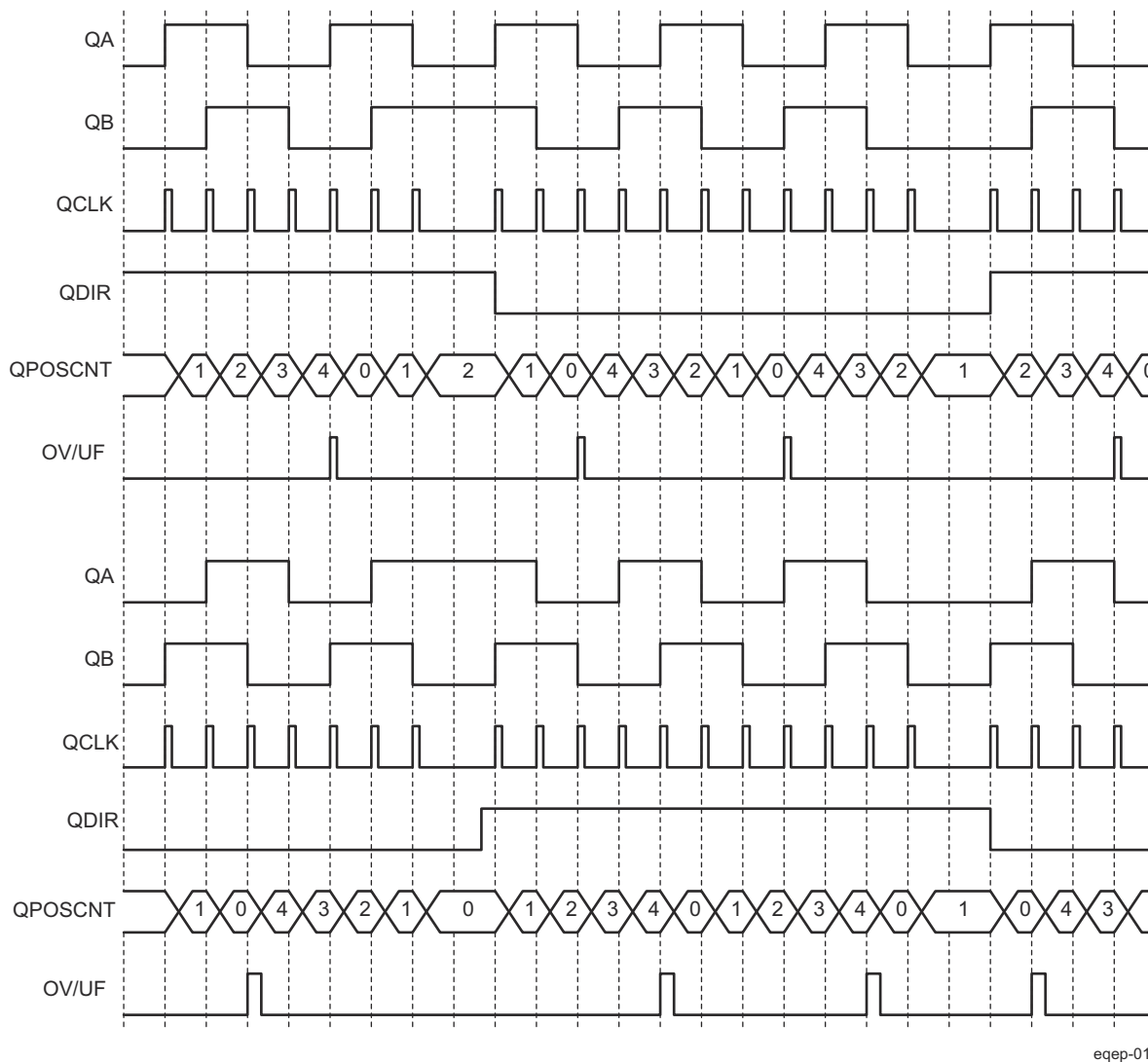
eqep-010

**Figure 12-448. Position Counter Reset by Index Pulse for 1000 Line Encoder (QPOS MAX = 3999 or F9Fh)**

#### 12.4.3.4.3.1.2 EQEP Position Counter Reset on Maximum Position (EQEP\_QDEC\_QEP\_CTL[29-28] PCRM=0b01)

If the position counter is equal to QPOSMAX (in EQEP\_QPOSMAX register), then the position counter is reset to 0 on the next EQEP clock for forward movement and position counter overflow flag is set. If the position counter is equal to 0, then the position counter is reset to QPOSMAX on the next QEP clock for reverse movement and position counter underflow flag is set. Figure 12-449 shows the position counter reset operation in this mode.

First index marker is defined as the quadrature edge following the first index edge. The EQEP peripheral records the occurrence of the first index marker (EQEP\_QEP\_STS\_CT[1] FIMF) and direction on the first index event marker (EQEP\_QEP\_STS\_CT[6] FIDF); it also remembers the quadrature edge on the first index marker so that the same relative quadrature transition is used for the software index marker (EQEP\_QDEC\_QEP\_CTL[21-20] IEL= 0b11).



**Figure 12-449. Position Counter Underflow/Overflow (QPOSMAX = 4)**

eqep-011

#### 12.4.3.4.3.1.3 Position Counter Reset on the First Index Event (EQEP\_QDEC\_QEP\_CTL[29-28] PCRM = 0b10)

If the index event occurs during forward movement, then the position counter is reset to 0 on the next EQEP clock. If the index event occurs during the reverse movement, then the position counter is reset to the value in the EQEP\_QPOSMAX register on the next EQEP clock. Note that this is done only on the first occurrence and subsequently the position counter value is not reset on an index event; rather, it is reset based on maximum position as described in [Section 12.4.3.4.3.1.2](#).

First index marker is defined as the quadrature edge following the first index edge. The EQEP peripheral records the occurrence of the first index marker (EQEP\_QEP\_STS\_CT[1] FIMF) and direction on the first index event marker (EQEP\_QEP\_STS\_CT[6] FIDF) in EQEP\_QEP\_STS\_CT registers. It also remembers the quadrature edge on the first index marker so that same relative quadrature transition is used for software index marker (EQEP\_QDEC\_QEP\_CTL[21-20] IEL = 0b11).

#### 12.4.3.4.3.1.4 Position Counter Reset on Unit Time out Event (EQEP\_QDEC\_QEP\_CTL[29-28] PCRM = 0b11)

In this mode, the QPOSCNT value is latched to the EQEP\_QPOSILAT register and then the QPOSCNT field is reset (to 0 or the QPOSMAX value in the EQEP\_QPOSMAX register, depending on the direction mode selected by EQEP\_QDEC\_QEP\_CTL[15-14] QSRC bits on a unit time event). This is useful for frequency measurement.

#### 12.4.3.4.3.2 EQEP Position Counter Latch

The EQEP index and strobe input can be configured to latch the position counter QPOSCNT (EQEP\_QPOSCNT) into QPOSILAT (EQEP\_QPOSILAT register) and QPOSSLAT (EQEP\_QPOSSLAT register) bitfields, respectively, on occurrence of a definite event on these pins.

##### 12.4.3.4.3.2.1 Index Event Latch

In some applications, it may not be desirable to reset the position counter on every index event and instead it may be required to operate the position counter in full 32-bit mode (EQEP\_QDEC\_QEP\_CTL[29-28] PCRM = 0b01 and EQEP\_QDEC\_QEP\_CTL[29-28] PCRM = 0b10 modes).

In such cases, the EQEP position counter can be configured to latch on the following events and direction information is recorded in the EQEP\_QEP\_STS\_CT[4] QDLF bit on every index event marker.

- Latch on Rising edge (EQEP\_QDEC\_QEP\_CTL[21-20] IEL = 0b01)
- Latch on Falling edge (EQEP\_QDEC\_QEP\_CTL[21-20] IEL = 0b10)
- Latch on Index Event Marker (EQEP\_QDEC\_QEP\_CTL[21-20] IEL = 0b11)

This is particularly useful as an error checking mechanism to check if the position counter accumulated the correct number of counts between index events. As an example, the 1000-line encoder must count 4000 times when moving in the same direction between the index events.

The index event latch interrupt flag (EQEP\_QINT\_EN\_FLG[26] IELI\_FLG) is set when the position counter is latched to the EQEP\_QPOSILAT register. The index event latch configuration bits are ignored when EQEP\_QDEC\_QEP\_CTL[29-28] PCRM = 0b00.

When Position counter reset on an index event mode is selected through EQEP\_QDEC\_QEP\_CTL[29-28] PCRM bit field (value: 0h), EQEP\_QDEC\_QEP\_CTL[21-20] IEL bit field must be configured to 0h and position counter value is latched into the EQEP\_QPOSILAT[31-0] QPOSILAT register for every index marker.

##### Latch on Rising Edge

(EQEP\_QDEC\_QEP\_CTL[21-20] IEL = 0b01)

The position counter value (QPOSCNT) is latched to the EQEP\_QPOSILAT register on every rising edge of an index input.

##### Latch on Falling Edge

(EQEP\_QDEC\_QEP\_CTL[21-20] IEL = 0b10)

The position counter value (QPOSCNT) is latched to the EQEP\_QPOSILAT register on every falling edge of index input.

##### Latch on Index Event

Marker/Software Index Marker  
(EQEP\_QDEC\_QEP\_CTL[21-20] IEL = 0b11)

The first index marker is defined as the quadrature edge following the first index edge. The EQEP peripheral records the occurrence of the first index marker (EQEP\_QEP\_STS\_CT[1] FIMF) and

Figure 12-450 shows the position counter latch using an index event marker.



2358 J721E DRA829/TDA4VM/AM68P Processors Silicon Revision 1.1 Texas  
Instruments Families of Products

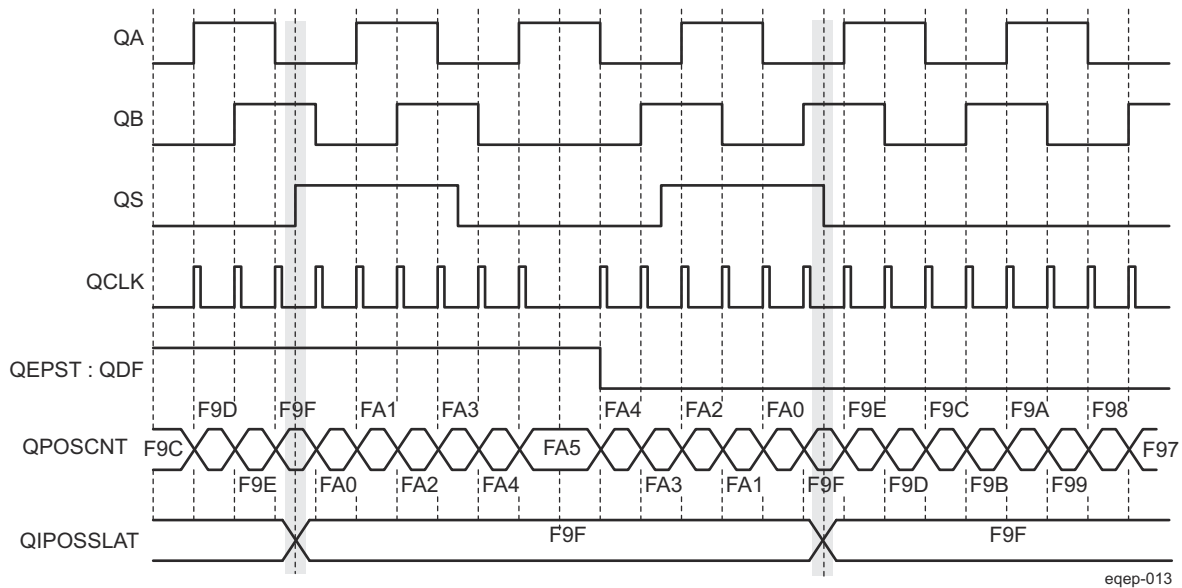


#### 12.4.3.4.3.2.2 EQEP Strobe Event Latch

The position-counter value is latched to the EQEP\_QPOSSLAT register on the rising edge of the strobe input (QCLK) by clearing the EQEP\_QDEC\_QEP\_CTL[22] SEL bit. Latching on the falling edge of the strobe input (QCLK) can be done by inverting the strobe input using the EQEP\_QDEC\_QEP\_CTL[5] QSP bit.

If the EQEP\_QDEC\_QEP\_CTL[22] SEL bit is set, then the position counter value is latched to the EQEP\_QPOSSLAT register on the rising edge of the strobe input for forward direction and on the falling edge of the strobe input for reverse direction as shown in [Figure 12-451](#).

The strobe event latch interrupt flag (EQEP\_QINT\_EN\_FLG[25] SELI\_FLG) is set when the position counter is latched to the EQEP\_QPOSSLAT register.



**Figure 12-451. EQEP Strobe Event Latch (EQEP\_QDEC\_QEP\_CTL[22] SEL = 0b1)**

### 12.4.3.4.3.3 EQEP Position Counter Initialization

The position counter can be initialized using following events:

- Index event
- Strobe event
- Software initialization.

#### Note

When all of the above events occur simultaneously, the sequence of priority is as follows: 1) Software Initialization, 2) strobe event initialization, 3) Index Event Initialization.

#### Index Event Initialization (IEI)

The EQEPx\_I index input can be used to trigger the initialization of the position counter at the rising or falling edge of the index input.

If the EQEP\_QDEC\_QEP\_CTL[25-24] IEI bits are 2h, then the position counter (EQEP\_QPOSCNT) is initialized with a value in the EQEP\_QPOSINIT register on the rising edge of strobe input for forward direction and on the falling edge of strobe input for reverse direction.

The index event initialization interrupt flag (EQEP\_QDEC\_QEP\_CTL[25-24] IEI) is set when the position counter is initialized with a value in the EQEP\_QPOSINIT register.

If EQEP\_QDEC\_QEP\_CTL[25-24] IEI bit field is configured with value 0h (default) or 1h, the index event initialization of position counter is disabled.

If EQEP\_QDEC\_QEP\_CTL[25-24] IEI bit field is configured with value 3h, then the the position counter (EQEP\_QPOSCNT) is initialized with a value in the EQEP\_QPOSINIT register on the falling edge of strobe input signal.

#### Strobe Event Initialization (SEI)

If the EQEP\_QDEC\_QEP\_CTL[27-26] SEI bits are 2h, then the position counter is initialized with a value in the EQEP\_QPOSINIT register on the rising edge of strobe input.

If the EQEP\_QDEC\_QEP\_CTL[27-26] SEI bits are 3h, then the position counter (EQEP\_QPOSCNT) is initialized with a value in the EQEP\_QPOSINIT register on the rising edge of strobe input for forward direction and on the falling edge of strobe input for reverse direction.

The strobe event initialization interrupt flag (EQEP\_QDEC\_QEP\_CTL[27-26] SEI) is set when the position counter is initialized with a value in the EQEP\_QPOSINIT register.

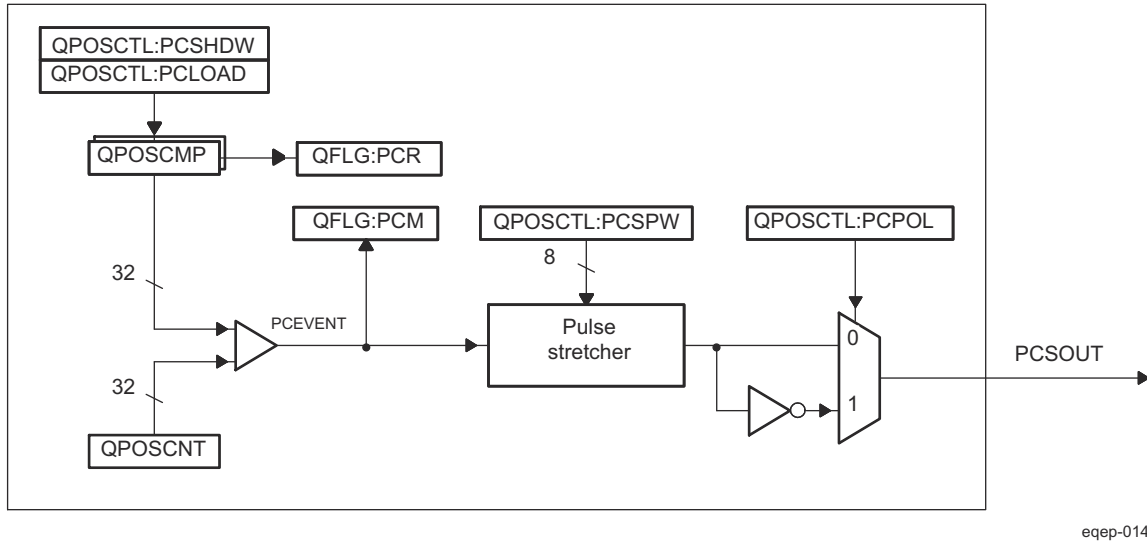
If EQEP\_QDEC\_QEP\_CTL[27-26] SEI bit field is configured with value 0h (default) or 1h, the strobe event initialization of position counter is disabled.

#### Software Initialization (SWI)

The position counter can be initialized in software by writing a 1h to the EQEP\_QDEC\_QEP\_CTL[23] SWI bit, which will automatically be cleared after initialization.

### 12.4.3.4.3.4 EQEP Position-Compare Unit

The EQEP peripheral includes a position-compare unit that is used to generate a sync output and/or interrupt on a position-compare match. [Figure 12-452](#) shows a diagram. The position-compare (EQEP\_QPOSCMP) register is shadowed and shadow mode can be enabled or disabled using the EQEP\_QCAP\_QPOS\_CTL[31] PCSHDW bit. If the shadow mode is not enabled, the CPU writes directly to the active position compare register.



**Figure 12-452. EQEP Position-compare Unit**

In shadow mode, SW can configure the position-compare unit (EQEP\_QCAP\_QPOS\_CTL[30] PCLOAD) to load the shadow register value into the active register on the following events and to generate the position-compare ready (EQEP\_QINT\_EN\_FLG[23] PCRI\_FLG) interrupt after loading.

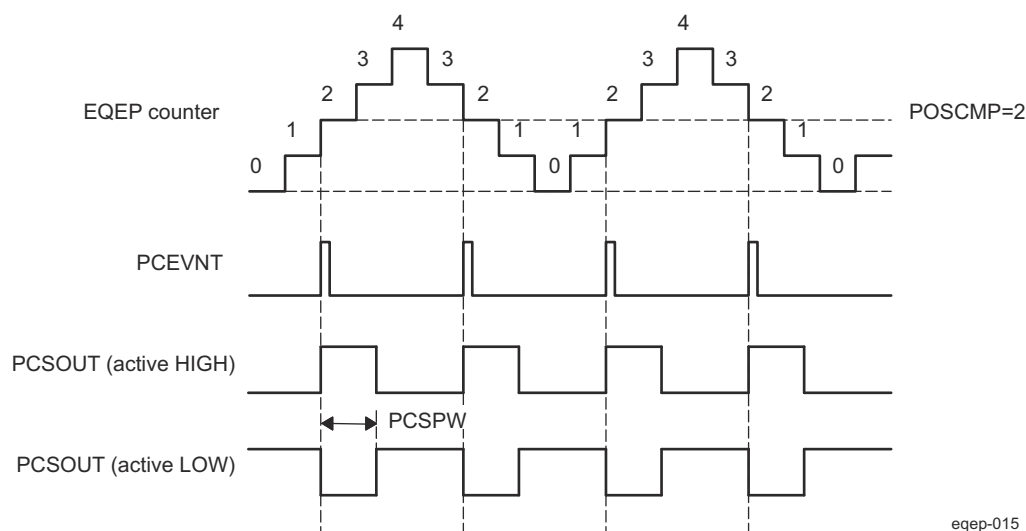
- Load on compare match
- Load on position-counter zero event.

The position-compare match (EQEP\_QINT\_EN\_FLG[24] PCMI\_FLG) is set when the position-counter value (QPOSCNT) matches with the active position-compare register (EQEP\_QPOSCMP) and the position-compare sync output of the programmable pulse width is generated on compare match to trigger an external device.

For example, if EQEP\_QPOSCMP bitfield POSCMP = 0x2, the position-compare unit generates a position-compare event on the transition from 1 to 2 of the EQEP position counter for forward counting direction and on the transition from 3 to 2 of the EQEP position counter for reverse counting direction (see [Figure 12-453](#)).

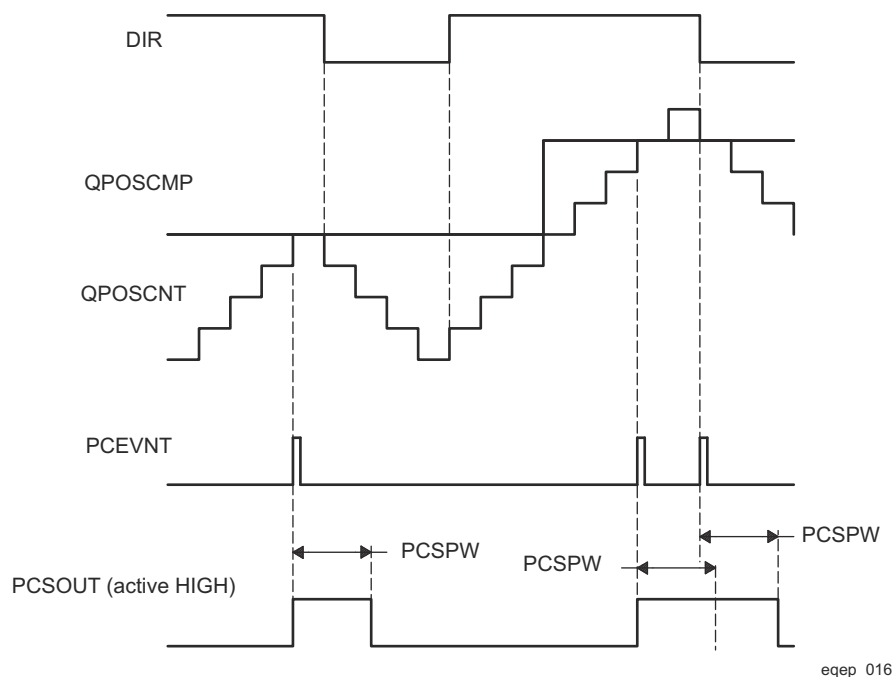
#### Note

When EQEP\_QCAP\_QPOS\_CTL[30] PCLOAD=0h, the shadow register is loaded into the active register as soon as POSCNT becomes zero, and it should not generate another shadow load if the POSCNT continues to stay at zero



**Figure 12-453. EQEP Position-compare Event Generation Points**

The pulse stretcher logic in the position-compare unit generates a programmable position-compare sync pulse output on the position-compare match. In the event of a new position-compare match while a previous position-compare pulse is still active, then the pulse stretcher generates a pulse of specified duration from the new position-compare event as shown in Figure 12-454.



**Figure 12-454. EQEP Position-compare Sync Output Pulse Stretcher**

#### 12.4.3.4.4 EQEP Edge Capture Unit

The EQEP peripheral includes an integrated edge capture unit to measure the elapsed time between the unit position events as shown in Figure 12-455. This feature is typically used for low speed measurement using the following equation:

$$V(k) = \frac{X}{t(k) - t(k-1)} = \frac{X}{\Delta T} \quad \text{eqep-017} \quad (21)$$

where,

- X - Unit position is defined by integer multiple of quadrature edges (see [Figure 12-456](#))
- $\Delta T$  - Elapsed time between unit position events
- v(k) - Velocity at time instant "k"

The EQEP capture timer (QCTMR bitfield in EQEP\_QCTMR register) runs from prescaled SYSCLKOUT and the prescaler is programmed by the EQEP\_QCAP\_QPOS\_CTL[6-4] CCPS bits. The capture timer QCTMR value is latched into the capture period register (EQEP\_QC\_PRD\_TLAT) on every unit position event and then the capture timer is reset.

Time measurement ( $\Delta T$ ) between unit position events will be correct if the following conditions are met:

- No more than 65,535 counts have occurred between unit position events.
- No direction change between unit position events.

The capture unit sets the EQEP overflow error flag (EQEP\_QEP\_STS\_CT[3] COEF) in the event of capture timer overflow between unit position events. If a direction change occurs between the unit position events, then an error flag is set in the status register (EQEP\_QEP\_STS\_CT[2] CDEF).

Capture Timer (EQEP\_QCTMR register) and Capture period register (EQEP\_QC\_PRD\_TLAT) can be configured to latch on following events.

- CPU read of EQEP\_QPOSCNT register
- Unit time-out event

If the EQEP\_QDEC\_QEP\_CTL[18] QCLM bit is cleared, then the capture timer and capture period values are latched into the EQEP\_QC\_PRD\_TLAT and EQEP\_QCPRDLAT registers, respectively, when the CPU reads the position counter in EQEP\_QPOSCNT.

---

#### Note

Capture timer and capture period values are not latched for an emulation read.

---

If the EQEP\_QDEC\_QEP\_CTL[18] QCLM bit is set, then the position counter, capture timer, and capture period values are latched into the EQEP\_QPOSLAT, EQEP\_QC\_PRD\_TLAT and EQEP\_QCPRDLAT registers, respectively, on unit time out.

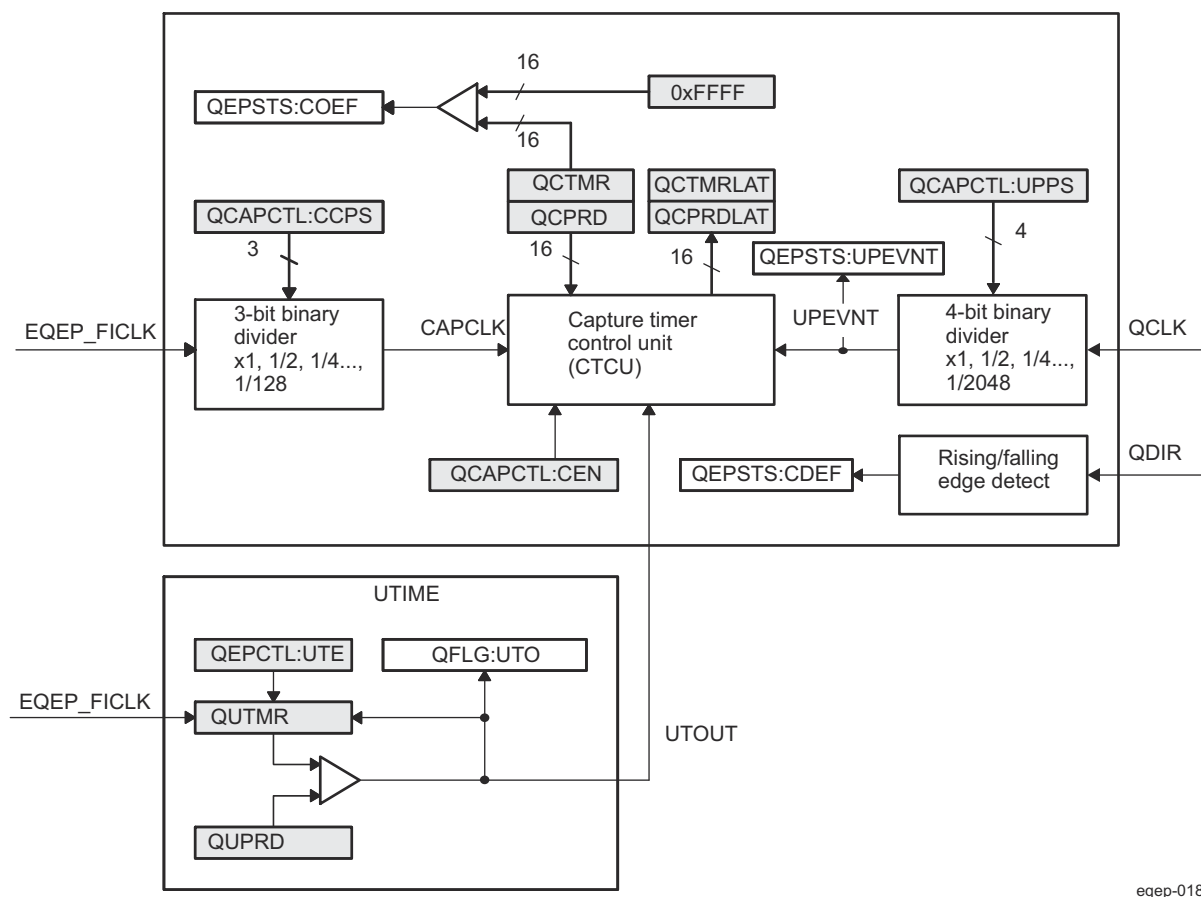
[Figure 12-457](#) shows the capture unit operation along with the position counter.

---

#### Note

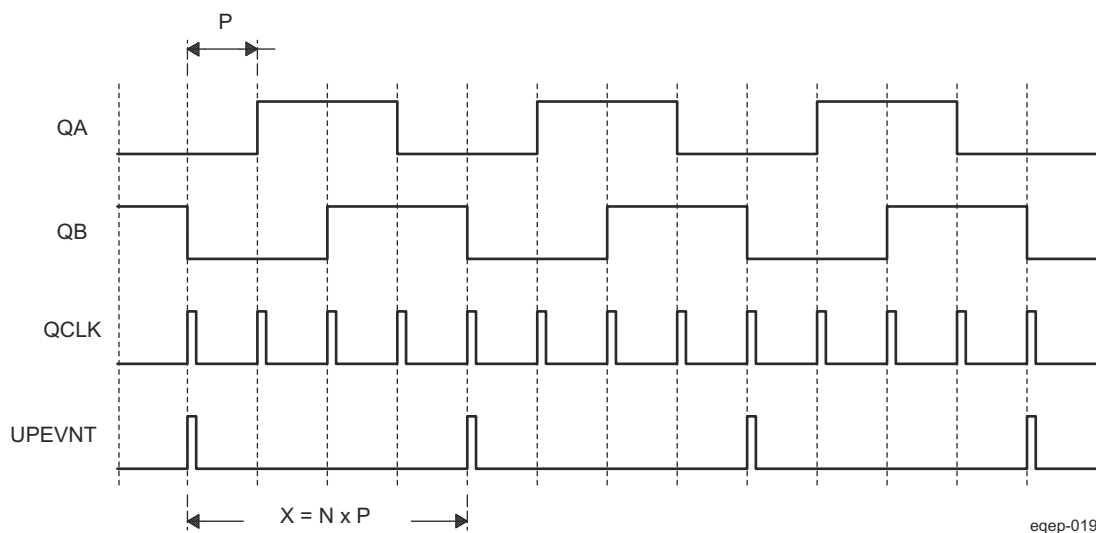
The EQEP\_QCAP\_QPOS\_CTL register should not be modified dynamically (such as switching CAPCLK prescaling mode from SYSCLKOUT/4 to SYSCLKOUT/8, where SYSCLKOUT is equivalent to EQEPx\_FICLK). The capture unit must be disabled before changing the prescaler.

---



eqep-018

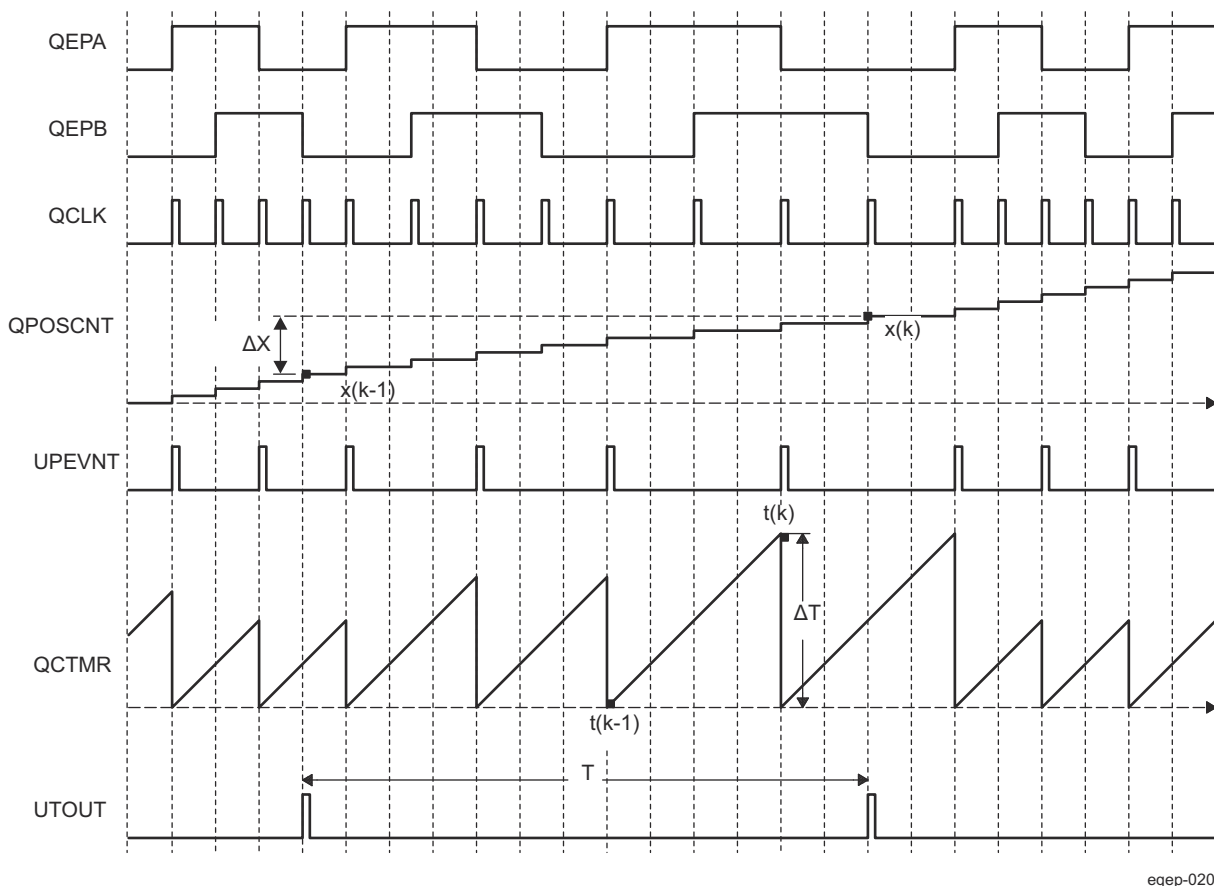
Figure 12-455. EQEP Edge Capture Unit



eqep-019

N - Number of quadrature periods selected using EQEP\_QCAP\_QPOS\_CTL[3-0] UPPS bits

Figure 12-456. Unit Position Event for Low Speed Measurement (EQEP\_QCAP\_QPOS\_CTL[UPPS] = 0010)



**Figure 12-457. EQEP Edge Capture Unit - Timing Details**

Velocity Calculation Equations:

$$V(k) = \frac{x(k) - x(k-1)}{T} = \frac{\Delta X}{T} \quad \text{or}$$

$$V(k) = \frac{X}{t(k) - t(k-1)} = \frac{X}{\Delta T}$$

eqep\_021

(22)

where

$v(k)$ : Velocity at time instant  $k$

$x(k)$ : Position at time instant  $k$

$x(k-1)$ : Position at time instant  $k - 1$

$T$ : Fixed unit time or inverse of velocity calculation rate

$\Delta X$ : Incremental position movement in unit time

$X$ : Fixed unit position

$\Delta T$ : Incremental time elapsed for unit position movement

$t(k)$ : Time instant " $k$ "

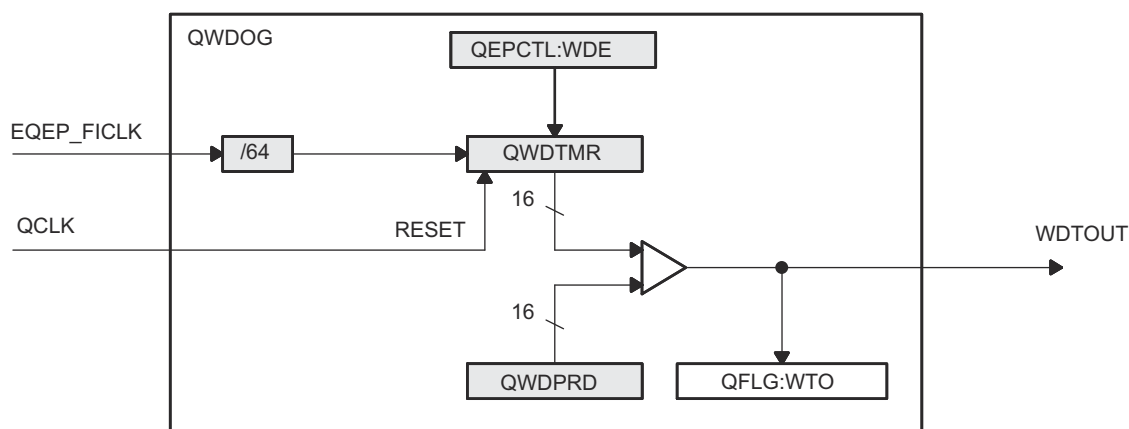
$t(k-1)$ : Time instant " $k - 1$ "

Unit time (T) and unit period (X) are configured using the EQEP\_QUPRD and EQEP\_QCAP\_QPOS\_CTL[3-0] UPPS registers. Incremental position output and incremental time output is available in the EQEP\_QOSLAT and EQEP\_QCPRLAT registers.

Parameter	Relevant Register to Configure or Read the Information
T	Unit Period Register (EQEP_QUPRD)
$\Delta X$	Incremental Position = QOSLAT(k) - QOSLAT(k - 1)
X	Fixed unit position defined by sensor resolution and QCAPCTL[3-0] UPPS bits
$\Delta T$	Capture Period Latch (QCPRLAT)

#### 12.4.3.4.5 EQEP Watchdog

The EQEP peripheral contains a 16-bit watchdog timer that monitors the quadrature-clock to indicate proper operation of the motion-control system. The EQEP watchdog timer is clocked from SYSCLKOUT/64 and the quadrature clock event (pulse) resets the watchdog timer. If no quadrature-clock event is detected until a period match (QWDPRD = QWDTMR), then the watchdog timer will time out and the watchdog interrupt flag will be set (EQEP\_QINT\_EN\_FLG[20] WDOI\_FLG). The time-out value is programmable through the watchdog period bit field (EQEP\_QWD\_TMR\_PRD[31-16] QWDPRD).



eqep-022

**Figure 12-458. EQEP Watchdog Timer**





#### 12.4.3.4.8 Summary of EQEP Functional Registers

Table 12-468 lists the registers with their memory locations, sizes, and reset values.

**Table 12-468. EQEP Control and Status Functional Registers**

Offset	Acronym	Register Description	Size (×16)/ #shadow
0h	EQEP_QPOSCNT	EQEP Position Counter Register	2/0
4h	EQEP_QPOSINIT	EQEP Position Counter Initialization Register	2/0
8h	EQEP_QPOSMAX	EQEP Maximum Position Count Register	2/0
Ch	EQEP_QPOSCMP	EQEP Position-Compare Register	2/1
10h	EQEP_QPOSILAT	EQEP Index Position Latch Register	2/0
14h	EQEP_QPOSSLAT	EQEP Strobe Position Latch Register	2/0
18h	EQEP_QPOSLAT	EQEP Position Counter Latch Register	2/0
1Ch	EQEP_QUTMR	EQEP Unit Timer Register	2/0
20h	EQEP_QUPRD	EQEP Unit Period Register	2/0
24h	EQEP_QWD_TMR_PRD	EQEP Watchdog Timer and Period Register	2/0
28h	EQEP_QDEC_QEP_CTL	EQEP Decoder and EQEP Control Register	2/0
2Ch	EQEP_QCAP_QPOS_CTL	EQEP Capture and Position Compare Control Register	2/0
30h	EQEP_QINT_EN_FLG	EQEP Interrupt Enable and Flag Register	2/0
34h	EQEP_QINT_CLR_FRC	EQEP Interrupt Clear and Forcing Register	2/0
38h	EQEP_QEP_STS_CT	EQEP Status and Capture Timer Register	2/0
3Ch	EQEP_QC_PRD_TLAT	EQEP Capture Period and Timer Latch Register	2/0
40h	EQEP_QCPRDLAT	EQEP Capture Period Latch Register	1/0
5Ch	EQEP_PID	EQEP Revision ID Register	2/0

## 12.4.4 Controller Area Network (MCAN)

This section describes the Controller Area Network (MCAN) modules in the device.

### 12.4.4.1 MCAN Overview

The Controller Area Network (CAN) is a serial communications protocol which efficiently supports distributed real-time control. CAN has high immunity to electrical interference. In a CAN network, many short messages are broadcast to the entire network, which provides for data consistency in every node of the system.

The MCAN module supports both classic CAN and CAN FD (CAN with Flexible Data-Rate) specifications. CAN FD feature allows high throughput and increased payload per data frame. The classic CAN and CAN FD devices can coexist on the same network without any conflict.

The device supports 16 MCAN modules:

- 2 in the device MCU domain
- 14 in the device MAIN domain

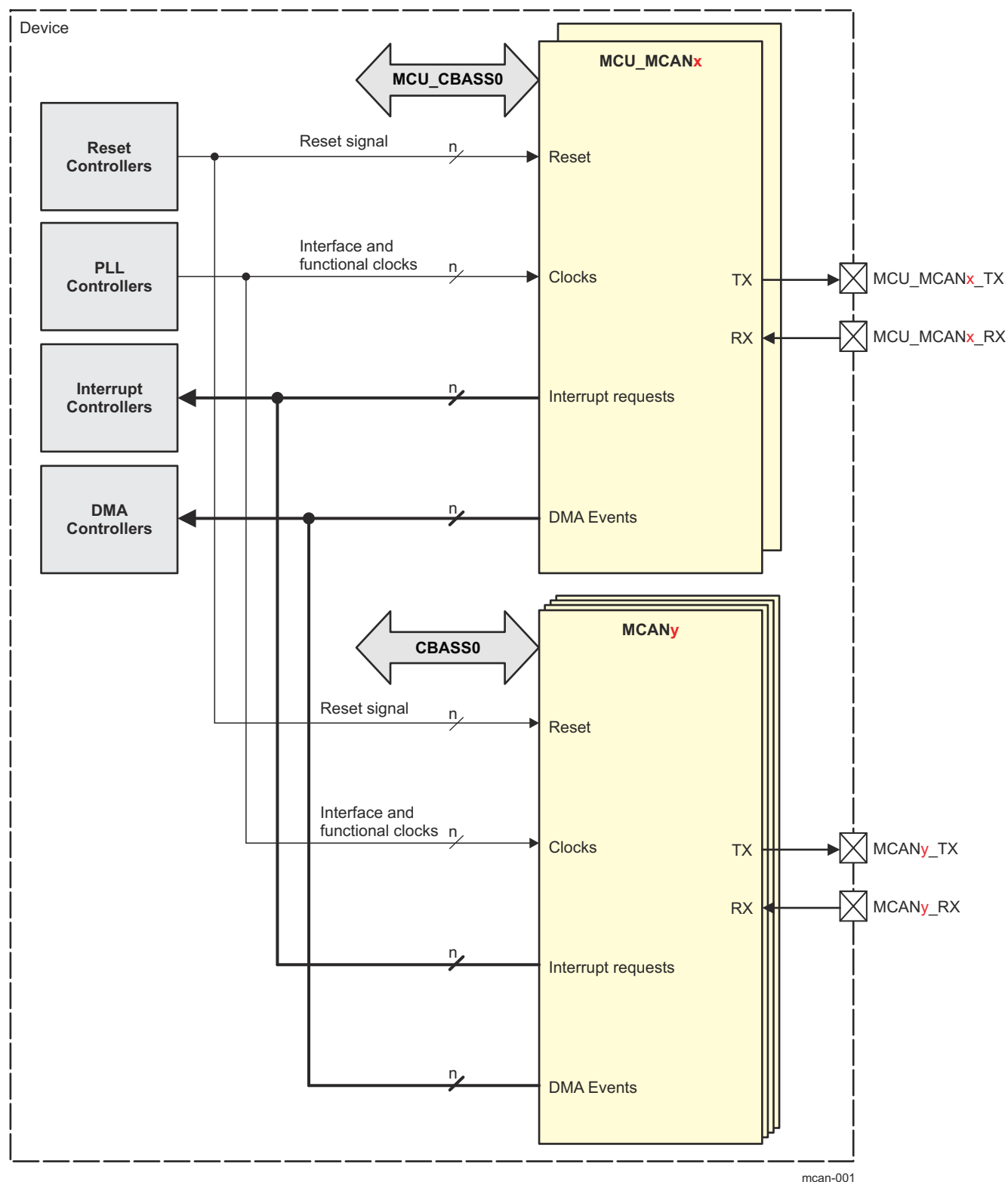
They connect to the physical layer of the CAN network through external (for the device) transceivers. Each MCAN module supports flexible bit rates greater than 1 Mbps and is compliant to ISO 11898-1:2015.

[Table 12-469](#) shows MCAN allocation across device domains.

**Table 12-469. MCAN Allocation Across Device Domains**

Instance	Domain		
	WKUP	MCU	MAIN
MCU_MCAN0	-	✓	-
MCU_MCAN1	-	✓	-
MCAN0	-	-	✓
MCAN1	-	-	✓
MCAN2	-	-	✓
MCAN3	-	-	✓
MCAN4	-	-	✓
MCAN5	-	-	✓
MCAN6	-	-	✓
MCAN7	-	-	✓
MCAN8	-	-	✓
MCAN9	-	-	✓
MCAN10	-	-	✓
MCAN11	-	-	✓
MCAN12	-	-	✓
MCAN13	-	-	✓

[Figure 12-461](#) shows the MCAN modules overview.



- A.  $x = 0 - 1$   
B.  $y = 0 - 13$

Figure 12-461. MCAN Modules Overview

#### **12.4.4.1.1 MCAN Features**

Each MCAN module implements the following features:

- Conforms with CAN Protocol 2.0 A, B and ISO 11898-1:2015
- Full CAN FD support (up to 64 data bytes)
- SAE J1939 support
- AUTOSAR support (only on DRA829 and TDA4VM family of devices)
- Up to 32 dedicated Transmit Buffers
- Configurable Transmit FIFO, up to 32 elements
- Configurable Transmit Queue, up to 32 elements
- Configurable Transmit Event FIFO, up to 32 elements
- Up to 64 dedicated Receive Buffers
- Two configurable Receive FIFOs, up to 64 elements each
- Up to 128 filter elements
- Internal Loopback mode for self-test
- Maskable interrupts, two interrupt lines
- Two clock domains (CAN clock/Host clock)
- Parity/ECC support - Message RAM single error correction and double error detection (SECDED) mechanism
- Local power-down and wakeup support
- Timestamp Counter

#### **12.4.4.1.2 MCAN Not Supported Features**

- Host bus firewall
- GPIO mode
- Clock calibration
- External (IO) Loopback mode
- Debug DMA (see [Section 12.4.4.4.7.4](#))
- TX DMA channels [31:3]

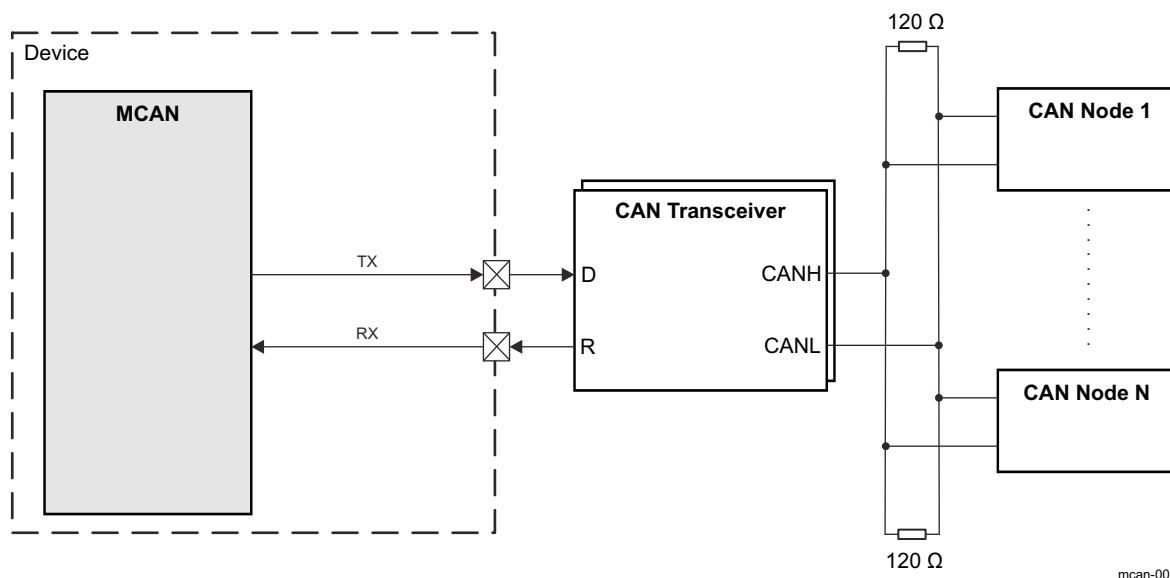
### 12.4.4.2 MCAN Environment

The MCU\_MCAN[0-1] and MCAN[0-13] modules are hereinafter referred to as MCAN module.

This section describes the MCAN external connections (environment).

CAN network physical layer consists of two-wire differential bus, usually twisted pair, and provides high level of interference immunity. External CAN transceiver IC is needed to access a CAN bus by the MCAN.

Figure 12-462 shows the MCAN typical application.



**Figure 12-462. MCAN Typical Application**

Table 12-470 describes the MCAN I/O signals.

**Table 12-470. MCAN I/O Signals**

Module Pin	Device Level Signal	I/O <sup>(1)</sup>	Description	Module Pin Reset Value <sup>(1)</sup>
<b>MCU_MCAN[0-1]</b>				
RX	MCU_MCAN0_RX	I	Serial data input from external CAN transceiver	HiZ
TX	MCU_MCAN0_TX	O	Serial data output to external CAN transceiver	1
RX	MCU_MCAN1_RX	I	Serial data input from external CAN transceiver	HiZ
TX	MCU_MCAN1_TX	O	Serial data output to external CAN transceiver	1
<b>MCAN[0-13]</b>				
RX	MCAN0_RX	I	Serial data input from external CAN transceiver	HiZ
TX	MCAN0_TX	O	Serial data output to external CAN transceiver	1
RX	MCAN1_RX	I	Serial data input from external CAN transceiver	HiZ
TX	MCAN1_TX	O	Serial data output to external CAN transceiver	1
RX	MCAN2_RX	I	Serial data input from external CAN transceiver	HiZ
TX	MCAN2_TX	O	Serial data output to external CAN transceiver	1
RX	MCAN3_RX	I	Serial data input from external CAN transceiver	HiZ
TX	MCAN3_TX	O	Serial data output to external CAN transceiver	1
RX	MCAN4_RX	I	Serial data input from external CAN transceiver	HiZ
TX	MCAN4_TX	O	Serial data output to external CAN transceiver	1
RX	MCAN5_RX	I	Serial data input from external CAN transceiver	HiZ
TX	MCAN5_TX	O	Serial data output to external CAN transceiver	1
RX	MCAN6_RX	I	Serial data input from external CAN transceiver	HiZ

**Table 12-470. MCAN I/O Signals (continued)**

Module Pin	Device Level Signal	I/O <sup>(1)</sup>	Description	Module Pin Reset Value <sup>(1)</sup>
TX	MCAN6_TX	O	Serial data output to external CAN transceiver	1
RX	MCAN7_RX	I	Serial data input from external CAN transceiver	HiZ
TX	MCAN7_TX	O	Serial data output to external CAN transceiver	1
RX	MCAN8_RX	I	Serial data input from external CAN transceiver	HiZ
TX	MCAN8_TX	O	Serial data output to external CAN transceiver	1
RX	MCAN9_RX	I	Serial data input from external CAN transceiver	HiZ
TX	MCAN9_TX	O	Serial data output to external CAN transceiver	1
RX	MCAN10_RX	I	Serial data input from external CAN transceiver	HiZ
RX	MCAN10_RX	I	Serial data input from external CAN transceiver	HiZ
TX	MCAN11_TX	O	Serial data output to external CAN transceiver	1
RX	MCAN11_RX	I	Serial data input from external CAN transceiver	HiZ
TX	MCAN12_TX	O	Serial data output to external CAN transceiver	1
RX	MCAN12_RX	I	Serial data input from external CAN transceiver	HiZ
TX	MCAN13_TX	O	Serial data output to external CAN transceiver	1
RX	MCAN13_RX	I	Serial data input from external CAN transceiver	HiZ

(1) I = Input; O = Output; HiZ = High Impedance

### Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables Pin Attributes and Pin Multiplexing in the device-specific Datasheet.

#### 12.4.4.2.1 CAN Network Basics

- CAN bus is a 2-wire differential bus using Non-Return-to-Zero (NRZ) encoding and has two states:
  - Recessive state (logical 1)
  - Dominant state (logical 0)
- The network is multimaster. When two or more nodes (ECUs) attempt to transmit at the same time, a non-destructive arbitration technique guarantees messages are sent in order of priority and no messages are lost.
- The message transmission is multicast. Data messages transmitted are identifier based, not address based.
- Content of message is labeled by the identifier that is unique throughout the network (for example: rpm, temperature, position, pressure, and so forth).
- All nodes on network receive the message and each performs an acceptance test on the identifier. If message is relevant, it is processed, otherwise it is ignored.
- The unique identifier also determines the priority of the message (the lower the numerical value of the identifier, the higher the priority is).
- Data is transmitted and received using message frames, consisting of the following basic fields:
  - Arbitration field
  - Control field
  - Data field (up to 8 bytes for Classical CAN and up to 64 bytes for CAN FD)
  - CRC field
  - ACK field

For more information, see *ISO 11898-1:2015: CAN data link layer and physical signaling*.



### 12.4.4.3 MCAN Integration

This section describes the MCAN integration in the device, including information about clocks, resets, and hardware requests.

#### 12.4.4.3.1 MCAN Integration in MCU Domain

There are two MCAN modules integrated in the device MCU domain - MCU\_MCAN0 and MCU\_MCAN1.

Figure 12-463 shows the integration of MCU\_MCAN0.

Figure 12-464 shows the integration of MCU\_MCAN1.

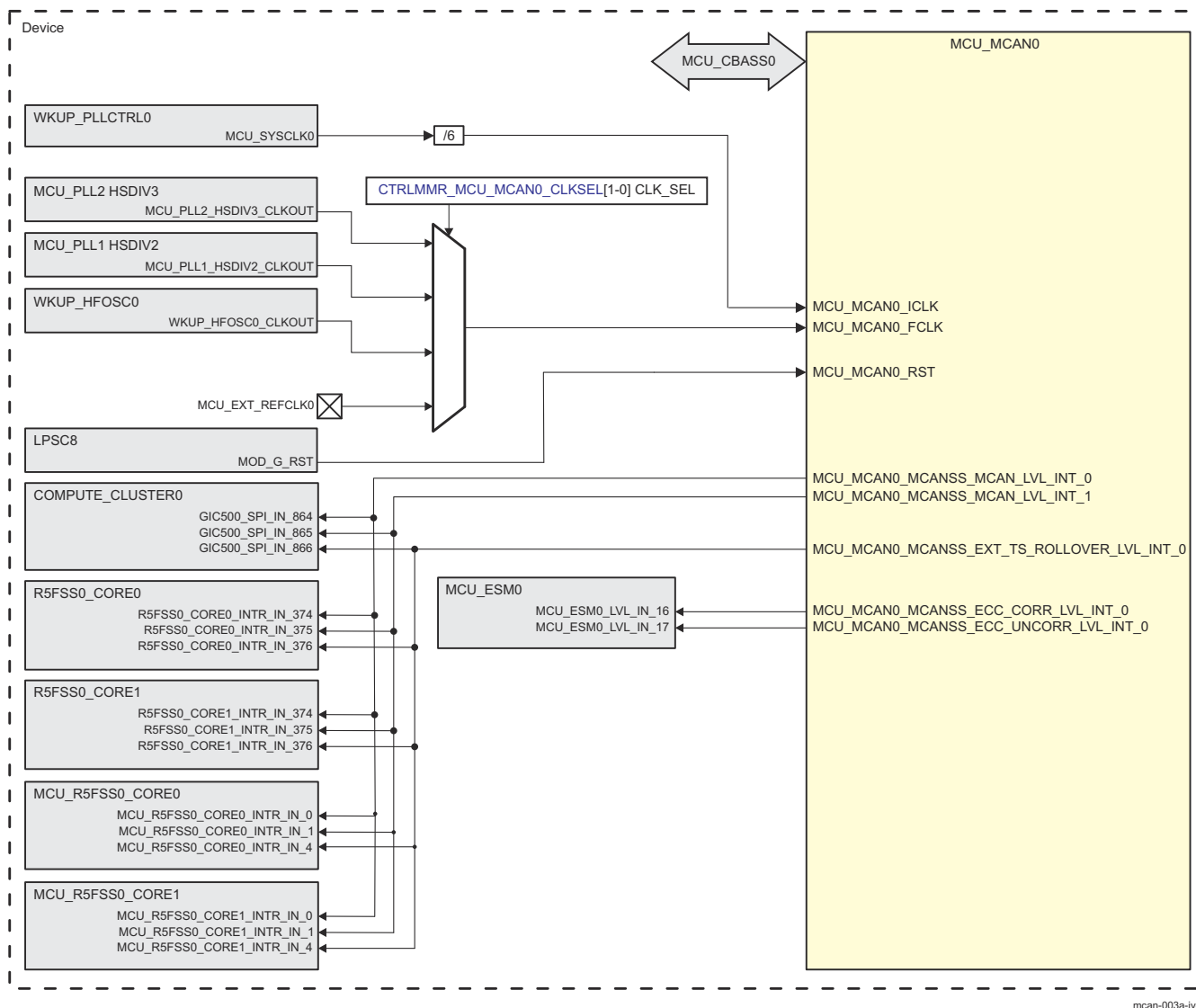


Figure 12-463. MCU\_MCAN0 Integration

mcan-003a-jvcl

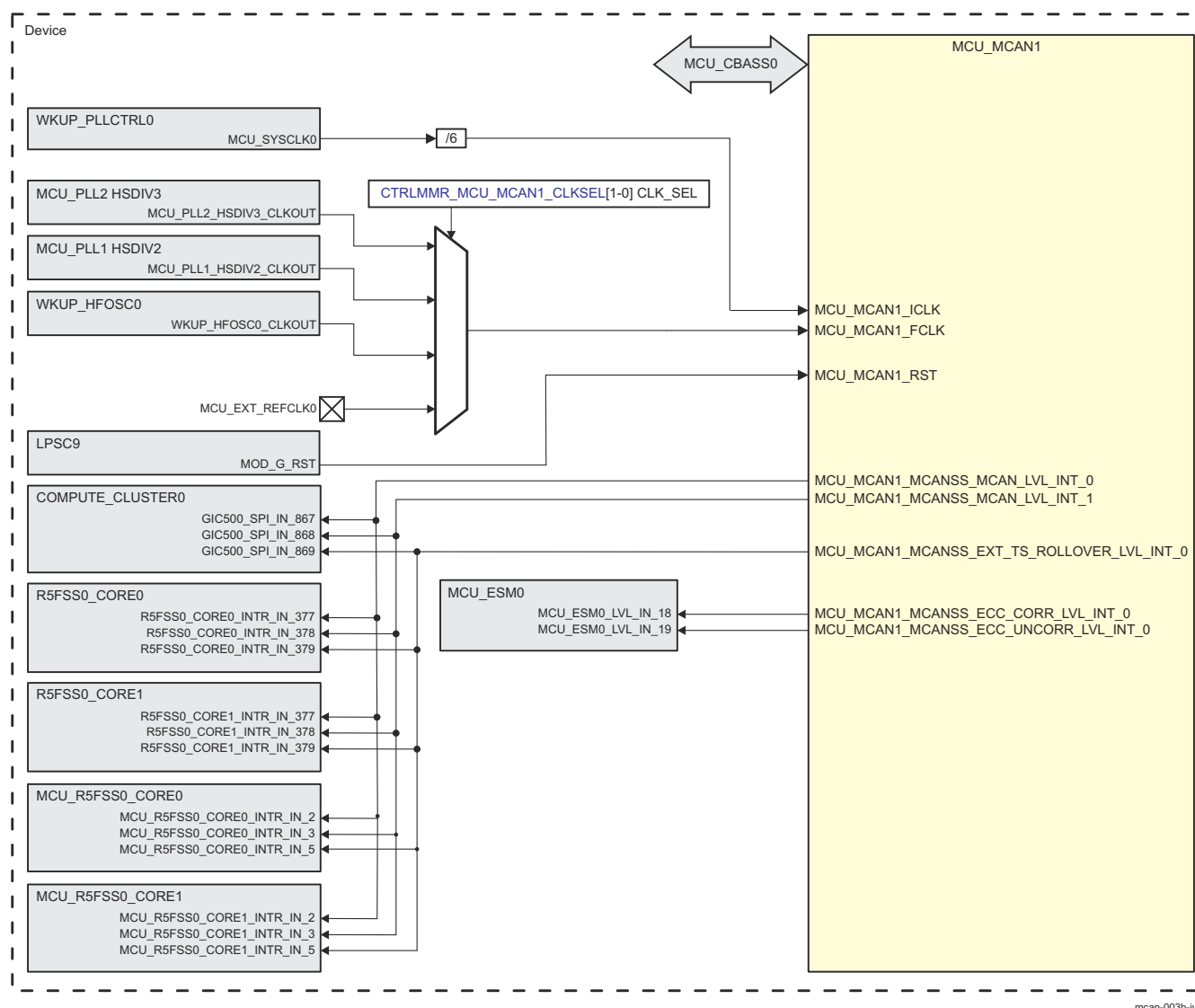

**Figure 12-464. MCU\_MCAN1 Integration**

Table 12-471 through Table 12-473 summarize the integration of MCU\_MCAN0 and MCU\_MCAN1 in the device MCU domain.

**Table 12-471. MCU\_MCAN Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
MCU_MCAN0	WKUP_PSC0	PD0	LPSC8	MCU_CBASS0
MCU_MCAN1	WKUP_PSC0	PD0	LPSC9	MCU_CBASS0

**Table 12-472. MCU\_MCAN Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
MCU_MCAN0	MCU_MCAN0_ICLK	MCU_SYSCLK0/6	WKUP_PLLCTRL0	Interface Clock

**Table 12-472. MCU\_MCAN Clocks and Resets (continued)**

	MCU_MCAN0_FCLK	MCU_PLL2_HSDIV3_CLKOUT (default)	MCU_PLL2 HSDIV3	Functional Clock (for more information about clock multiplexing, see CTRLMMR_MCU_MCAN0_CLK_SEL[1-0] CLK_SEL bit field in <i>Control Module (CTRL_MMR)</i> . Default: CTRLMMR_MCU_MCAN0_CLK_SEL[1-0] CLK_SEL = 0h, MCU_PLL2_HSDIV3_CLKOUT is selected)
		MCU_EXT_REFCLK0	I/O pin	
		MCU_PLL1_HSDIV2_CLKOUT	MCU_PLL1 HSDIV2	
		WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	
MCU_MCAN1	MCU_MCAN1_ICLK	MCU_SYSCLK0/6	WKUP_PLLCTRL0	Interface Clock
	MCU_MCAN1_FCLK	MCU_PLL2_HSDIV3_CLKOUT (default)	MCU_PLL2 HSDIV3	Functional Clock (for more information about clock multiplexing, see CTRLMMR_MCU_MCAN1_CLK_SEL[1-0] CLK_SEL bit field in <i>Control Module (CTRL_MMR)</i> . Default: CTRLMMR_MCU_MCAN1_CLK_SEL[1-0] CLK_SEL = 0h, MCU_PLL2_HSDIV3_CLKOUT is selected)
		MCU_EXT_REFCLK0	I/O pin	
		MCU_PLL1_HSDIV2_CLKOUT	MCU_PLL1 HSDIV2	
		WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	
	Resets			
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
MCU_MCAN0	MCU_MCAN0_RST	MOD_G_RST	LPSC8	Asynchronous Module Reset
MCU_MCAN1	MCU_MCAN1_RST	MOD_G_RST	LPSC9	Asynchronous Module Reset

**Table 12-473. MCU\_MCAN Hardware Requests**

Interrupt Requests						
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type	
MCU_MCA N0	MCU_MCAN0_MCANSS_MCAN_LVL_INT_0	GIC500_SPI_IN_864	COMPUTE_CLUSTER0	MCU_MCAN0 Line 0 Interrupt Request	Level	
		R5FSS0_INTRTR0_IN_335	R5FSS0_INTRTR0			
		R5FSS1_INTRTR0_IN_335	R5FSS1_INTRTR0			
		MCU_R5FSS0_CORE0_INTR_IN_0	MCU_R5FSS0_CORE0			
		MCU_R5FSS0_CORE1_INTR_IN_0	MCU_R5FSS0_CORE1			
MCU_MCAN0_MCANSS_MCAN_LVL_INT_1	MCU_MCAN0_MCANSS_MCAN_LVL_INT_1	GIC500_SPI_IN_865	COMPUTE_CLUSTER0	MCU_MCAN0 Line 1 Interrupt Request	Level	
		R5FSS0_INTRTR0_IN_336	R5FSS0_INTRTR0			
		R5FSS1_INTRTR0_IN_336	R5FSS1_INTRTR0			
		MCU_R5FSS0_CORE0_INTR_IN_1	MCU_R5FSS0_CORE0			
		MCU_R5FSS0_CORE1_INTR_IN_1	MCU_R5FSS0_CORE1			
MCU_MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	MCU_MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	GIC500_SPI_IN_866	COMPUTE_CLUSTER0	MCU_MCAN0 External TimeStamp Counter Rollover Interrupt	Level	
		R5FSS0_INTRTR0_IN_337	R5FSS0_INTRTR0			
		R5FSS1_INTRTR0_IN_337	R5FSS1_INTRTR0			
		MCU_R5FSS0_CORE0_INTR_IN_4	MCU_R5FSS0_CORE0			

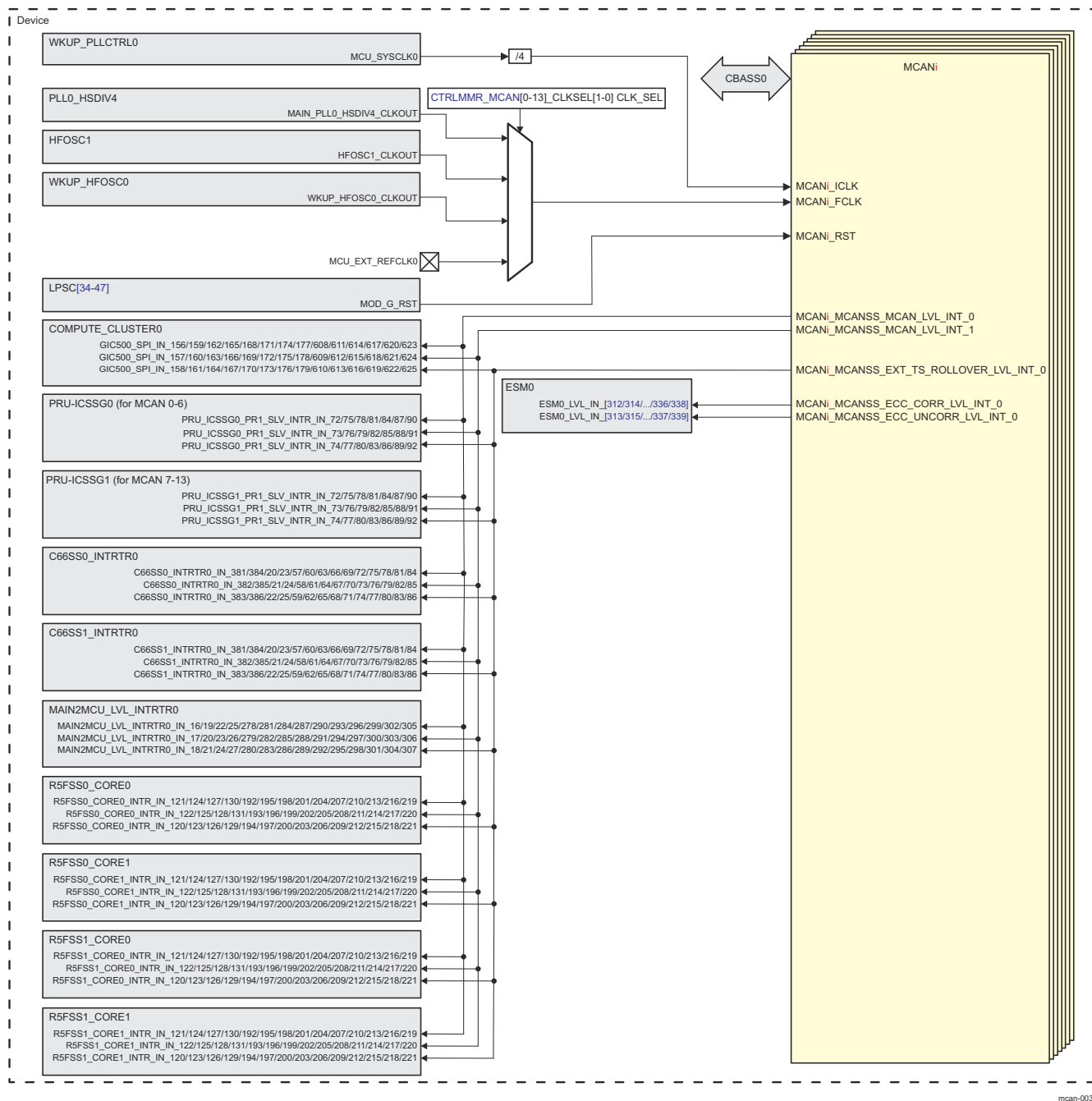
**Table 12-473. MCU\_MCAN Hardware Requests (continued)**

		MCU_R5FSS0_CORE1_INTR_IN_4	MCU_R5FSS0_CORE1		
MCU_MCAN0_MCANSS_ECC_CORR_LVL_INT_0		MCU_ESM0_LVL_IN_16	MCU_ESM0	MCU_MCAN 0 ECC Correctable Error Interrupt Request	Level
MCU_MCAN0_MCANSS_ECC_UNCORR_LVL_INT_0		MCU_ESM0_LVL_IN_17	MCU_ESM0	MCU_MCAN 0 ECC Uncorrectable Error Interrupt Request	Level
MCU_MCAN1_MCANSS_MCAN_LVL_INT_0	MCU_MCAN1_MCANSS_MCAN_LVL_INT_0	GIC500_SPI_IN_867	COMPUTE_CLUSTER0	MCU_MCAN 1 Line 0 Interrupt Request	Level
		R5FSS0_INTRTR0_IN_338	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_338	R5FSS1_INTRTR0		
		MCU_R5FSS0_CORE0_INTR_IN_2	MCU_R5FSS0_CORE0		
		MCU_R5FSS0_CORE1_INTR_IN_2	MCU_R5FSS0_CORE1		
MCU_MCAN1_MCANSS_MCAN_LVL_INT_1	MCU_MCAN1_MCANSS_MCAN_LVL_INT_1	GIC500_SPI_IN_868	COMPUTE_CLUSTER0	MCU_MCAN 1 Line 1 Interrupt Request	Level
		R5FSS0_INTRTR0_IN_339	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_339	R5FSS1_INTRTR0		
		MCU_R5FSS0_CORE0_INTR_IN_3	MCU_R5FSS0_CORE0		
		MCU_R5FSS0_CORE1_INTR_IN_3	MCU_R5FSS0_CORE1		
MCU_MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	MCU_MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	GIC500_SPI_IN_869	COMPUTE_CLUSTER0	MCU_MCAN 1 External TimeStamp Counter Rollover Interrupt	Level
		R5FSS0_INTRTR0_IN_340	R5FSS0_INTRTR0		
		R5FSS1_INTRTR0_IN_340	R5FSS1_INTRTR0		
		MCU_R5FSS0_CORE0_INTR_IN_5	MCU_R5FSS0_CORE0		
		MCU_R5FSS0_CORE1_INTR_IN_5	MCU_R5FSS0_CORE1		
MCU_MCAN1_MCANSS_ECC_CORR_LVL_INT_0		MCU_ESM0_LVL_IN_18	MCU_ESM0	MCU_MCAN 1 ECC Correctable Error Interrupt Request	Level
MCU_MCAN1_MCANSS_ECC_UNCORR_LVL_INT_0		MCU_ESM0_LVL_IN_19	MCU_ESM0	MCU_MCAN 1 ECC Uncorrectable Error Interrupt Request	Level

### 12.4.4.3.2 MCAN Integration in MAIN Domain

There are fourteen MCAN modules integrated in the device MAIN domain - MCAN[0-13].

Figure 12-465 shows the integration of MCAN[0-13].



A.  $i = 0 - 13$

**Figure 12-465. MCAN[0-13] Integration**

Table 12-474 through Table 12-476 summarize the integration of MCAN[0-13] in the device MAIN domain.

**Table 12-474. MCAN Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
MCAN0	PSC0	PD1	LPSC34	CBASS0
MCAN1	PSC0	PD1	LPSC35	CBASS0
MCAN2	PSC0	PD1	LPSC36	CBASS0
MCAN3	PSC0	PD1	LPSC37	CBASS0
MCAN4	PSC0	PD1	LPSC38	CBASS0
MCAN5	PSC0	PD1	LPSC39	CBASS0
MCAN6	PSC0	PD1	LPSC40	CBASS0
MCAN7	PSC0	PD1	LPSC41	CBASS0
MCAN8	PSC0	PD1	LPSC42	CBASS0
MCAN9	PSC0	PD1	LPSC43	CBASS0
MCAN10	PSC0	PD1	LPSC44	CBASS0
MCAN11	PSC0	PD1	LPSC45	CBASS0
MCAN12	PSC0	PD1	LPSC46	CBASS0
MCAN13	PSC0	PD1	LPSC47	CBASS0

**Table 12-475. MCAN Clocks and Resets**

Module Instance	Clocks			
	Module Clock Input	Source Clock Signal	Source	Description
MCAN0	MCAN0_ICLK	MCU_SYSCLK0/4	WKUP_PLLCTRL0	Interface Clock
	MCAN0_FCLK	MAIN_PLL0_HSDIV4_CLKOUT (default)	PLL0_HSDIV4	Functional Clock (for more information about clock multiplexing, see CTRLMMR_MCAN0_CLKSEL[1-0] CLK_SEL bit field in <i>Control Module (CTRL_MMR)</i> . Default: CTRLMMR_MCAN0_CLKSEL[1-0] CLK_SEL = 0h, MAIN_PLL0_HSDIV4_CLKOUT is selected)
		MCU_EXT_REFCLK0	I/O pin	
		HFOSC1_CLKOUT	HFOSC1	
		HFOSC0_CLKOUT	WKUP_HFOSC0	
MCAN1	MCAN1_ICLK	MCU_SYSCLK0/4	WKUP_PLLCTRL0	Interface Clock
	MCAN1_FCLK	MAIN_PLL0_HSDIV4_CLKOUT (default)	PLL0_HSDIV4	Functional Clock (for more information about clock multiplexing, see CTRLMMR_MCAN1_CLKSEL[1-0] CLK_SEL bit field in <i>Control Module (CTRL_MMR)</i> . Default: CTRLMMR_MCAN1_CLKSEL[1-0] CLK_SEL = 0h, MAIN_PLL0_HSDIV4_CLKOUT is selected)
		MCU_EXT_REFCLK0	I/O pin	
		HFOSC1_CLKOUT	HFOSC1	
		HFOSC0_CLKOUT	WKUP_HFOSC0	
MCAN2	MCAN2_ICLK	MCU_SYSCLK0/4	WKUP_PLLCTRL0	Interface Clock
	MCAN2_FCLK	MAIN_PLL0_HSDIV4_CLKOUT (default)	PLL0_HSDIV4	Functional Clock (for more information about clock multiplexing, see CTRLMMR_MCAN2_CLKSEL[1-0] CLK_SEL bit field in <i>Control Module (CTRL_MMR)</i> . Default: CTRLMMR_MCAN2_CLKSEL[1-0] CLK_SEL = 0h, MAIN_PLL0_HSDIV4_CLKOUT is selected)
		MCU_EXT_REFCLK0	I/O pin	
		HFOSC1_CLKOUT	HFOSC1	
		HFOSC0_CLKOUT	WKUP_HFOSC0	

**Table 12-475. MCAN Clocks and Resets (continued)**

MCAN3	MCAN3_ICLK	MCU_SYSCLK0/4	WKUP_PLLCTRL0	Interface Clock
	MCAN3_FCLK	MAIN_PLL0_HSDIV4_CLKOUT (default)	PLL0_HSDIV4	Functional Clock (for more information about clock multiplexing, see CTRLMMR_MCAN3_CLKSEL[1-0] CLK_SEL bit field in <i>Control Module (CTRL_MMR)</i> . Default: CTRLMMR_MCAN3_CLKSEL[1-0] CLK_SEL = 0h, MAIN_PLL0_HSDIV4_CLKOUT is selected)
		MCU_EXT_REFCLK0	I/O pin	
		HFOSC1_CLKOUT	HFOSC1	
		HFOSC0_CLKOUT	WKUP_HFOSC0	
MCAN4	MCAN4_ICLK	MCU_SYSCLK0/4	WKUP_PLLCTRL0	Interface Clock
	MCAN4_FCLK	MAIN_PLL0_HSDIV4_CLKOUT (default)	PLL0_HSDIV4	Functional Clock (for more information about clock multiplexing, see CTRLMMR_MCAN4_CLKSEL[1-0] CLK_SEL bit field in <i>Control Module (CTRL_MMR)</i> . Default: CTRLMMR_MCAN4_CLKSEL[1-0] CLK_SEL = 0h, MAIN_PLL0_HSDIV4_CLKOUT is selected)
		MCU_EXT_REFCLK0	I/O pin	
		HFOSC1_CLKOUT	HFOSC1	
		HFOSC0_CLKOUT	WKUP_HFOSC0	
MCAN5	MCAN5_ICLK	MCU_SYSCLK0/4	WKUP_PLLCTRL0	Interface Clock
	MCAN5_FCLK	MAIN_PLL0_HSDIV4_CLKOUT (default)	PLL0_HSDIV4	Functional Clock (for more information about clock multiplexing, see CTRLMMR_MCAN5_CLKSEL[1-0] CLK_SEL bit field in <i>Control Module (CTRL_MMR)</i> . Default: CTRLMMR_MCAN5_CLKSEL[1-0] CLK_SEL = 0h, MAIN_PLL0_HSDIV4_CLKOUT is selected)
		MCU_EXT_REFCLK0	I/O pin	
		HFOSC1_CLKOUT	HFOSC1	
		HFOSC0_CLKOUT	WKUP_HFOSC0	
MCAN6	MCAN6_ICLK	MCU_SYSCLK0/4	WKUP_PLLCTRL0	Interface Clock
	MCAN6_FCLK	MAIN_PLL0_HSDIV4_CLKOUT (default)	PLL0_HSDIV4	Functional Clock (for more information about clock multiplexing, see CTRLMMR_MCAN6_CLKSEL[1-0] CLK_SEL bit field in <i>Control Module (CTRL_MMR)</i> . Default: CTRLMMR_MCAN6_CLKSEL[1-0] CLK_SEL = 0h, MAIN_PLL0_HSDIV4_CLKOUT is selected)
		MCU_EXT_REFCLK0	I/O pin	
		HFOSC1_CLKOUT	HFOSC1	
		HFOSC0_CLKOUT	WKUP_HFOSC0	
MCAN7	MCAN7_ICLK	MCU_SYSCLK0/4	WKUP_PLLCTRL0	Interface Clock
	MCAN7_FCLK	MAIN_PLL0_HSDIV4_CLKOUT (default)	PLL0_HSDIV4	Functional Clock (for more information about clock multiplexing, see CTRLMMR_MCAN7_CLKSEL[1-0] CLK_SEL bit field in <i>Control Module (CTRL_MMR)</i> . Default: CTRLMMR_MCAN7_CLKSEL[1-0] CLK_SEL = 0h, MAIN_PLL0_HSDIV4_CLKOUT is selected)
		MCU_EXT_REFCLK0	I/O pin	
		HFOSC1_CLKOUT	HFOSC1	
		HFOSC0_CLKOUT	WKUP_HFOSC0	
MCAN8	MCAN8_ICLK	MCU_SYSCLK0/4	WKUP_PLLCTRL0	Interface Clock

**Table 12-475. MCAN Clocks and Resets (continued)**

MCAN8_FCLK		MAIN_PLL0_HSDIV4_CLKOUT (default)	PLL0_HSDIV4	Functional Clock (for more information about clock multiplexing, see CTRLMMR_MCAN8_CLKSEL[1-0] CLK_SEL bit field in <i>Control Module (CTRL_MMR)</i> . Default: CTRLMMR_MCAN8_CLKSEL[1-0] CLK_SEL = 0h, MAIN_PLL0_HSDIV4_CLKOUT is selected)
		MCU_EXT_REFCLK0	I/O pin	
		HFOSC1_CLKOUT	HFOSC1	
		HFOSC0_CLKOUT	WKUP_HFOSC0	
MCAN9	MCAN9_ICLK	MCU_SYSCLK0/4	WKUP_PLLCTRL0	Interface Clock
	MCAN9_FCLK	MAIN_PLL0_HSDIV4_CLKOUT (default)	PLL0_HSDIV4	Functional Clock (for more information about clock multiplexing, see CTRLMMR_MCAN9_CLKSEL[1-0] CLK_SEL bit field in <i>Control Module (CTRL_MMR)</i> . Default: CTRLMMR_MCAN9_CLKSEL[1-0] CLK_SEL = 0h, MAIN_PLL0_HSDIV4_CLKOUT is selected)
		MCU_EXT_REFCLK0	I/O pin	
		HFOSC1_CLKOUT	HFOSC1	
		HFOSC0_CLKOUT	WKUP_HFOSC0	
MCAN10	MCAN10_ICLK	MCU_SYSCLK0/4	WKUP_PLLCTRL0	Interface Clock
	MCAN10_FCLK	MAIN_PLL0_HSDIV4_CLKOUT (default)	PLL0_HSDIV4	Functional Clock (for more information about clock multiplexing, see CTRLMMR_MCAN10_CLKSEL[1-0] CLK_SEL bit field in <i>Control Module (CTRL_MMR)</i> . Default: CTRLMMR_MCAN10_CLKSEL[1-0] CLK_SEL = 0h, MAIN_PLL0_HSDIV4_CLKOUT is selected)
		MCU_EXT_REFCLK0	I/O pin	
		HFOSC1_CLKOUT	HFOSC1	
		HFOSC0_CLKOUT	WKUP_HFOSC0	
MCAN11	MCAN11_ICLK	MCU_SYSCLK0/4	WKUP_PLLCTRL0	Interface Clock
	MCAN11_FCLK	MAIN_PLL0_HSDIV4_CLKOUT (default)	PLL0_HSDIV4	Functional Clock (for more information about clock multiplexing, see CTRLMMR_MCAN11_CLKSEL[1-0] CLK_SEL bit field in <i>Control Module (CTRL_MMR)</i> . Default: CTRLMMR_MCAN11_CLKSEL[1-0] CLK_SEL = 0h, MAIN_PLL0_HSDIV4_CLKOUT is selected)
		MCU_EXT_REFCLK0	I/O pin	
		HFOSC1_CLKOUT	HFOSC1	
		HFOSC0_CLKOUT	WKUP_HFOSC0	
MCAN12	MCAN12_ICLK	MCU_SYSCLK0/4	WKUP_PLLCTRL0	Interface Clock
	MCAN12_FCLK	MAIN_PLL0_HSDIV4_CLKOUT (default)	PLL0_HSDIV4	Functional Clock (for more information about clock multiplexing, see CTRLMMR_MCAN12_CLKSEL[1-0] CLK_SEL bit field in <i>Control Module (CTRL_MMR)</i> . Default: CTRLMMR_MCAN12_CLKSEL[1-0] CLK_SEL = 0h, MAIN_PLL0_HSDIV4_CLKOUT is selected)
		MCU_EXT_REFCLK0	I/O pin	
		HFOSC1_CLKOUT	HFOSC1	
		HFOSC0_CLKOUT	WKUP_HFOSC0	
MCAN13	MCAN13_ICLK	MCU_SYSCLK0/4	WKUP_PLLCTRL0	Interface Clock



**Table 12-475. MCAN Clocks and Resets (continued)**

MCAN13_FCLK	MAIN_PLL0_HSDIV4_CLKOUT (default)	PLL0_HSDIV4	Functional Clock (for more information about clock multiplexing, see CTRLMMR_MCAN13_CLKSEL[ 1-0] CLK_SEL bit field in <i>Control Module (CTRL_MMR)</i> . Default: CTRLMMR_MCAN13_CLKSEL[ 1-0] CLK_SEL = 0h, MAIN_PLL0_HSDIV4_CLKOUT is selected)
	MCU_EXT_REFCLK0	I/O pin	
	HFOSC1_CLKOUT	HFOSC1	
	HFOSC0_CLKOUT	WKUP_HFOSC0	

Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
MCAN0	MCAN0_RST	MOD_G_RST	LPSC34	Asynchronous Module Reset
MCAN1	MCAN1_RST	MOD_G_RST	LPSC35	Asynchronous Module Reset
MCAN2	MCAN2_RST	MOD_G_RST	LPSC36	Asynchronous Module Reset
MCAN3	MCAN3_RST	MOD_G_RST	LPSC37	Asynchronous Module Reset
MCAN4	MCAN4_RST	MOD_G_RST	LPSC38	Asynchronous Module Reset
MCAN5	MCAN5_RST	MOD_G_RST	LPSC39	Asynchronous Module Reset
MCAN6	MCAN6_RST	MOD_G_RST	LPSC40	Asynchronous Module Reset
MCAN7	MCAN7_RST	MOD_G_RST	LPSC41	Asynchronous Module Reset
MCAN8	MCAN8_RST	MOD_G_RST	LPSC42	Asynchronous Module Reset
MCAN9	MCAN9_RST	MOD_G_RST	LPSC43	Asynchronous Module Reset
MCAN10	MCAN10_RST	MOD_G_RST	LPSC44	Asynchronous Module Reset
MCAN11	MCAN11_RST	MOD_G_RST	LPSC45	Asynchronous Module Reset
MCAN12	MCAN12_RST	MOD_G_RST	LPSC46	Asynchronous Module Reset
MCAN13	MCAN13_RST	MOD_G_RST	LPSC47	Asynchronous Module Reset

**Table 12-476. MCAN Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCAN0	MCAN0_MCANSS_MCAN_LVL_INT_0	GIC500_SPI_IN_156	COMPUTE_CLUSTER_0	MCAN0 Line 0 Interrupt Request	Level
		R5FSS0_CORE0_INTR_IN_1 21	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_1 21	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_1 21	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_1 21	R5FSS1_CORE1		
		C66SS0_INTRTR0_IN_381	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_381	C66SS1_INTRTR0		
		PRU_ICSSG0_PR1_SLV_INT R_IN_72	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INT R_IN_72	PRU-ICSSG1		
		MAIN2MCU_LVL_INTRTR0_I N_16	MAIN2MCU_LVL_INT RTR0		

**Table 12-476. MCAN Hardware Requests (continued)**

MCAN0	MCANSS_MCAN_LVL_INT_1	GIC500_SPI_IN_157	COMPUTE_CLUSTER 0	MCAN0 Line 1 Interrupt Request	Level
		R5FSS0_CORE0_INTR_IN_1 22	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_1 22	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_1 22	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_1 22	R5FSS1_CORE1		
		C66SS0_INTRTR0_IN_382	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_382	C66SS1_INTRTR0		
		PRU_ICSSG0_PR1_SLV_INT R_IN_73	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INT R_IN_73	PRU-ICSSG1		
		MAIN2MCU_LVL_INTRTR0_I N_17	MAIN2MCU_LVL_INT RTR0		
MCAN0	MCANSS_EXT_TS_ROLLOVE R_LVL_INT_0	GIC500_SPI_IN_158	COMPUTE_CLUSTER 0	MCAN0 External TimeStamp Counter Rollover Interrupt	Level
		R5FSS0_CORE0_INTR_IN_1 20	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_1 20	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_1 20	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_1 20	R5FSS1_CORE1		
		C66SS0_INTRTR0_IN_383	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_383	C66SS1_INTRTR0		
		PRU_ICSSG0_PR1_SLV_INT R_IN_74	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INT R_IN_74	PRU-ICSSG1		
		MAIN2MCU_LVL_INTRTR0_I N_18	MAIN2MCU_LVL_INT RTR0		
MCAN0	MCANSS_ECC_CORR_LVL_IN T_0	ESM0_LVL_IN_312	ESM0	MCAN0 ECC Correctable Error Interrupt Request	Level
MCAN0	MCANSS_ECC_UNCORR_LVL _INT_0	ESM0_LVL_IN_313	ESM0	MCAN0 ECC Uncorrectable Error Interrupt Request	Level
MCAN1	MCAN1_MCANSS_MCAN_LVL_INT_0	GIC500_SPI_IN_159	COMPUTE_CLUSTER 0	MCAN1 Line 0 Interrupt Request	Level
		R5FSS0_CORE0_INTR_IN_1 24	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_1 24	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_1 24	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_1 24	R5FSS1_CORE1		
		C66SS0_INTRTR0_IN_384	C66SS0_INTRTR0		

**Table 12-476. MCAN Hardware Requests (continued)**

MCAN1_MCANSS_MCAN_LVL_INT_1	C66SS1_INTRTR0_IN_384	C66SS1_INTRTR0	MCAN1 Line 1 Interrupt Request	Level
	PRU_ICSSG0_PR1_SLV_INT R_IN_75	PRU-ICSSG0		
	PRU_ICSSG1_PR1_SLV_INT R_IN_75	PRU-ICSSG1		
	MAIN2MCU_LVL_INTRTR0_I N_19	MAIN2MCU_LVL_INT RTR0		
	GIC500_SPI_IN_160	COMPUTE_CLUSTER 0		
	R5FSS0_CORE0_INTR_IN_1 25	R5FSS0_CORE0		
	R5FSS0_CORE1_INTR_IN_1 25	R5FSS0_CORE1		
	R5FSS1_CORE0_INTR_IN_1 25	R5FSS1_CORE0		
	R5FSS1_CORE1_INTR_IN_1 25	R5FSS1_CORE1		
	C66SS0_INTRTR0_IN_385	C66SS0_INTRTR0		
MCAN1_MCANSS_EXT_TS_ROLLOVE R_LVL_INT_0	C66SS1_INTRTR0_IN_385	C66SS1_INTRTR0	MCAN1 External TimeStamp Counter Rollover Interrupt	Level
	PRU_ICSSG0_PR1_SLV_INT R_IN_76	PRU-ICSSG0		
	PRU_ICSSG1_PR1_SLV_INT R_IN_76	PRU-ICSSG1		
	MAIN2MCU_LVL_INTRTR0_I N_20	MAIN2MCU_LVL_INT RTR0		
	GIC500_SPI_IN_161	COMPUTE_CLUSTER 0		
	R5FSS0_CORE0_INTR_IN_1 23	R5FSS0_CORE0		
	R5FSS0_CORE1_INTR_IN_1 23	R5FSS0_CORE1		
	R5FSS1_CORE1_INTR_IN_1 23	R5FSS1_CORE0		
	R5FSS1_CORE1_INTR_IN_1 23	R5FSS1_CORE1		
	C66SS0_INTRTR0_IN_386	C66SS0_INTRTR0		
MCAN1_MCANSS_ECC_CORR_LVL_IN T_0	C66SS1_INTRTR0_IN_386	C66SS1_INTRTR0	MCAN1 ECC Correctable Error Interrupt Request	Level
	PRU_ICSSG0_PR1_SLV_INT R_IN_77	PRU-ICSSG0		
	PRU_ICSSG1_PR1_SLV_INT R_IN_77	PRU-ICSSG1		
	MAIN2MCU_LVL_INTRTR0_I N_21	MAIN2MCU_LVL_INT RTR0		
	ESM0_LVL_IN_314	ESM0		
MCAN1_MCANSS_ECC_UNCORR_LVL _INT_0	ESM0_LVL_IN_315	ESM0	MCAN1 ECC Uncorrectable Error Interrupt Request	Level

**Table 12-476. MCAN Hardware Requests (continued)**

MCAN2	MCAN2_MCANSS_MCAN_LVL_INT_0	GIC500_SPI_IN_162	COMPUTE_CLUSTER 0	MCAN2 Line 0 Interrupt Request	Level
		R5FSS0_CORE0_INTR_IN_1 27	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_1 27	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_1 27	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_1 27	R5FSS1_CORE1		
		C66SS0_INTRTR0_IN_20	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_20	C66SS1_INTRTR0		
		PRU_ICSSG0_PR1_SLV_INT R_IN_78	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INT R_IN_78	PRU-ICSSG1		
		MAIN2MCU_LVL_INTRTR0_I N_22	MAIN2MCU_LVL_INT RTR0		
	MCAN2_MCANSS_MCAN_LVL_INT_1	GIC500_SPI_IN_163	COMPUTE_CLUSTER 0	MCAN2 Line 1 Interrupt Request	Level
		R5FSS0_CORE0_INTR_IN_1 28	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_1 28	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_1 28	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_1 28	R5FSS1_CORE1		
		C66SS0_INTRTR0_IN_21	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_21	C66SS1_INTRTR0		
		PRU_ICSSG0_PR1_SLV_INT R_IN_79	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INT R_IN_79	PRU-ICSSG1		
		MAIN2MCU_LVL_INTRTR0_I N_23	MAIN2MCU_LVL_INT RTR0		
	MCAN2_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	GIC500_SPI_IN_164	COMPUTE_CLUSTER 0	MCAN2 External TimeStamp Counter Rollover Interrupt	Level
		R5FSS0_CORE0_INTR_IN_1 26	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_1 26	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_1 26	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_1 26	R5FSS1_CORE1		
		C66SS0_INTRTR0_IN_22	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_22	C66SS1_INTRTR0		
		PRU_ICSSG0_PR1_SLV_INT R_IN_80	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INT R_IN_80	PRU-ICSSG1		
		MAIN2MCU_LVL_INTRTR0_I N_24	MAIN2MCU_LVL_INT RTR0		

**Table 12-476. MCAN Hardware Requests (continued)**

MCAN2	MCAN2_MCANSS_ECC_CORR_LVL_IN_T_0	ESM0_LVL_IN_316	ESM0	MCAN2 ECC Correctable Error Interrupt Request	Level
	MCAN2_MCANSS_ECC_UNCORR_LVL_INT_0	ESM0_LVL_IN_317	ESM0	MCAN2 ECC Uncorrectable Error Interrupt Request	Level
MCAN3	MCAN3_MCANSS_MCAN_LVL_INT_0	GIC500_SPI_IN_165	COMPUTE_CLUSTER0	MCAN3 Line 0 Interrupt Request	Level
		R5FSS0_CORE0_INTR_IN_1_30	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_1_30	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_1_30	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_1_30	R5FSS1_CORE1		
		C66SS0_INTRTR0_IN_23	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_23	C66SS1_INTRTR0		
		PRU_ICSSG0_PR1_SLV_INT_R_IN_81	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INT_R_IN_81	PRU-ICSSG1		
		MAIN2MCU_LVL_INTRTR0_IN_25	MAIN2MCU_LVL_INTRTR0		
MCAN3	MCAN3_MCANSS_MCAN_LVL_INT_1	GIC500_SPI_IN_166	COMPUTE_CLUSTER0	MCAN3 Line 1 Interrupt Request	Level
		R5FSS0_CORE0_INTR_IN_1_31	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_1_31	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_1_31	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_1_31	R5FSS1_CORE1		
		C66SS0_INTRTR0_IN_24	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_24	C66SS1_INTRTR0		
		PRU_ICSSG0_PR1_SLV_INT_R_IN_82	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INT_R_IN_82	PRU-ICSSG1		
		MAIN2MCU_LVL_INTRTR0_IN_26	MAIN2MCU_LVL_INTRTR0		
MCAN3	MCAN3_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	GIC500_SPI_IN_167	COMPUTE_CLUSTER0	MCAN3 External TimeStamp Counter Rollover Interrupt	Level
		R5FSS0_CORE0_INTR_IN_1_29	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_1_29	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_1_29	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_1_29	R5FSS1_CORE1		
		C66SS0_INTRTR0_IN_25	C66SS0_INTRTR0		

**Table 12-476. MCAN Hardware Requests (continued)**

		C66SS1_INTRTR0_IN_25	C66SS1_INTRTR0		
		PRU_ICSSG0_PR1_SLV_INT R_IN_83	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INT R_IN_83	PRU-ICSSG1		
		MAIN2MCU_LVL_INTRTR0_I N_27	MAIN2MCU_LVL_INT RTR0		
MCAN3	MCANSS_ECC_CORR_LVL_IN T_0	ESM0_LVL_IN_318	ESM0	MCAN3 ECC Correctable Error Interrupt Request	Level
MCAN3	MCANSS_ECC_UNCORR_LVL _INT_0	ESM0_LVL_IN_319	ESM0	MCAN3 ECC Uncorrectable Error Interrupt Request	Level
MCAN4	MCAN4_MCANSS_MCAN_LVL_INT_0	GIC500_SPI_IN_168	COMPUTE_CLUSTER 0	MCAN4 Line 0 Interrupt Request	Level
		R5FSS0_CORE0_INTR_IN_1 92	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_1 92	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_1 92	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_1 92	R5FSS1_CORE1		
		C66SS0_INTRTR0_IN_57	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_57	C66SS1_INTRTR0		
		PRU_ICSSG0_PR1_SLV_INT R_IN_84	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INT R_IN_84	PRU-ICSSG1		
		MAIN2MCU_LVL_INTRTR0_I N_278	MAIN2MCU_LVL_INT RTR0		
MCAN4	MCANSS_MCAN_LVL_INT_1	GIC500_SPI_IN_169	COMPUTE_CLUSTER 0	MCAN4 Line 1 Interrupt Request	Level
		R5FSS0_CORE0_INTR_IN_1 93	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_1 93	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_1 93	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_1 93	R5FSS1_CORE1		
		C66SS0_INTRTR0_IN_58	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_58	C66SS1_INTRTR0		
		PRU_ICSSG0_PR1_SLV_INT R_IN_85	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INT R_IN_85	PRU-ICSSG1		
		MAIN2MCU_LVL_INTRTR0_I N_279	MAIN2MCU_LVL_INT RTR0		

**Table 12-476. MCAN Hardware Requests (continued)**

MCAN4	MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	GIC500_SPI_IN_170	COMPUTE_CLUSTER0	MCAN4 External TimeStamp Counter Rollover Interrupt	Level
		R5FSS0_CORE0_INTR_IN_194	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_194	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_194	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_194	R5FSS1_CORE1		
		C66SS0_INTRTR0_IN_59	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_59	C66SS1_INTRTR0		
		PRU_ICSSG0_PR1_SLV_INT_R_IN_86	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INT_R_IN_86	PRU-ICSSG1		
MCAN4	MCANSS_ECC_CORR_LVL_INT_0	MAIN2MCU_LVL_INTRTR0_IN_280	MAIN2MCU_LVL_INTRTR0	MCAN4 ECC Correctable Error Interrupt Request	Level
		ESM0_LVL_IN_320	ESM0		
MCAN4	MCANSS_ECC_UNCORR_LVL_INT_0	ESM0_LVL_IN_321	ESM0	MCAN4 ECC Uncorrectable Error Interrupt Request	Level
MCAN5	MCANSS_MCAN_LVL_INT_0	GIC500_SPI_IN_171	COMPUTE_CLUSTER0	MCAN5 Line 0 Interrupt Request	Level
		R5FSS0_CORE0_INTR_IN_195	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_195	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_195	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_195	R5FSS1_CORE1		
		C66SS0_INTRTR0_IN_60	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_60	C66SS1_INTRTR0		
		PRU_ICSSG0_PR1_SLV_INT_R_IN_87	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INT_R_IN_87	PRU-ICSSG1		
	MCANSS_MCAN_LVL_INT_1	MAIN2MCU_LVL_INTRTR0_IN_281	MAIN2MCU_LVL_INTRTR0	MCAN5 Line 1 Interrupt Request	Level
		GIC500_SPI_IN_172	COMPUTE_CLUSTER0		
		R5FSS0_CORE0_INTR_IN_196	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_196	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_196	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_196	R5FSS1_CORE1		
		C66SS0_INTRTR0_IN_61	C66SS0_INTRTR0		

**Table 12-476. MCAN Hardware Requests (continued)**

MCAN5	MCAN5_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	C66SS1_INTRTR0_IN_61	C66SS1_INTRTR0	MCAN5 External TimeStamp Counter Rollover Interrupt	Level
		PRU_ICSSG0_PR1_SLV_INT_R_IN_88	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INT_R_IN_88	PRU-ICSSG1		
		MAIN2MCU_LVL_INTRTR0_IN_282	MAIN2MCU_LVL_INTRTR0		
		GIC500_SPI_IN_173	COMPUTE_CLUSTER0		
		R5FSS0_CORE0_INTR_IN_1_97	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_1_97	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_1_97	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_1_97	R5FSS1_CORE1		
		C66SS0_INTRTR0_IN_62	C66SS0_INTRTR0		
MCAN5	MCAN5_MCANSS_ECC_CORR_LVL_INT_0	C66SS1_INTRTR0_IN_62	C66SS1_INTRTR0	MCAN5 ECC Correctable Error Interrupt Request	Level
		PRU_ICSSG0_PR1_SLV_INT_R_IN_89	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INT_R_IN_89	PRU-ICSSG1		
		MAIN2MCU_LVL_INTRTR0_IN_283	MAIN2MCU_LVL_INTRTR0		
		ESM0_LVL_IN_322	ESM0		
		ESM0_LVL_IN_323	ESM0		
		ESM0_LVL_IN_322	ESM0		
		ESM0_LVL_IN_323	ESM0		
		ESM0_LVL_IN_322	ESM0		
		ESM0_LVL_IN_323	ESM0		
MCAN6	MCAN6_MCANSS_MCAN_LVL_INT_0	GIC500_SPI_IN_174	COMPUTE_CLUSTER0	MCAN6 Line 0 Interrupt Request	Level
		R5FSS0_CORE0_INTR_IN_1_98	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_1_98	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_1_98	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_1_98	R5FSS1_CORE1		
		C66SS0_INTRTR0_IN_63	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_63	C66SS1_INTRTR0		
		PRU_ICSSG0_PR1_SLV_INT_R_IN_90	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INT_R_IN_90	PRU-ICSSG1		
		MAIN2MCU_LVL_INTRTR0_IN_284	MAIN2MCU_LVL_INTRTR0		



**Table 12-476. MCAN Hardware Requests (continued)**

MCAN6_MCANSS_MCAN_LVL_INT_1	GIC500_SPI_IN_175	COMPUTE_CLUSTER	MCAN6 Line 1 Interrupt Request	Level
		0		
		R5FSS0_CORE0_INTR_IN_1		
		99		
		R5FSS0_CORE1_INTR_IN_1		
		99		
		R5FSS1_CORE0_INTR_IN_1		
		99		
		R5FSS1_CORE1_INTR_IN_1		
		99		
MCAN6_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	GIC500_SPI_IN_176	C66SS0_INTRTR0_IN_64	MCAN6 External TimeStamp Counter Rollover Interrupt	Level
		C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_64		
		C66SS1_INTRTR0		
		PRU_ICSSG0_PR1_SLV_INT		
		R_IN_91		
		PRU_ICSSG1_PR1_SLV_INT		
		R_IN_91		
		MAIN2MCU_LVL_INTRTR0_I		
		N_285		
MCAN6_MCANSS_ECC_CORR_LVL_INT_0	ESM0_LVL_IN_324	COMPUTE_CLUSTER	MCAN6 ECC Correctable Error Interrupt Request	Level
		0		
		R5FSS0_CORE0_INTR_IN_2		
		00		
		R5FSS0_CORE1_INTR_IN_2		
		00		
		R5FSS1_CORE0_INTR_IN_2		
		00		
		R5FSS1_CORE1_INTR_IN_2		
		00		
MCAN6_MCANSS_ECC_UNCORR_LVL_INT_0	ESM0_LVL_IN_325	C66SS0_INTRTR0_IN_65	MCAN6 ECC Uncorrectable Error Interrupt Request	Level
		C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_65		
		C66SS1_INTRTR0		
		PRU_ICSSG0_PR1_SLV_INT		
		R_IN_92		
		PRU_ICSSG1_PR1_SLV_INT		
		R_IN_92		
		MAIN2MCU_LVL_INTRTR0_I		
		N_286		
MCAN7	MCAN7_MCANSS_MCAN_LVL_INT_0	GIC500_SPI_IN_177	MCAN7 Line 0 Interrupt Request	Level
		COMPUTE_CLUSTER		
		0		
		R5FSS0_CORE0_INTR_IN_2		
		01		
		R5FSS0_CORE1_INTR_IN_2		
MCAN7	MCAN7_MCANSS_MCAN_LVL_INT_0	R5FSS0_CORE1_INTR_IN_2		
		01		
		R5FSS1_CORE0_INTR_IN_2		
		01		
		R5FSS1_CORE1_INTR_IN_2		
		01		
MCAN7	MCAN7_MCANSS_MCAN_LVL_INT_0	C66SS0_INTRTR0_IN_66	MCAN7 Line 0 Interrupt Request	Level
		C66SS0_INTRTR0		

**Table 12-476. MCAN Hardware Requests (continued)**

MCAN7_MCANSS_MCAN_LVL_INT_1	C66SS1_INTRTR0_IN_66	C66SS1_INTRTR0	MCAN7 Line 1 Interrupt Request	Level
	PRU_ICSSG0_PR1_SLV_INT R_IN_72	PRU-ICSSG0		
	PRU_ICSSG1_PR1_SLV_INT R_IN_72	PRU-ICSSG1		
	MAIN2MCU_LVL_INTRTR0_I N_287	MAIN2MCU_LVL_INT RTR0		
	GIC500_SPI_IN_178	COMPUTE_CLUSTER 0		
	R5FSS0_CORE0_INTR_IN_2 02	R5FSS0_CORE0		
	R5FSS0_CORE1_INTR_IN_2 02	R5FSS0_CORE1		
	R5FSS1_CORE0_INTR_IN_2 02	R5FSS1_CORE0		
	R5FSS1_CORE1_INTR_IN_2 02	R5FSS1_CORE1		
	C66SS0_INTRTR0_IN_67	C66SS0_INTRTR0		
MCAN7_MCANSS_EXT_TS_ROLLOVE R_LVL_INT_0	C66SS1_INTRTR0_IN_67	C66SS1_INTRTR0	MCAN7 External TimeStamp Counter Rollover Interrupt	Level
	PRU_ICSSG0_PR1_SLV_INT R_IN_73	PRU-ICSSG0		
	PRU_ICSSG1_PR1_SLV_INT R_IN_73	PRU-ICSSG1		
	MAIN2MCU_LVL_INTRTR0_I N_288	MAIN2MCU_LVL_INT RTR0		
	GIC500_SPI_IN_179	COMPUTE_CLUSTER 0		
	R5FSS0_CORE0_INTR_IN_2 03	R5FSS0_CORE0		
	R5FSS0_CORE1_INTR_IN_2 03	R5FSS0_CORE1		
	R5FSS1_CORE0_INTR_IN_2 03	R5FSS1_CORE0		
	R5FSS1_CORE1_INTR_IN_2 03	R5FSS1_CORE1		
	C66SS0_INTRTR0_IN_68	C66SS0_INTRTR0		
MCAN7_MCANSS_ECC_CORR_LVL_IN T_0	C66SS1_INTRTR0_IN_68	C66SS1_INTRTR0	MCAN7 ECC Correctable Error Interrupt Request	Level
	PRU_ICSSG0_PR1_SLV_INT R_IN_74	PRU-ICSSG0		
	PRU_ICSSG1_PR1_SLV_INT R_IN_74	PRU-ICSSG1		
	MAIN2MCU_LVL_INTRTR0_I N_289	MAIN2MCU_LVL_INT RTR0		
	ESM0_LVL_IN_326	ESM0		
MCAN7_MCANSS_ECC_UNCORR_LVL _INT_0	ESM0_LVL_IN_327	ESM0	MCAN7 ECC Uncorrectable Error Interrupt Request	Level

**Table 12-476. MCAN Hardware Requests (continued)**

MCAN8	MCAN8_MCANSS_MCAN_LVL_INT_0	GIC500_SPI_IN_608	COMPUTE_CLUSTER 0	MCAN8 Line 0 Interrupt Request	Level
		R5FSS0_CORE0_INTR_IN_2 04	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_2 04	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_2 04	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_2 04	R5FSS1_CORE1		
		C66SS0_INTRTR0_IN_69	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_69	C66SS1_INTRTR0		
		PRU_ICSSG0_PR1_SLV_INT R_IN_75	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INT R_IN_75	PRU-ICSSG1		
		MAIN2MCU_LVL_INTRTR0_I N_290	MAIN2MCU_LVL_INT RTR0		
	MCAN8_MCANSS_MCAN_LVL_INT_1	GIC500_SPI_IN_609	COMPUTE_CLUSTER 0	MCAN8 Line 1 Interrupt Request	Level
		R5FSS0_CORE0_INTR_IN_2 05	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_2 05	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_2 05	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_2 05	R5FSS1_CORE1		
		C66SS0_INTRTR0_IN_70	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_70	C66SS1_INTRTR0		
		PRU_ICSSG0_PR1_SLV_INT R_IN_76	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INT R_IN_76	PRU-ICSSG1		
		MAIN2MCU_LVL_INTRTR0_I N_291	MAIN2MCU_LVL_INT RTR0		
	MCAN8_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	GIC500_SPI_IN_610	COMPUTE_CLUSTER 0	MCAN8 External TimeStamp Counter Rollover Interrupt	Level
		R5FSS0_CORE0_INTR_IN_2 06	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_2 06	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_2 06	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_2 06	R5FSS1_CORE1		
		C66SS0_INTRTR0_IN_71	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_71	C66SS1_INTRTR0		
		PRU_ICSSG0_PR1_SLV_INT R_IN_77	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INT R_IN_77	PRU-ICSSG1		
		MAIN2MCU_LVL_INTRTR0_I N_292	MAIN2MCU_LVL_INT RTR0		

**Table 12-476. MCAN Hardware Requests (continued)**

MCAN8_MCANSS_ECC_CORR_LVL_IN_T_0		ESM0_LVL_IN_328	ESM0	MCAN8 ECC Correctable Error Interrupt Request	Level
MCAN8_MCANSS_ECC_UNCORR_LVL_INT_0		ESM0_LVL_IN_329	ESM0	MCAN8 ECC Uncorrectable Error Interrupt Request	Level
MCAN9	MCAN9_MCANSS_MCAN_LVL_INT_0	GIC500_SPI_IN_611	COMPUTE_CLUSTER0	MCAN9 Line 0 Interrupt Request	Level
		R5FSS0_CORE0_INTR_IN_207	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_207	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_207	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_207	R5FSS1_CORE1		
		C66SS0_INTRTR0_IN_72	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_72	C66SS1_INTRTR0		
		PRU_ICSSG0_PR1_SLV_INT_R_IN_78	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INT_R_IN_78	PRU-ICSSG1		
	MCAN9_MCANSS_MCAN_LVL_INT_1	MAIN2MCU_LVL_INTRTR0_IN_293	MAIN2MCU_LVL_INTRTR0	MCAN9 Line 1 Interrupt Request	Level
		GIC500_SPI_IN_612	COMPUTE_CLUSTER0		
		R5FSS0_CORE0_INTR_IN_208	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_208	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_208	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_208	R5FSS1_CORE1		
		C66SS0_INTRTR0_IN_73	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_73	C66SS1_INTRTR0		
	MCAN9_MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	PRU_ICSSG0_PR1_SLV_INT_R_IN_79	PRU-ICSSG0	MCAN9 External TimeStamp Counter Rollover Interrupt	Level
		PRU_ICSSG1_PR1_SLV_INT_R_IN_79	PRU-ICSSG1		
		MAIN2MCU_LVL_INTRTR0_IN_294	MAIN2MCU_LVL_INTRTR0		
		GIC500_SPI_IN_613	COMPUTE_CLUSTER0		
		R5FSS0_CORE0_INTR_IN_209	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_209	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_209	R5FSS1_CORE0		

**Table 12-476. MCAN Hardware Requests (continued)**

		R5FSS1_CORE1_INTR_IN_2 09	R5FSS1_CORE1		
		C66SS0_INTRTR0_IN_74	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_74	C66SS1_INTRTR0		
		PRU_ICSSG0_PR1_SLV_INT R_IN_80	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INT R_IN_80	PRU-ICSSG1		
		MAIN2MCU_LVL_INTRTR0_I N_295	MAIN2MCU_LVL_INT RTR0		
	MCAN9_MCANSS_ECC_CORR_LVL_IN T_0	ESM0_LVL_IN_330	ESM0	MCAN9 ECC Correctable Error Interrupt Request	Level
	MCAN9_MCANSS_ECC_UNCORR_LVL _INT_0	ESM0_LVL_IN_331	ESM0	MCAN9 ECC Uncorrectable Error Interrupt Request	Level
MCAN10	MCAN10_MCANSS_MCAN_LVL_INT_0	GIC500_SPI_IN_614	COMPUTE_CLUSTER 0	MCAN10 Line 0 Interrupt Request	Level
		R5FSS0_CORE0_INTR_IN_2 10	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_2 10	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_2 10	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_2 10	R5FSS1_CORE1		
		C66SS0_INTRTR0_IN_75	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_75	C66SS1_INTRTR0		
		PRU_ICSSG0_PR1_SLV_INT R_IN_81	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INT R_IN_81	PRU-ICSSG1		
		MAIN2MCU_LVL_INTRTR0_I N_296	MAIN2MCU_LVL_INT RTR0		
	MCAN10_MCANSS_MCAN_LVL_INT_1	GIC500_SPI_IN_615	COMPUTE_CLUSTER 0	MCAN10 Line 1 Interrupt Request	Level
		R5FSS0_CORE0_INTR_IN_2 11	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_2 11	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_2 11	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_2 11	R5FSS1_CORE1		
		C66SS0_INTRTR0_IN_76	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_76	C66SS1_INTRTR0		
		PRU_ICSSG0_PR1_SLV_INT R_IN_82	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INT R_IN_82	PRU-ICSSG1		
		MAIN2MCU_LVL_INTRTR0_I N_297	MAIN2MCU_LVL_INT RTR0		

**Table 12-476. MCAN Hardware Requests (continued)**

MCAN10	MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	GIC500_SPI_IN_616	COMPUTE_CLUSTER0	MCAN10 External TimeStamp Counter Rollover Interrupt	Level
		R5FSS0_CORE0_INTR_IN_212	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_212	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_212	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_212	R5FSS1_CORE1		
		C66SS0_INTRTR0_IN_77	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_77	C66SS1_INTRTR0		
		PRU_ICSSG0_PR1_SLV_INT_R_IN_83	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INT_R_IN_83	PRU-ICSSG1		
MCAN10	MCANSS_ECC_CORR_LVL_INT_0	MAIN2MCU_LVL_INTRTR0_IN_298	MAIN2MCU_LVL_INTRTR0	MCAN10 ECC Correctable Error Interrupt Request	Level
		ESM0_LVL_IN_332	ESM0		
MCAN10	MCANSS_ECC_UNCORR_LVL_INT_0	ESM0_LVL_IN_333	ESM0	MCAN10 ECC Uncorrectable Error Interrupt Request	Level
MCAN11	MCAN11_MCANSS_MCAN_LVL_INT_0	GIC500_SPI_IN_617	COMPUTE_CLUSTER0	MCAN11 Line 0 Interrupt Request	Level
		R5FSS0_CORE0_INTR_IN_213	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_213	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_213	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_213	R5FSS1_CORE1		
		C66SS0_INTRTR0_IN_78	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_78	C66SS1_INTRTR0		
		PRU_ICSSG0_PR1_SLV_INT_R_IN_84	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INT_R_IN_84	PRU-ICSSG1		
MCAN11	MCAN11_MCANSS_MCAN_LVL_INT_1	MAIN2MCU_LVL_INTRTR0_IN_299	MAIN2MCU_LVL_INTRTR0	MCAN11 Line 1 Interrupt Request	Level
		GIC500_SPI_IN_618	COMPUTE_CLUSTER0		
		R5FSS0_CORE0_INTR_IN_214	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_214	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_214	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_214	R5FSS1_CORE1		
		C66SS0_INTRTR0_IN_79	C66SS0_INTRTR0		

**Table 12-476. MCAN Hardware Requests (continued)**

MCAN11	MCANSS_EXT_TS_ROLLOVER_LVL_INT_0	C66SS1_INTRTR0_IN_79	C66SS1_INTRTR0	MCAN11 External TimeStamp Counter Rollover Interrupt	Level
		PRU_ICSSG0_PR1_SLV_INT_R_IN_85	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INT_R_IN_85	PRU-ICSSG1		
		MAIN2MCU_LVL_INTRTR0_I_N_300	MAIN2MCU_LVL_INT_RTR0		
		GIC500_SPI_IN_619	COMPUTE_CLUSTER0		
		R5FSS0_CORE0_INTR_IN_2_15	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_2_15	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_2_15	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_2_15	R5FSS1_CORE1		
		C66SS0_INTRTR0_IN_80	C66SS0_INTRTR0		
MCAN11	MCANSS_ECC_CORR_LVL_INT_0	C66SS1_INTRTR0_IN_80	C66SS1_INTRTR0	MCAN11 ECC Correctable Error Interrupt Request	Level
		PRU_ICSSG0_PR1_SLV_INT_R_IN_86	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INT_R_IN_86	PRU-ICSSG1		
		MAIN2MCU_LVL_INTRTR0_I_N_301	MAIN2MCU_LVL_INT_RTR0		
		ESM0_LVL_IN_334	ESM0		
		ESM0_LVL_IN_335	ESM0		
		GIC500_SPI_IN_620	COMPUTE_CLUSTER0		
		R5FSS0_CORE0_INTR_IN_2_16	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_2_16	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_2_16	R5FSS1_CORE0		
MCAN12	MCAN12_MCANSS_MCAN_LVL_INT_0	R5FSS1_CORE1_INTR_IN_2_16	R5FSS1_CORE1	MCAN12 Line 0 Interrupt Request	Level
		C66SS0_INTRTR0_IN_81	C66SS0_INTRTR0		
		C66SS1_INTRTR0_IN_81	C66SS1_INTRTR0		
		PRU_ICSSG0_PR1_SLV_INT_R_IN_87	PRU-ICSSG0		
		PRU_ICSSG1_PR1_SLV_INT_R_IN_87	PRU-ICSSG1		
		MAIN2MCU_LVL_INTRTR0_I_N_302	MAIN2MCU_LVL_INT_RTR0		

**Table 12-476. MCAN Hardware Requests (continued)**

MCAN12_MCANSS_MCAN_LVL_INT_1	GIC500_SPI_IN_621	COMPUTE_CLUSTER 0	MCAN12 Line 1 Interrupt Request	Level	
	R5FSS0_CORE0_INTR_IN_2 17	R5FSS0_CORE0			
	R5FSS0_CORE1_INTR_IN_2 17	R5FSS0_CORE1			
	R5FSS1_CORE0_INTR_IN_2 17	R5FSS1_CORE0			
	R5FSS1_CORE1_INTR_IN_2 17	R5FSS1_CORE1			
	C66SS0_INTRTR0_IN_82	C66SS0_INTRTR0			
	C66SS1_INTRTR0_IN_82	C66SS1_INTRTR0			
	PRU_ICSSG0_PR1_SLV_INT R_IN_88	PRU-ICSSG0			
	PRU_ICSSG1_PR1_SLV_INT R_IN_88	PRU-ICSSG1			
	MAIN2MCU_LVL_INTRTR0_I N_303	MAIN2MCU_LVL_INT RTR0			
MCAN12_MCANSS_EXT_TS_ROLLOV ER_LVL_INT_0	GIC500_SPI_IN_622	COMPUTE_CLUSTER 0	MCAN12 External TimeStamp Counter Rollover Interrupt	Level	
	R5FSS0_CORE0_INTR_IN_2 18	R5FSS0_CORE0			
	R5FSS0_CORE1_INTR_IN_2 18	R5FSS0_CORE1			
	R5FSS1_CORE0_INTR_IN_2 18	R5FSS1_CORE0			
	R5FSS1_CORE1_INTR_IN_2 18	R5FSS1_CORE1			
	C66SS0_INTRTR0_IN_83	C66SS0_INTRTR0			
	C66SS1_INTRTR0_IN_83	C66SS1_INTRTR0			
	PRU_ICSSG0_PR1_SLV_INT R_IN_89	PRU-ICSSG0			
	PRU_ICSSG1_PR1_SLV_INT R_IN_89	PRU-ICSSG1			
	MAIN2MCU_LVL_INTRTR0_I N_304	MAIN2MCU_LVL_INT RTR0			
MCAN12_MCANSS_ECC_CORR_LVL_I NT_0	ESM0_LVL_IN_336	ESM0	MCAN12 ECC Correctable Error Interrupt Request	Level	
MCAN12_MCANSS_ECC_UNCORR_LV L_INT_0	ESM0_LVL_IN_337	ESM0	MCAN12 ECC Uncorrectable Error Interrupt Request	Level	
MCAN13	MCAN13_MCANSS_MCAN_LVL_INT_0	GIC500_SPI_IN_623	COMPUTE_CLUSTER 0	MCAN13 Line 0 Interrupt Request	Level
		R5FSS0_CORE0_INTR_IN_2 19	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_2 19	R5FSS0_CORE1		
		R5FSS1_CORE0_INTR_IN_2 19	R5FSS1_CORE0		
		R5FSS1_CORE1_INTR_IN_2 19	R5FSS1_CORE1		
		C66SS0_INTRTR0_IN_84	C66SS0_INTRTR0		



**Table 12-476. MCAN Hardware Requests (continued)**

MCAN13_MCANSS_MCAN_LVL_INT_1	C66SS1_INTRTR0_IN_84	C66SS1_INTRTR0	MCAN13 Line 1 Interrupt Request	Level
	PRU_ICSSG0_PR1_SLV_INT R_IN_90	PRU-ICSSG0		
	PRU_ICSSG1_PR1_SLV_INT R_IN_90	PRU-ICSSG1		
	MAIN2MCU_LVL_INTRTR0_I N_305	MAIN2MCU_LVL_INT RTR0		
	GIC500_SPI_IN_624	COMPUTE_CLUSTER 0		
	R5FSS0_CORE0_INTR_IN_2 20	R5FSS0_CORE0		
	R5FSS0_CORE1_INTR_IN_2 20	R5FSS0_CORE1		
	R5FSS1_CORE0_INTR_IN_2 20	R5FSS1_CORE0		
	R5FSS1_CORE1_INTR_IN_2 20	R5FSS1_CORE1		
	C66SS0_INTRTR0_IN_85	C66SS0_INTRTR0		
MCAN13_MCANSS_EXT_TS_ROLLOV ER_LVL_INT_0	C66SS1_INTRTR0_IN_85	C66SS1_INTRTR0	MCAN13 External TimeStamp Counter Rollover Interrupt	Level
	PRU_ICSSG0_PR1_SLV_INT R_IN_91	PRU-ICSSG0		
	PRU_ICSSG1_PR1_SLV_INT R_IN_91	PRU-ICSSG1		
	MAIN2MCU_LVL_INTRTR0_I N_306	MAIN2MCU_LVL_INT RTR0		
	GIC500_SPI_IN_625	COMPUTE_CLUSTER 0		
	R5FSS0_CORE0_INTR_IN_2 21	R5FSS0_CORE0		
	R5FSS0_CORE1_INTR_IN_2 21	R5FSS0_CORE1		
	R5FSS1_CORE0_INTR_IN_2 21	R5FSS1_CORE0		
	R5FSS1_CORE1_INTR_IN_2 21	R5FSS1_CORE1		
	C66SS0_INTRTR0_IN_86	C66SS0_INTRTR0		
MCAN13_MCANSS_ECC_CORR_LVL_I NT_0	C66SS1_INTRTR0_IN_86	C66SS1_INTRTR0	MCAN13 ECC Correctable Error Interrupt Request	Level
	PRU_ICSSG0_PR1_SLV_INT R_IN_92	PRU-ICSSG0		
	PRU_ICSSG1_PR1_SLV_INT R_IN_92	PRU-ICSSG1		
	MAIN2MCU_LVL_INTRTR0_I N_307	MAIN2MCU_LVL_INT RTR0		
	ESM0_LVL_IN_338	ESM0		
MCAN13_MCANSS_ECC_UNCORR_LV L_INT_0	ESM0_LVL_IN_339	ESM0	MCAN13 ECC Uncorrectable Error Interrupt Request	Level

#### 12.4.4.4 MCAN Functional Description

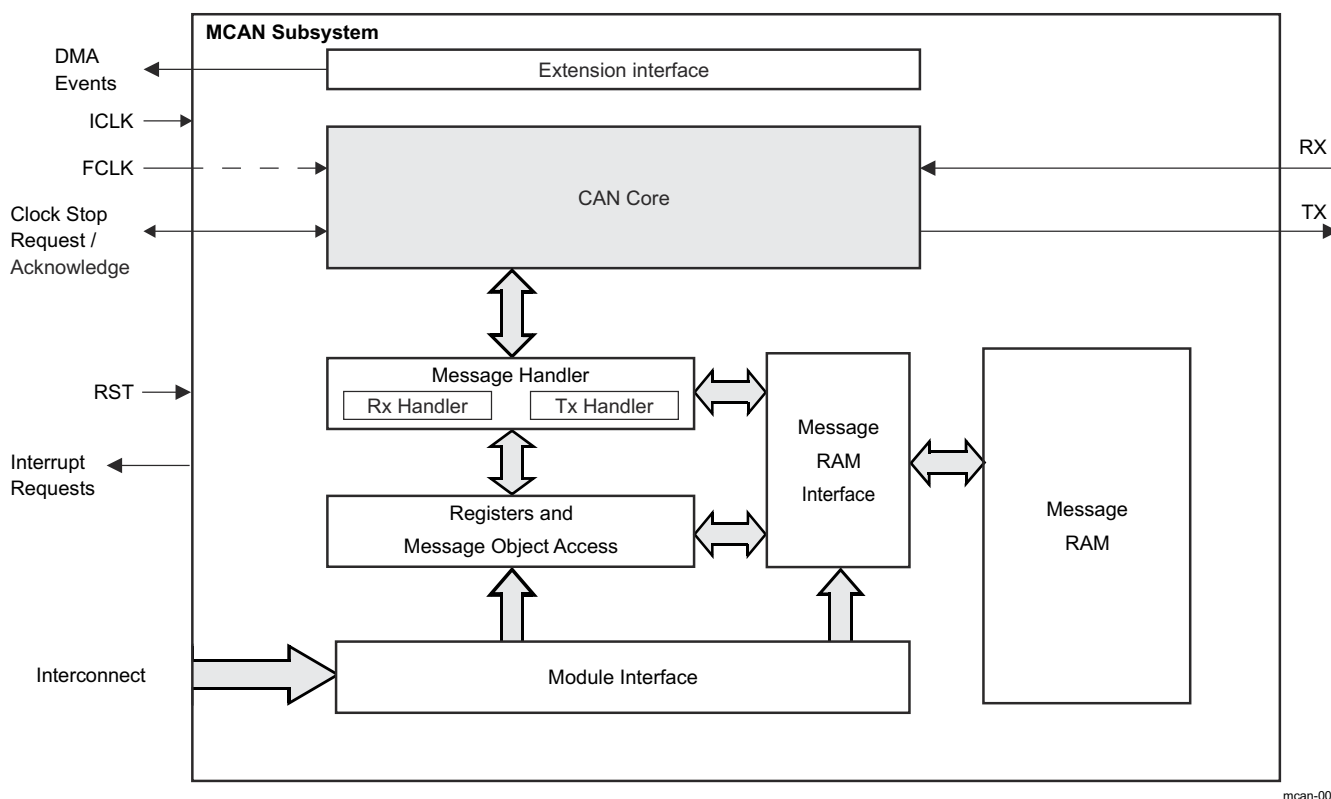
The MCAN module performs CAN protocol communication according to ISO 11898-1:2015. The bit rate can be programmed to values greater than 1 Mbps. Additional transceiver hardware is required for the connection to the physical layer (CAN bus).

For communication on a CAN network, individual message frames can be configured. The message frames and identifier masks are stored in the Message RAM.

All functions concerning the handling of messages are implemented in the Message Handler.

The register set of the MCAN module can be accessed directly via the module interface. These registers are used to control and configure the CAN core and the Message Handler, and to access the Message RAM.

Figure 12-466 shows the MCAN module block diagram.



**Figure 12-466. MCAN Block Diagram**

The MCAN module blocks description:

- **CAN Core:** the CAN core consists of the CAN protocol controller and the Rx/Tx shift register. It handles all ISO 11898-1:2015 protocol functions and supports 11-bit and 29-bit identifiers.
- **Message Handler:** the Message Handler (Rx Handler and Tx Handler) is a state machine that controls the data transfer between the single-ported Message RAM and the CAN core's Rx/Tx shift register. It also handles the acceptance filtering and the Interrupt/DMA request generation as programmed in the control registers.
- **Message RAM:** the main purpose of the Message RAM is to store Rx/Tx messages, Tx Event elements, and Message ID Filter elements (for more information, see *Message RAM*).
- **Message RAM Interface:** enables connection between the Message RAM and the other blocks in the MCAN module.
- **Registers and Message Object Access:** data consistency is ensured by indirect accesses to the message objects. The interface registers have the same word-length as the Message RAM.

- **Module Interface:** provides connection to the Registers and Message Object Access block and Message RAM Interface block
- **Clocking:** two clocks are provided to the MCAN module: the peripheral synchronous clock (interface clock ICLK) and the peripheral asynchronous clock (functional clock - FCLK). For more information, see *MCU\_MCAN Clocks and Resets* and *MCAN Clocks and Resets*.
- **Extension Interface:** this interface is used for DMA requests signaling (see *DMA Requests*).

#### 12.4.4.4.1 Module Clocking Requirements

Two clocks are provided to the MCAN module:

- the peripheral synchronous clock (ICLK) as the general module clock source
- and the peripheral asynchronous clock (FCLK) provided to the CAN core for generating the CAN bit timing.

Within the MCAN module there is a synchronization mechanism implemented to ensure safe data transfer between the two clock domains. There is synchronization between the signals from the Host clock domain to the CAN clock domain and vice versa and between the reset signal to the Host clock domain and to the CAN clock domain.

#### Note

ICLK must always be higher or equal to FCLK, in order to achieve a stable functionality of the MCAN module. Here, also the frequency shift of the modulated ICLK has to be considered:

$$f_{0,ICLK} \pm \Delta f_{FM,ICLK} \geq f_{FCLK}$$

For more information on how to configure the relevant clock source registers, see *Clocking* and the device-specific Datasheet.

#### 12.4.4.4.2 Interrupt and DMA Requests

The MCAN module provides interrupt and DMA requests. They are configured via the Host CPU. The Suspend Mode is requesting or forcing (based on MCANSS\_CTRL[3] DBGSUSP\_FREE bit) the MCAN module to go into initialization mode (see MCAN\_CCCR[0] INIT bit) in which new interrupts and DMA requests will not be issued, that is to prevent the interrupt and DMA requests from propagating to the Host CPU (for more information, see [Section 12.4.4.4.3.8.2, Suspend Mode](#)).

##### 12.4.4.4.2.1 Interrupt Requests

The MCAN module has two interrupt lines. There are 30 internal interrupt sources. Each source can be configured to drive one of the two interrupt lines. The interrupts are 'level high' interrupts.

The MCAN core provides two interrupt requests (for Line 0 and Line 1).

For more information, see the following registers:

- Interrupt Register (MCAN\_IR)
- Interrupt Enable (MCAN\_IE)
- Interrupt Line Select (MCAN\_ILS)
- Interrupt Line Enable (MCAN\_ILE)

The MCAN module is capable of issuing ECC interrupts. After clearing the ECC interrupt source, the application software must also write 1 to EOI register (MCANSS\_ECC\_SEC\_EOI\_REG/MCANSS\_ECC\_DED\_EOI\_REG). For more information, see *ECC Aggregator*.

The MCAN module supports External Timestamp Counter. When the External Timestamp Counter rolls over it produces an interrupt (see *External Timestamp Counter*).

For more information, see the following registers:

- Interrupt Clear Shadow Register (MCANSS\_ICS)
- Interrupt Raw Status Register (MCANSS\_IRS)
- Interrupt Enable Clear Shadow Register (MCANSS\_IECS)

- Interrupt Enable Register (MCANSS\_IE)
- Interrupt Enable Status Register (MCANSS\_IES)
- End Of Interrupt Register (MCANSS\_EOI)
- External Timestamp Prescaler Register (MCANSS\_EXT\_TS\_PRESCALER)
- External Timestamp Unserved Interrupts Counter Register (MCANSS\_EXT\_TS\_UNSERVICED\_INTR\_CNTR)

For more information about available Interrupt Requests, see *MCU\_MCAN Hardware Requests* and *MCAN Hardware Requests*.

#### 12.4.4.4.2.2 DMA Requests

Functional transmit and Filter DMA requests are generated by the MCAN module based on the signaling in the Extension Interface. The DMA signaling uses a simple DMA request active high pulse.

The active high pulse indicates a pending message is transmitted (see MCAN\_TXBRP). This pulse can be used to transfer another message to the Tx Buffer, which would need to be followed by writing 1 to the corresponding MCAN\_TXBAR[0] AR bit to mark a new Tx message pending transmission.

The Parity on Tx DMA Events is available using an EDC Controller which can be accessed through the ECC Aggregator.

Standard and Extended message filters can be set to issue a pulse when a filter match occurs. These 'Filter Events' can be used to DMA messages from the Rx FIFO. The events are high level single clock cycle pulses (ICLK).

#### 12.4.4.4.2.3

For more information about available DMA Requests, see [Table 12-473](#) and [Table 12-476](#).

#### 12.4.4.4.3 Operating Modes

##### 12.4.4.4.3.1 Software Initialization

Setting the MCAN\_CCCR[0] INIT bit to 1 starts a software initialization. This is done either by software or by a hardware reset, when an uncorrected bit error was detected in the Message RAM, or by going Bus\_Off state. While the MCAN\_CCCR[0] INIT bit is set, the message transfer is stopped and the status of the output TX pin is recessive (high). The counters of the Error Management Logic (EML) are unchanged. Setting the MCAN\_CCCR[0] INIT bit does not change any configuration register. Resetting the MCAN\_CCCR[0] INIT bit finishes the software initialization. After waiting for the occurrence of a sequence of 11 consecutive recessive bits (indication for Bus\_Idle state) the message transfer starts.

Access to the MCAN configuration registers is only enabled when both MCAN\_CCCR[0] INIT and MCAN\_CCCR[1] CCE bits are set (write protection).

The MCAN\_CCCR[1] CCE bit can only be set/reset while the MCAN\_CCCR[0] INIT = 1. The MCAN\_CCCR[1] CCE bit is automatically reset when the MCAN\_CCCR[0] INIT bit is reset.

The following registers are reset when the MCAN\_CCCR[1] CCE bit is set:

- MCAN\_HPMS - High Priority Message Status
- MCAN\_RXF0S - Rx FIFO 0 Status
- MCAN\_RXF1S - Rx FIFO 1 Status
- MCAN\_TXFQS - Tx FIFO/Queue Status
- MCAN\_TXBRP - Tx Buffer Request Pending
- MCAN\_TXBTO - Tx Buffer Transmission Occurred
- MCAN\_TXBCF - Tx Buffer Cancellation Finished
- MCAN\_TXEFS - Tx Event FIFO Status

The Timeout Counter value MCAN\_TOCV[15:0] TOC field is preset to the value configured by the MCAN\_TOCC[31:16] TOP field when the MCAN\_CCCR[1] CCE bit is set.

In addition the Tx Handler and Rx Handler are held in idle state while MCAN\_CCCR[1] CCE = 1.

The following registers are only writeable while MCAN\_CCCR[1] CCE = 0

- MCAN\_TXBAR - Tx Buffer Add Request
- MCAN\_TXBCR - Tx Buffer Cancellation Request

MCAN\_CCCR[7] TEST and MCAN\_CCCR[5] MON bits can only be set by the Host CPU while MCAN\_CCCR[0] INIT = 1 and MCAN\_CCCR[1] CCE = 1. Both bits may be reset at any time. The MCAN\_CCCR[6] DAR bit can only be set/reset while MCAN\_CCCR[0] INIT = 1 and MCAN\_CCCR[1] CCE = 1.

Table 12-477 shows the steps to configure the MCAN module.

**Table 12-477. Steps to Configure MCAN Module**

Step	Operation	Description	Pseudo Code
1	Initialize MCAN_CCCR	Set MCAN_CCCR[0] INIT bit and check that it has been set	INIT = 1; If INIT ≠ 1, wait until it is
2	Unlock protected registers	Set MCAN_CCCR[1] CCE bit	CCE = 1;
3	Configure CAN mode	Set MCAN_CCCR[8] FDOE bit to CAN FD	FDOE = 1 for CAN FD FDOE = 0 for CAN
4	Configure Bit Rate Switching	Set MCAN_CCCR[9] BRSE bit	BRSE = 1 with bit rate switching BRSE = 0 without bit rate switching
5	Set bit timing	Set MCAN_NBTP register	
6	Lock protected registers	Clear MCAN_CCCR[1] CCE bit	CCE = 0;
7	Return MCAN module to normal operation	Clear MCAN_CCCR[0] INIT bit and check it has been cleared	INIT = 0; If INIT ≠ 0, wait until it is

#### 12.4.4.4.3.2 Normal Operation

Once the MCAN module is initialized and the MCAN\_CCCR[0] INIT bit is reset to zero, the MCAN module synchronizes itself to the CAN bus and is ready for communication. After passing the acceptance filtering, received messages including Message Identifier (ID) and Data Length Code (DLC) are stored into a dedicated Rx Buffer or into Rx FIFO 0/Rx FIFO 1.

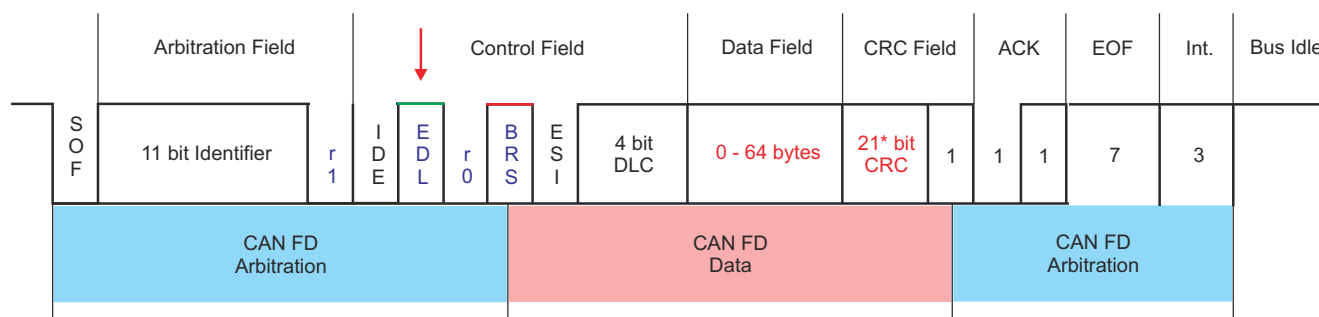
For messages to be transmitted dedicated Tx Buffers and/or a Tx FIFO or a Tx Queue can be initialized or updated.

#### Note

Automated transmission on reception of remote frames is not supported.

#### 12.4.4.4.3.3 CAN FD Operation

The CAN FD standard allows extended frames to be sent, up to 64 data bytes in a single frame at a higher bit rate for the data phase of a frame, up to 8 Mbps. The CAN FD standard introduces the ability to switch from one bit rate to another. Extended Data Length (EDL), as shown in Figure 12-467, sets a data length of up to 8 or up to 64 data bytes. Bit Rate Switching (BRS) indicates whether two bit rates (the data phase is transmitted at a different bit rate to the arbitration phase) are enabled.



\* 17 bit CRC for data fields with up to 16 bytes

mcan-004a

**Figure 12-467. CAN FD Frame**

### Note

Figure 12-467 presents CAN FD frame according to the Non-ISO CAN FD (legacy) protocol. In the new ISO CAN FD protocol the CRC Field includes additional 5 bits (three stuff bit counter (SBC) bits and two parity bits). With these additional bits, a weakness identified in the error detection scheme chosen by the original protocol is removed. By setting MCAN\_CCCR[15] NISO bit, the ISO or Non-ISO CAN FD format can be chosen. In CAN network ISO CAN FD and non-ISO CAN FD devices should never mix.

There are two variants of CAN FD frame transmission:

- CAN FD frame transmission without bit rate switching
- CAN FD frame transmission where control field, data field, and CRC field are transmitted with a higher bit rate than the beginning and the end of the frame

In the CAN frames FDF = recessive (logical 1) signifies a CAN FD frame, FDF = dominant (logical 0) signifies a Classic CAN frame. In a CAN FD frame, the two bits following FDF - res and BRS, decide whether the bit rate inside of this CAN FD frame is switched. A CAN FD bit rate switch is signified by res = dominant and BRS = recessive. Note that the coding of res = recessive is reserved for future expansion of the protocol.

In case the MCAN module receives a frame with FDF = recessive and res = recessive, it will signal a Protocol Exception Event by setting the MCAN\_PSR[14] EXE bit. When Protocol Exception Handling is enabled (MCAN\_CCCR[12] PXHD = 0), this causes the operation state to change from Receiver (MCAN\_PSR[4-3] ACT = 10) to Integrating (MCAN\_PSR[4-3] ACT = 00) at the next sample point. In case Protocol Exception Handling is disabled (MCAN\_CCCR[12] PXHD = 1), the MCAN will treat a recessive res bit as an form error and will respond with an error frame.

CAN FD operation is enabled by programming the MCAN\_CCCR[8] FDOE bit. In case MCAN\_CCCR[8] FDOE = 1, transmission and reception of CAN FD frames is enabled. Transmission and reception of Classic CAN frames is always possible. Whether a CAN FD frame or a Classic CAN frame is transmitted can be configured via the FDF bit in the respective Tx Buffer element.

With MCAN\_CCCR[8] FDOE = 0, received frames are interpreted as Classic CAN frames, which leads to the transmission of an error frame when receiving a CAN FD frame. When CAN FD operation is disabled, no CAN FD frames are transmitted even if the FDF bit of a Tx Buffer element is set. The MCAN\_CCCR[8] FDOE and MCAN\_CCCR[9] BRSE bits can only be changed while the MCAN\_CCCR[0] INIT and MCAN\_CCCR[1] CCE bits are both set. With MCAN\_CCCR[8] FDOE = 0, the setting of bits FDF and BRS is ignored and frames are transmitted in Classic CAN format.

With MCAN\_CCCR[8] FDOE = 1 and MCAN\_CCCR[9] BRSE = 0, only FDF bit of a Tx Buffer element is evaluated. With MCAN\_CCCR[8] FDOE = 1 and MCAN\_CCCR[9] BRSE = 1, transmission of CAN FD frames with bit rate switching is enabled. All Tx Buffer elements with bits FDF and BRS set are transmitted in CAN FD format with bit rate switching.

A mode change during CAN operation is only recommended under the following conditions:

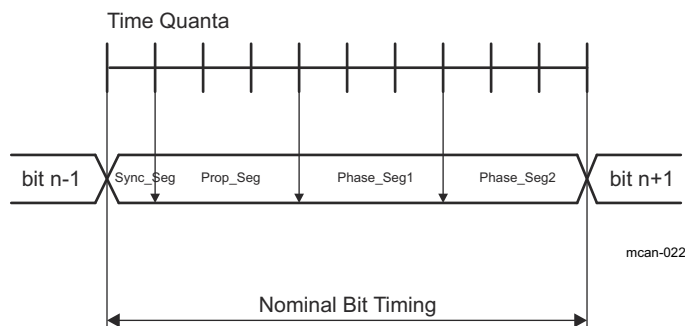
- The failure rate in the CAN FD data phase is significant higher than in the CAN FD arbitration phase. In this case disable the CAN FD bit rate switching option for transmissions.
- During system startup all nodes are transmitting Classic CAN messages until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation.
- Wakeup messages in CAN Partial Networking have to be transmitted in Classic CAN format.
- End-of-line programming in case not all nodes are CAN FD capable. Non CAN FD nodes are held in Silent mode until programming has completed. Then all nodes switch back to Classic CAN communication.

In the CAN FD format, the DLC coding differs from the standard CAN format (see [Table 12-478](#)).

**Table 12-478. DLC Coding**

DLC	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Number of Data Bytes in Standard CAN	0	1	2	3	4	5	6	7	8	8	8	8	8	8	8	8
Number of Data Bytes in CAN FD	0	1	2	3	4	5	6	7	8	12	16	20	24	32	48	64

For CAN FD frames, the bit timing will be switched inside the frame after the BRS (Bit Rate Switch) bit in case this bit is recessive. In the CAN FD arbitration phase, before the BRS bit, the nominal CAN bit timing (see [Figure 12-468](#)) is used as configured by the Nominal Bit Timing and Prescaler Register MCAN\_NBTP. In the following CAN FD data phase, the data phase bit timing is used as configured by the Data Bit Timing and Prescaler Register MCAN\_DBTP. The bit timing is switched back from the data phase timing at the CRC delimiter or when an error is detected, whichever occurs first.



**Figure 12-468. CAN Bit Timing**

The maximum configurable data phase bit timing depends on the CAN clock frequency (FCLK). Example: with FCLK = 20 MHz and the shortest configurable bit time of 4  $t_q$  (time quanta), the bit rate in the data phase is 5 Mbps.

In both data frame formats, CAN FD and CAN FD with bit rate switching, the value of the ESI (Error Status Indicator) bit depends on transmitter's error state (see MCAN\_PSR[11] RESI bit) monitored at the start of the transmission. If the transmitter has error passive flag the ESI bit is transmitted recessive, else it is transmitted dominant.

#### 12.4.4.4.3.4 Transmitter Delay Compensation

##### 12.4.4.4.3.4.1 Description

When only one CAN FD node is transmitting and all others are receivers the length of the bus line has no impact. When transmitting via the TX pin the MCAN module receives the transmitted data from its local CAN transceiver via the RX pin. The received data is delayed. If the transmitter delay is greater than TSEG1 (time segment before sample point), a bit error is detected.

The MCAN module provides a delay compensation mechanism to compensate the transmitter delay. The compensation mechanism enables transmission with higher bit rates during the CAN FD data phase



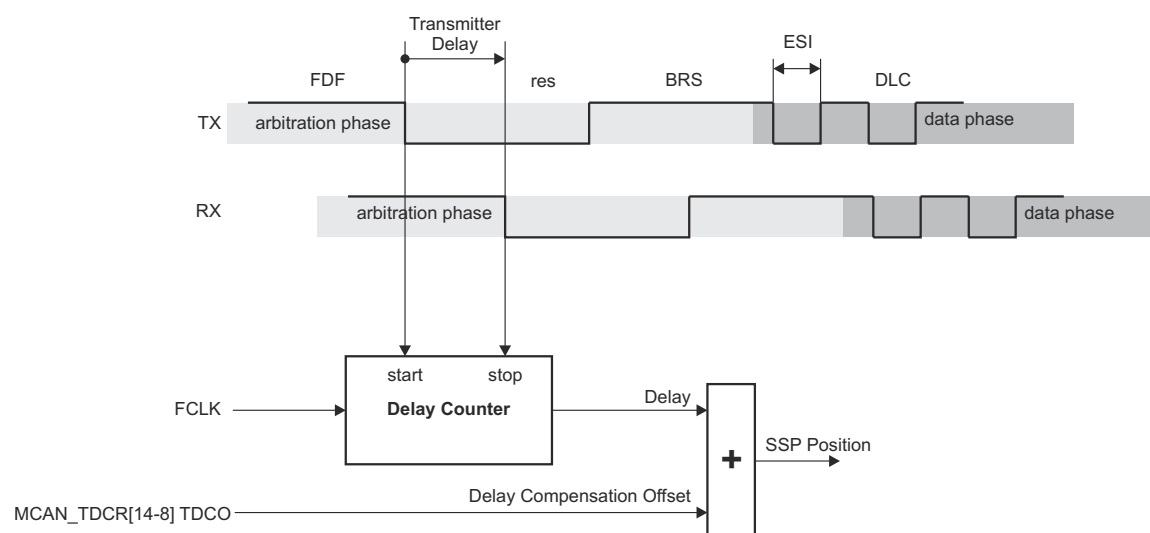
independent of the delay of a specific CAN transceiver. Without transmitter delay compensation the bit rate in the data phase is limited by the transmitter delay.

The mechanism enables configurations where the data bit time is shorter than the transmitter delay (it is described in detail in ISO 11898-1:2015). The transmitter delay compensation is enabled by setting the MCAN\_DBTP[23] TDC bit to 1.

The delayed transmit data is compared against the received data at the Secondary Sample Point (SSP) in order to check for bit errors during the data phase of transmitting nodes. If a bit error is detected, the transmitter will react on this bit error at the next following regular sample point. During arbitration phase the delay compensation is always disabled.

The received bit is compared against the transmitted bit at the SSP. The SSP position is defined as the sum of the measured delay from the MCAN's transmit output TX pin through the transceiver to the receive input RX pin plus the transmitter delay compensation offset configured by the MCAN\_TDCR[14-8] TDCO field (see Figure 12-469). The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit (example: half of the bit time in the data phase). The position of the SSP is rounded down to the next integer number of mtq.

The actual transmitter delay compensation value can be checked by reading the MCAN\_PSR[22-16] TDCV field. This field is cleared when the MCAN\_CCCR[0] INIT bit is set and is updated at each transmission of CAN FD frame while the MCAN\_DBTP[23] TDC bit is set.



mcan-005

**Figure 12-469. Transmitter Delay Measurement**

#### 12.4.4.3.4.2 Transmitter Delay Compensation Measurement

When transmitter delay compensation is enabled (by programming MCAN\_DBTP[23] TDC = 1), the measurement is started within each transmitted CAN FD frame at the falling edge of FDF bit to bit res. The measurement is stopped when this edge is seen at the receive input RX pin of the transmitter. The resolution of this measurement is one mtq (see Figure 12-469). The mtq (minimum time quantum) dimension is equal to the CAN clock period (FCLK).

The use of a transmitter delay compensation filter window can be enabled by programming MCAN\_TDCR[6-0] TDCF field. This filter feature defines a minimum value for the SSP position to avoid the case in which a dominant glitch inside the received FDF bit ends the delay compensation measurement before the falling edge of the received res bit, resulting in an early taken SSP position. Dominant edges on the RX pin, that would result in an earlier SSP position are ignored for transmitter delay measurement. The measurement is stopped when the SSP position is at least MCAN\_TDCR[6-0] TDCF field and the RX pin is low.





#### 12.4.4.4.3.7 Disabled Automatic Retransmission (DAR) Mode

According to the CAN Specification (see ISO11898-1:2015, *Recovery Management* section), the MCAN module provides means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. By default automatic retransmission is enabled (see the MCAN\_CCCR[6] DAR bit).

##### 12.4.4.4.3.7.1 Frame Transmission in DAR Mode

In DAR mode all transmissions are automatically cancelled after they started on the CAN bus. A Tx Buffer's Tx Request Pending MCAN\_TXBRP[xx] TRPx bit is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, has been aborted due to lost arbitration, or when an error occurred during frame transmission.

Successful transmission:

- Corresponding Tx Buffer Transmission Occurred MCAN\_TXBTO[xx] TOx bit is set
- Corresponding Tx Buffer Cancellation Finished MCAN\_TXBCF[xx] CFx bit is not set

Successful transmission in spite of cancellation:

- Corresponding Tx Buffer Transmission Occurred MCAN\_TXBTO[xx] TOx bit is set
- Corresponding Tx Buffer Cancellation Finished MCAN\_TXBCF[xx] CFx bit is set

Arbitration lost or frame transmission disturbed:

- Corresponding Tx Buffer Transmission Occurred MCAN\_TXBTO[xx] TOx bit is not set
- Corresponding Tx Buffer Cancellation Finished MCAN\_TXBCF[xx] CFx bit is set

In case of a successful frame transmission, and if storage of Tx events is enabled, a Tx Event FIFO element is written with Event Type ET = 10 (transmission in spite of cancellation).

#### 12.4.4.4.3.8 Power Down (Sleep Mode)

The entering in Power Down mode is controlled via two sources:

- PSC (via clock stop request signal)
- Software (by writing to the MCAN\_CCCR[4] CSR bit)

As long as the clock stop request signal is active, the MCAN\_CCCR[4] CSR bit is read as 1.

When all pending transmission requests have completed, the MCAN module waits until bus idle state is detected. Then the MCAN module sets the MCAN\_CCCR[0] INIT bit to 1 to prevent any further CAN transfers.

The MCAN module acknowledges that it is ready for power down:

- By asserting clock stop acknowledge signal to the PSC (in case of PSC source).
- By setting the MCAN\_CCCR[3] CSA flag bit to 1 (in case of Software source).

In this state, before the clocks are switched off, further register accesses can be made. Now the module clock inputs ICLK and FCLK may be switched off.

To leave power down mode, the application has to turn on the module clocks before resetting the input clock stop request signal respectively the MCAN\_CCCR[4] CSR flag bit. The MCAN will acknowledge this by resetting the output clock stop acknowledge signal respectively the MCAN\_CCCR[3] CSA flag bit. Afterwards, the application can restart CAN communication by resetting the MCAN\_CCCR[0] INIT bit.

Restoring the clocks from clock stop mode, needs to be done according to how the clock stop was initiated:

- If Software asserts the MCAN\_CCCR[3] CSA flag bit, once the MCAN module goes idle, the MCAN\_CCCR[0] INIT bit is set. To get it started again, Software needs to write 0 to the MCAN\_CCCR[0] INIT bit.
- If PSC is issuing a clock stop request, than there are two options for waking up:
  - After removing clock stop request signal, Software would need to write 0 to the MCAN\_CCCR[0] INIT bit, or
  - If the MCANSS\_CTRL[5] AUTOWAKEUP bit is set, than after removing clock stop request signal, an FSM inside the MCAN module will reset the MCAN\_CCCR[0] INIT bit (without Software).

#### 12.4.4.4.3.8.1 External Clock Stop Mode

The MCAN module supports two external clock stop modes:

- Immediate
- Graceful

In a graceful clock stop mode, when the clock stop request is asserted, the MCAN core will respond with clock stop acknowledge when all pending Tx messages have been processed and an Idle line had been detected. The MCAN\_CCCR[0] INIT bit will be set, the MCAN core will go and stay Idle.

The automatic wakeup feature is enabled by setting the MCANSS\_CTRL[5] AUTOWAKEUP and MCANSS\_CTRL[4] WAKEUPREQEN bits to 1 (for more information, see [Section 12.4.4.4.3.8.3, Wakeup request](#)). When external clock stop request is removed and no suspend request is active, a read-modify-write to the MCAN\_CCCR[0] INIT bit is performed to clear it.

#### 12.4.4.4.3.8.2 Suspend Mode

The MCAN module supports two suspend modes:

- Immediate
- Graceful

In a graceful suspend mode (see the MCANSS\_CTRL[3] DBGSUSP\_FREE bit), when the suspend request is asserted, a clock stop request to the MCAN core is performed. The MCAN core will respond with clock stop acknowledge when all pending Tx messages have been processed and an Idle line had been detected. At that point the MCAN\_CCCR[0] INIT bit will be set, the MCAN core will go and stay Idle. The suspend state can be verified by reading MCAN\_CCCR[0] INIT bit.

The automatic wakeup feature is enabled by setting the MCANSS\_CTRL[5] AUTOWAKEUP and MCANSS\_CTRL[4] WAKEUPREQEN bits to 1 (for more information, see [Section 12.4.4.4.3.8.3, Wakeup request](#)). When suspend request is removed, if no external clock stop request is active, a read-modify-write to the MCAN\_CCCR[0] INIT bit is performed to clear it.

During suspend mode the auto-clear feature is disabled. The following register fields have an auto-clear feature:

- MCAN\_ECR[23-16] CEL
- MCAN\_PSR[2-0] LEC
- MCAN\_PSR[10-8] DLEC
- MCAN\_PSR[11] RESI
- MCAN\_PSR[12] RBRS
- MCAN\_PSR[13] RFDF
- MCAN\_PSR[14] PXE

#### 12.4.4.4.3.8.3 Wakeup request

Issuing a clock stop request puts the MCAN module into Power Down mode (Sleep Mode). During transition from IDLE to ACTIVE, if the MCANSS\_CTRL[5] AUTOWAKEUP and MCANSS\_CTRL[4] WAKEUPREQEN bits are enabled, after the MCAN Core respond to the removal of the clock stop request with removing the clock stop acknowledge, a read-modify-write will be issued to clear the MCAN\_CCCR[0] INIT bit and the MCAN core will resume operation.

If the MCANSS\_CTRL[4] WAKEUPREQEN bit is set, the MCAN module provides a wakeup request on the following wakeup event:

- The receive RX pin is dominant (logical 0)

The wakeup request is de-asserted when any of the following conditions occur:

- Clock stop request is removed and clock stop acknowledge is de-asserted
- A reset is applied to the MCAN module

#### 12.4.4.4.3.9 Test Modes

The MCAN\_TEST register write access is enabled by setting the test mode enable MCAN\_CCCR[7] TEST bit to 1. The MCAN\_TEST register allows the configuration of the test modes and test functions.

The CAN transmit TX pin has four output functions. One of those functions can be selected by programming the MCAN\_TEST[6-5] TX field. Additionally to its default function (the serial data output) it can drive the CAN Sample Point signal to monitor the MCAN's bit timing and it can drive constant dominant or recessive values.

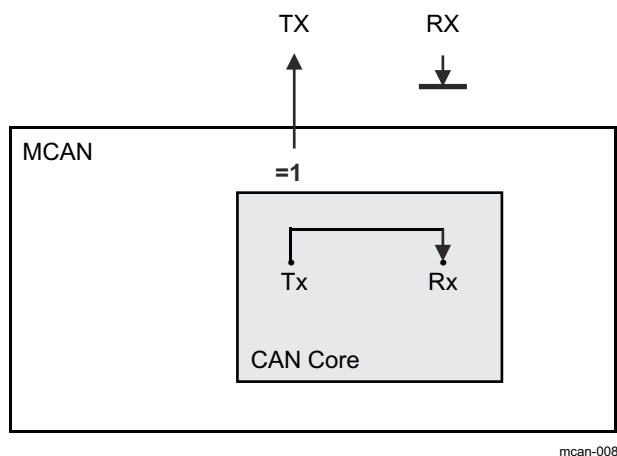
The actual value of the CAN receive RX pin can be monitored from MCAN\_TEST[7] RX bit. Both functions can be used to check the CAN bus physical layer. Due to the synchronization mechanism between CAN clock (FCLK) and Host clock (ICLK) domain, there may be a delay of several Host clock periods between writing to the MCAN\_TEST[6-5] TX field until the new configuration is visible at the output TX pin. This applies also when reading input RX pin via the MCAN\_TEST[7] RX bit.

### Note

Test modes should be used for self test only. The software control for TX pin interferes with all CAN protocol functions. It is not recommended to use test modes for application.

#### 12.4.4.4.3.9.1 Internal Loopback Mode

The MCAN module can be set into Internal Loopback Mode by programming MCAN\_TEST[4] LBCK and MCAN\_CCCR[5] MON bits to 1. The Internal Loopback Mode is used for a 'Hot Selftest'. The 'Hot Selftest' allows the MCAN module to be tested without affecting a running CAN system connected to the TX and RX pins. In this mode RX pin is disconnected from the MCAN module and TX pin is held recessive. [Figure 12-471](#) shows the connection of the TX and RX pins to the MCAN module in case of Internal Loopback Mode.



**Figure 12-471. Internal Loopback Mode**

#### 12.4.4.4.4 Timestamp Generation

The MCAN module has integrated a 16-bit wrap-around counter for timestamp generation. The timestamp counter prescaler MCAN\_TSCC[19-16] TCP field can be configured to clock the counter in multiples of CAN bit times (1-16). The counter is readable via the MCAN\_TSCV[15-0] TSC field. A write access to the MCAN\_TSCV register resets the counter to zero. When the timestamp counter wraps around the interrupt MCAN\_IR[16] TSW flag is set. On start of a frame reception/transmission the counter value is captured and stored into the timestamp section of an Rx Buffer/Rx FIFO (RXTS[15-0]) or Tx Event FIFO (TXTS[15-0]) element. For more information, see [Section 12.4.4.4.10, Message RAM](#).

#### 12.4.4.4.1 External Timestamp Counter

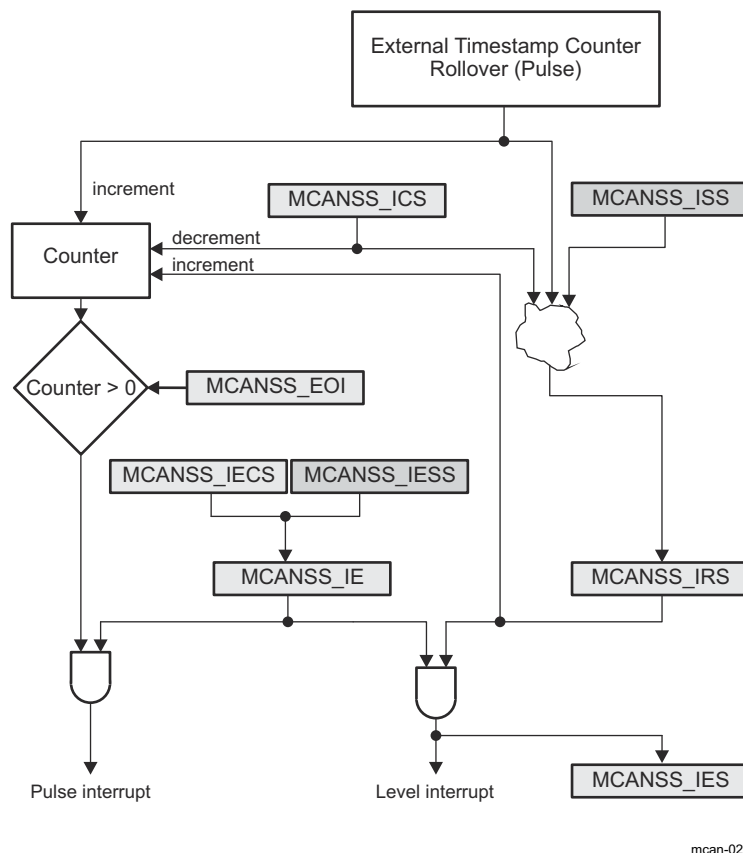
For CAN FD operation mode the MCAN core requires an External Timestamp Counter. An externally generated 16-bit vector may substitute the integrated 16-bit CAN bit time counter (internal timestamp counter) for receive

and transmit timestamp generation. An external 16-bit timestamp counter can be used by programming the MCAN\_TSCC[1-0] TSS field.

The External Timestamp Counter uses the interface clock (ICLK) as a reference clock. The MCAN Core accepts a 16-bit timestamp. A 24-bit prescaler provides a programmable resolution for the timestamp (see MCANSS\_EXT\_TS\_PRESCALER[23-0] PRESCALER bit field). The External Timestamp Counter counter can be enabled or disabled through the MCANSS\_CTRL[6] EXT\_TS\_CNTR\_EN bit. When disabled the counter is reset back to zero. While enabled the counter keeps incrementing. When the timestamp rolls over the MCAN timestamp interrupt is generated.

When the timestamp rolls over the MCANSS\_IRS register is set (see Figure 12-472). The MCANSS\_IE register can be affected by writing to the MCANSS\_IESS register to set or to the MCANSS\_IECS register to clear. The MCANSS\_IESS register is a shadow register mapped to the same address as the MCANSS\_IE register. The level interrupt is a reflection of both MCANSS\_IRS and MCANSS\_IE being set. The MCANSS\_IES register reflects the level interrupt. When an rollover event occurs the interrupt counter is incremented. Writing to the MCANSS\_ICS register to clear the MCANSS\_IRS register will also decrement the interrupt counter. Writing to the MCANSS\_EOI register will issue another pulse if the interrupt counter is not zero.

The rollover event can be artificially simulated by software through writing to the Interrupt Set Shadow register (MCANSS\_ISS). The MCANSS\_ISS register is a shadow register mapped to the same address as the MCANSS\_IRS register.



**Figure 12-472. External Timestamp Counter Interrupt**

#### 12.4.4.4.5 Timeout Counter

The MCAN module has integrated a 16-bit Timeout Counter. It is used to signal timeout conditions for the Rx FIFO 0, Rx FIFO 1, and Tx Event FIFO Message RAM elements. The Timeout Counter is configured via the MCAN\_TOCC register. It is enabled via the MCAN\_TOCC[0] ETOC bit. The Timeout Counter operates

as down-counter and uses the same prescaler programmed by the MCAN\_TSCC[19-16] TCP field as the Timestamp Counter. The actual counter value can be monitored from the MCAN\_TOCV[15-0] TOC field. The Timeout Counter can be started only when MCAN\_CCCR[0] INIT = 0 and stopped when MCAN\_CCCR[0] INIT = 1 (example: when the MCAN enters Bus\_Off state). The operation mode is selected by the MCAN\_TOCC[2-1] TOS field. When Continuous Mode is selected, the counter starts when MCAN\_CCCR[0] INIT = 0, a write to the MCAN\_TOCV register presets the counter to the value configured by the MCAN\_TOCC[31-16] TOP field and continues down-counting.

In case the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by the MCAN\_TOCC[31-16] TOP field. Down-counting is started when the first FIFO element is stored. Writing to the MCAN\_TOCV register has no effect. When the counter reaches zero, the interrupt MCAN\_IR[18] TOO flag is set.

In Continuous Mode, the counter is immediately restarted at the value configured by the MCAN\_TOCC[31-16] TOP field.

#### 12.4.4.4.6 ECC Support

The Message Memory is wrapped in an ECC wrapper providing SECDED parity functionality. The ECC wrapper is controlled by an ECC Aggregator.

##### 12.4.4.4.6.1 ECC Wrapper

The ECC wrapper provides Single Error Correction (SEC) and Double Error Detection (DED) parity to the Message Memory content. It has side band signals for error notification. The ECC Wrapper implements an error injection test mode.

The error correction is done using a lazy write back. When an error is detected, it is noted in a FIFO Queue which waits for an access gap to write the data back and refresh the memory. If a transaction writes new data to the compromised entry before the lazy write back completes, the write back is discarded.

##### 12.4.4.4.6.2 ECC Aggregator

---

#### Note

For more information about ECC Aggregator, refer to [Section 12.11.4, ECC Aggregator](#).

---

#### 12.4.4.4.7 Rx Handling

The Rx Handler controls the following operations:

- Acceptance filtering
- The transfer of received messages to the Rx Buffers or to one of the two Rx FIFOs (Rx FIFO 0 or Rx FIFO 1)
- Rx FIFO Put and Get Index operations

##### 12.4.4.4.7.1 Acceptance Filtering

The MCAN module is capable to configure two sets of acceptance filters - one set for standard and one set for extended identifiers. These filters can be assigned to an Rx Buffer or to one of the two Rx FIFOs.

The main features of the filter elements are:

- Each filter element can be configured as:
  - Range Filter (from - to)
  - Filter for specific IDs (for one or two dedicated IDs)
  - Classic Bit Mask Filter
- Each filter element can be enabled/disabled individually
- Each filter element can be configured for acceptance or rejection filtering
- Filters are checked sequentially and execution (acceptance filtering procedure) stops at the first matching filter element or when the end of the filter list is reached



Related configuration registers are:

- Global Filter Configuration (MCAN\_GFC) register
- Standard ID Filter Configuration (MCAN\_SIDFC) register
- Extended ID Filter Configuration (MCAN\_XIDFC) register
- Extended ID AND Mask (MCAN\_XIDAM) register

Depending on the configuration of the filter element (see SFEC/EFEC in [Section 12.4.4.4.10, Message RAM](#)) if filter matches, one of the following actions is performed:

- Received frame is stored in FIFO 0 or FIFO 1
- Received frame is stored in Rx Buffer
- Received frame is stored in Rx Buffer and generation of pulse at filter event pin is performed. This is high level single ICLK pulse. For more information, see [Section 12.4.4.4.2.1, DMA Requests](#).
- Received frame is rejected
- Set High Priority Message interrupt flag MCAN\_IR[8] HPM
- Set High Priority Message interrupt flag MCAN\_IR[8] HPM and store received frame in FIFO 0 or FIFO 1

Acceptance filtering starts when complete Message ID is received. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If a filter element matches - the Rx Handler starts writing the received message data in portions of 32 bit to the matching Rx Buffer or Rx FIFO. If an error condition occurs (for example: CRC error), this message is rejected with the following impact on the affected Rx Buffer or Rx FIFO:

- Rx Buffer:  
New Data flag (MCAN\_NDAT1/MCAN\_NDAT2) of matching Rx Buffer is not set, but Rx Buffer (partly) overwritten with received data (for error type see MCAN\_PSR[2-0] LEC respectively MCAN\_PSR[10-8] DLEC fields).
- Rx FIFO:  
Put index of matching Rx FIFO is not updated, but related Rx FIFO element (partly) overwritten with received data (for error type see MCAN\_PSR[2-0] LEC respectively MCAN\_PSR[10-8] DLEC fields). If matching Rx FIFO is configured to operate in overwrite mode, the boundary conditions described in [Section 12.4.4.4.7.2.2](#) have to be considered.

#### 12.4.4.4.7.1.1 Range Filter

Each filter element can be configured to operate as Range Filter (Standard Filter Type SFT = 00/Extended Filter Type EFT = 00). The filter matches for all received message frames with IDs in the range from SFID1 to SFID2 (SFID2 ≥ SFID1) respectively in the range from EFID1 to EFID2 (EFID2 ≥ EFID1). For more information see [Section 12.4.4.4.10.5, Standard Message ID Filter Element](#) and [Section 12.4.4.4.10.6, Extended Message ID Filter Element](#).

There are two options for range filtering of extended frames:

- Extended Filter Type EFT = 00: The Extended ID AND Mask (MCAN\_XIDAM) is used for Range Filtering. The Message ID of received frames is ANDed with the Extended ID AND Mask (MCAN\_XIDAM) before the range filter is applied.
- Extended Filter Type EFT = 11: The Extended ID AND Mask (MCAN\_XIDAM) is not used for Range Filtering.

#### 12.4.4.4.7.1.2 Filter for specific IDs

Each filter element can be configured to filter one or two dedicated Message IDs (Standard Filter Type SFT = 01/Extended Filter Type EFT = 01). To filter only one specific Message ID, the filter element has to be configured with SFID1 = SFID2 respectively EFID1 = EFID2. For more information see [Section 12.4.4.4.10.5, Standard Message ID Filter Element](#) and [Section 12.4.4.4.10.6, Extended Message ID Filter Element](#).

#### **12.4.4.4.7.1.3 Classic Bit Mask Filter**

Classic bit mask filtering can filter groups of Message IDs (Standard Filter Type SFT =10/Extended Filter Type EFT =10). This is done by masking single bits of a received Message ID. In this case SFID1/EFID1 element is used as Message ID filter, while SFID2/EFID2 element is used as filter mask.

A 0 bit at the filter mask (SFID2/EFID2) will mask out the corresponding bit position of the configured Message ID filter (SFID1/EFID1) and the value of the received Message ID at that bit position is not relevant for acceptance filtering. Only those bits of the received Message ID where the corresponding mask bits are 1 are relevant for acceptance filtering.

There are two interesting cases:

- All mask bits are 1: a match occurs only when the received Message ID and the configured Message ID filter are identical.
- All mask bits are 0: all Message IDs match.

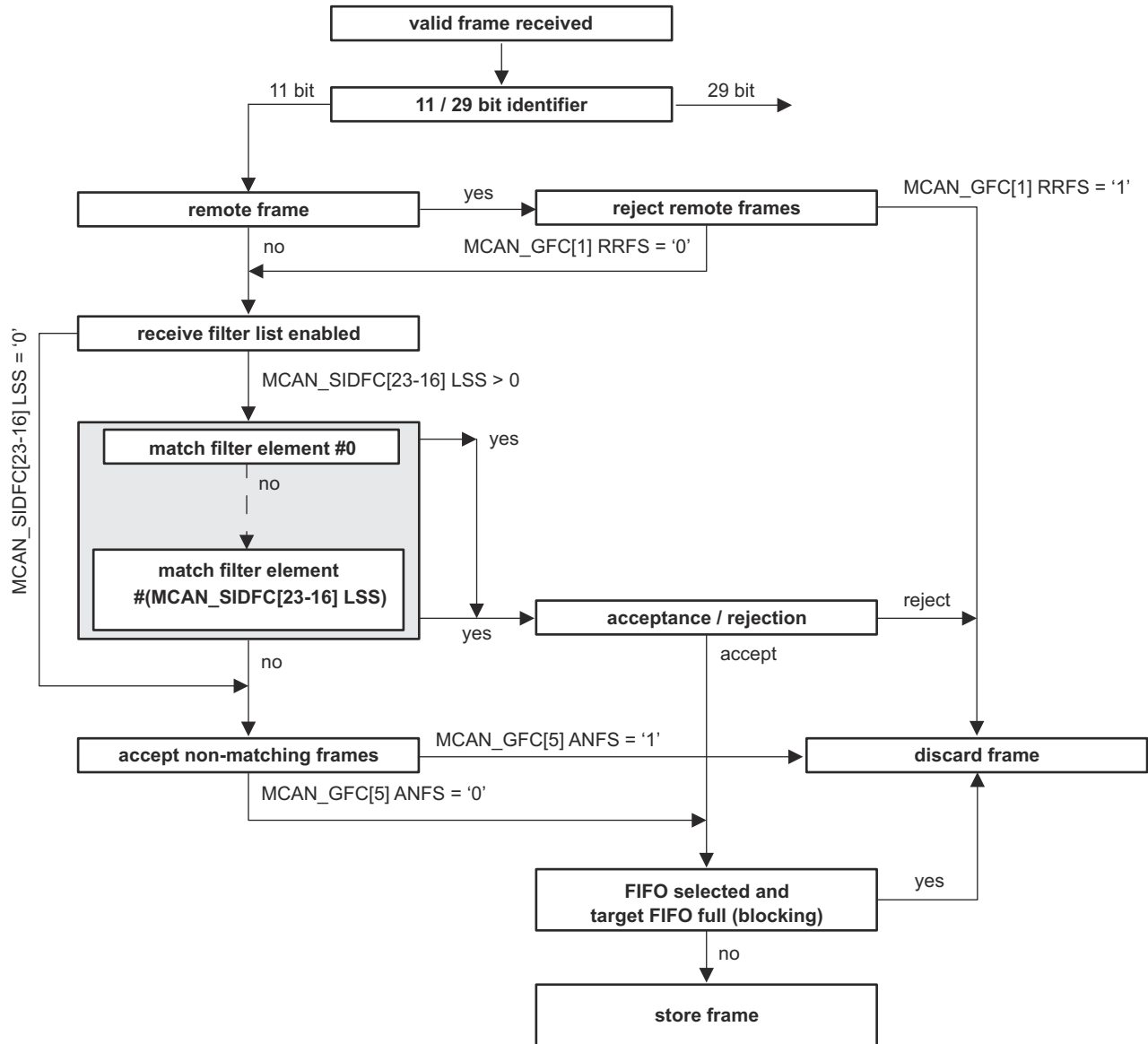
#### **12.4.4.4.7.1.4 Standard Message ID Filtering**

The standard Message ID (11-bit ID) filtering flow is shown in [Figure 12-473](#). [Section 12.4.4.4.10.5](#), *Standard Message ID Filter Element* describes the standard Message ID filter element.

The Remote Transmission Request (RTR) and Extended Identifier (XTD) bits of the received frames are compared against the list of configured filter elements. This is controlled by the following registers:

- Global Filter Configuration (MCAN\_GFC) register
- Standard ID Filter Configuration (MCAN\_SIDFC) register





mcan-009

**Figure 12-473. Standard Message ID Filter Path**

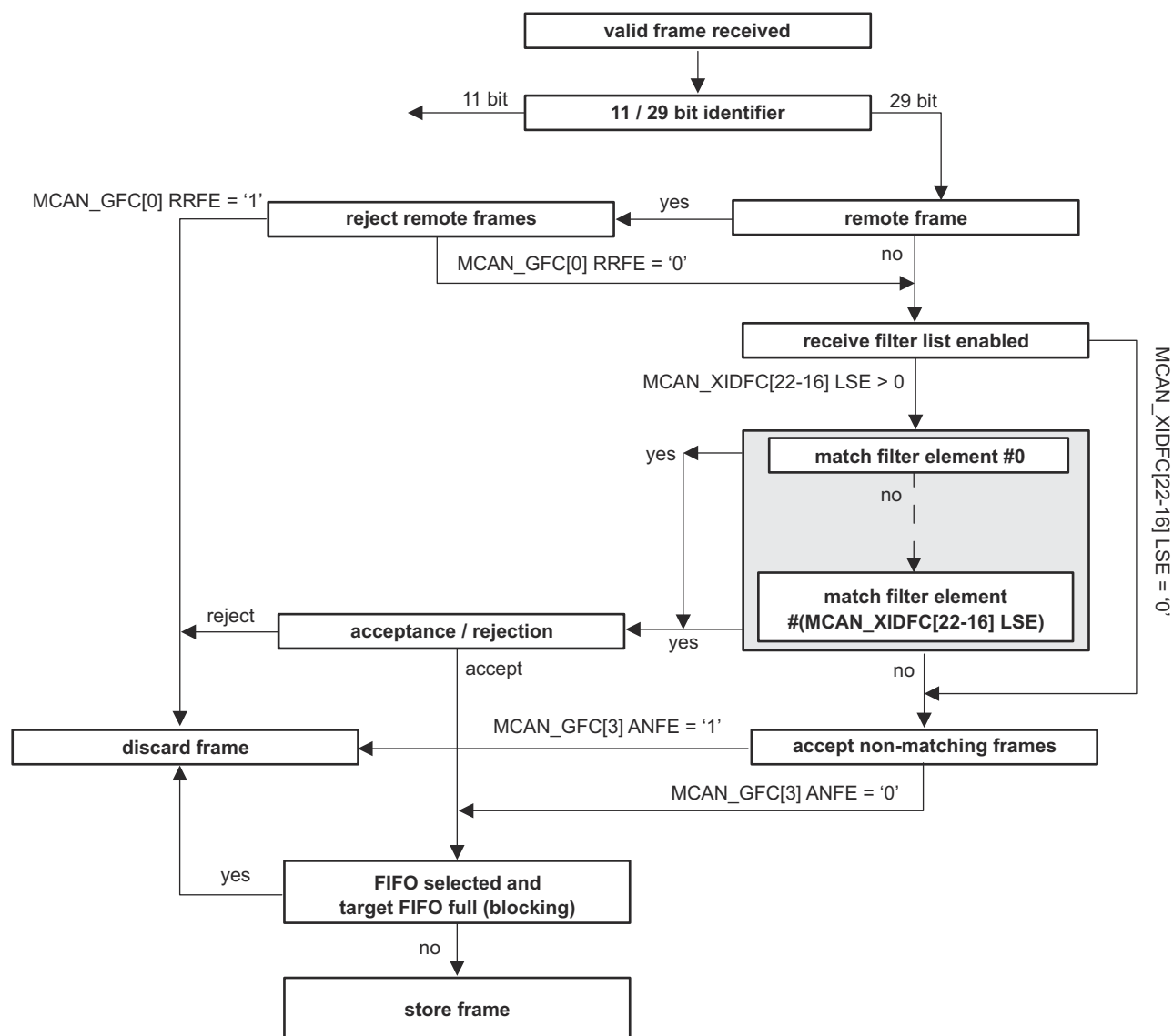
#### 12.4.4.4.7.1.5 Extended Message ID Filtering

The extended Message ID (29-bit ID) filtering flow is shown in [Figure 12-474](#). [Section 12.4.4.4.10.6](#), *Extended Message ID Filter Element* describes the extended Message ID filter element.

The Remote Transmission Request (RTR) and Extended Identifier (XTD) bits of the received frames are compared against the list of configured filter elements. This is controlled by the following registers:

- Global Filter Configuration (MCAN\_GFC) register
- Extended ID Filter Configuration (MCAN\_XIDFC) register

Note that before the filter list is executed the received identifier is ANDed with the Extended ID AND Mask (MCAN\_XIDAM).



mcan-010

**Figure 12-474. Extended Message ID Filter Path**

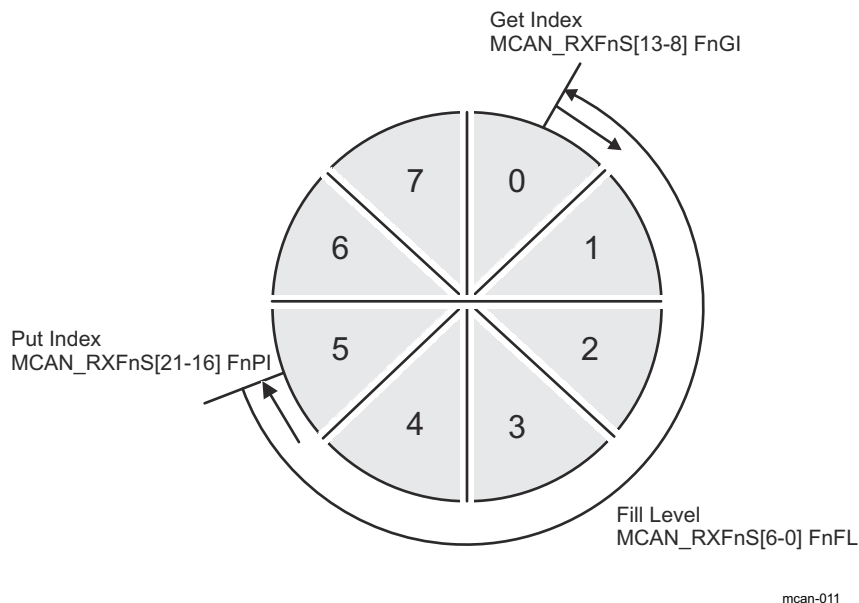
#### 12.4.4.4.7.2 Rx FIFOs

The configuration of the Rx FIFOs (Rx FIFO 0 and Rx FIFO 1) can be done via the MCAN\_RXF0C and MCAN\_RXF1C registers. Each Rx FIFO can be configured to store up to 64 received messages.

After acceptance filtering the received messages that passed are transferred to the Rx FIFO. The filter mechanisms available for the Rx FIFO 0 and Rx FIFO 1 is described in [Section 12.4.4.4.7.1, Acceptance Filtering](#). [Section 12.4.4.4.10.2, Rx Buffer and FIFO Element](#) describes the Rx FIFO element.

The Rx FIFO watermark can be used to prevent an Rx FIFO overflow. If the Rx FIFO fill level reaches the Rx FIFO watermark configured by the MCAN\_RXFnC[30-24] FnWM field (where: n = 0 or 1) an interrupt flag MCAN\_IR[1] RF0W/MCAN\_IR[5] RF1W is set.

When the Rx FIFO Put Index reaches the Rx FIFO Get Index (MCAN\_RXFnS[21-16] FnPI = MCAN\_RXFnS[13-8] FnGI) an Rx FIFO Full condition is signalled by the MCAN\_RXFnS[24] FnF status bit and interrupt flag MCAN\_IR[2] RF0F/MCAN\_IR[6] RF1F is set. [Figure 12-475](#) shows Rx FIFO Status. The FIFOs fill level is presented in the MCAN\_RXFnS[6-0] FnFL field (the number of elements stored in Rx FIFO).



**Figure 12-475. Rx FIFO Status**

Rx FIFOs start address in the Message RAM (MCAN\_RXFnC[15-2] FnSA field) have to be configured when reading from an Rx FIFO (Rx FIFO Get Index - MCAN\_RXFnS[13-8] FnGI). [Table 12-479](#) presents Rx Buffer/Rx FIFO Element Size for different Rx Buffer/Rx FIFO Data Field Size which is configured via the MCAN\_RXESC register.

**Table 12-479. Rx Buffer/Rx FIFO Element Size**

MCAN_RXESC[10-8] RBDS MCAN_RXESC[2-0] F0DS/ MCAN_RXESC[6-4] F1DS	Data Field [bytes]	FIFO Element Size [RAM words]
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

#### 12.4.4.4.7.2.1 Rx FIFO Blocking Mode

The Rx FIFO blocking mode is the default operation mode for the Rx FIFOs. It is configured by the MCAN\_RXFnC[31] FnOM = 0.

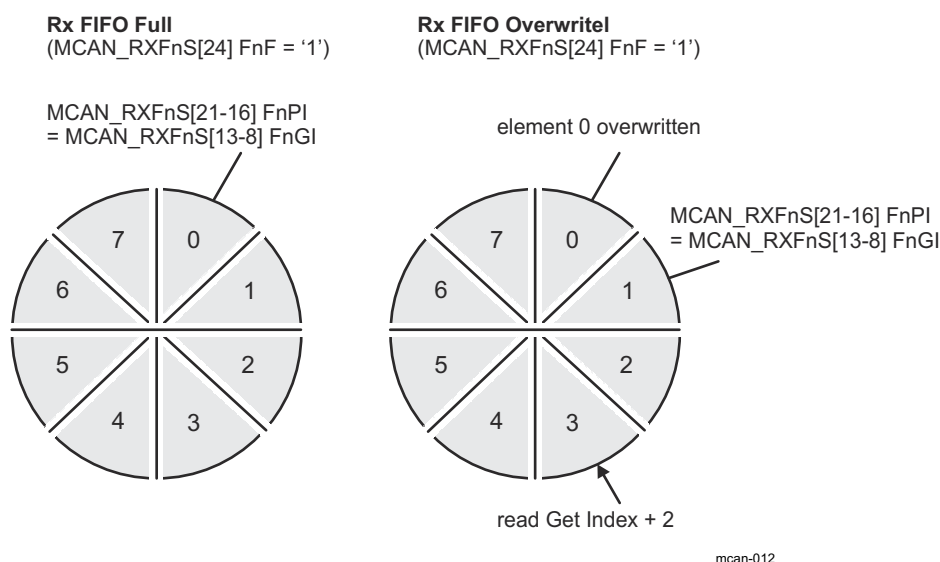
If an Rx FIFO full condition is reached (MCAN\_RXFnS[21-16] FnPI = MCAN\_RXFnS[13-8] FnGI), no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO Get Index has been incremented. An Rx FIFO full condition is signalled by the MCAN\_RXFnS[24] FnF = 1 and interrupt flag MCAN\_IR[2] RF0F/MCAN\_IR[6] RF1F is set.

In case a message is received while the corresponding Rx FIFO is full, this message is rejected and the message lost condition is signalled by MCAN\_RXFnS[25] RFnL = 1 and interrupt flag MCAN\_IR[3] RFnL/MCAN\_IR[25] RFnL is set.

#### 12.4.4.4.7.2.2 Rx FIFO Overwrite Mode

The Rx FIFO overwrite mode is configured by the MCAN\_RXFnC[31] FnOM = 1. When an Rx FIFO full condition is reached (MCAN\_RXFnS[21-16] FnPI = MCAN\_RXFnS[13-8] FnGI) signalled by MCAN\_RXFnS[24] FnF = 1, the next accepted message for the FIFO will overwrite the oldest FIFO message. Put index/Get index are both incremented by one.

In overwrite mode if an Rx FIFO full condition is signalled, reading of the Rx FIFO elements should start at least at get index + 1. The reason for that is, that it might happen, that a received message is written to the Message RAM (Put index) while the Host CPU is reading from the Message RAM (Get index). In this case inconsistent data may be read from the respective Rx FIFO element. The problem is solved by adding an offset to the Get index when reading from the Rx FIFO. The offset depends on how fast the Host CPU accesses the Rx FIFO. [Figure 12-476](#) shows an offset of two with respect to the Get index when reading the Rx FIFO. In this case the two messages stored in element 1 and 2 are lost.



**Figure 12-476. Rx FIFO Overflow Handling**

After reading from the Rx FIFO, the number of the last element read has to be written to the Rx FIFO Acknowledge Index MCAN\_RXFnA[5-0] FnAI. This increments the get index to that element number. In case the Put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (MCAN\_RXFnS[24] FnF = 0).

#### 12.4.4.4.7.3 Dedicated Rx Buffers

The MCAN supports up to 64 dedicated Rx Buffers. The start address of the Rx Buffers section in the Message RAM is configured via MCAN\_RXBC[15-2] RBSA field. To store in an Rx Buffer a Standard or Extended Message ID Filter Element with SFEC/EFEC = 111 and SFID2/EFID2[10-9] = 00 has to be configured (see [Section 12.4.4.4.10.5, Standard Message ID Filter Element](#) and [Section 12.4.4.4.10.6, Extended Message ID Filter Element](#)).

After a received message has been accepted by a filter element, the message is stored into the Rx Buffer in the Message RAM referenced by the filter element (the format is the same as for an Rx FIFO element). In addition the flag MCAN\_IR[19] DRX (Message stored in Dedicated Rx Buffer) is set.

[Table 12-480](#) shows Example Filter Configuration for Rx Buffers.

**Table 12-480. Example Filter Configuration for Rx Buffers**

Filter Element	SFID1[10-0] EFID1[28-0]	SFID2[10-9] EFID2[10-9]	SFID2[5-0] EFID2[5-0]
0	ID message 1	00	00 0000

**Table 12-480. Example Filter Configuration for Rx Buffers (continued)**

Filter Element	SFID1[10-0] EFID1[28-0]	SFID2[10-9] EFID2[10-9]	SFID2[5-0] EFID2[5-0]
1	ID message 2	00	00 0001
2	ID message 3	00	00 0010

After the last word of a matching received message has been written to the Message RAM, the respective New Data flag in register MCAN\_NDAT1/MCAN\_NDAT2 is set. As long as the New Data flag is set, the respective Rx Buffer is locked against updates from received matching frames. The New Data flags have to be reset by the Host CPU by writing a 1 to the respective bit position.

While an Rx Buffer's New Data flag is set, a Message ID Filter Element referencing this specific Rx Buffer will not match, causing the acceptance filtering to continue. Following Message ID Filter Elements may cause the received message to be stored into another Rx Buffer, or into an Rx FIFO, or the message may be rejected, depending on filter configuration.

#### 12.4.4.7.3.1 Rx Buffer Handling

Rx Buffer Handling include the following steps:

- Reset interrupt flag MCAN\_IR[19] DRX
- Read New Data registers
- Read messages from Message RAM
- Reset New Data flags of processed messages

#### 12.4.4.7.4 Debug on CAN Support

Debug DMA is not supported feature. Debug messages can be traced through the RX FIFO (see [Section 12.4.4.7.2](#)).

#### 12.4.4.8 Tx Handling

The Tx Handler is used to handle the Tx requests. It controls the transfer of transmit messages from the dedicated Tx Buffers, the Tx FIFO, and the Tx Queue to the CAN Core, the Tx Event FIFO, and the Put and Get Index operations. The MCAN module supports up to 32 Tx Buffers. These Tx Buffers can be configured as dedicated Tx Buffers, Tx FIFO, or Tx Queue and as combination of dedicated Tx Buffers/Tx FIFO or dedicated Tx Buffers/Tx Queue. For each Tx Buffer element Classical CAN or CAN FD transmission mode can be configured. [Section 12.4.4.10.3](#) describes the Tx Buffer Element. [Table 12-481](#) shows the possible configurations for message transmission.

**Table 12-481. Possible Configurations for Message Transmission**

MCAN_CCCR		Tx Buffer Element		Frame Transmission
MCAN_CCCR[9] BRSE	MCAN_CCCR[8] FDOE	FDF	BRS	
ignored	0	ignored	ignored	Classic CAN
0	1	0	ignored	Classic CAN
0	1	1	ignored	CAN FD without bit rate switching
1	1	0	ignored	Classic CAN
1	1	1	0	CAN FD without bit rate switching
1	1	1	1	CAN FD with bit rate switching

When the Tx Buffer Request Pending MCAN\_TXBRP register is updated, or when a transmission has been started the Tx Handler starts scanning to check for the highest priority pending Tx request. The Tx Buffer with lowest Message ID has highest priority.

### Note

AUTOSAR requires at least three Tx Queue Buffers and support of transmit cancellation.

#### 12.4.4.4.8.1 Transmit Pause

The transmit pause feature is intended for use in CAN networks where the CAN Message IDs are specific and cannot easily be changed. These Message IDs may have a higher priority than other defined Message IDs, while in a specific application their relative priority should be inverse. This allows for a case where one ECU sends a burst of CAN messages that cause another ECU's CAN messages to be delayed (paused).

The transmit pause feature is enabled by the MCAN\_CCCR[14] TXP bit. By default this bit is disabled (MCAN\_CCCR[14] TXP = 0). Each time after successfully transmitted message, a pause for two CAN bit times occurs before the start of the next transmission. This allows the other CAN nodes in the network to transmit messages even if their Message IDs have lower priority.

#### 12.4.4.4.8.2 Dedicated Tx Buffers

Dedicated Tx Buffers are intended for message transmission under complete control of the Host CPU.

There are two options:

- Each dedicated Tx Buffer is configured with a specific Message ID.
- Two or more dedicated Tx Buffers are configured with the same Message ID. In this case the Tx Buffer with the lowest buffer number is transmitted first.

After the data section has been updated, a transmission is requested by an Add Request. This is done via the MCAN\_TXBAR[x]ARn bit (where x = 0 - 31). The requested messages arbitrate internally with messages from an optional Tx FIFO or Tx Queue and externally with messages on the CAN bus, and are sent out according to their Message ID.

Table 12-482 shows Tx Buffer/Tx FIFO/Tx Queue Element Size. A Dedicated Tx Buffer allocates Element Size 32-bit words in the Message RAM. The start address of a dedicated Tx Buffer in the Message RAM is calculated by adding transmit buffer index from 0 to 31 (MCAN\_TXFQS[20-16] TFQPI) × Element Size to the Tx Buffer Start Address MCAN\_TXBC[15-2] TBSA field.

**Table 12-482. Tx Buffer/Tx FIFO/Tx Queue Element Size**

MCAN_TXESC[2-0] TBDS	Data Field [bytes]	Element Size [RAM words]
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

#### 12.4.4.4.8.3 Tx FIFO

Tx FIFO mode is configured by setting bit MCAN\_TXBC[30] TFQM = 0. The stored in the Tx FIFO messages are transmitted starting with the message referenced by the Get Index MCAN\_TXFQS[12-8] TFGI field. After each transmission the Get Index is incremented until the Tx FIFO is empty. The Tx FIFO Free Level MCAN\_TXFQS[5-0] TFFL field indicates the number of the available free Tx FIFO elements. The Tx FIFO allows transmission of messages with the same Message ID from different Tx Buffers in the order these messages have been written to the Tx FIFO.

New transmit messages have to be written to the Tx FIFO starting with the Tx Buffer referenced by the Put Index MCAN\_TXFQS[20-16] TFQPI field. After each Add Request (MCAN\_TXBAR[x] ARn = 1) the

Put Index is incremented to the next free Tx FIFO element. When the Put Index reaches the Get Index (MCAN\_TXFQS[20-16] TFQPI = MCAN\_TXFQS[12-8] TFGI), Tx FIFO Full condition is signalled by bit MCAN\_TXFQS[21] TFQF = 1. In this case no further messages should be written to the Tx FIFO until the next message has been transmitted and the Get Index has been incremented.

The number of requested Tx buffers should not exceed the number of free Tx Buffers as indicated by the Tx FIFO Free Level MCAN\_TXFQS[5-0] TFFL field.

In case a transmission request for the Tx Buffer referenced by the Get Index is cancelled, the Get Index is incremented to the next Tx Buffer with pending transmission request and the Tx FIFO Free Level MCAN\_TXFQS[5-0] TFFL field is recalculated. In case transmission cancellation is applied to any other Tx Buffer - the Get Index and the FIFO Free Level remain unchanged.

A Tx FIFO element allocates Element Size 32-bit words in the Message RAM (see [Table 12-482](#)). The start address of the next available (free) Tx FIFO Buffer is calculated by adding Tx FIFO/Queue Put Index MCAN\_TXFQS[20-16] TFQPI (from 0 to 31) × Element Size to the Tx Buffer Start Address MCAN\_TXBC[15-2] TBSA field.

#### 12.4.4.4.8.4 Tx Queue

Tx Queue mode is configured by setting bit MCAN\_TXBC[30] TFQM = 1. The stored in the Tx Queue messages are transmitted starting with the highest priority message (lowest Message ID). In case two or more Queue Buffers are configured with the same Message ID, the Queue Buffer with the lowest buffer number is transmitted first.

New transmit messages have to be written to the Tx FIFO starting with the Tx Buffer referenced by the Put Index MCAN\_TXFQS[20-16] TFQPI field. Each Add Request cyclically increments the Put Index to the next free Tx Buffer. In case of Tx Queue Full condition (MCAN\_TXFQS[21] TFQF = 1), the Put Index is not valid and no further message should be written to the Tx Queue until at least one of the requested messages has been sent out or a pending transmission request has been cancelled.

The application may use the MCAN\_TXBRP register instead of the Put Index and may place messages to any Tx Buffer without pending transmission request.

A Tx Queue Buffer allocates Element Size 32-bit words in the Message RAM (see [Table 12-482](#)). The start address of the next available (free) Tx Queue Buffer is calculated by adding Tx FIFO/Queue Put Index MCAN\_TXFQS[20-16] TFQPI (from 0 to 31) × Element Size to the Tx Buffer Start Address MCAN\_TXBC[15-2] TBSA field.

#### 12.4.4.4.8.5 Mixed Dedicated Tx Buffers/Tx FIFO

For this combination the Tx Buffers section in the Message RAM is separated in two parts:

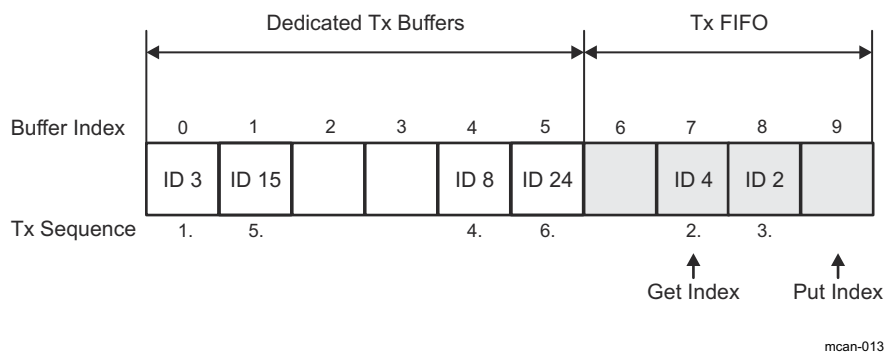
- Dedicated Tx Buffers: the number of Dedicated Tx Buffers is configured by the MCAN\_TXBC[21-16] NDTB field
- Tx FIFO: the number of Tx Buffers assigned to the Tx FIFO is configured by the MCAN\_TXBC[29-24] TFQS field

If the MCAN\_TXBC[29-24] TFQS field is empty (zero) - only Dedicated Tx Buffers are used.

Tx prioritization:

- Scan Dedicated Tx Buffers and oldest pending Tx FIFO Buffer (referenced by the MCAN\_TXFQS[12-8] TFGI field)
- Buffer with lowest Message ID gets highest priority and is transmitted next

[Figure 12-477](#) shows Mixed Dedicated Tx Buffers/Tx FIFO example.



**Figure 12-477. Mixed Dedicated Tx Buffers/Tx FIFO (example)**

#### 12.4.4.4.8.6 Mixed Dedicated Tx Buffers/Tx Queue

For this combination the Tx Buffers section in the Message RAM is separated in two parts:

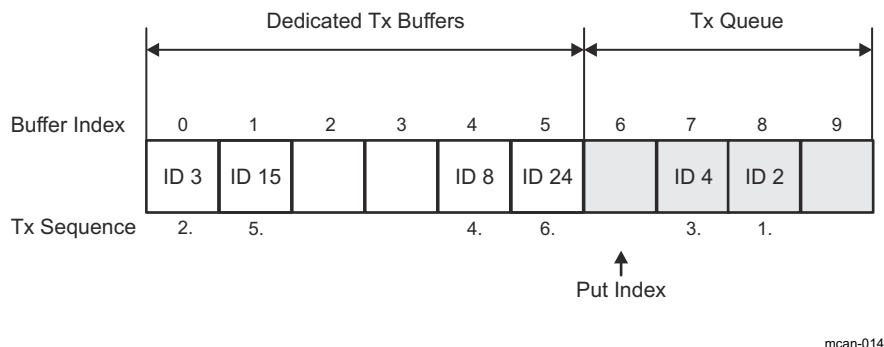
- Dedicated Tx Buffers: the number of Dedicated Tx Buffers is configured by the MCAN\_TXBC[21-16] NDTB field
- Tx Queue: the number of Tx Buffers assigned to the Tx Queue is configured by the MCAN\_TXBC[29-24] TFQS field

If MCAN\_TXBC[29-24] TFQS field is empty (zero) - only Dedicated Tx Buffers are used.

Tx prioritization:

- Scan all Tx Buffers with activated transmission request
- Tx Buffer with lowest Message ID gets highest priority and is transmitted next

Figure 12-478 shows Mixed Dedicated Tx Buffers/Tx Queue example.



**Figure 12-478. Mixed Dedicated Tx Buffers/Tx Queue (example)**

#### 12.4.4.4.8.7 Transmit Cancellation

This feature is especially intended for gateway and AUTOSAR based applications. The Host CPU can cancel a requested transmission from a dedicated Tx Buffer or a Tx Queue Buffer by setting bit MCAN\_TXBCR[n] CRn = 1 (where n = 0 - 31). The corresponding bit position n is equivalent to the number of the Tx Buffer.

Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signalled by setting the corresponding bit of the MCAN\_TXBCF register (MCAN\_TXBCF[n] CFn = 1).

If transmission from a Tx Buffer is already ongoing and a transmit cancellation is requested, the corresponding MCAN\_TXBRP[n] TRPn bit remains set as long as the transmission is in progress. If the transmission was successful, the corresponding MCAN\_TXBTO[n] TOn and MCAN\_TXBCF[n] CFn bits are set. If the transmission was not successful, only the corresponding bit MCAN\_TXBCF[n] CFn = 1.



---

**Note**

If pending transmission is cancelled immediately before this transmission could have been started, a short time window occurs where no transmission is started even if another message is also pending in this node. This may enable another node to transmit a message which may have a lower priority than the second message in this node.

---

**12.4.4.4.8 Tx Event Handling**

To support Tx Event Handling the Message RAM has implemented a Tx Event FIFO section. Up to 32 Tx Event FIFO elements can be configured. [Section 12.4.4.4.10.4](#) describes the Tx Event FIFO element. After message transmission on the CAN bus, Message ID and Timestamp are stored in a Tx Event FIFO element. To link a Tx Event to a Tx Event FIFO element, the Message Marker from the transmitted Tx Buffer is copied into the Tx Event FIFO element.

A Tx Event FIFO full condition is signalled by the MCAN\_IR[14] TEFF bit. In this case no further elements are written to the Tx Event FIFO until at least one element has been read out and the Tx Event FIFO Get Index has been incremented (MCAN\_TXEFS[12-8] EFGI). In case a Tx Event occurs while the Tx Event FIFO is full, this event is rejected and interrupt flag MCAN\_IR[15] TEFL bit is set.

The Tx Event FIFO watermark can be configured to avoid a Tx Event FIFO overflow. When the Tx Event FIFO fill level reaches the Tx Event FIFO watermark configured by the MCAN\_TXEFC[29-24] EFWM field, interrupt flag MCAN\_IR[13] TEFW is set. When reading from the Tx Event FIFO, two times the Tx Event FIFO Get Index MCAN\_TXEFS[12-8] EFGI field has to be added to the Tx Event FIFO start address MCAN\_TXEFC[15-2] EFSA field.

**12.4.4.4.9 FIFO Acknowledge Handling**

The Get Indices of the two Rx FIFOs (Rx FIFO 0 or Rx FIFO 1) and the Tx Event FIFO are controlled by writing to the corresponding FIFO Acknowledge Index (see MCAN\_RXF0A, MCAN\_RXF1A, and MCAN\_TXEFA). Writing to the FIFO Acknowledge Index will set the FIFO Get Index to the FIFO Acknowledge Index plus one and thereby updates the FIFO Fill Level.

There are two use cases:

- A single element has been read from the FIFO: the Get Index value is written to the FIFO Acknowledge Index.
- A sequence of elements has been read from the FIFO: the Get Index value (Index of the last element read) is written to the FIFO Acknowledge Index at the end of that read sequence.

The Host CPU has free access to the Message RAM. The special care has to be taken when reading FIFO elements in an arbitrary order (Get Index not considered). This can be useful when reading a High Priority Message from one of the two Rx FIFOs. In this case the FIFO's Acknowledge Index should not be written because this would set the Get Index to a wrong position and also changes the FIFO's Fill Level. In this case some of the older FIFO elements would be lost.

---

**Note**

The application has to ensure that a valid value is written to the FIFO Acknowledge Index. The MCAN module does not check for erroneous values.

---

#### 12.4.4.4.10 Message RAM

The MCAN module has implemented Message RAM. The main purpose of the Message RAM is to store:

- Receive Messages
- Transmit Messages
- Tx Event Elements
- Message ID Filter Elements

##### 12.4.4.4.10.1 Message RAM Configuration

The MCAN module is configured to allocate 4352 words in the Message RAM. The Message RAM has a width of 32 bits.

[Table 12-483](#) presents the Message RAM Address Range for all device MCAN instances.

**Table 12-483. Message RAM Address Range**

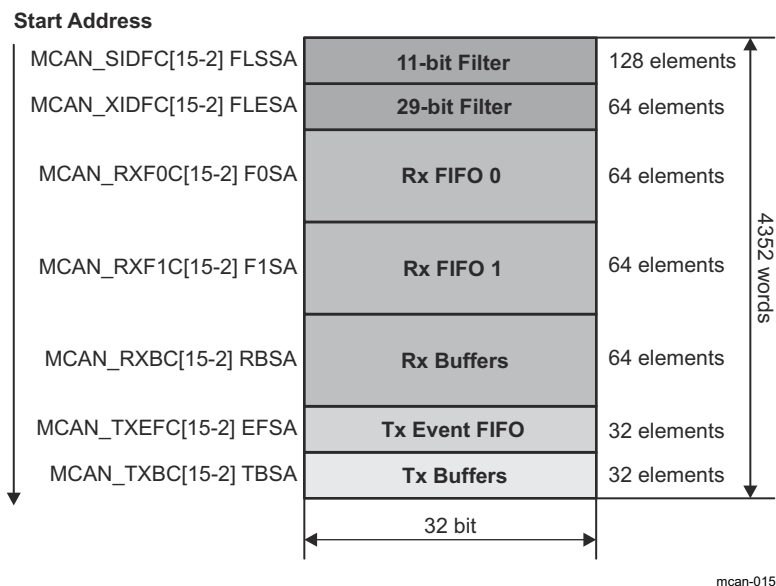
Module Instance	Region Name	Address Range		Size
		Start	End	
MCU_MCAN0	MCU_MCAN0_MSGMEM_RAM	4050 0000h	4050 7FFFh	32 KB
MCU_MCAN1	MCU_MCAN1_MSGMEM_RAM	4054 0000h	4054 7FFFh	32 KB
MCAN0	MCAN0_MSGMEM_RAM	0270 8000h	0270 FFFFh	32 KB
MCAN1	MCAN1_MSGMEM_RAM	0271 8000h	0271 FFFFh	32 KB
MCAN2	MCAN2_MSGMEM_RAM	0272 8000h	0272 FFFFh	32 KB
MCAN3	MCAN3_MSGMEM_RAM	0273 8000h	0273 FFFFh	32 KB
MCAN4	MCAN4_MSGMEM_RAM	0274 8000h	0274 FFFFh	32 KB
MCAN5	MCAN5_MSGMEM_RAM	0275 8000h	0275 FFFFh	32 KB
MCAN6	MCAN6_MSGMEM_RAM	0276 8000h	0276 FFFFh	32 KB
MCAN7	MCAN7_MSGMEM_RAM	0277 8000h	0277 FFFFh	32 KB
MCAN8	MCAN8_MSGMEM_RAM	0278 8000h	0278 FFFFh	32 KB
MCAN9	MCAN9_MSGMEM_RAM	0279 8000h	0279 FFFFh	32 KB
MCAN10	MCAN10_MSGMEM_RAM	027A 8000h	027A FFFFh	32 KB
MCAN11	MCAN11_MSGMEM_RAM	027B 8000h	027B FFFFh	32 KB
MCAN12	MCAN12_MSGMEM_RAM	027C 8000h	027C FFFFh	32 KB
MCAN13	MCAN13_MSGMEM_RAM	027D 8000h	027D FFFFh	32 KB

The Message RAM is capable to include each of the sections listed in [Figure 12-479](#). It is not necessary to configure each of the sections (a section in the Message RAM may be 0) and there is not restriction with respect to the sequence of the sections. For parity checking or ECC a respective number of bits has to be added to each word.

When the MCAN module addresses the Message RAM it addresses 32-bit words. The start addresses are configurable and they are 32-bit word addresses.

The element size can be configured for:

- Rx FIFO 0 via the MCAN\_RXESC[2-0] F0DS field
- Rx FIFO 1 via the MCAN\_RXESC[6-4] F1DS field
- Rx Buffers via the MCAN\_RXESC[10-8] RBDS field
- Tx Buffers via the MCAN\_TXESC[2-0] TBDS field



**Figure 12-479. Message RAM Configuration**

The Host CPU configures the following information in the Message RAM:

- Start addresses of the memory sections
- Number of elements in each section
- The size of the elements in some sections

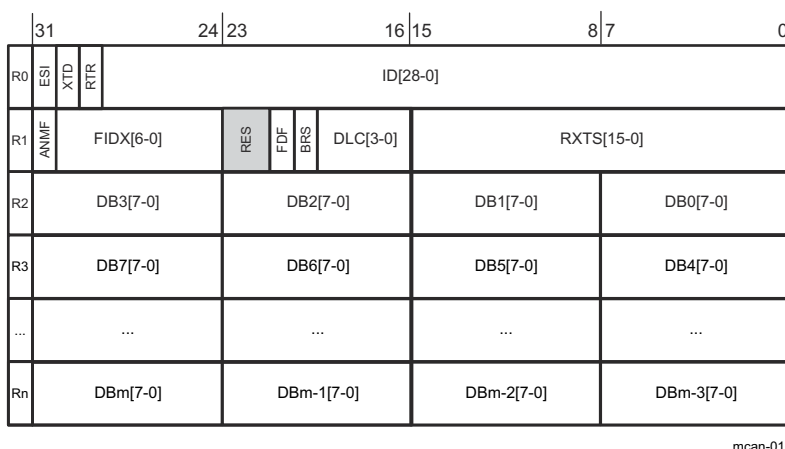
**Note**

The MCAN module does not check for errors in the Message RAM configuration. The configuration of the start addresses of the different sections and the number of elements of each section has to be done carefully. This will prevent falsification or loss of data.

**12.4.4.4.10.2 Rx Buffer and FIFO Element**

Up to 64 Rx Buffers and two Rx FIFOs can be configured in the Message RAM. Each Rx FIFO section can be configured to store up to 64 received messages. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via the MCAN\_RXESC register.

Figure 12-480 shows Rx Buffer/Rx FIFO element structure.



**Figure 12-480. Rx Buffer/Rx FIFO Element Structure**

Table 12-484 shows Rx Buffer/Rx FIFO element field descriptions.

**Table 12-484. Rx Buffer/Rx FIFO Element Field Descriptions**

Word	Bits	Field Name	Description
R0	31	ESI	Error State Indicator <ul style="list-style-type: none"> <li>0h = Transmitting node is error active</li> <li>1h = Transmitting node is error passive</li> </ul>
	30	XTD	Extended Identifier Signals to the Host CPU whether the received frame has a standard or extended identifier. <ul style="list-style-type: none"> <li>0h = 11-bit standard identifier</li> <li>1h = 29-bit extended identifier</li> </ul>
	29	RTR	Remote Transmission Request Signals to the Host CPU whether the received frame is a data frame or a remote frame. <ul style="list-style-type: none"> <li>0h = Received frame is a data frame</li> <li>1h = Received frame is a remote frame</li> </ul> <p><b>Note:</b> There are no remote frames in CAN FD format. In case a CAN FD frame was received (FDF = 1), RTR bit reflects the state of the reserved r1 bit (RES[23]).</p>
	28-0	ID[28-0]	Identifier Standard or extended identifier depending on XTD bit. A standard identifier is stored into ID[28-18].

**Table 12-484. Rx Buffer/Rx FIFO Element Field Descriptions (continued)**

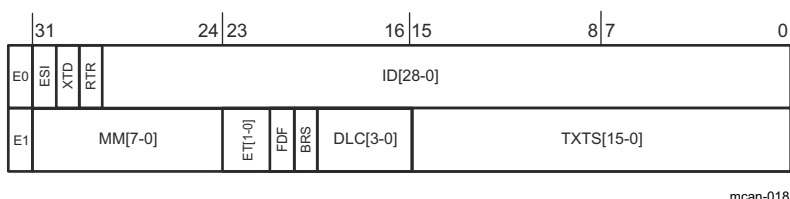
Word	Bits	Field Name	Description
R1	31	ANMF	Accepted Non-matching Frame Acceptance of non-matching frames may be enabled via the MCAN_GFC[5-4] ANFS and MCAN_GFC[3-2] ANFE fields. <ul style="list-style-type: none"> <li>0h = Received frame matching filter index FIDX field</li> <li>1h = Received frame did not match any Rx filter element</li> </ul>
	30-24	FIDX[6-0]	Filter Index 0h-7Fh (0-127): Index of matching Rx acceptance filter element (invalid if ANMF = 1). Range is 0 to MCAN_SIDFC[23-16] LSS - 1 respectively MCAN_XIDFC[22-16] LSE - 1.
	23-22	RES	Reserved
	21	FDF	FD Format <ul style="list-style-type: none"> <li>0h = Standard frame format</li> <li>1h = CAN FD frame format (new DLC-coding and CRC)</li> </ul>
	20	BRS	Bit Rate Switch <ul style="list-style-type: none"> <li>0h = Frame received without bit rate switching</li> <li>1h = Frame received with bit rate switching</li> </ul>
	19-16	DLC[3-0]	Data Length Code <ul style="list-style-type: none"> <li>0h-8h (0-8) = CAN + CAN FD: received frame has 0-8 data bytes</li> <li>9h-Fh (9-15) = CAN: received frame has 8 data bytes</li> <li>9h-Fh (9-15) = CAN FD: received frame has 12/16/20/24/32/48/64 data bytes</li> </ul>
R2	15-0	RXTS[15-0]	Rx Timestamp Timestamp Counter value captured on start of frame reception. Resolution depending on configuration of the Timestamp Counter Prescaler MCAN_TSCC[19-16] TCP.
	31-24	DB3[7-0]	Data Byte 3
	23-16	DB2[7-0]	Data Byte 2
	15-8	DB1[7-0]	Data Byte 1
R3	7-0	DB0[7-0]	Data Byte 0
	31-24	DB7[7-0]	Data Byte 7
	23-16	DB6[7-0]	Data Byte 6
	15-8	DB5[7-0]	Data Byte 5
Rn	7-0	DB4[7-0]	Data Byte 4
	...	...	...
	31-24	DBm[7-0]	Data Byte m
	23-16	DBm-1[7-0]	Data Byte m-1
Rn	15-8	DBm-2[7-0]	Data Byte m-2
	7-0	DBm-3[7-0]	Data Byte m-3

**Note:** Depending on the configuration of the element size (MCAN\_RXESC), between two and sixteen 32-bit words (Rn = 3-17) are used for storage of a CAN message's data field.

### 12.4.4.4.10.3 Tx Buffer Element

The Tx Buffers section can be configured to hold dedicated Tx Buffers as well as a Tx FIFO/Tx Queue. In case that the Tx Buffers section is shared by dedicated Tx buffers and a Tx FIFO/Tx Queue, the dedicated Tx Buffers start at the beginning of the Tx Buffers section followed by the buffers assigned to the Tx FIFO or Tx Queue. The Tx Handler makes difference between dedicated Tx Buffers and Tx FIFO/Tx Queue via the MCAN\_TXBC[29-24] TFQS and MCAN\_TXBC[21-16] NDTB fields. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via the MCAN\_TXESC register.

Figure 12-481 shows Tx Buffer element structure.



mcan-018

**Figure 12-481. Tx Buffer Element Structure**

Table 12-485 shows Tx Buffer element field descriptions.

**Table 12-485. Tx Buffer Element Field Descriptions**

Word	Bits	Field Name	Description
T0	31	ESI	<p>Error State Indicator</p> <ul style="list-style-type: none"> <li>0h = ESI bit in CAN FD format depends only on error passive flag</li> <li>1h = ESI bit in CAN FD format transmitted recessive</li> </ul> <p><b>Note:</b> The ESI bit of the transmit buffer is or'ed with the error passive flag to decide the value of the ESI bit in the transmitted CAN FD frame. As required by the CAN FD protocol specification, an error active node may optionally transmit the ESI bit recessive, but an error passive node will always transmit the ESI bit recessive.</p>
	30	XTD	<p>Extended Identifier</p> <ul style="list-style-type: none"> <li>0h = 11-bit standard identifier</li> <li>1h = 29-bit extended identifier</li> </ul>
	29	RTR	<p>Remote Transmission Request</p> <ul style="list-style-type: none"> <li>0h = Transmit data frame</li> <li>1h = Transmit remote frame</li> </ul> <p><b>Note:</b> When RTR = 1, the MCAN module transmits a remote frame according to ISO11898-1:2015, even if the MCAN_CCCR[8] FDOE bit enables the transmission in CAN FD format.</p>
	28-0	ID[28-0]	<p>Identifier</p> <p>Standard or extended identifier depending on XTD bit. A standard identifier has to be written to ID[28-18].</p>

**Table 12-485. Tx Buffer Element Field Descriptions (continued)**

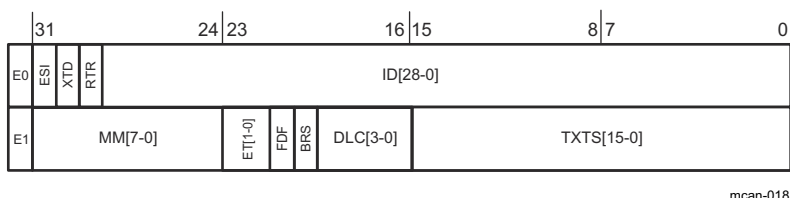
Word	Bits	Field Name	Description
T1	31-24	MM[7-0]	Message Marker Written by Host CPU during Tx Buffer configuration. Copied into Tx Event FIFO element for identification of Tx message status (see also MM[7-0] field in <a href="#">Table 12-486</a> ).
	23	EFC	Event FIFO Control <ul style="list-style-type: none"> <li>0h = Don't store Tx events</li> <li>1h = Store Tx events</li> </ul>
	22	RES	Reserved
	21	FDF	FD Format <ul style="list-style-type: none"> <li>0h = Frame transmitted in Classic CAN format</li> <li>1h = Frame transmitted in CAN FD format</li> </ul>
	20	BRS	Bit Rate Switch <ul style="list-style-type: none"> <li>0h = CAN FD frames transmitted without bit rate switching</li> <li>1h = CAN FD frames transmitted with bit rate switching</li> </ul> <p><b>Note:</b> ESI, FDF, and BRS bits are only evaluated when CAN FD operation is enabled via the MCAN_CCCR[8] FDOE bit. BRS bit is only evaluated when in addition the MCAN_CCCR[9] BRSE = 1.</p>
	19-16	DLC[3-0]	Data Length Code <ul style="list-style-type: none"> <li>0h-8h (0-8) = CAN + CAN FD: transmit frame has 0-8 data bytes</li> <li>9h-Fh (9-15) = CAN: transmit frame has 8 data bytes</li> <li>9h-Fh (9-15) = CAN FD: transmit frame has 12/16/20/24/32/48/64 data bytes</li> </ul>
	15-0	RES	Reserved
T2	31-24	DB3[7-0]	Data Byte 3
	23-16	DB2[7-0]	Data Byte 2
	15-8	DB1[7-0]	Data Byte 1
	7-0	DB0[7-0]	Data Byte 0
T3	31-24	DB7[7-0]	Data Byte 7
	23-16	DB6[7-0]	Data Byte 6
	15-8	DB5[7-0]	Data Byte 5
	7-0	DB4[7-0]	Data Byte 4
...	...	...	...
Tn	31-24	DBm[7-0]	Data Byte m
	23-16	DBm-1[7-0]	Data Byte m-1
	15-8	DBm-2[7-0]	Data Byte m-2
	7-0	DBm-3[7-0]	Data Byte m-3

**Note:** Depending on the configuration of the element size (MCAN\_TXESC), between two and sixteen 32-bit words (Tn = 3-17) are used for storage of a CAN message's data field.

#### 12.4.4.4.10.4 Tx Event FIFO Element

Each element stores information about transmitted messages. By reading the Tx Event FIFO the Host CPU gets this information in the order the messages were transmitted. Status information about the Tx Event FIFO can be obtained from the MCAN\_TXEFS register.

Figure 12-482 shows Tx Event FIFO element structure.



**Figure 12-482. Tx Event FIFO Element Structure**

Table 12-486 shows Tx Event FIFO element field descriptions.

**Table 12-486. Tx Event FIFO Element Field Descriptions**

Word	Bits	Field Name	Description
E0	31	ESI	Error State Indicator <ul style="list-style-type: none"> <li>0h = Transmitting node is error active</li> <li>1h = Transmitting node is error passive</li> </ul>
	30	XTD	Extended Identifier <ul style="list-style-type: none"> <li>0h = 11-bit standard identifier</li> <li>1h = 29-bit extended identifier</li> </ul>
	29	RTR	Remote Transmission Request <ul style="list-style-type: none"> <li>0h = Data frame transmitted</li> <li>1h = Remote frame transmitted</li> </ul>
	28-0	ID[28-0]	Identifier <p>Standard or extended identifier depending on XTD bit. A standard identifier has to be written to ID[28-18].</p>



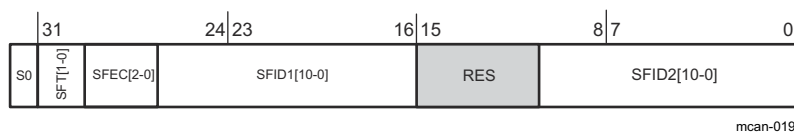
**Table 12-486. Tx Event FIFO Element Field Descriptions (continued)**

Word	Bits	Field Name	Description
E1	31-24	MM[7-0]	Message Marker Copied from Tx Buffer into Tx Event FIFO element for identification of Tx message status (see also MM[7-0] field in <a href="#">Table 12-485</a> ).
	23-22	ET[1-0]	Event Type <ul style="list-style-type: none"> <li>0h = Reserved</li> <li>1h = Tx event</li> <li>2h = Transmission in spite of cancellation (always set for transmissions in DAR mode)</li> <li>3h = Reserved</li> </ul>
	21	FDF	FD Format <ul style="list-style-type: none"> <li>0h = Standard frame format</li> <li>1h = CAN FD frame format (new DLC-coding and CRC)</li> </ul>
	20	BRS	Bit Rate Switch <ul style="list-style-type: none"> <li>0h = Frame transmitted without bit rate switching</li> <li>1h = Frame transmitted with bit rate switching</li> </ul>
	19-16	DLC[3-0]	Data Length Code <ul style="list-style-type: none"> <li>0h-8h (0-8) = CAN + CAN FD: frame with 0-8 data bytes transmitted</li> <li>9h-Fh (9-15) = CAN: frame with 8 data bytes transmitted</li> <li>9h-Fh (9-15) = CAN FD: frame with 12/16/20/24/32/48/64 data bytes transmitted</li> </ul>
	15-0	TXTS[15-0]	Tx Timestamp Timestamp Counter value captured on start of frame transmission. Resolution depending on configuration of the Timestamp Counter Prescaler MCAN_TSCC[19-16] TCP field.

#### 12.4.4.4.10.5 Standard Message ID Filter Element

Up to 128 filter elements can be configured for 11-bit standard IDs. When accessing a Standard Message ID Filter element, its address is the Filter List Standard Start Address MCAN\_SIDFC[15-2] FLSSA field plus the index of the filter element (0-127).

[Figure 12-483](#) shows Standard Message ID Filter element structure.


**Figure 12-483. Standard Message ID Filter Element Structure**

[Table 12-487](#) shows Standard Message ID Filter element field descriptions.

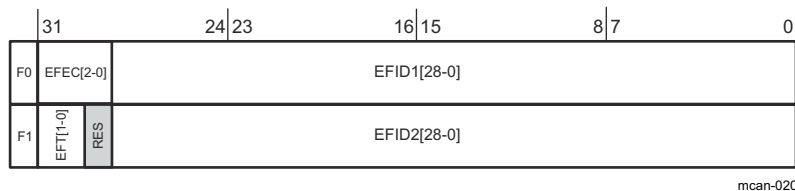
**Table 12-487. Standard Message ID Filter Element Field Descriptions**

Word	Bits	Field Name	Description
S0	31-30	SFT[1-0]	<p>Standard Filter Type</p> <ul style="list-style-type: none"> <li>0h = Range filter from SFID1 to SFID2 (SFID2 ≥ SFID1)</li> <li>1h = Dual ID filter for SFID1 or SFID2</li> <li>2h = Classic filter: SFID1 = filter; SFID2 = mask</li> <li>3h = Filter element disabled</li> </ul> <p><b>Note:</b> With SFT = 11 the filter element is disabled and the acceptance filtering continues (same behaviour as with SFEC = 000)</p>
	29-27	SFEC[2-0]	<p>Standard Filter Element Configuration</p> <p>All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC = 100, 101, or 110 a match sets interrupt flag MCAN_IR[8]HPM and, if enabled, an interrupt is generated. In this case the MCAN_HPMS register is updated with the status of the priority match.</p> <ul style="list-style-type: none"> <li>0h = Disable filter element</li> <li>1h = Store in Rx FIFO 0 if filter matches</li> <li>2h = Store in Rx FIFO 1 if filter matches</li> <li>3h = Reject ID if filter matches</li> <li>4h = Set priority if filter matches</li> <li>5h = Set priority and store in FIFO 0 if filter matches</li> <li>6h = Set priority and store in FIFO 1 if filter matches</li> <li>7h = Store into Rx Buffer , configuration of SFT[1-0] ignored</li> </ul>
	26-16	SFID1[10-0]	<p>Standard Filter ID 1</p> <p>When filtering for Rx Buffers this field defines the ID of a standard message to be stored. The received identifiers must match exactly, no masking mechanism is used.</p>
	15-11	RES	Reserved
		SFID2[10-0]	<p>Standard Filter ID 2</p> <p>This bit field has a different meaning depending on the configuration of SFEC:</p> <ul style="list-style-type: none"> <li>1) SFEC = 001 - 110 Second ID of standard ID filter element</li> <li>2) SFEC = 111 Filter for Rx Buffers</li> </ul>
	10-0	SFID2[10-9]	<p>This field is decides whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence.</p> <ul style="list-style-type: none"> <li>0h = Store message into an Rx Buffer</li> <li>1h = Debug Message A</li> <li>2h = Debug Message B</li> <li>3h = Debug Message C</li> </ul> <p><b>Note:</b> Debug feature is not supported.</p>
		SFID2[8-6]	<p>This field is used to control the filter event pins at the Extension Interface. A one at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one MCAN_ICKL period in case the filter matches.</p> <p><b>Note:</b> Only three filter event pins are supported.</p>
		SFID2[5-0]	<p>This field defines the offset to the Rx Buffer Start Address</p>

#### 12.4.4.4.10.6 Extended Message ID Filter Element

Up to 64 filter elements can be configured for 29-bit extended IDs. When accessing an Extended Message ID Filter element, its address is the Filter List Extended Start Address MCAN\_XIDFC[15-2] FLESA field plus two times the index of the filter element (0-63).

Figure 12-484 shows Extended Message ID Filter element structure.



**Figure 12-484. Extended Message ID Filter Element Structure**

Table 12-488 shows Extended Message ID Filter element field descriptions.

**Table 12-488. Extended Message ID Filter Element Field Descriptions**

Word	Bits	Field Name	Description
F0	31-29	EFEC[2-0]	<p>Extended Filter Element Configuration</p> <p>All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC = 100, 101, or 110 a match sets interrupt flag MCAN_IR[8]HPM and, if enabled, an interrupt is generated. In this case the MCAN_HPMS register is updated with the status of the priority match.</p> <ul style="list-style-type: none"> <li>0h = Disable filter element</li> <li>1h = Store in Rx FIFO 0 if filter matches</li> <li>2h = Store in Rx FIFO 1 if filter matches</li> <li>3h = Reject ID if filter matches</li> <li>4h = Set priority if filter matches</li> <li>5h = Set priority and store in FIFO 0 if filter matches</li> <li>6h = Set priority and store in FIFO 1 if filter matches</li> <li>7h = Store into Rx Buffer or as debug message, configuration of EFT[1-0] ignored</li> </ul>
	28-0	EFID1[28-0]	<p>Extended Filter ID 1</p> <p>First ID of extended ID filter element.</p> <p>When filtering for Rx Buffers this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only XIDAM masking mechanism (see <a href="#">Section 12.4.4.4.7.1.5, Extended Message ID Filtering</a>) is used.</p>

**Table 12-488. Extended Message ID Filter Element Field Descriptions (continued)**

Word	Bits	Field Name	Description
F1	31-30	EFT[1-0]	Extended Filter Type <ul style="list-style-type: none"> <li>0h = Range filter from EFID1 to EFID2 (EFID2 ≥ EFID1)</li> <li>1h = Dual ID filter for EFID1 or EFID2</li> <li>2h = Classic filter: EFID1 = filter, EFID2 = mask</li> <li>3h = Range filter from EFID1 to EFID2 (EFID2 ≥ EFID1), XIDAM mask not applied</li> </ul>
	29	RES	Reserved
		EFID2[28-0]	Extended Filter ID 2 This bit field has a different meaning depending on the configuration of EFEC: <ul style="list-style-type: none"> <li>1) EFEC = 001 - 110 Second ID of extended ID filter element</li> <li>2) EFEC = 111 Filter for Rx Buffers</li> </ul>
	28-0	EFID2[10-9]	This field decides whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence. <ul style="list-style-type: none"> <li>0h = Store message into an Rx Buffer</li> <li>1h = Debug Message A</li> <li>2h = Debug Message B</li> <li>3h = Debug Message C</li> </ul> <b>Note:</b> Debug feature is not supported.
		EFID2[8-6]	This field is used to control the filter event pins at the Extension Interface. A one at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one MCAN_ICKL period in case the filter matches. <b>Note:</b> Only three filter event pins are supported.
		EFID2[5-0]	This field defines the offset to the Rx Buffer Start Address MCAN_RXBC[15-2] RBSA field for storage of a matching message.

## 12.5 Audio Interfaces

## 12.5.1 Audio Tracking Logic (ATL)

This chapter describes the integration of the Audio Tracking Logic (ATL) subsystem in the device.

### 12.5.1.1 ATL Overview

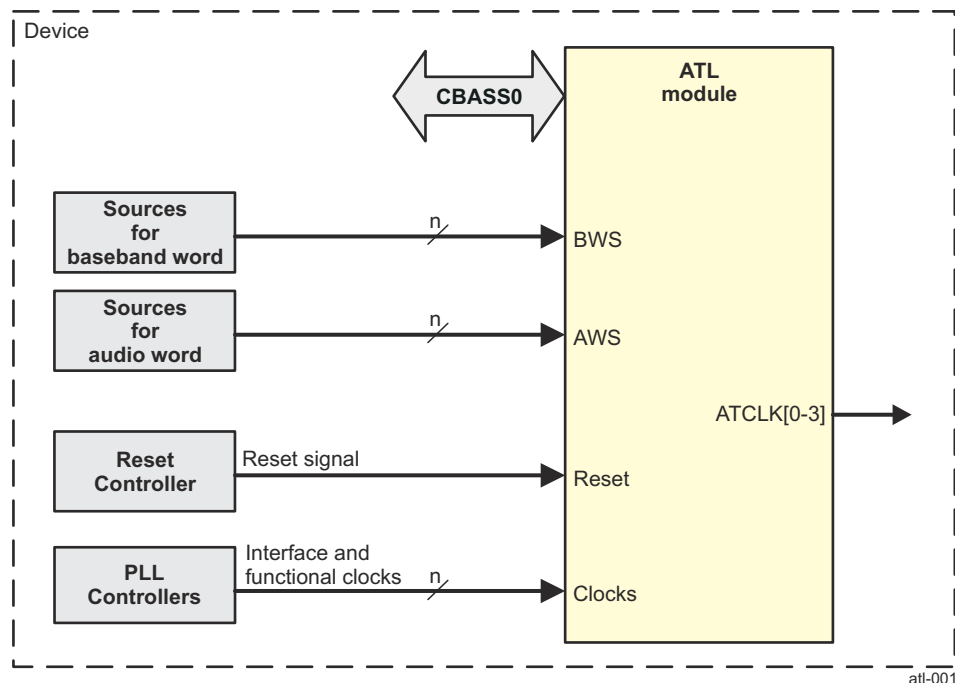
The Audio Tracking Logic (ATL) is used by HD Radio™ applications to synchronize the digital audio output to the baseband clock. This same IP can also be used generically to track errors between two reference signals (such as frame syncs) and generate a modulated clock output (using software-controlled cycle stealing) which averages to some desired frequency. This process can be used as a hardware assist for asynchronous sample rate conversion algorithms.

Table 12-489 shows ATL module allocation across device domains.

**Table 12-489. ATL Allocation Across Device Domains**

Instance	Domain		
	WKUP	MCU	MAIN
ATL	-	-	✓

Figure 12-485 shows the ATL module overview.



**Figure 12-485. ATL Module Overview**

#### 12.5.1.1.1 ATL Features Overview

The ATL module supports the following features:

- One ATL module, containing four ATL instances, for HD Radio support and asynchronous sample rate conversion assistance
- Each instance tracks the time error between two syncs (local audio word select [AWS] and baseband word select [BWS])
- Each instance selects between 28 mux choices for BWS and 30 mux choices for AWS.
- Each instance generates modulated ATCLK clock signals with software-initiated pulse stealing.

The intention is to use these clocks to derive audio output bit clock on MCASP

- Selection between ATL\_VCLK clock or functional ATL\_PCLK to run error counting timers and to derive modulated ATCLK clock outputs.
- Clock and reset management: Receives clock and reset signals from the device PSC module. The ATL module receives hardware reset.
- Power management: The ATL belongs to the PD0 power domain.
- Local software reset

#### **12.5.1.1.2 ATL Not Supported Features**

The ATL module not supports the following features:

- Smart ATL\_CLK\_STOP
- ATL\_MOD\_OUT[3-0] clock outputs

## 12.5.2 Multichannel Audio Serial Port (MCASP)

This section describes the Multichannel Audio Serial Port (MCASP).

### 12.5.2.1 MCASP Overview

This section introduces the Multichannel Audio Serial Port (MCASP) module and describes its main functions and connections in the device.

The MCASP functions as a general-purpose audio serial port are optimized to the requirements of various audio applications. The MCASP module can operate in both transmit and receive modes. The MCASP is useful for time-division multiplexed (TDM) stream, Inter-IC Sound (I2S) protocols reception and transmission as well as for an inter-component digital audio interface transmission (DIT). The MCASP has the flexibility to gluelessly connect to a Sony/Philips digital interface (S/PDIF) transmit physical layer component.

Although inter-component digital audio interface reception (DIR) mode (this is, S/PDIF stream receiving) is not natively supported by the MCASP module, a specific TDM mode implementation for the MCASP receivers allows an easy connection to external DIR components (for example, S/PDIF to I2S format converters).

shows MCASP modules allocation across device domains.

[Figure 12-486](#) shows the MCASP modules overview.

**Figure 12-486. MCASP Modules Overview**

#### 12.5.2.1.1 MCASP Features

Each MCASP module includes the following main features:

- Independent serializer for each AXRx channel of each MCASP module
- Clock stop request/acknowledge protocol
- A single 32-bit buffer per serializer for transmit and receive operations
- Interconnect interface port for CBASS0
- Two independent clock generator modules for transmit and receive (clocking flexibility allows the MCASP to receive and transmit at different rates. For example, the MCASP can receive data at 48 kHz, but output up-sampled data at 96 kHz or 192 kHz)
- Each MCASP module functional clock can be generated:
  - Internally (master mode)
  - Supplied over MCASP serial interface (slave mode)
  - Has a controllable functional clock divide ratio
- Independent transmit and receive modules, each includes:
  - Programmable clock and frame sync generator
  - TDM streams from 2 to 32, and 384 time slots
  - Support for time slot sizes of 8, 12, 16, 20, 24, 28, and 32 bits
  - Data formatter for bit manipulation
- Glueless connection to audio Analog-to-Digital Converters (ADC), Digital-to-Analog Converters (DAC), codec, digital audio interface receiver (DIR), and S/PDIF transmit physical layer components
- Wide variety of I2S and similar bit-stream format
- Integrated digital audio interface transmitter (DIT):
  - S/PDIF, IEC60958-1, AES-3 formats
  - Enhanced channel status/user data RAM
- 384-slot TDM with external digital audio interface receiver (DIR) device
  - For DIR reception, an external DIR receiver integrated circuit should be used with I2S output format and connected to the MCASP receive section
- Support for 2 × DMA requests (one per direction):
  - 1 level-sensitive transmit direct memory access (DMA) request common for all of the MCASP serializers
  - 1 level-sensitive receive direct memory access (DMA) request common for all of the MCASP serializers
  - All transmit DMA requests are mapped to the device DMA controllers
- One transmit interrupt request common for all serializers



- One receive interrupt request common for all serializers
- Each of the Rx and Tx interrupts is propagated to different host processors via the device Interrupts

---

**Note**

Because a serializer receive and transmit channels data is shared on the same MCASP data pin, user can choose to have either Tx or Rx function from a serializer, not both at the same time.

---

**12.5.2.1.2 MCASP Not Supported Features**

Each MCASP module does not support the following features:

- Muting output (AMUTE)
- Muting input (AMUTEIN)

### 12.5.2.2 MCASP Environment

This section describes the MCASP application fields from an environment point of view (external connections). This section also lists the possible interfaces and describes the protocol and data format used in each case.

#### 12.5.2.2.1 MCASP Signals

Figure 12-487 shows all of the MCASP interface signals.

**Figure 12-487. MCASP Interface Signals**

Table 12-490 describes the MCASP I/O signals.

**Table 12-490. MCASP I/O Signals**

Module Pin	Device Level Signal	I/O <sup>(1)</sup>	Description	Module Pin Reset Value <sup>(2)</sup>
<b>MCASP0 module</b>				
AXR0	MCASP0_AXR0	I/O	Audio transmit/receive data - channel 0	HiZ
AXR1	MCASP0_AXR1	I/O	Audio transmit/receive data - channel 1	HiZ
AXR2	MCASP0_AXR2	I/O	Audio transmit/receive data - channel 2	HiZ
AXR3	MCASP0_AXR3	I/O	Audio transmit/receive data - channel 3	HiZ
AXR4	MCASP0_AXR4	I/O	Audio transmit/receive data - channel 4	HiZ
AXR5	MCASP0_AXR5	I/O	Audio transmit/receive data - channel 5	HiZ
AXR6	MCASP0_AXR6	I/O	Audio transmit/receive data - channel 6	HiZ
AXR7	MCASP0_AXR7	I/O	Audio transmit/receive data - channel 7	HiZ
AXR8	MCASP0_AXR8	I/O	Audio transmit/receive data - channel 8	HiZ
AXR9	MCASP0_AXR9	I/O	Audio transmit/receive data - channel 9	HiZ
AXR10	MCASP0_AXR10	I/O	Audio transmit/receive data - channel 10	HiZ
AXR11	MCASP0_AXR11	I/O	Audio transmit/receive data - channel 11	HiZ
AXR12	MCASP0_AXR12	I/O	Audio transmit/receive data - channel 12	HiZ
AXR13	MCASP0_AXR13	I/O	Audio transmit/receive data - channel 13	HiZ
AXR14	MCASP0_AXR14	I/O	Audio transmit/receive data - channel 14	HiZ
AXR15	MCASP0_AXR15	I/O	Audio transmit/receive data - channel 15	HiZ
ACLKX	MCASP0_ACLKX	I/O	Transmit bit clock	HiZ
AFSX	MCASP0_AFSX	I/O	Transmit frame synchronization	HiZ
ACLKR	MCASP0_ACLKR	I/O	Receive bit clock	HiZ
AFSR	MCASP0_AFSR	I/O	Receive frame synchronization	HiZ
MCASP0_AHCLKX_I/O	AUDIO_EXT_REFCLK[0-1]	I/O	Transmit high-frequency master clock. See <a href="#">Figure 12-496</a>	HiZ
MCASP0_AHCLKR_I/O	AUDIO_EXT_REFCLK[0-1]	I/O	Receive high-frequency master clock. See <a href="#">Figure 12-496</a>	HiZ
<b>MCASP1 module</b>				
AXR0	MCASP1_AXR0	I/O	Audio transmit/receive data - channel 0	HiZ
AXR1	MCASP1_AXR1	I/O	Audio transmit/receive data - channel 1	HiZ
AXR2	MCASP1_AXR2	I/O	Audio transmit/receive data - channel 2	HiZ
AXR3	MCASP1_AXR3	I/O	Audio transmit/receive data - channel 3	HiZ
AXR4	MCASP1_AXR4	I/O	Audio transmit/receive data - channel 4	HiZ
ACLKX	MCASP1_ACLKX	I/O	Transmit bit clock	HiZ
AFSX	MCASP1_AFSX	I/O	Transmit frame synchronization	HiZ
ACLKR	MCASP1_ACLKR	I/O	Receive bit clock	HiZ
AFSR	MCASP1_AFSR	I/O	Receive frame synchronization	HiZ

**Table 12-490. MCASP I/O Signals (continued)**

Module Pin	Device Level Signal	I/O <sup>(1)</sup>	Description	Module Pin Reset Value <sup>(2)</sup>
MCASP1_AHCLKX_I/O	AUDIO_EXT_REFCLK[0-1]	I/O	Transmit high-frequency master clock. See <a href="#">Figure 12-496</a>	HiZ
MCASP1_AHCLKR_I/O	AUDIO_EXT_REFCLK[0-1]	I/O	Receive high-frequency master clock. See <a href="#">Figure 12-496</a>	HiZ
<b>MCASP2 module</b>				
AXR0	MCASP2_AXR0	I/O	Audio transmit/receive data - channel 0	HiZ
AXR1	MCASP2_AXR1	I/O	Audio transmit/receive data - channel 1	HiZ
AXR2	MCASP2_AXR2	I/O	Audio transmit/receive data - channel 2	HiZ
AXR3	MCASP2_AXR3	I/O	Audio transmit/receive data - channel 3	HiZ
AXR4	MCASP2_AXR4	I/O	Audio transmit/receive data - channel 4	HiZ
ACLKX	MCASP2_ACLKX	I/O	Transmit bit clock	HiZ
AFSX	MCASP2_AFSX	I/O	Transmit frame synchronization	HiZ
ACLKR	MCASP2_ACLKR	I/O	Receive bit clock	HiZ
AFSR	MCASP2_AFSR	I/O	Receive frame synchronization	HiZ
MCASP2_AHCLKX_I/O	AUDIO_EXT_REFCLK[0-1]	I/O	Transmit high-frequency master clock. See <a href="#">Figure 12-496</a>	HiZ
MCASP2_AHCLKR_I/O	AUDIO_EXT_REFCLK[0-1]	I/O	Receive high-frequency master clock. See <a href="#">Figure 12-496</a>	HiZ

(1) I = Input; O = Output; I/O = Bidirectional

(2) HiZ = High Impedance

#### Note

signals are multiplexed to AUDIO\_EXT\_REFCLK[0-1] device pins.

#### Note

For signals to work properly, the RXACTIVE bit of the appropriate CTRLMMR\_PADCONFIGy registers should be set to 0x1 because of retiming purposes.

#### Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

#### Note

A serializer AXR data pin is shared between the transmit and receive logic of that serializer. The direction of data is determined in the MCASP\_PDIR register and the function (Tx or Rx) is selected in the corresponding serializer control register MCASP\_SRCTLn (n = 0 to 15).

### 12.5.2.2.2 MCASP Protocols and Data Formats

#### 12.5.2.2.2.1 Protocols Supported

The MCASP supports a wide variety of protocols:

- Transmit section supports:
  - Wide variety of I2S and similar bit-stream formats
  - TDM streams from 2 to 32 time slots
  - S/PDIF, IEC60958-1, AES-3 formats
- Receive section supports:

- Wide variety of I2S and similar bit-stream formats
- TDM streams from 2 to 32 time slots
- TDM stream of 384 time slots specifically designed for easy interface to external digital interface receiver (DIR) device transmitting DIR frames to MCASP using the I2S protocol (one time slot for each DIR subframe)

The transmit and receive sections of the module may be individually programmed to support the following options on the basic serial protocol:

- Programmable clock and frame sync polarity (rising or falling edge): ACLKR/X, AFSSR/X and AHCLKR/X\_I/O
- Slot length (number of bits per time slot): 2, 4, 8, 12, 16, 20, 24, 28, 32 bits supported
- Word length (bits per word): 2, 4, 8, 12, 16, 20, 24, 28, 32 bits; always less than or equal to the time slot length
- First-bit data delay: 0, 1, 2 bit clocks
- Left/right alignment of word inside slot
- Bit order: MSB first or LSB first
- Bit mask/pad/rotate function
  - Automatically aligns data internally in either Q31 or integer formats
  - Automatically masks nonsignificant bits (sets to 0, 1, or extends value of another bit)

---

#### Note

In I2S mode, the transmit and receive sections can support simultaneous transfers on up to all serial data pins operating as 192 kHz stereo channels.

---

In DIT mode for MCASP, additional features of the transmitters are:

- Transmit-only mode 384 time slots (subframe) per frame
- Biphase encoded 3.3 V Output
- Channel status RAM (384 bits)
- User data RAM (384 bits)
- Support for consumer and professional applications
- Separate valid bit (V) for subframe A, B
- Stereo Support Only (Mono means send data 2 × via software)

In DIT mode, the transmitter can support a 192 kHz frame rate (stereo) on up to all serial data pins simultaneously (note that the internal bit clock for DIT runs two times faster than the equivalent bit clock for I2S mode, due to the need to generate Biphase Mark Encoded Data).

---

#### Note

The MCASP does NOT natively support DIR-mode reception (this is, receiving in the S/PDIF format). To allow this, the MCASP can use a DIR-input to I2S-output converter implemented by an external device (this is, external DIR component). To facilitate reception in this case, the TDM mode of MCASP receivers logic is extended to support a non-standard 384-slot TDM stream.

---



---

#### Note

An external transceiver must be connected to the MCASP port in the device to translate the electrical signals delivered by the MCASP (1.2 V or 1.8 V LVCMOS levels) to the electrical levels of the S/PDIF standard.

---

#### 12.5.2.2.2 Definition of Terms

The serial bitstream transmitted or received by a MCASP serializer is a long sequence of 1s and 0s on an audio transmit/receive pins AXRn. However, the sequence has a hierarchical organization that can be described in terms of frames of data, slots, words, and bits.

A basic synchronous serial interface consists of three important components: clock, frame sync, and data. [Figure 12-488](#) shows two of the three basic components: the clock signal (ACLKX/ACLKR) and the data signals AXRn. [Figure 12-488](#) does not specify whether the clock is for transmit (ACLKX) or receive (ACLKR) because the definitions of terms apply to both receive and transmit interfaces. In operation, each transmitter and receiver uses the signals ACLKX and ACLKR as serial clock, respectively. Optionally, a receiver can use ACLKX as the serial clock when a transmitter and receiver (not from the same serializer) of the MCASP are configured to operate synchronously.

- Bit:

A bit is the smallest entity in the serial data stream. The beginning and end of each bit is marked by an edge of the serial clock. The duration of a bit is a serial clock period. A '1' is represented by a logic high on AXRn pins for the entire duration of the bit. A 0 is represented by a logic low on an AXRn pin for the entire duration of the bit.

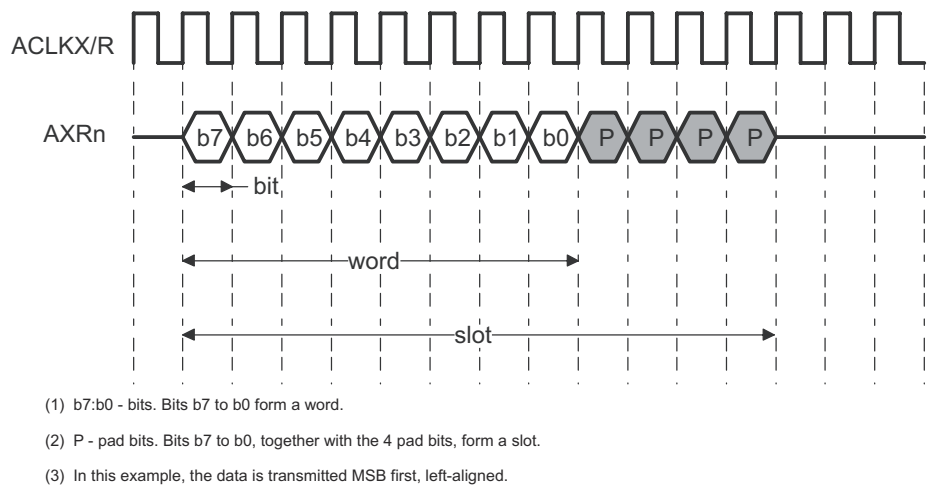
- Word:

A word is a group of bits that make up the data being transferred between the MCASP and the external device. [Figure 12-488](#) shows an 8-bit word.

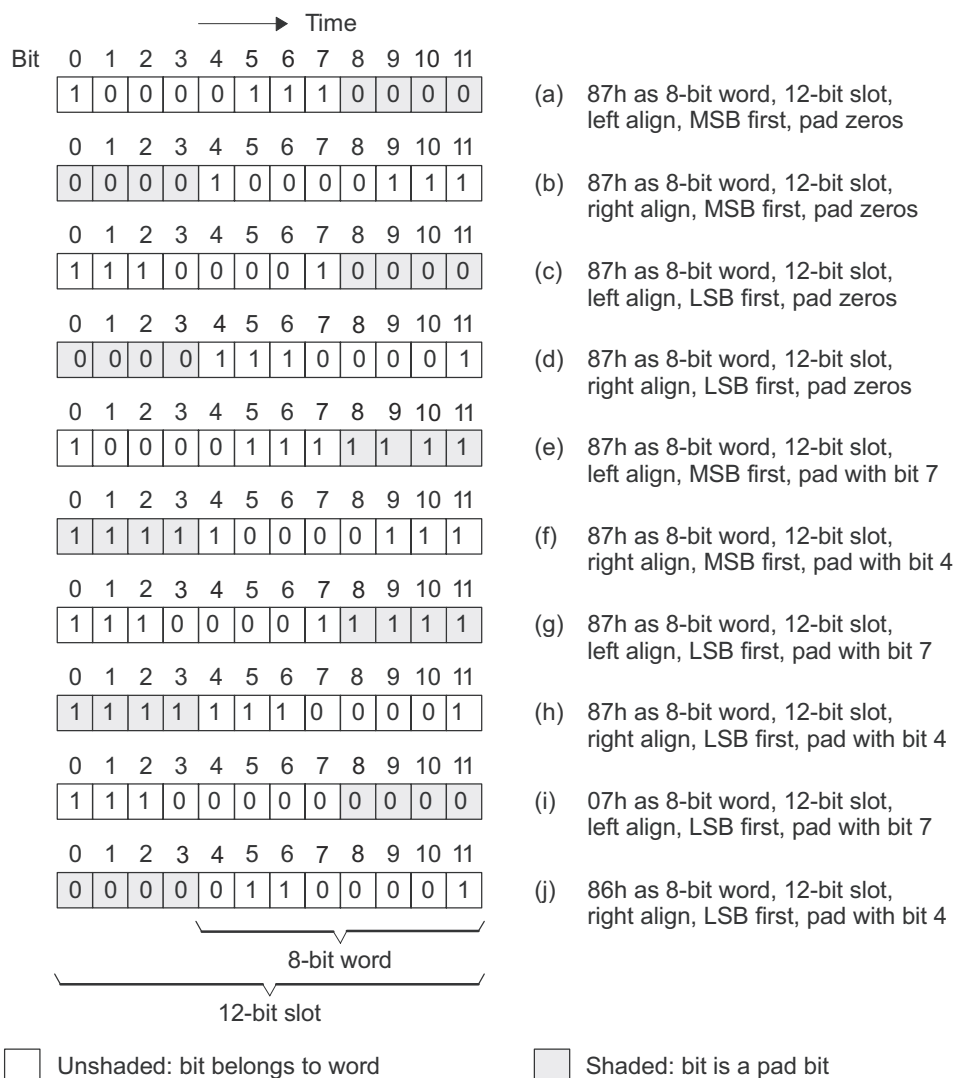
- Slot:

A slot consists of the bits that make up the word and can consist of additional bits used to pad the word to a convenient number of bits for the interface between the MCASP and the external device. In [Figure 12-488](#), the audio data consists of only 8 bits of useful data (8-bit word), but it is padded with four 0s (12-bit slot) to satisfy the desired protocol in interfacing to an external device. Within a slot, the bits can be shifted out of the MCASP on an AXRn pin with either MSB or LSB first.

When the word size is smaller than the slot size, the word can be aligned to the left of the slot (beginning) or to the right of the slot (end). The additional bits in the slot not belonging to the word can be padded with 0, 1, or with one of the bits (typically, the MSB or LSB) from the data word, this is, left-aligned words within a slot are terminated with padding bits and right-aligned words within a slot are preceded by padding bits to fit in the slot size. [Figure 12-489](#) shows these options.



**Figure 12-488. Definition of Bit, Word, and Slot**



**Figure 12-489. Bit Order and Word Alignment Within a Slot Examples**

- Frame

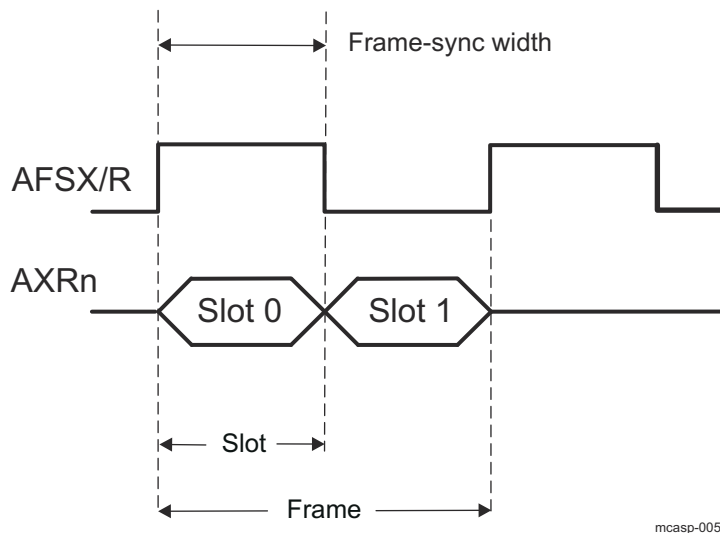
The third basic element of a synchronous serial interface is the frame synchronization signal, also referred to as frame sync in this chapter. A frame contains one or multiple slots, as determined by the desired protocol. [Figure 12-490](#) shows an example frame of data and the frame definitions. In operation, the transmitter uses AFSX, and the receiver - AFSR signal. [Figure 12-490](#) does NOT specify whether the frame sync (FS) is for transmit (AFSX) or receive (AFSR) because the definitions of terms apply to both receive and transmit interfaces. In operation, each transmitter/receiver uses AFSX/AFSR as a frame synchronization signal, respectively. Optionally, the receiver can use AFSX as the frame sync when the transmitter and receiver of the MCASP are configured to operate synchronously. This example shows two slots in a frame (I2S format) and a frame-sync (FS) duration of a slot length.

This section shows only the generic definition of the frame sync. For more information about the frame-sync formats required for the transfer modes and protocols (TDM-mode and DIT-mode supported formats), see [Section 12.5.2.2.2.3, TDM Format](#) and [Section 12.5.2.2.2.5, S/PDIF-Coding Format](#).

### Note

All of the MCASP serializers share the same, device pad accessible, clock and frame signals, as follows:

- AHCLKX\_I/O, ACLKX and AFSX for the transmitting section
- AHCLKR\_I/O, ACLKR and AFSR for the receiving section



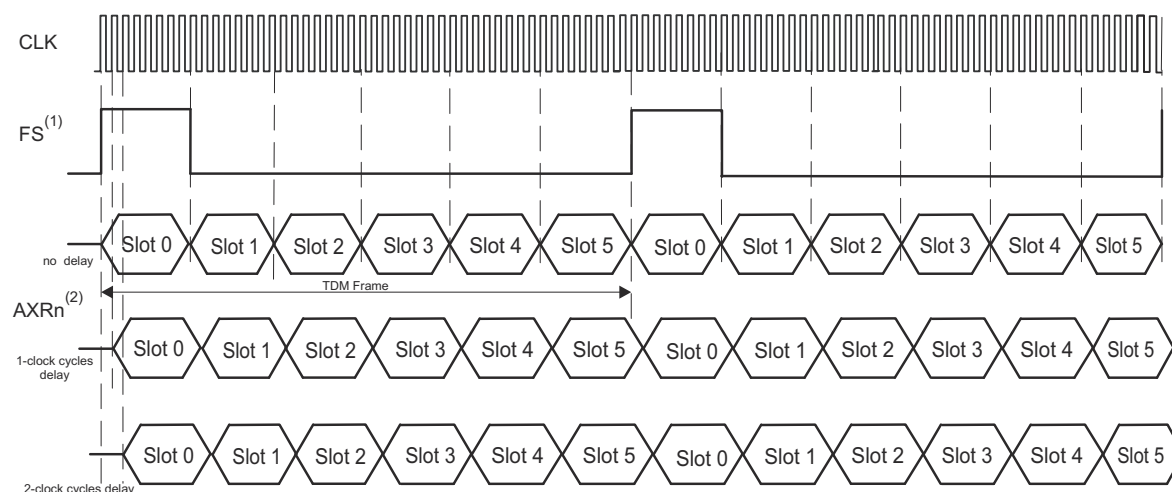
**Figure 12-490. Definition of Frame and Frame-Sync Width**

The following terms are used throughout this chapter:

- TDM: Time-division multiplexed. See [Section 12.5.2.2.2.3, TDM Format](#) for details on the TDM protocol.
- I2S: Inter-Integrated Sound protocol, commonly used on audio interfaces. The MCASP supports the I2S protocol as part of the TDM mode (when configured as a 2-slot frame).
- DIT: Digital audio interface transmit. The MCASP supports transmitting in S/PDIF format on each AXRn data pin.
- DIR: Digital audio interface receive. The MCASP does NOT natively support receiving in S/PDIF format on AXRn data pins and requires an external DIR-to-TDM or DIR-to-I2S converter chip for a DIR-frame reception.
- Slot or time slot: For DIT/DIR format, a MCASP time slot corresponds to a DIT/DIR subframe.

#### 12.5.2.2.2.3 TDM Format

The TDM format is used to transfer data between the host CPU and one or more analog-to-digital converter (ADC), digital-to-analog converter (DAC), or S/PDIF receiver (DIR) devices. An example for a 6-slot (channel) TDM transmission on one MCASP data pin - AXRn is illustrated on [Figure 12-491](#).



- (1) - Frame sync duration of 1 slot - length is shown. A single bit - duration of FS is also supported
- (2) - Slot 0 of AXRn stream is being offset with 0-, 1-, and 2- clock cycle delay from the frame sync, respectively.

mcasp-006

**Figure 12-491. TDM Format - 6 channel example**

The TDM format uses three signals in a basic synchronous serial interface: data (AXRn), clock (CLK) and frame sync (FS). The data signal present on AXRn pin is fully synchronous to the serial clock (ACLKX or ACLKR). The data bits are grouped into words and slots (see also [Section 12.5.2.2.2](#)), the latter being also referred to as the "time-slots" or "channels" in TDM terminology. A frame consists of multiple time-slots. Each TDM frame is marked by the frame sync signal (AFSX or AFSR). The TDM transfer is continuous and periodic, with no delays between slots.

Within a certain frame, the last bit of slot N is followed immediately on the next serial clock with the first bit of the next slot N+1. On the boundary between two adjacent TDM-frames, the last bit of the last slot from the frame M, is followed immediately on the next clock cycle with the first bit of the first slot from the next frame M+1. For MCASP, there is an option to offset the first bit of the first slot with a 0-, 1- or 2-cycle delay from the frame sync signal.

The frame sync - AFSX/AFSR only marks the beginning of slot 0 and start of a new frame. Since it does not determine the boundaries of a slot, there is a requirement for a connected transmitter and receiver to agree on the number of transferred bits per slot.

In a typical audio system involving MCASP module, a single TDM data frame is transferred during each sample period  $T_s$  of a data converter. The user has following choices to implement multiple channels:

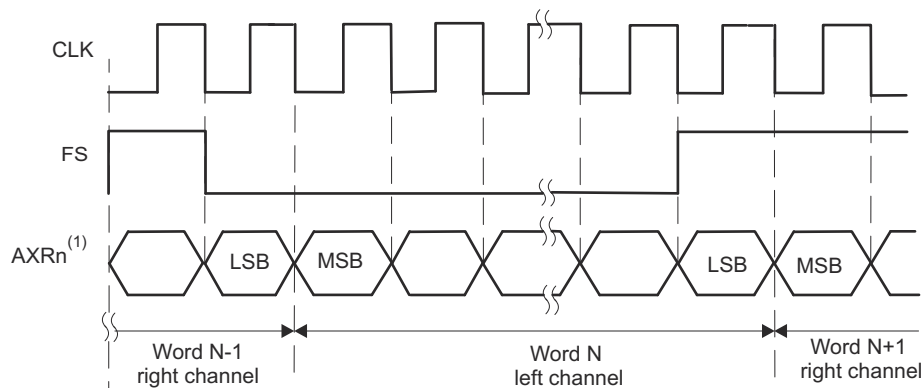
- Use more data slots (on a price of higher speed serial clock) per frame transmitted/received on just one of the available MCASP data pins AXRn.
- Use less number of slots per TDM frame (requires a slower serial clock), making them available on several of the MCASP pins AXRn.

#### 12.5.2.2.4 I2S Format

The TDM transfer mode of the MCASP supports the I2S format when frame is configured to have 2 slots. The I2S format is specifically designed to transfer a stereo channel (left and right) over a single data pin AXRn. The "Slots" are also commonly referred to as "channels". The frame width duration in the I2S format equals size of a slot. The frame signal is also referred to as "word select" in the I2S format.

The I2S protocol is illustrated on [Figure 12-492](#).





(1) - The example shows I2S data MSB-first transmission with 1-clock cycle delay between FS and data MSB

mcasp-007

**Figure 12-492. I2S Format Overview**

#### 12.5.2.2.2.5 S/PDIF Coding Format

The MCASP transmitter supports the S/PDIF format with 3.3 V biphase-mark encoded output. The S/PDIF format is supported by the DIT- transfer mode of the MCASP. This section briefly discusses the S/PDIF coding format.

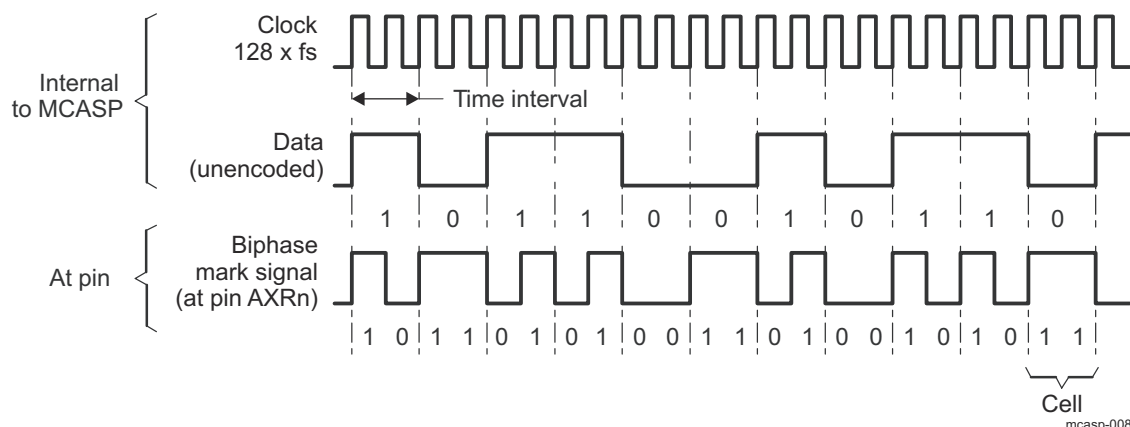
#### Note

The DIR- reception of S/PDIF format frames is NOT natively supported from the device MCASP. For this purpose, an external DIR-to-TDM transfer mode adapter can be used between the remote device S/PDIF transmitter output and the MCASP TDM-only compatible receiver input.

#### 12.5.2.2.2.5.1 Biphase-Mark Code

In S/PDIF format, the digital signal is coded using the biphase-mark code (BMC). For each serializer transmitter  $n$ , the clock, frame, and data are embedded in only one signal - the data signal AXR $n$ . In the BMC system, each data bit is encoded into two logical states (00, 01, 10, or 11) at the pin. These two logical states form a cell. The duration of the cell, which equals the duration of the data bit, is called a time interval. A logical 1 is represented by two transitions of the signal within a time interval, which corresponds to a cell with logical states 01 or 10. A logical 0 is represented by one transition within a time interval, which corresponds to a cell with logical states 00 or 11. In addition, the logical level at the start of a cell is inverted from the level at the end of the previous cell. [Figure 12-493](#) and [Table 12-491](#) show how data is encoded to the BMC format.

As shown in [Figure 12-493](#), the clock frequency is twice the unencoded data bit rate. In addition, the clock is always programmed to  $128 \times f_s$ , where  $f_s$  is the sample rate (see [Section 12.5.2.2.2.5.3, Frame Format](#), for details on how this clock rate is derived based on the S/PDIF format). The device receiving in S/PDIF format can recover the clock and frame information from the BMC signal.



**Figure 12-493. Biphase-Mark Code**

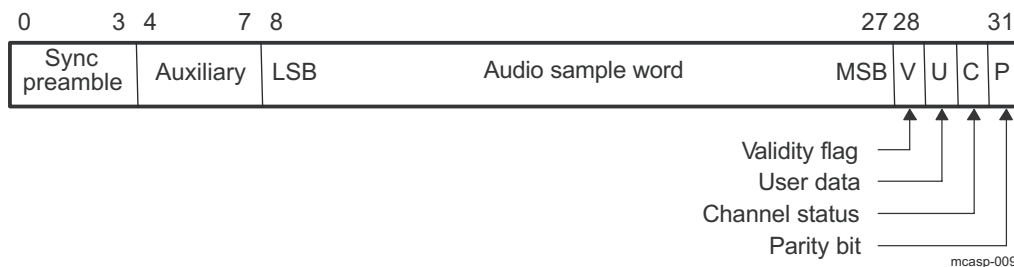
**Table 12-491. Biphase-Mark Encoder**

Data (Unencoded)	Previous State at Pin AXRn	BMC-Encoded Cell Output at Pin AXRn
0	0	11
0	1	00
1	0	10
1	1	01

#### 12.5.2.2.2.5.2 S/PDIF Subframe Format

Every audio sample transmitted in a subframe consists of 32 S/PDIF time intervals (or cells), numbered 0 to 31. [Figure 12-494](#) shows a subframe.

- Time intervals 0–3 carry one of the three permitted preambles to signify the type of audio sample in the current subframe. The preamble is not encoded in BMC format, and therefore the preamble code can contain more than two consecutive 0 or 1 logical states in a row. See [Table 12-492](#).
- Time intervals 4–27 carry the audio sample word in linear 2s-complement representation. The MSB is carried by time interval 27. When a 24-bit coding range is used, the LSB is in time interval 4. When a 20-bit coding range is used, time intervals 8–27 carry the audio sample word with the LSB in time interval 8. Time intervals 4–7 may be used for other applications and are designated auxiliary sample bits.
- If the source provides fewer bits than the interface allows (20 or 24), the unused LSBs are set to logical 0. For a nonlinear PCM audio application or a data application, the main data field can carry any other information.
- Time interval 28 carries the validity bit (V) associated with the main data field in the subframe.
- Time interval 29 carries the user data channel (U) associated with the main data field in the subframe.
- Time interval 30 carries the channel status information (C) associated with the main data field in the subframe. The channel status indicates if the data in the subframe is digital audio or some other type of data.
- Time interval 31 carries a parity bit (P) such that time intervals 4–31 carry an even number of 1s and an even number of 0s (even parity). As listed in [Table 12-492](#), the preambles (time intervals 0–3) are also defined with even parity.



**Figure 12-494. S/PDIF Subframe Format**

As listed in [Table 12-492](#), the MCASP DIT generates only one polarity of preambles, and it assumes the previous logical state is 0. This is because the MCASP assures an even-polarity encoding scheme when transmitting in DIT mode. If an underrun condition occurs, the DIT resynchronizes to the correct logic level on the AXRn pin before continuing with the next transmission.

**Table 12-492. Preamble Codes**

Preamble Code <sup>(1)</sup>	Previous Logical State	Logical States on pin AXRn <sup>(2)</sup>	Description
B (or Z)	0	1110 1000	Start of a block and subframe 1
M (or X)	0	1110 0010	Subframe 1
W (or Y)	0	1110 0100	Subframe 2

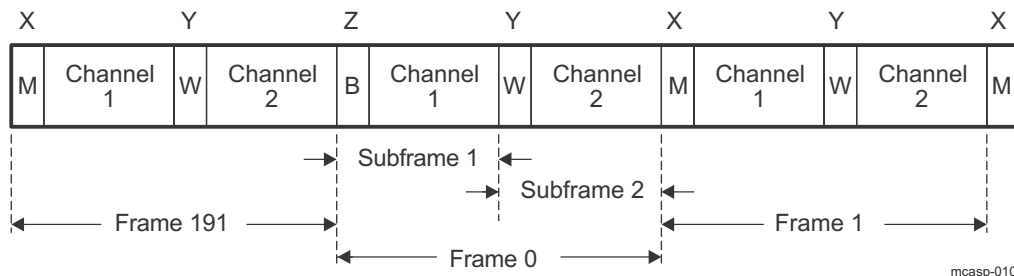
- (1) Historically, preamble codes are referred to as B, M, and W. For use in professional applications, preambles are referred to as Z, X, and Y, respectively.
- (2) The preamble is not BMC-encoded. Each logical state is synchronized to the serial clock. These eight logical states make up time slots (cells) 0 to 3 in the S/PDIF stream.

#### 12.5.2.2.5.3 Frame Format

An S/PDIF frame is composed of two subframes (see [Figure 12-495](#)). For linear coded audio applications, the rate of frame transmission normally corresponds exactly to the source sampling frequency  $f_s$ . The S/PDIF format clock rate is therefore  $128 \times f_s$  ( $128 = 32$  cells per subframe  $\times 2$  clocks per cell  $\times 2$  subframes per sample). For example, for an S/PDIF stream at a 192-kHz sampling frequency, the serial clock is  $128 \times 192$  kHz = 24.58 MHz.

In 2-channel operation mode, the samples taken from both channels are transmitted by time multiplexing in consecutive subframes. Both subframes contain valid data (cell 28 validity bits for A- and B- channels, both set to '0'). The first subframe (left or A channel in stereophonic operation and primary channel in monophonic operation) normally starts with preamble M. However, the preamble of the first subframe changes to preamble B once every 192 frames to identify the start of the block structure used to organize the channel status information. The second subframe (right or B channel in stereophonic operation and secondary channel in monophonic operation) always starts with preamble W.

In single-channel operation mode in a professional application, the frame format is the same as in the 2-channel mode. Data is carried in the first subframe and may be duplicated in the second subframe. If the second subframe is not carrying duplicate data, cell 28 (validity bit) is set to logical 1.



**Figure 12-495. S/PDIF Frame Format**

### 12.5.2.3 MCASP Integration

This section describes MCASP integration in the device, including information about clocks, resets, and hardware requests.

#### 12.5.2.3.1 MCASP Integration in MAIN Domain

There are MCASP modules integrated in the device MAIN domain - . [Figure 12-496](#) shows the integration of MCASP[0-].

**Figure 12-496. MCASP Integration**

[Table 12-493](#) through [Table 12-495](#) summarize the integration of MCASP[0-] in the device MAIN domain.

**Table 12-493. MCASP Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
MCASP0	PSC0	PD0	LPSC3	CBASS0
MCASP1	PSC0	PD0	LPSC3	CBASS0
MCASP2	PSC0	PD0	LPSC3	CBASS0

**Table 12-494. MCASP Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
MCASPi	MCASPi_ICLK	MAIN_SYSCLK0/2	PLLCTRL0	MCASPi interface clock
	MCASPi_AUX_CLK	MAIN_PLL4_HSDIV0_CLKOUT	PLL4	MCASPi functional clock (Master Mode). Output of multiplexer, see <i>MCASP Integration</i> . <sup>(1)</sup>
		MAIN_PLL2_HSDIV2_CLKOUT	PLL2	
		ATCLK0	ATL	
		ATCLK1	ATL	
		ATCLK2	ATL	
		ATCLK3	ATL	
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
MCASPi	MCASPi_RST	MOD_G_RST	LPSC3	MCASPi reset

(1) Multiplexers control is provided via CTRLMMR\_MCASPi\_CLKSEL[2-0] AUXCLK\_SEL bit-field for module MCASPi (). For more information, see *Control Module (CTRL\_MMR)* of the chapter *Device Configuration*.

**Table 12-495. MCASP Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCASP0	MCASP0_REC_INTR_PEND_0	GIC500_SPI_IN_577	COMPUTE_CLUSTER0	MCASP0 receive interrupt request	Level
		MAIN2MCU_LVL_INTRTR0_IN_177	MAIN2MCU_LVL_INTRTR0		
		R5FSS0_CORE0_INTR_IN_133	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_133	R5FSS0_CORE1		
	MCASP0_XMIT_INTR_PEND_0	GIC500_SPI_IN_576	COMPUTE_CLUSTER0	MCASP0 transmit interrupt request	Level
		MAIN2MCU_LVL_INTRTR0_IN_176	MAIN2MCU_LVL_INTRTR0		
		R5FSS0_CORE0_INTR_IN_132	R5FSS0_CORE0		

**Table 12-495. MCASP Hardware Requests (continued)**

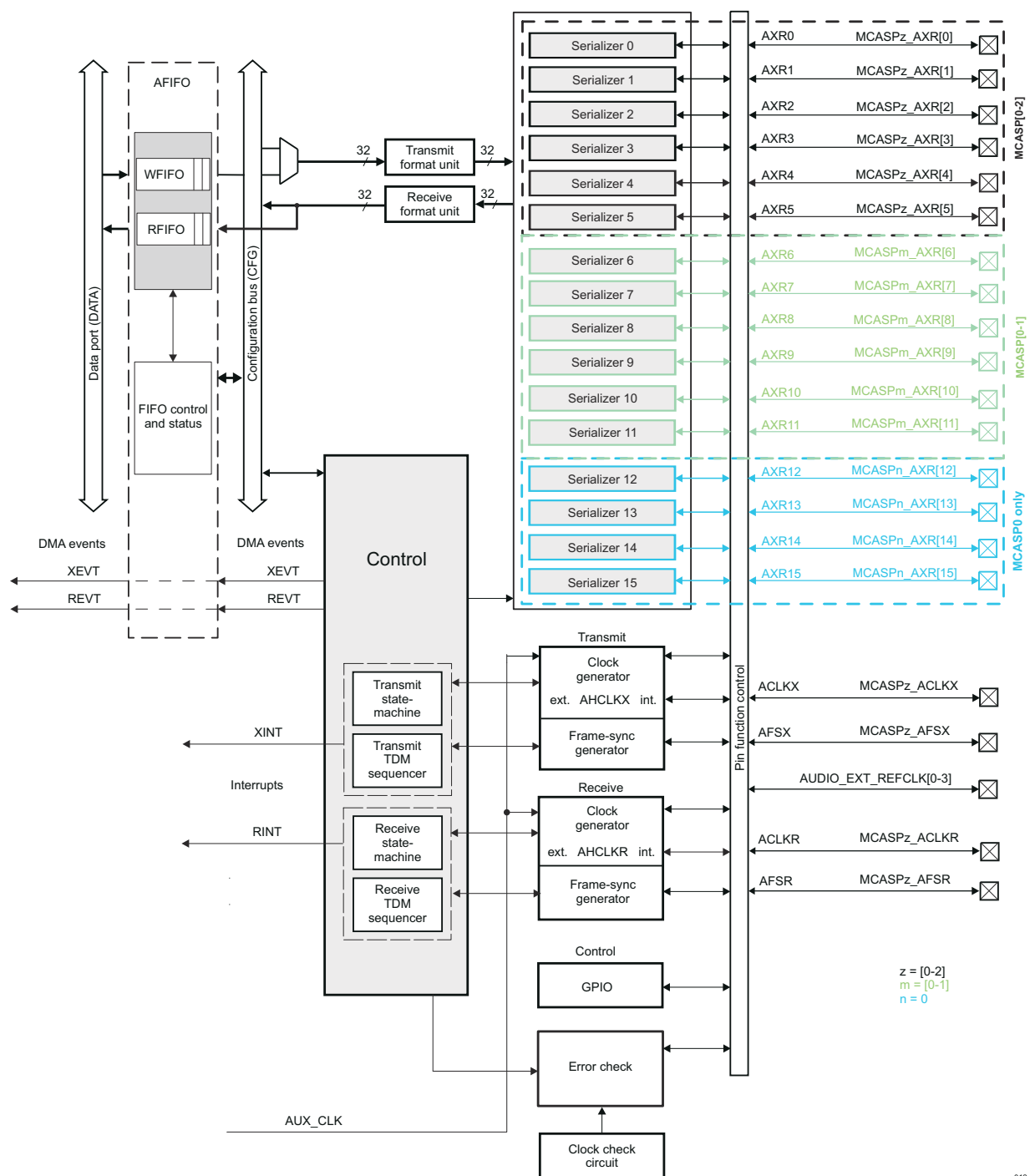
		R5FSS0_CORE1_INTR_IN_132	R5FSS0_CORE1		
MCASP1	MCASP1_REC_INTR_PEND_0	GIC500_SPI_IN_579	COMPUTE_CLUSTER0	MCASP1 receive interrupt request	Level
		MAIN2MCU_LVL_INTRTR0_IN_179	MAIN2MCU_LVL_INTRTR0		
		R5FSS0_CORE0_INTR_IN_135	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_135	R5FSS0_CORE1		
	MCASP1_XMIT_INTR_PEND_0	GIC500_SPI_IN_578	COMPUTE_CLUSTER0	MCASP1 transmit interrupt request	Level
		MAIN2MCU_LVL_INTRTR0_IN_178	MAIN2MCU_LVL_INTRTR0		
		R5FSS0_CORE0_INTR_IN_134	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_134	R5FSS0_CORE1		
MCASP2	MCASP2_REC_INTR_PEND_0	GIC500_SPI_IN_581	COMPUTE_CLUSTER0	MCASP2 receive interrupt request	Level
		R5FSS0_CORE0_INTR_IN_277	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_277	R5FSS0_CORE1		
		MAIN2MCU_LVL_INTRTR0_IN_181	MAIN2MCU_LVL_INTRTR0		
	MCASP2_XMIT_INTR_PEND_0	GIC500_SPI_IN_580	COMPUTE_CLUSTER0	MCASP2 transmit interrupt request	Level
		R5FSS0_CORE0_INTR_IN_276	R5FSS0_CORE0		
		R5FSS0_CORE1_INTR_IN_276	R5FSS0_CORE1		
		MAIN2MCU_LVL_INTRTR0_IN_180	MAIN2MCU_LVL_INTRTR0		
DMA Events					
Module Instance	Module DMA Event	Destination DMA Event Input	Destination	Description	Type
MCASP0	MCASP0_REC_DMA_EVT	MCASP0_RX0	PDMA0_MCASP_G0	MCASP0 receive request line	Pulse
	MCASP0_XMIT_DMA_EVT	MCASP0_TX0	PDMA0_MCASP_G0	MCASP0 transmit request line	Pulse
MCASP1	MCASP1_REC_DMA_EVT	MCASP1_RX0	PDMA0_MCASP_G0	MCASP1 receive request line	Pulse
	MCASP1_XMIT_DMA_EVT	MCASP1_TX0	PDMA0_MCASP_G0	MCASP1 transmit request line	Pulse
MCASP2	MCASP2_REC_DMA_EVT	MCASP2_RX0	PDMA0_MCASP_G0	MCASP2 receive request line	Pulse
	MCASP2_XMIT_DMA_EVT	MCASP2_TX0	PDMA0_MCASP_G0	MCASP2 transmit request line	Pulse

### 12.5.2.4 MCASP Functional Description

In the text throughout this section a single instance of MCASP is described assuming that all modules are functionally identical. For module availability and integration differences, see *MCASP Environment* and *MCASP Integration*.

#### 12.5.2.4.1 MCASP Block Diagram

Figure 12-497 shows the major blocks of the MCASP module.



mcasp-012

A. i represents a valid instance of MCASP in a domain. See the device datasheet for valid instances.

- B. All signals might not be valid for an instance. See the device datasheet for specifics.

**Figure 12-497. MCASP Module Block Diagram**

### Note

The internal and external clocks mentioned in this section are with respect to clock and frame-sync generator modules.

#### 12.5.2.4.2 MCASP Clock and Frame-Sync Configurations

There are three scenarios to provide clock source signals for the Tx part and four scenarios for the Rx part of the MCASP serializers. The first three scenarios are identical between the Tx and Rx part of the MCASP. They feature an asynchronous operation between receiver and transmitter channels using independent Tx/Rx bit rate clock sources (either internal or external).

In the first scenario, the transmit - XCLK and receive - RCLK serial clocks (clock at the bit rate) are generated internally by passing through a couple of clock dividers off the internal functional clock source (AUXCLK). In this case, the bit rate clock is generated internally and is driven out on the pin ACLKX for the Tx part and pin ACLKR for the Rx part, respectively. An internally generated high-frequency clock can be optionally driven out onto the AHCLKX pin for the Tx part to serve as a reference clock for other components in the system.

In the second scenario, an external for the device clock, is passed on the ACLKX (for the TX part) and ACLKR (for the RX part) pins which are configured as inputs. In this case the Rx- /Tx- high-speed clock logic is bypassed for the XCLK/RCLK generation.

In the third (mixed) scenario, an externally driven (master) high-frequency clock is applied on the AHCLKX (for the TX part) pin, which is configured as input. In this case the AHCLKX clock frequency can be divided down via programming the ACLKX associated divider to produce the necessary bit rate clock. The high-speed clock divider can NOT be used.

In the fourth clock generation scenario the bit rate clock for MCASP receivers - RCLK is derived from the bit rate clock of the MCASP transmitters - XCLK for a synchronous operation between transmitters and receivers. Hence, the whole receiver clock generator logic is bypassed.

A typical role of the MCASP frame sync signal is to carry the left/right clock (LRCLK) signal when transmitting and receiving stereo data.

For an asynchronous operation, the AFSX (Tx part) and AFSR (Rx part) frame synchronization signals can be sourced internally or delivered externally independently for the Tx and Rx channels. During synchronous operation the receive frame sync - AFSR signal is derived from the transmit frame sync - AFSX signal. A synchronous and asynchronous mode applies to bit rate clock and frame sync signals at the same time.

##### 12.5.2.4.2.1 MCASP Transmit Clock

The transmit high-speed and transmit clock configuration is controlled by the following registers:

- MCASP\_ACLKXCTL
- MCASP\_AHCLKXCTL

In case, the transmit bit clock, ACLKX, is generated internally, the MCASP\_ACLKXCTL[5] CLKXM bit must be set to 1. Thus, the clock is divided down by a programmable bit clock divider (the MCASP\_ACLKXCTL[4-0] CLKXDIV bit field) from the source signal.

If the transmit high-frequency master clock, AHCLKX, is also sourced internally (that is first scenario described in [Section 12.5.2.4.2](#), the MCASP\_AHCLKXCTL[15] HCLKXM bit must be set to 1. Thus, the clock is divided down by a programmable high-clock divider (the MCASP\_AHCLKXCTL[11-0] HCLKXDIV bit field) from the MCASP internal clock source AUXCLK.

Internally, the MCASP always shifts transmit data at the rising edge of the internal transmit clock - XCLK, (see [Figure 12-498](#)). The CLKXP mux determines if ACLKX needs to be inverted to become XCLK. If MCASP\_ACLKXCTL[7] CLKXP = 0, the CLKXP mux directly passes ACLKX signal to XCLK. As a result, the

MCASP shifts transmit data at the rising edge of ACLKX. If MCASP\_ACLKXCTL[7] CLKXP = 1, the CLKX mux passes the inverted version of ACLKX to XCLK. As a result, the MCASP shifts transmit data at the falling edge of ACLKX.

It can be seen in [Figure 12-498](#) that XCLK is propagated to the Rx clock logic, to allow an internally synchronous operation between MCASP transmitters and receivers. This is used for example in the MCASP loopback mode.

#### Note

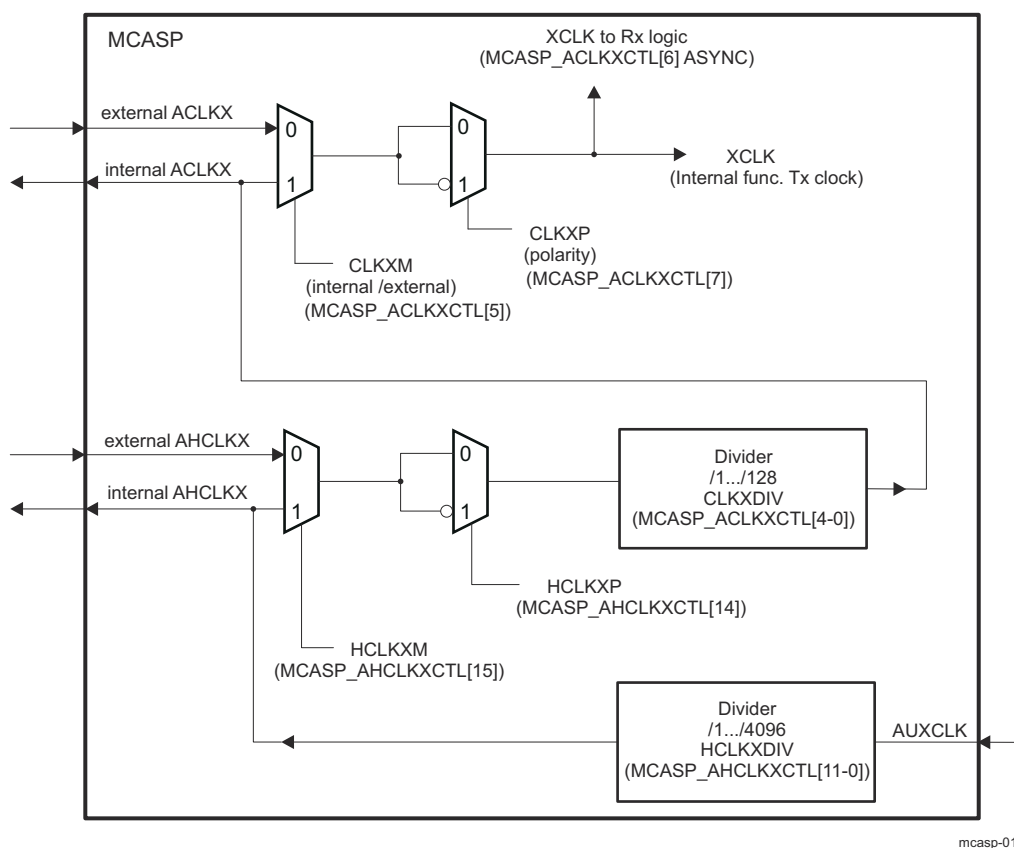
The polarity of ACLKX can be controlled in the MCASP\_ACLKXCTL[7] CLKXP bit, regardless of ACLKX signal being internally or externally sourced.

In addition, there is an option to invert polarity of the AHCLKX master high speed clock via writing the MCASP\_AHCLKXCTL[14] HCLKXP bit.

#### Note

In a similar way, the polarity of AHCLKX clock can be controlled in the MCASP\_AHCLKXCTL[14] HCLKXP bit, regardless of the AHCLKX signal being internally or externally sourced.

[Figure 12-498](#) presents the block diagram of the transmit clock generator.



**Figure 12-498. Transmit Clock Generator Block Diagram**



---

### Note

In this device:

- ACLKX is mapped on the device ball ACLKX
- internal AHCLKX is mapped on the device balls AUDIO\_EXT\_REFCLK[0-1]
- external AHCLKX is mapped on MCASPi\_AHCLKX clock from the Device Configuration

For more information about MCASP integration, see *MCASP Environment* and *MCASP Integration*.

---

#### 12.5.2.4.2.2 MCASP Receive Clock

The MCASP receive clock generator is built on a very similar to the transmit clock generator (but independent) circuit.

The receive clock configuration is controlled by the following registers:

- MCASP\_ACLKRCTL
- MCASP\_AHCLKRCTL

In case, the receive bit clock, ACLKR, is generated internally (but asynchronously to XCLK), the MCASP\_ACLKRCTL[5] CLKRM bit must be set to 1. Thus, the clock is divided down by a programmable bit clock divider (the MCASP\_ACLKRCTL[4-0] CLKRDIV bit field) from the source signal.

If the receive high-frequency master clock, AHCLKR, is also sourced internally (that is, first scenario described in [Section 12.5.2.4.2](#)) and the MCASP\_AHCLKRCTL[15] HCLKRM bit must be set to 1. Thus, the clock is divided down by a programmable high-clock divider (the MCASP\_AHCLKRCTL[11-0] HCLKRDIV bit field) from the MCASP internal clock source AUXCLK.

---

### Note

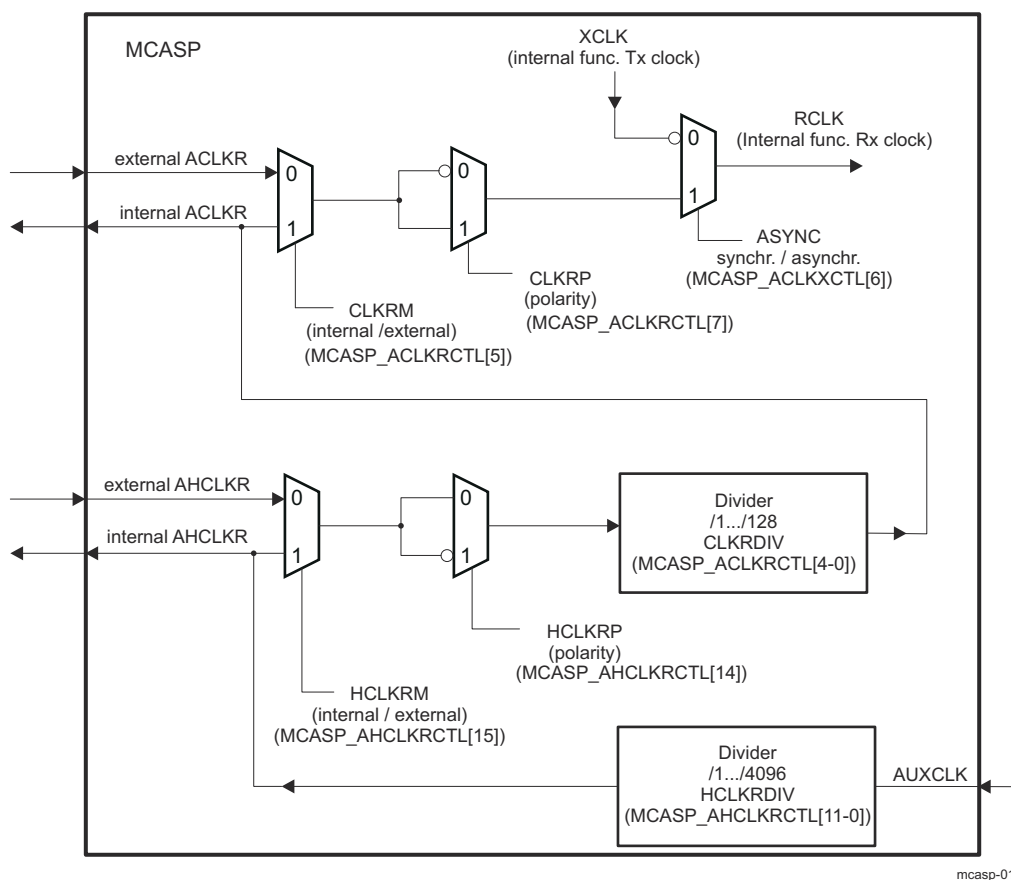
The polarity of ACLKR can be controlled in the MCASP\_ACLKRCTL[7] CLKRP bit, regardless of ACLKR signal being internally or externally sourced.

In a similar way, the polarity of AHCLKR clock can be controlled in the MCASP\_AHCLKRCTL[14] HCLKRP bit, regardless of the AHCLKR signal being internally or externally sourced.

---

There is an option for the MCASP receiver to be configured to operate synchronously to the ACLKX and AFSX signals. The XCLK output of the Tx Clock generator (see [Figure 12-498](#) and [Figure 12-499](#)) becomes source of the receive clock (RCLK output), when the MCASP\_ACLKXCTL[6] ASYNC bit in the transmit clock control register is set to '0b0'. For more information, refer to [Section 12.5.2.4.2.4](#).

[Figure 12-499](#) presents the block diagram of the receive clock generator.



**Figure 12-499. Receive Clock Generator Block Diagram**

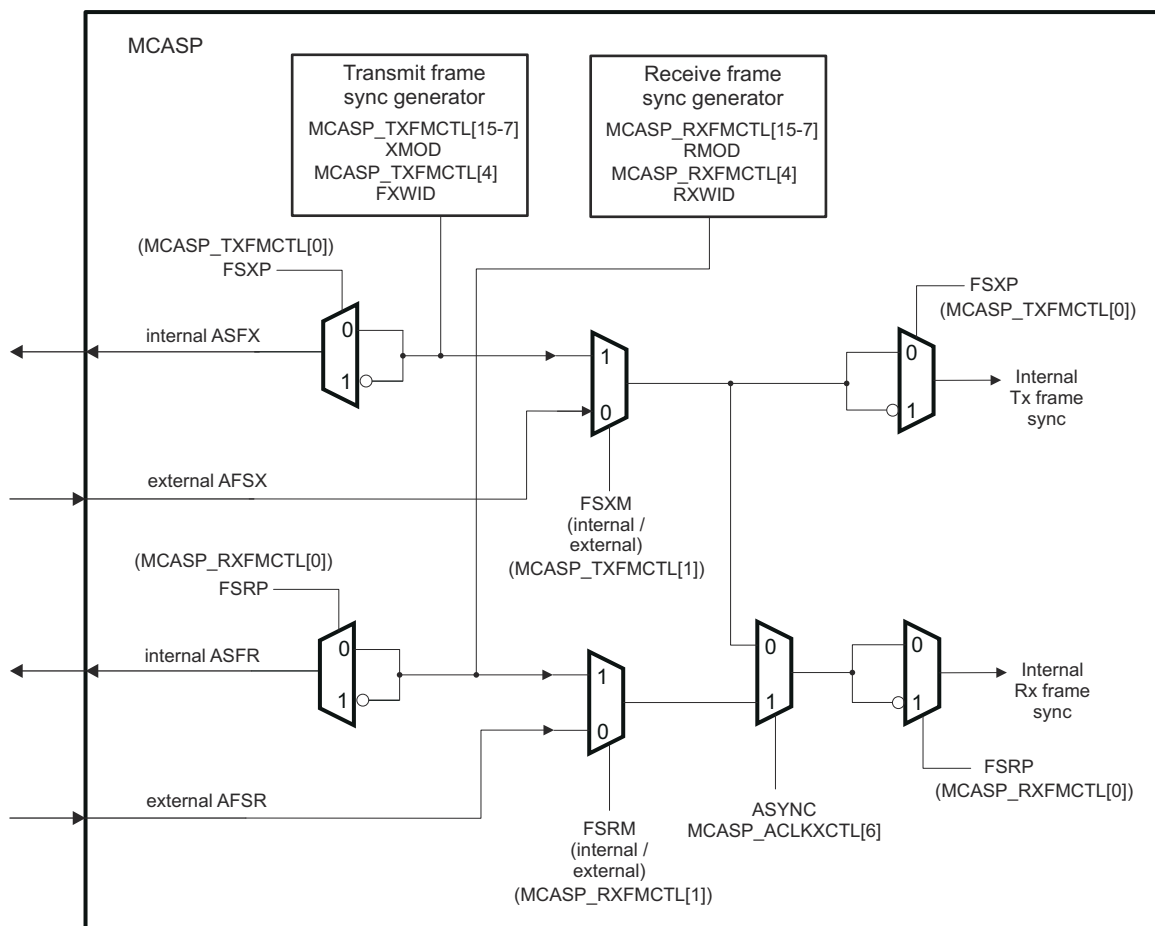
### Note

In this device:

- ACLKR is mapped on the device ball ACLKR
- internal AHCLKX is mapped on the device balls AUDIO\_EXT\_REFCLK[0-1]
- external AHCLKR is mapped on Device Configuration MCASPi\_AHCLKR from Device Configuration

#### 12.5.2.4.2.3 Frame-Sync Generator

There are two different modes for frame sync: burst and TDM. The MCASP frame sync generator logic is illustrated in [Figure 12-500](#). The I/O buffers are not part of the MCASP module, and are not shown in the figure.



mcasep-015

**Figure 12-500. Frame Sync Generator Block Diagram**

**For the transmit logic**, following frame-sync generator configurations can be selected:

- Internally/externally generated frame-sync via configuring MCASP\_AFSXCTL[1] FSXM bit
- Frame-sync polarity: Rising edge or falling edge via configuring MCASP\_AFSXCTL[0] FSXP bit
- Frame-sync width: "single bit" or "single word" via configuring MCASP\_AFSXCTL[4] FXWID bit
- Frame sync mode - the appropriate frame sync generation pattern for the selected transfer mode is defined in the MCASP\_AFSXCTL[15-7] XMOD bit field, as follows:
  - For DIT mode (384 slots) - MCASP\_AFSXCTL[15-7] XMOD = 0x180
  - For I2S mode (2 TDM slots) - MCASP\_AFSXCTL[15-7] XMOD = 0x2
  - For TDM mode (from 3 to 32 TDM slots) - MCASP\_AFSXCTL[15-7] XMOD bit field set in range 0x3 - 0x20
- Bit delay: 0, 1, or 2 cycles before the first data bit. This delay is defined in MCASP\_XFMT[17-16] XDADLY bit field

**For the receive logic**, following frame-sync generator configurations can be selected:

- Internally/externally generated frame-sync via configuring MCASP\_AFSRCTL[1] FSRM bit
- Frame-sync polarity: Rising edge or falling edge via configuring MCASP\_AFSRCTL[0] FSRP bit
- Frame-sync width: "single bit" or "single word" via configuring MCASP\_AFSRCTL[4] FRWID bit
- Frame sync mode - the appropriate frame sync generation pattern for the selected transfer mode is defined in the MCASP\_AFSRCTL[15-7] RMOD bit field, as follows:
  - For I2S mode (2 TDM slots) - MCASP\_AFSRCTL[15-7] RMOD = 0x2
  - For TDM mode (from 3 to 32 TDM slots) - MCASP\_AFSRCTL[15-7] RMOD set in range 0x3 - 0x20
  - For the special 384-slot TDM mode - MCASP\_AFSRCTL[15-7] RMOD = 0x180

- Bit delay: 0, 1, or 2 cycles before the first data bit. This delay is defined in the MCASP\_RFMT[17-16] RDATA bit field
- Selecting the source (AFSX or AFSR) of receiver internal frame synchronization. This is done in the same bit - MCASP\_ACLKXCTL[6] ASYNC, used to define the receiver internal clock source. For more details, refer to [Section 12.5.2.4.2.4, Synchronous and Asynchronous Transmit and Receive Operations](#).

Regardless of the AFSX/AFSR being internally generated or externally sourced, the polarity of AFSX/AFSR is determined by FSXP/FSRP, respectively, to be either rising or falling edge. If FSXP/FSRP = 0, the frame sync polarity is rising edge. If FSXP/FSRP = 1, the frame sync polarity is falling edge.

#### Note

Certain restrictions apply to the receive and transmit logic settings, when the MCASP\_ACLKXCTL[6] ASYNC bit is set to 0b0. They are described in [Section 12.5.2.4.2.4](#).

#### 12.5.2.4.2.4 Synchronous and Asynchronous Transmit and Receive Operations

##### Synchronous Transmit and Receive Operations -

When the MCASP\_ACLKXCTL[6] ASYNC bit is written to 0b0, the transmit and receive sections operate synchronously to the transmit section clock and transmit frame sync signals.

Though Rx section may have a different data format, it has to be configured to have the same slot size than the transmit section one. As shown on the [Figure 12-499](#), with the ASYNC bit set to 0b0, the RCLK becomes an inverted version of the transmit clock generator XCLK output.

When the MCASP\_ACLKXCTL[6] ASYNC = 0b0, both Rx and Tx sections use the same clock and frame sync signals. For this reason, they must be aligned on the following settings:

- MCASP\_DITCTL[0] DITEN = 0 (that is, transmission in TDM mode is enabled)
- The total number of bits per frame must be the same (that is, RSSZ \* RMOD product value must equal XSSZ \* XMOD product value)
- The settings in the MCASP\_ACLKRCTL register are NOT considered
- FSXM must match FSRM
- FXWID must match FRWID

For all other settings, the transmit and receive sections may be programmed independently.

##### Asynchronous Transmit and Receive Operations -

When the MCASP\_ACLKXCTL[6] ASYNC = 0b1, Tx and Rx operate independently from each other with separate clock and frame sync signals.

#### Note

Synchronous transmit and receive operations are allowed only in the MCASP TDM (I2S) mode (this is, when MCASP\_DITCTL[0] DITEN = 0b0).

#### 12.5.2.4.3 MCASP Frame Sync Feedback for Cross Synchronization

The device level requirement is to align frame syncs across McASP that may be feeding DACs with different formats but the same frequency.

The intent of is to provide a mechanism to check for the alignment of frame syncs. This works by feeding the frame sync of McASP into the a receive data pin of McASP. If this serializer is enabled as a receiver, then the frame sync of McASP will appear as receive data for McASP.

To adjust the McASP transmit bit clock, the MCASP\_ACLKXCTL[17-16] CLKXADJ register bit field can be used to lengthen (write 0x2) or reduce (write 0x1) the McASP bit clock period in a one-shot fashion. After each adjustment the frame sync should be checked again and the process repeated until the two frame syncs are aligned.

This process is expected to be carried out once, before any audio transfers occur. It is not intended to be used for sample rate conversion but rather initial phase alignment of transmit data across MCASP.

#### 12.5.2.4.4 MCASP Serializers

The MCASP serializers shift serial data in (Rx) and out (Tx) of the MCASP. A given serializer  $n$  consists of a shift register (XRSR $n$ ) with a single-entry data buffer XRBUFF $n$  used either for transmitting (write accessible in the MCASP\_XBUFF $n$  register) or for receiving (read accessible in the MCASP\_RBUFF $n$  register) data. In addition, each serializer has a dedicated control register (MCASP\_SRCTL $n$ ) and a serial bidirectional data pin - AXR $n$ . The register MCASP\_SRCTL $n$  allows  $n$ -th serializer to be configured as a transmitter, receiver, or as inactive. There are transmit and receive data formatting units to support data alignment options of the MCASP which are shared between all Tx and Rx serializers, respectively.

A given serializer XRSR $n$  shifter configured as a receiver in the MCASP\_SRCTL $n$  register, shifts in data through MCASP corresponding device level bidirectional data pad AXR $n$ . A given serializer XRSR $n$  shifter configured as a transmitter in the MCASP\_SRCTL $n$  register, shifts out data on MCASP corresponding device level bidirectional data pad AXR $n$  ( $n = 0$  to  $15$ ).

The serializer is clocked from the transmit section clock (ACLKX signal) if configured to transmit or clocked from the receive section clock (ACLKR signal) if configured to receive. A serializer configured to transmit and receive operates in lockstep, which means that for MCASP there are at most a couple of zones, one for transmit and one for receive.

Figure 12-501 shows the serializer block diagram.

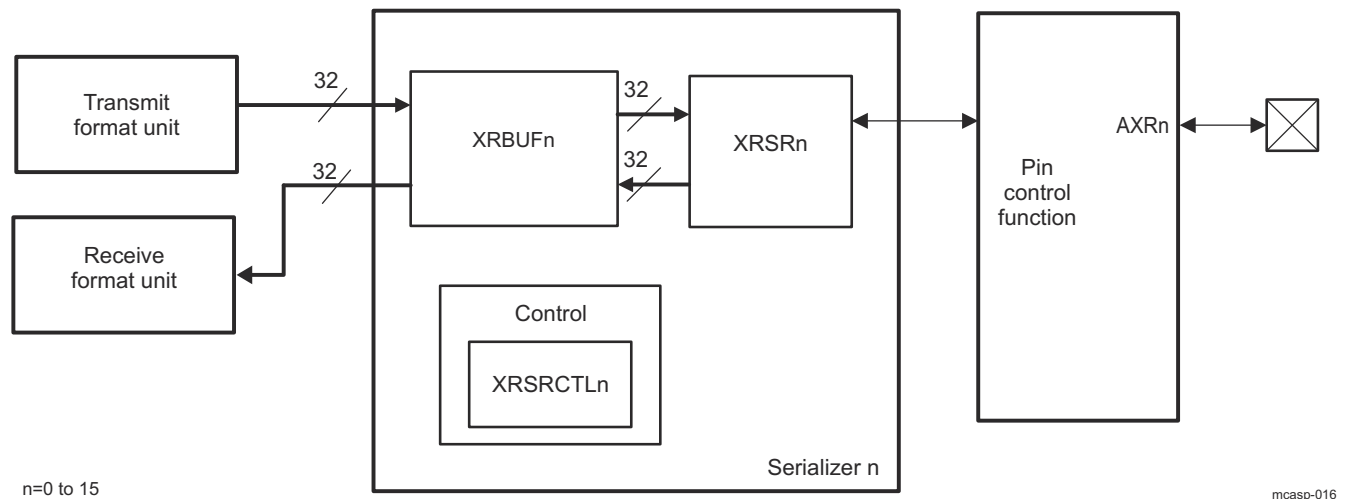


Figure 12-501. Individual Serializer and Connections Within MCASP

#### Transmission on the $n$ -th serializer is performed as follows:

The MCASP is serviced by writing data into the MCASP\_XBUFF $n$  register, which is an alias of the serializer data buffer - XRBUFF $n$  for transmit function. The data automatically passes through the transmit format unit before reaching the XRBUFF $n$  register in the serializer. The data is then copied from the XRBUFF $n$  to XRSR $n$  and shifted out from AXR $n$  synchronously to the serial clock.

#### Reception from the $n$ -th serializer is performed as follows:

The data is shifted into the MCASP XRSR $n$  serializer register through the AXR $n$  pin, bit by bit. Once the entire slot becomes available within the XRSR $n$  shift register, the data is copied into the serializer data buffer - XRBUFF $n$ , and can be accessed in the MCASP\_RBUFF $n$  register, which is an alias of the serializer data buffer - XRBUFF $n$  for receive function. When software reads the data from this register, the MCASP passes the data through the receive format unit, hence it returns the formatted data.

#### Serializer controls:

A serializer  $n$  is configured as inactive via setting MCASP\_SRCTL $n$ [1-0] SRMOD bit field to 0x0.

For a transmitting serializer, the MCASP\_SRCTL $n$ [3-2] DISMOD bitfield, defines the AXR $n$  pin output state, during inactive slots (HIGH, LOW or Hi-Z).

Transmit function for the  $n$ -th serializer is selected via setting MCASP\_SRCTL $n$ [1-0] SRMOD bit field to 0x1.

Receive function for the  $n$ -th serializer is selected via setting MCASP\_SRCTL $n$ [1-0] SRMOD bit field to 0x2 ( $n = 0$  to 15).

In the DIT-transmission mode (that is S/PDIF format data transmission): in addition to the data, the serializer shifts out other DIT-specific information accordingly (preamble, user data, etc.). For more information, see *S/PDIF Coding Format*.

#### 12.5.2.4.5 MCASP Format Units

The MCASP has one transmit data formatting unit and one receive data formatting unit, shared between the device MCASP serializers. These units automatically remap the data bits within the transmitted or received words between a natural format for the device processors (for example, a Q31 representation) and the required format for the external serial device (for example I2S format). During the remapping process, the format unit can also mask off certain bits.

Since all transmitters share the same data formatting unit, the MCASP only supports one transmit format at a time. For example, the MCASP does NOT transmit in "I2S format" on serializer 0, while transmitting "Left Justified" on serializer 1. Likewise, the receiver section of the MCASP only supports one data format at a time, and this format applies to all receiving serializers.

#### Note

The MCASP can transmit in one format while receiving in a completely different format.

The bit mask and pad stage of each of Tx and Rx format units includes a full 32-bit mask register, allowing selected individual bits to either pass through the stage unchanged, or be masked off. The bit mask and pad then pad the value of the masked off bits by inserting either a 0, a 1, or one of the original 32 bits as the pad value. The last option allows for sign-extension when the sign bit is selected to pad the remaining bits. The rotate right stage performs bitwise rotation by a multiple of 4 bits (between 0 and 28 bits), programmable by the MCASP\_RFMT/MCASP\_XFMT register. Note that this is a rotation process, not a shifting process, so bit 0 gets shifted back into bit 31 during the rotation. The bit order - reversal stage either passes all 32 bits directly through, or swaps them. This allows for either MSB or LSB first data formats. If bit order reversal is not enabled, then the MCASP will naturally transmit and receive in an LSB first order. Finally, note that the RDATDLY/XDATDLY bits in the MCASP\_RFMT/MCASP\_XFMT also determine the data format. For example, the difference between I2S format and left-justified is determined by the delay between the frame sync edge and the first data bit of a given time slot. For I2S format, RDATDLY/XDATDLY should be set to a 1-bit delay, whereas for left-justified format, it should be set to a 0-bit delay. The combination of all the options in the MCASP\_RFMT/MCASP\_XFMT register means that the MCASP supports a wide variety of data formats, both on the serial data lines, and in the device CPU data representation.

##### 12.5.2.4.5.1 Transmit Format Unit

The MCASP transmit formatting unit consists of three stages :

- Bit mask (masks off bits)
- Rotate right (aligns data within word)
- Bit reversal (selects between MSB-first or LSB-first)

Figure 12-502 shows the transmit formatting unit.

The MCASP transmitter supports serial formats of:

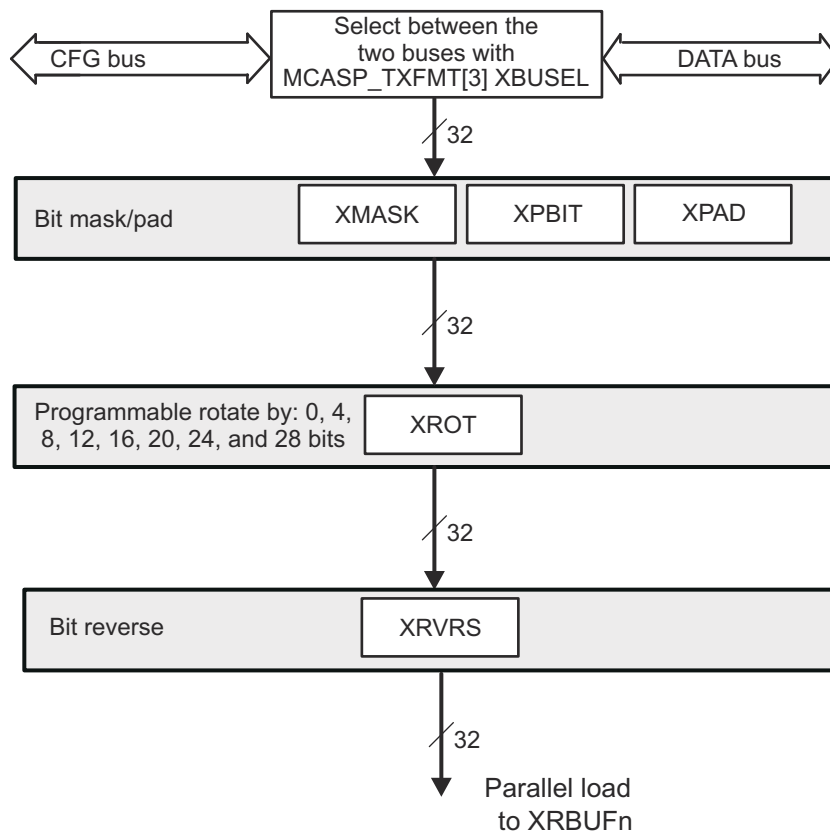
- Slot (or Time slot) size = 8, 12, 16, 20, 24, 28, 32 bits
- Word size  $\leq$  Slot size

- Alignment: when more bits/slot than bits/words, then:
  - Left aligned = word shifted first, remaining bits are pad
  - Right aligned = pad bits are shifted first, word occupies the last bits in slot
- Order of bits shifted out:
  - MSB: most-significant bit of word is shifted out first, last bit is LSB
  - LSB: least-significant bit of word is shifted out last, last bit is MSB

Hardware support for these serial formats comes from the programmable options in the bitstream format register

- MCASP\_XFMT:

- XRVRS: bit reverse (1) or no bit reverse (0)
- XROT: rotate right by 0, 4, 8, 12, 16, 20, 24, or 28 bits
- XSSZ: transmit slot size of 8, 12, 16, 20, 24, 28, or 32 bits



mcasp-017

**Figure 12-502. Transmit Format Unit**

As shown in [Figure 12-502](#), the data to the transmit format unit can come from the configuration port (CFG) or the data port (DATA). The selection is made through the MCASP\_XFMT[3] XBUSEL bit. According to port type selected, data transfer has different behaviour. For more details, refer to the [Section 12.5.2.4.11.1.3, Transfers Through the DATA Port](#), and [Section 12.5.2.4.11.1.4, Transfers Through the Configuration \(CFG\) Bus](#).

In the transmit format unit (TFU), the input data bits are first masked-off with the MCASP\_XMASK[31-0] XMASK contents. The masked data is then right-rotated to the MCASP\_XFMT[2-0] XROT bit field positions, to produce the output word for a TDM- or DIT- transmission.

The bit mask stage includes a full 32-bit mask register, allowing selected individual bits to pass through the stage unchanged or be masked off.



### 12.5.2.4.5.1.1 TDM Mode Transmission Data Alignment Settings

The TDM-mode transmission settings are relevant for I2S-protocol and protocols using more than 2 TDM-slots. XSSZ should always be programmed to match the slot size of the serial stream.

#### Note

Note that, TDM word size is not directly programmed into the MCASP, but rather is used to determine the rotation needed in the XROT field.

The [Table 12-496](#) show the XRVRS and XROT fields for each serial format and for both integer and Q31 fractional internal representations.

The [Table 12-496](#) assumes that all slot size (SLOT in [Table 12-496](#)) and word size (WORD in [Table 12-496](#)) options are multiples of 4, since the transmit rotate right unit only supports rotation by multiples of 4. However, the bit mask/pad unit does allow for any number of significant digits. For example, a Q31 number may have 19 significant digits (word) and be transmitted in a 24-bit slot; this would be formatted as a word size of 20 bits and a slot size of 24 bits. However, it is possible to set the bit mask unit to only pass the 19 most-significant digits (program the mask value to FFFF E000h). The digits that are not significant can be set to a selected pad value, which can be any one of the significant digits, a fixed value of 0, or a fixed value of 1.

The transmit bit mask/pad unit operates on data as an initial step of the transmit format unit, and the data is aligned in the same representation as it is written to the transmitter (typically Q31 or integer).

**Table 12-496. MCASP TFU TDM Mode Settings**

Bit Stream Order	Bit Stream Alignment	Internal Numeric Representation	MCASP_XFMT bits	
			XROT <sup>(1)</sup>	XRVRS
MSB first <sup>(2)</sup>	Left aligned	Q31 fraction	0	1
MSB first	Right aligned	Q31 fraction	SLOT - WORD	1
LSB first	Left aligned	Q31 fraction	32 - WORD	0
LSB first	Right aligned	Q31 fraction	32 - SLOT	0
MSB first <sup>(2)</sup>	Left aligned	Integer	WORD	1
MSB first	Right aligned	Integer	SLOT	1
LSB first	Left aligned	Integer	0	0
LSB first	Right aligned	Integer	(32 - (SLOT - WORD)) % 32	0

(1) WORD = Word size rounded up to the nearest multiple of 4; SLOT = slot size; % = modulo operator

(2) To transmit in I2S format, select MSB first, left aligned, and also select XDATDLY = 01 (1 bit delay)

### 12.5.2.4.5.1.2 DIT Mode Transmission Data Alignment Settings

In case of a DIT-mode (S/PDIF protocol) transmission, while left-aligned Q31 data should be right-rotated to a multiple by 4 positions, no right-rotation is required for a right-aligned Q31 data. Because this is a rotation process, not a shifting process, bit 0 gets shifted back into bit 31 during the process.

The MCASP\_XFMT[17-16] XDATDLY bit field must be set to a 0-bit delay (0x0 value).

For left-aligned Q31 data, the following transmit format unit settings process the data into right-aligned data, ready for transmission:

- MCASP\_XFMT[2-0] XROT =
  - 0x2 (rotate right by 8 bits) - for a 24-bit output audio data
  - 0x3 (rotate right by 12 bits) - for a 20-bit output audio data
  - 0x4 (rotate right by 16 bits) - for a 16-bit output audio data
- MCASP\_XFMT[15] XRVRS = 0x0 – Bit reversal is not enabled; the MCASP naturally transmits and receives in a LSB-first order.
- MCASP\_XMASK[32] XMASK = 0xFFFFFFFF00 – 0xFFFF0000
- MCASP\_XFMT[14-13] XPAD = 0x0 (Pad extra bits with 0s.)



For right-aligned data, the following transmit format unit settings process the data into right-aligned audio data ready for transmission:

- MCASP\_XFMT[2-0] XROT = 0x0 (rotate right by 0 bits regardless of the audio word length)
- MCASP\_XFMT[15] XRVRS = 0x0 – Bit reversal is not enabled; the MCASP naturally transmits and receives in a LSB-first order.
- MCASP\_XMASK[32] XMASK = 0x00FFFFFF – 0x0000FFFF
- MCASP\_XFMT[14-13] XPAD = 0x0 (Pad extra bits with 0s.)

The example settings provided in [Table 12-497](#) should be applied to MCASP in cases of DIT-transmitting a Q31 data as a 24-bit, 20-bit and 16-bit left- or right- aligned audio word, respectively. Note that the listed settings let the MCASP TFU preserve the most significant bits and cut only the LSBs of the original Q31 CPU data:

**Table 12-497. MCASP TFU DIT-Mode Example Settings**

Output Audio Word Alignment	Audio Word Length	Right-rotation (multiple of 4-bit positions)	XMASK	XROT
LEFT	16	16	0xFFFF0000	0x4
LEFT	20	12	0xFFFF000	0x3
LEFT	24	8	0xFFFF00	0x2
RIGHT	16	0	0x0000FFFF	0x0
RIGHT	20	0	0x000FFFFF	0x0
RIGHT	24	0	0x00FFFFFF	0x0

Assuming that a Q31 data word 0xFA5AFxxx (where x-marked nibbles of the data are applied as padding bits of the word) is generated on the MCASP CFG (peripheral) port. To transmit a left-aligned 20-bit version of same word, preserving the MSBs, according to the [Table 12-497](#), the user must set XMASK = 0xFFFF000, and to select a right-rotation to 12 positions (XROT = 0x3).

- After applying 0-s (XPAD = 0) as masking-off bits at the first TFU stage, word is transformed to the word 0xFA5AF000.
- After a rotation by 12 positions to the right is performed in TFU, the 20-bit output word obtained is: 0x000FA5AF. Thus the word gets ready for transmission being mapped with its LS-bit as bit 8 and its MS-bit as bit 27 within a S/PDIF bitstream. This word is shifted in a LSB-to-MSB order (XRVRS = 0x0) out of the XRSR register during a DIT-transmission.

Assuming that a right-aligned Q31 data word - 0x yyyyE4B4 is generated by software on the MCASP CFG (peripheral) port (with the presumption that y-marked nibbles of the input data are applied as padding bits). To transmit a right-aligned 16-bit version of same word, preserving the MSBs, according to the table MCASP TFU Example Settings, user is supposed to set XMASK = 0x0000FFFF, and to select right-rotation to 0 positions (XROT = 0x0).

- After masking-off with 0s at first TFU stage, word is transformed to 0x0000E4B4.
- Since no rotation is applied, the 16-bit output word obtained is actually the one obtained in the masking stage – 0x0000E4B4.

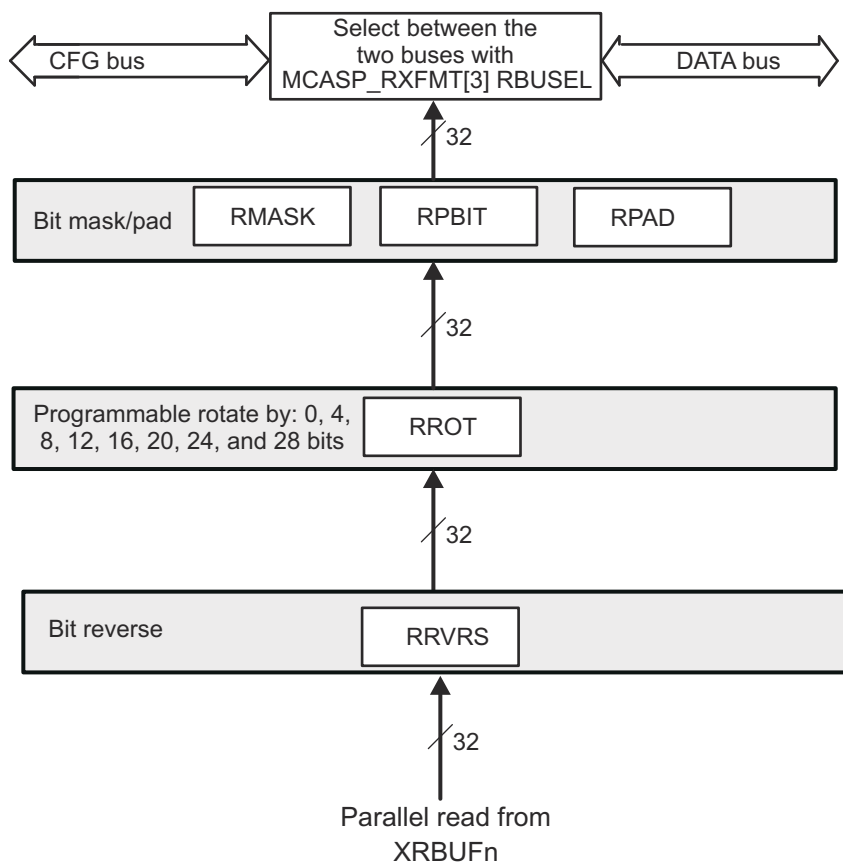
The above examples use internal representation in integer and Q31 notation, but other fractional notations are also possible.

#### 12.5.2.4.5.2 Receive Format Unit

The MCASP receive formatting unit consists of three stages:

- Bit mask (masks off bits)
- Rotate right (aligns data within word)
- Bit reversal (selects between MSB first or LSB first)

[Figure 12-503](#) shows the receive format unit (RFU).



mcasp-018

**Figure 12-503. Receive Format Unit**

The MCASP receiver supports serial formats of:

- Slot or time slot size = 8, 12, 16, 20, 24, 28, 32 bits
- Word size ≤ Slot size
- Alignment when more bits are available per slot than bits per word within the slot, then:
  - Left aligned = word shifted first, remaining bits are pad
  - Right aligned = pad bits are shifted first, word occupies the last bits in slot
- Order of bits shifted out:
  - MSB: most-significant bit of word is shifted out first, last bit is LSB
  - LSB: least-significant bit of word is shifted out last, last bit is MSB

Hardware support for these serial formats comes from the programmable options in the receive bitstream format register - MCASP\_RFMT:

- RRVRS: bit reverse (1) or no bit reverse (0)
- RROT: rotate right by 0, 4, 8, 12, 16, 20, 24, or 28 bits
- RSSZ: receive slot size of 8, 12, 16, 20, 24, 28, or 32 bits

As shown on [Figure 12-503](#), the data processed in the RFU can be output to host CPU through the configuration port (CFG) or the data port (DATA). The selection is made through the MCASP\_RFMT[3] RBUSEL bit. According to port type selected, data transfer has different behaviour. For more details, refer to the [Section 12.5.2.4.11.1.3, Transfers Through the DATA Port](#), and [Section 12.5.2.4.11.1.4, Transfers Through the Configuration \(CFG\) Bus](#).

#### 12.5.2.4.5.2.1 TDM Mode Reception Data Alignment Settings

RSSZ should always be programmed to match the slot size of the serial stream.

### Note

Note that the word size is not directly programmed into the MCASP, but rather is used to determine the rotation needed in the RROT field.

Table 12-498 shows the RRVRS and RROT fields for each serial format and for both integer and Q31 fractional internal representations.

**Table 12-498. MCASP RFU Settings**

Bit Stream Order	Bit Stream Alignment	Internal Numeric Representation	MCASP_RFMT bits	
			RROT <sup>(1)</sup>	RRVRS
MSB first <sup>(2)</sup>	Left aligned	Q31 fraction	SLOT	1
MSB first	Right aligned	Q31 fraction	WORD	1
LSB first	Left aligned	Q31 fraction	$(32 - (\text{SLOT} - \text{WORD})) \% 32$	0
LSB first	Right aligned	Q31 fraction	0	0
MSB first <sup>(2)</sup>	Left aligned	Integer	SLOT - WORD	1
MSB first	Right aligned	Integer	0	1
LSB first	Left aligned	Integer	32 - SLOT	0
LSB first	Right aligned	Integer	32 - WORD	0

(1) WORD = Word size rounded up to the nearest multiple of 4; SLOT = slot size; % = modulo operator

(2) To receive in I2S format, select MSB first, left aligned, and also select RDATDLY = 01 (1 bit delay)

The Table 12-498 assumes that all slot size and word size options are multiples of 4; since the receive rotate right unit only supports rotation by multiples of 4. However, the bit mask/pad unit does allow for any number of significant digits. For example, a Q31 number may have 19 significant digits (word) and be received in a 24-bit slot; this would be formatted as a word size of 20 bits and a slot size of 24 bits. However, it is possible to set the bit mask unit to only pass the 19 most-significant digits (program the mask value to FFFF E000h). The digits that are not significant can be set to a selected pad value, which can be any one of the significant digits, a fixed value of 0, or a fixed value of 1. The receive bit mask/pad unit operates on data as the final step of the receive format unit (see Figure 12-503), and the data is aligned in the same representation as it is read from the receiver (typically Q31 or integer).

#### 12.5.2.4.6 MCASP State-Machines

The receive and transmit sections have independent state machines.

Each state-machine controls the interactions between the various units in the MCASP Rx and Tx sections, respectively. In addition, each state-machine keeps track of error conditions and serial port status. No serial transfers can occur until the RX/TX state-machine is released from reset.

The transmit state-machine is controlled by the transmit bitstream format register (MCASP\_XFMT) and it reports the MCASP status and error conditions in the transmitter status register (MCASP\_XSTAT).

Similarly, the receive state-machine is controlled by the receive bitstream format register (MCASP\_RFMT) and it reports the MCASP status and error conditions in the receiver status register (MCASP\_RSTAT).

#### 12.5.2.4.7 MCASP TDM Sequencers

There are separate TDM sequencers for the transmit section and the receive section. Each TDM sequencer keeps track of the slot count. In addition, the TDM sequencer checks the bits of the MCASP\_RTDM/MCASP\_XTDM register and determines if the MCASP should receive/transmit in that time slot.

There are two possibilities for a slot: The MCASP either performs Rx/Tx operations during the time slot (transmit/receive bit is active), or the MCASP skips Rx/Tx operations during the time slot (transmit/receive bit is inactive). In the latter case, no transfers between the XRBUFF and XRSR registers in the serializer would occur during that time slot.

In addition, during time of inactive slots, the serializers programmed as transmitters place their data output pins - AXRn in a predetermined state - logic low, high, or high impedance (tri-stated) as programmed in each serializer control MCASP\_SRCTLn[3-2] DISMOD register. Refer also to [Section 12.5.2.4.10.2.1, TDM Time Slots Generation and Processing](#), for details on how DMA event or interrupt generations are handled during inactive time slots in TDM mode.

**In case of a DIT-transmission (S/PDIF transfers):** the time division multiplexing (TDM) sequencer is used to count the 384 subframes (slots) in the DIT block. If currently transmitting slot 1, slot 2 (next value of the TDM slot counter) should be used during the encode phase to select the appropriate C, V, and U bit, because the data encoded and written to a MCASP\_XBUFn register during the current time slot (slot 1) is actually shifted out on the next time slot (n = 0 to 15).

The transmit TDM sequencer is controlled by the MCASP\_XTDM register and reports the current transmit slot to the MCASP\_XTDMSLOT[9-0] XSLOT CNT bit field.

#### 12.5.2.4.8 MCASP Software Reset

The MCASP can be put into reset through the global transmit and receive control register (MCASP\_GBLCTL). A valid serial clock must be supplied to the MCASP to assert the software reset bits in the MCASP\_GBLCTL register.

#### 12.5.2.4.9 MCASP Power Management

[Table 12-499](#) describes power-management features available to the MCASP.

**Table 12-499. Local Power-Management Features**

Feature	Registers	Description
Slave clock stop modes	MCASP_PWRIDLESYSCONFIG[1-0] IDLE_MODE	Force-clock stop, no-clock stop, and smart-clock stop modes are available.

#### CAUTION

No wakeup schema is supported for the MCASP. To ensure a correct behavior after enabling MCASP at device Device Configuration level, the user software is strongly recommended to choose *No Idle* mode, setting MCASP\_PWRIDLESYSCONFIG[1-0] IDLE\_MODE bit field to 0x1. Before disabling MCASP at device Device Configuration level, user software is strongly recommended to choose a *Smart-Idle* mode, setting MCASP\_PWRIDLESYSCONFIG[1-0] IDLE\_MODE bit field to 0x2.

#### 12.5.2.4.10 MCASP Transfer Modes

##### 12.5.2.4.10.1 Burst Transfer Mode

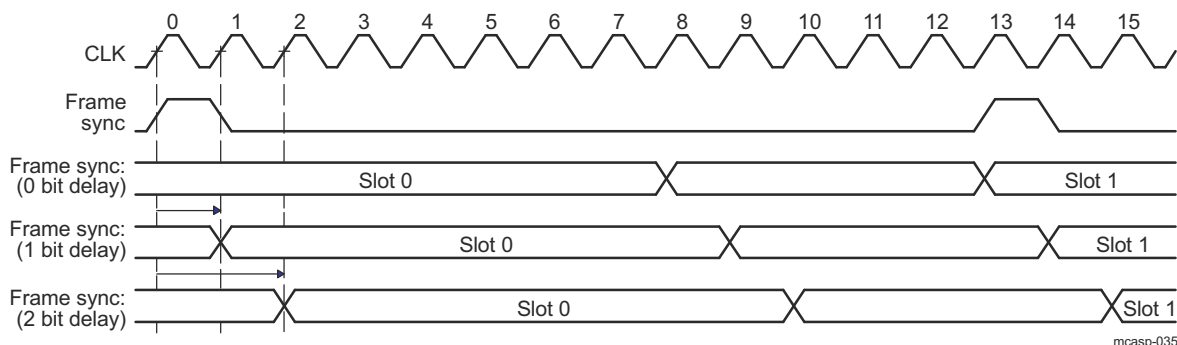
The MCASP supports a burst transfer mode, which is useful for nonaudio data such as passing control information between two processors. Burst transfer mode uses a synchronous serial format similar to the TDM mode. The frame sync generation is not periodic or time-driven as in TDM mode, but data driven, and the frame sync is generated for each data word transferred.

When operating in burst frame sync mode (see [Figure 12-504](#)), as specified for transmit (MCASP\_AFSXCTL[15-7] = 0) and receive (MCASP\_AFSRCTL[15-7] RMOD = 0), one slot is shifted for each active edge of the frame sync signal that is recognized. Additional clocks after the slot and before the next frame sync edge are ignored.

In burst frame sync mode, the frame sync delay may be specified as 0, 1, or 2 serial clock cycles. This is the delay between the frame sync active edge and the start of the slot. The frame sync signal lasts for a single bit clock duration (MCASP\_AFSRCTL[4] FRWID = 0, MCASP\_AFSXCTL[4] FXWID = 0).

For transmit, when generating the transmit frame sync internally, the frame sync begins when the previous transmission has completed and when all the XBUF<sub>n</sub> (for every serializer set to operate as a transmitter) has been updated with new data.

For receive, when generating the receive frame sync internally, frame sync begins when the previous transmission has completed and when all the RBUF<sub>n</sub> (for every serializer set to operate as a receiver) has been read.



**Figure 12-504. Burst Frame Sync Mode**

The control registers must be configured as follows for the burst transfer mode. The burst mode specific bit fields are in bold face:

- MCASP\_PFUNC: The clock, frame, data pins must be configured for MCASP function.
- MCASP\_PDIR: The clock, frame, data pins must be configured to the direction desired.
- MCASP\_PDOUT, MCASP\_PDIN, MCASP\_PDCLR: Not applicable. Leave at default.
- MCASP\_GBLCTL: Follow the initialization sequence in *MCASP Global Initialization*, to configure this register.
- MCASP\_AMUTE: Not applicable. Leave at default.
- MCASP\_DLBCTL: If loopback mode is desired, configure this register according to [Section 12.5.2.4.15, MCASP Loopback Modes](#), otherwise leave this register at default.
- MCASP\_DITCTL: DITEN must be left at default 0 to select non-DIT mode. Leave the register at default.
- MCASP\_RMASK/MCASP\_XMASK: Mask desired bits according to [Section 12.5.2.4.5, MCASP Format Units](#).
- MCASP\_RFMT/MCASP\_XFMT: Program all fields according to data format desired. See [Section 12.5.2.4.5, MCASP Format Units](#).
- MCASP\_RFMT/MCASP\_XFMT: Clear RMOD/XMOD bits to 0 to indicate burst mode. Clear FRWID/FXWID bits to 0 for single bit frame sync duration. Configure other fields as desired.
- MCASP\_ACLKRCTL/MCASP\_ACLKXCTL: Program all fields according to bit clock desired. See [Section 12.5.2.4.2, MCASP Clock and Frame-Sync Configurations](#).
- MCASP\_AHCLKRCTL/MCASP\_AHCLKXCTL: Program all fields according to high-frequency clock desired. See [Section 12.5.2.4.2, MCASP Clock and Frame-Sync Configurations](#).
- MCASP\_RTDM/MCASP\_XTDM: Program RTDMS0/XTDMS0 to 1 to indicate one active slot only. Leave other fields at default.
- MCASP\_RINTCTL/MCASP\_XINTCTL: Program all fields according to interrupts desired.
- MCASP\_RCLKCHK/MCASP\_XCLKCHK: Not applicable. Leave at default.
- MCASP\_SRCTL<sub>n</sub>: Program SRMOD to inactive/transmitter/receiver as desired. DISMOD is not applicable and should be left at default (n = 0 to 15).
- MCASP\_DITCSR<sub>Ai</sub>, MCASP\_DITCSR<sub>Bi</sub>, MCASP\_DITUDR<sub>Ai</sub>, MCASP\_DITUDR<sub>Bi</sub>: Not applicable. Leave at default (i = 0 to 5).

#### 12.5.2.4.10.2 Time-Division Multiplexed (TDM) Transfer Mode

The MCASP time-division multiplexed (TDM) transfer mode supports the TDM format discussed in *TDM Format*.

Transmitting data in the TDM transfer mode requires a minimum set of pins:

- ACLKX - transmit bit clock

- AFSX - transmit frame sync (or commonly called left/right clock)
- One or more serial data pins, AXRn, whose serializers are configured to transmit

For more details on MCASP transmitting serializers clock and frame sync options, refer to the [Section 12.5.2.4.2.1, Transmit Clock](#), and [Section 12.5.2.4.2.3, Frame-Sync Generator](#).

Similarly, to receive data in the TDM transfer mode requires a minimum set of pins:

- ACLKR - receive bit clock
- AFSR - receive frame sync (or commonly called left/right clock)
- One or more serial data pins, AXRn, whose serializers are configured to receive

For more details on MCASP receiving serializers clock and frame sync options, refer to [Section 12.5.2.4.2.2, Receive Clock](#), and [Section 12.5.2.4.2.3, Frame-Sync Generator](#).

The control registers must be configured as follows for the TDM mode. The TDM mode specific bit fields are highlighted in bold:

- MCASP\_PFUNC: The clock, frame, data pins must be configured for MCASP function.
- MCASP\_PDIR: The clock, frame, data pins must be configured to the direction desired.
- MCASP\_PDOUT, MCASP\_PDIN, MCASP\_PDCLR: Not applicable. Leave at default.
- MCASP\_GBLCTL: Follow the initialization sequence is described in the [Section 12.5.2.5.2, MCASP Operational Modes Configuration](#).
- MCASP\_AMUTE: Leave this register at default state.
- MCASP\_DLBCTL: If loopback mode is desired, configure this register according to [Section 12.5.2.4.15](#), otherwise leave this register at default.
- MCASP\_DITCTL: DITEN must be left at default 0 to select TDM mode (transmitters only).
- MCASP\_RMASK/MCASP\_XMASK: Mask desired bits according to [Section 12.5.2.4.5, MCASP Format Units](#).
- MCASP\_RFMT/MCASP\_XFMT: Program all fields according to data format desired. See the [Section 12.5.2.4.5, MCASP Format Units](#).
- MCASP\_AFSRCTL/MCASP\_AFSXCTL: Set RMOD/XMOD bits to (0x2 - 0x20) for Rx/Tx (2- 32 slots) TDM mode. In addition, set RMOD to 0x180 if 384-slot TDM stream has to be received by MCASP. Configure other fields as desired.
- MCASP\_ACLKRCTL/MCASP\_ACLKXCTL: Program all fields according to bit clock desired. For more information, refer to [Section 12.5.2.4.2, MCASP Clock and Frame-Sync Configurations](#).
- MCASP\_AHCLKRCTL/MCASP\_AHCLKXCTL: Program all fields according to high-frequency clock desired. For more details, refer to [Section 12.5.2.4.2, MCASP Clock and Frame-Sync Configurations](#).
- MCASP\_RTDM/MCASP\_XTDM: Program all fields according to the time slot characteristics desired.
- MCASP\_XINTCTL: Program all fields according to transmit interrupts desired.
- MCASP\_RCLKCHK/MCASP\_XCLKCHK: Program all fields according to clock checking desired.
- MCASP\_SRCTLn: Program all fields according to serializer operation desired (n = 0 to 15).

#### Note

The MCASP\_DITCSRAi, MCASP\_DITCSRBi, MCASP\_DITUDRAi, MCASP\_DITUDRBi (i = 0 to 5) settings are NOT applicable in TDM transfer modes. They have to be kept at their default values.

#### 12.5.2.4.10.2.1 TDM Time Slots Generation and Processing

TDM mode on the MCASP can extend to support multiprocessor applications, with up to 32 time slots per frame. For each of the time slots, the MCASP may be configured to participate or to be inactive by configuring MCASP\_XTDM and/or MCASP\_RTDM registers.

The TDM sequencer (separate ones for transmit and receive) functions in this mode. The TDM sequencer counts the slots beginning with the frame sync. For each slot, the TDM sequencer checks the respective bit in either MCASP\_XTDM or MCASP\_RTDM register to determine if the MCASP transmits/receives in that time slot.



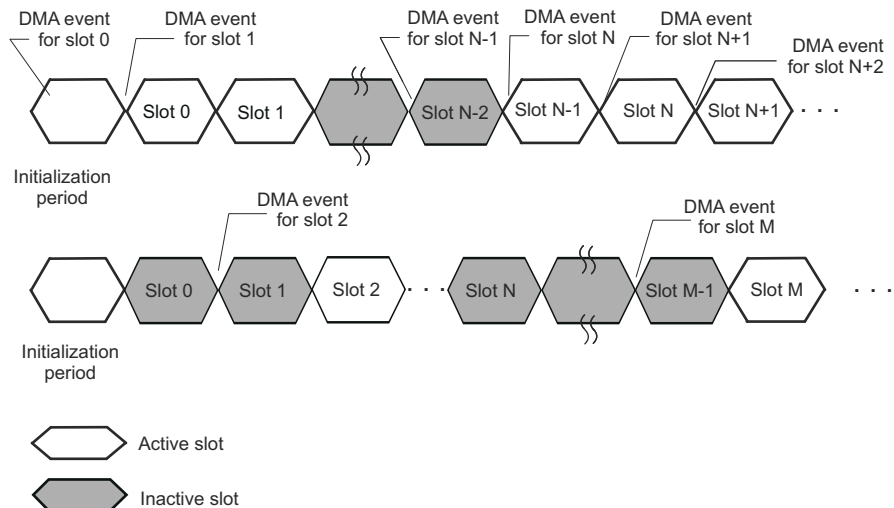
### Note

If a MCASP\_XTDM/MCASP\_RTDM bit defines an active slot (number of slot matches the bit position), the MCASP functions normally during that time slot; otherwise, the MCASP is inactive during that time slot; no update to the buffer occurs, and no event is generated. MCASP (transmit only) data pins are automatically set to a high-impedance state, 0, or 1 during that slot, as determined by bitfield MCASP\_SRCTLn[3-2] DISMOD (n = 0 to 15).

Figure 12-505 shows when the transmit DMA event - XINT is generated. See Section 12.5.2.4.11.1, *Data Ready Status and Event/Interrupt Generation* for details on data ready and the initialization period indication. The transmit DMA event for an active time slot (slot N) is generated during the previous time slot (slot N - 1), regardless of the previous time slot (slot N - 1) being active or inactive.

During an active transmit time slot (slot N), if the next time slot (slot N + 1) is configured to be active, the copy from XRBUFFn to XRSRn generates the DMA event for time slot N + 1. If the next time slot (slot N + 1) is configured to be inactive, then the DMA event will be delayed to time slot M - 1. In this case, slot M is the next active time slot. The DMA event for time slot M is generated during the first bit time of slot M - 1.

The receive DMA event is generated after data is received in the buffer (looks back in time). If a time slot is disabled, then no data is copied to the buffer for that time slot and no DMA event is generated.



mcasp-019

**Figure 12-505. Transmit DMA Event (XINT) Generation in TDM Time Slots**

#### 12.5.2.4.10.2.2 Special 384-Slot TDM Mode for Connection to External DIR

The MCASP receiver also supports a 384 time slot TDM mode (DIR mode), to support S/PDIF receiver ICs whose natural block (block corresponds to MCASP frame) size is 384 samples. The receive TDM time slot register (MCASP\_RTDM) should be programmed to all 1s during reception of a DIR block. Other TDM functionalities (for example, inactive slots) are not supported (only the slot counter counts the 384 subframes in a block). To receive data in DIR mode, the following pins are typically needed:

- ACLKR - receive bit clock
- AFSR - receive frame sync (or commonly called left/right clock)
- In this mode, AFSR should be connected to a DIR which outputs a start of block signal, instead of LRCLK
- One or more serial data pins, AXRn, whose serializers have been configured to receive
- For this special DIR mode, the control registers can be configured just as for TDM mode, except set MCASP\_AFSRCTL[15-7] RMOD bit field to 384 (0x180) to receive 384 time slots

### 12.5.2.4.10.3 DIT Transfer Mode

The DIT transfer mode of the MCASP also supports transmission of audio data in S/PDIF, AES-3, and IEC-60958 formats. These formats are designed to carry audio data between different systems through an optical or coaxial cable. The DIT mode applies only to a serializer configured as transmitter, not as receiver. For a description of the S/PDIF format, see *S/PDIF Coding Format*.

#### 12.5.2.4.10.3.1 Transmit DIT Encoding

When the MCASP operates in DIT mode, the data transmitted is output as a biphasemark encoded bitstream, with preamble, channel status, user data, validity, and parity automatically stuffed into the bitstream by the MCASP. The MCASP includes separate validity bits for even/odd subframes and two 384-bit RAM modules to hold channel status and user data bits.

#### Note

The transmit TDM time slot register (MCASP\_XTDM) should be programmed to all 1s during DIT mode. TDM functionality is not supported in DIT mode, except that the TDM slot counter counts the DIT subframes.

To transmit data in DIT mode, the following pins are typically required:

- AHCLKX – transmit high-frequency master clock (The internal clock source can be used instead.)
- One serial data pin (AXRn) of a serializer n configured to transmit.

For DIT Mode Transmission Data Alignment Settings see [Section 12.5.2.4.5.1.2, DIT Mode Transmission Data Alignment Settings](#).

If the MCASP is configured to transmit in the DIT mode on more than one serial data pin, the bit streams on all pins will be synchronized. In addition, although they will carry unique audio data, they will carry the same channel status, user data, and validity information.

The actual 24-bit audio data must always be in bit positions 23–0 after passing through the first three stages of the transmit format unit.

#### 12.5.2.4.10.3.2 Transmit DIT Clock and Frame-Sync Generation

The DIT transmitter works only in the following configuration:

- In the transmit frame control register (MCASP\_AFSXCTL):
  - Internally generated transmit frame sync, FSXM = 1
  - Rising-edge frame sync, FSXP = 0
  - Bit-width frame sync, FXWID = 0
  - 384-slot TDM, XMOD = 1 1000 0000b
- In the transmit clock control register (MCASP\_ACLKXCTL), ASYNC = 1
- In the transmit bitstream format register (MCASP\_XFMT), XSSZ = 1111 (32-bit slot size)

All combinations of AHCLKX and ACLKX are supported.

The following summarizes the register configurations required for DIT mode. DIT mode-specific bit fields are in bold face:

- MCASP\_PFUNC: The data pin - AXRn must be configured for MCASP function. If AHCLKX is used, it must also be configured for MCASP function. Other pins can be configured to function as GPIOs, if desired.
- MCASP\_PDIR: The data pin must be configured as output. If internal clock source AUXCLK is used as the reference clock, it may be output as the AHCLKX device level signal by configuring AHCLKX pin as an output.
- MCASP\_GBLCTL: Global initialization
- MCASP\_AMUTE: Leave this register at default state.
- MCASP\_DITCTL: The DITEN bit must be set to 0b1 to enable DIT mode. Configure other bits as desired.
- MCASP\_XMASK: Mask the desired bits, depending upon left-aligned or right-aligned internal data.



- MCASP\_XFMT: XDATDLY = 0. XRVRS = 0. XPAD = 0. XSSZ = Fh (32-bit slot). XBUSEL = configured as desired. The XROT bit is configured, as described in the [Section 12.5.2.4.5.1.2](#).
- MCASP\_AFSXCTL: Configure the bits according to former discussions.
- MCASP\_ACLKXCTL: ASYNC = 1. Program the CLKXDIV bits to obtain the bit clock rate desired. CLKXM = 1.
- MCASP\_AHCLKXCTL: Program the HCLKXDIV bits to obtain the high-frequency bit clock rate desired.
- MCASP\_XTDM: Set to FFFF FFFFh for all active slots for DIT transfers.
- MCASP\_XINTCTL: Program all fields according to the interrupts desired.
- MCASP\_XCLKCHK: Program all fields according to the clock checking desired.
- MCASP\_SRCTLn: Set SRMOD = 1 (transmitter) for the DIT pins (n = 0 to 15).
- MCASP\_DITCSRAi and MCASP\_DITCSRBi: Program the channel status bits as desired (i = 0 to 5).
- MCASP\_DITUDRAi and MCASP\_DITUDRBi: Program the user data bits as desired (i = 0 to 5).

### Note

In DIT mode, the transmitter can support a 192 kHz frame rate (stereo) on up to 2 serial data pins simultaneously (note that the internal bit clock for DIT runs two times faster than the equivalent bit clock for TDM (I2S) mode, due to the need to generate Biphase Mark Encoded Data - see *Biphase-Mark Code*).

#### 12.5.2.4.10.3.3 DIT Channel Status and User Data Register Files

The channel status registers (MCASP\_DITCSRAi and MCASP\_DITCSRBi) and user data registers (MCASP\_DITUDRAi and MCASP\_DITUDRBi) are not double-buffered. Typically, programmers use one of the synchronizing interrupts, such as the last slot, to create an event at a safe time so the register may be updated. In addition, the software reads the transmit TDM slot counter to determine which word of the register is being used (i = 0 to 5).

It is a software requirement to avoid writing to the word of user data and channel status that are being used to encode the current time slot; otherwise, it is undetermined whether old or new data is used to encode the bitstream.

The DIT subframe format is defined in *S/PDIF Subframe Format*. The channel status information (C) and user data (U) are defined in the following DIT control registers:

- MCASP\_DITCSRA0 to MCASP\_DITCSRA5: The 192 bits in these six registers contain the channel status information for the left channel within each frame.
- MCASP\_DITCSR0 to MCASP\_DITCSR5: The 192 bits in these six registers contain the channel status information for the right channel within each frame.
- MCASP\_DITUDRA0 to MCASP\_DITUDRA5: The 192 bits in these six registers contain the user data information for the left channel within each frame.
- MCASP\_DITUDRB0 to MCASP\_DITUDRB5: The 192 bits in these six registers contain the user data information for the right channel within each frame.
- The S/PDIF block format is shown in *S/PDIF Frame Format*. There are 192 frames within a block (frame 0 to frame 191). There are two subframes within each frame (subframes 1 and 2 for the left and right channels, respectively).

The channel status and user data information sent on each subframe is summarized in [Table 12-500](#).

**Table 12-500. Channel Status and User Data for Each DIT Block**

Frame	Subframe	Preamble	Channel Status Defined in:	User Data Defined in:
<b>Defined by DITCSRA0, DITCSR0, DITUDRA0, DITUDRB0</b>				
0	1 (L)	B	DITCSRA0[0]	DITUDRA0[0]
0	2 (R)	W	DITCSR0[0]	DITUDRB0[0]
1	1 (L)	M	DITCSRA0[1]	DITUDRA0[1]
1	2 (R)	W	DITCSR0[1]	DITUDRB0[1]
2	1 (L)	M	DITCSRA0[2]	DITUDRA0[2]

**Table 12-500. Channel Status and User Data for Each DIT Block (continued)**

Frame	Subframe	Preamble	Channel Status Defined in:	User Data Defined in:
2	2 (R)	W	DITCSRB0[2]	DITUDRB0[2]
...	...	...	...	...
31	1 (L)	M	DITCSRA0[31]	DITUDRA0[31]
31	2 (R)	W	DITCSRB0[31]	DITUDRB0[31]
<b>Defined by DITCSRA1, DITCSRB1, DITUDRA1, DITUDRB1</b>				
32	1 (L)	M	DITCSRA1[0]	DITUDRA1[0]
32	2 (R)	W	DITCSRB1[0]	DITUDRB1[0]
...	...	...	...	...
63	1 (L)	M	DITCSRA1[31]	DITUDRA1[31]
63	2 (R)	W	DITCSRB1[31]	DITUDRB1[31]
<b>Defined by DITCSRA2, DITCSRB2, DITUDRA2, DITUDRB2</b>				
64	1 (L)	M	DITCSRA2[0]	DITUDRA2[0]
64	2 (R)	W	DITCSRB2[0]	DITUDRB2[0]
...	...	...	...	...
95	1 (L)	M	DITCSRA2[31]	DITUDRA2[31]
95	2 (R)	W	DITCSRB2[31]	DITUDRB2[31]
<b>Defined by DITCSRA3, DITCSRB3, DITUDRA3, DITUDRB3</b>				
96	1 (L)	M	DITCSRA3[0]	DITUDRA3[0]
96	2 (R)	W	DITCSRB3[0]	DITUDRB3[0]
...	...	...	...	...
127	1 (L)	M	DITCSRA3[31]	DITUDRA3[31]
127	2 (R)	W	DITCSRB3[31]	DITUDRB3[31]
<b>Defined by DITCSRA4, DITCSRB4, DITUDRA4, DITUDRB4</b>				
128	1 (L)	M	DITCSRA4[0]	DITUDRA4[0]
128	2 (R)	W	DITCSRB4[0]	DITUDRB4[0]
...	...	...	...	...
159	1 (L)	M	DITCSRA4[31]	DITUDRA4[31]
159	2 (R)	W	DITCSRB4[31]	DITUDRB4[31]
<b>Defined by DITCSRA5, DITCSRB5, DITUDRA5, DITUDRB5</b>				
160	1 (L)	M	DITCSRA5[0]	DITUDRA5[0]
160	2 (R)	W	DITCSRB5[0]	DITUDRB5[0]
...	...	...	...	...
191	1 (L)	M	DITCSRA5[31]	DITUDRA5[31]
191	2 (R)	W	DITCSRB5[31]	DITUDRB5[31]

#### 12.5.2.4.11 MCASP Data Transmission and Reception

The MCASP is serviced by writing data to the MCASP\_XBUF<sub>n</sub> registers for transmit operations, and by reading data from the MCASP\_RBUF<sub>n</sub> registers for receive operations. The MCASP sets status flags and notifies the software whenever data is ready to be serviced. The [Section 12.5.2.4.11.1, Data Ready Status and Event/Interrupt Generation](#), discusses data-ready status in details (n = 0 to 15).

The MCASP transmit/receive XRBUF<sub>n</sub> buffer can be accessed through one of the two peripheral ports of the device:

- **DATA port:** This port is dedicated to DMA initiated data transfers on the device for MCASP transmit (Tx) purposes.
- **Configuration bus (CFG):** The configuration bus- CFG port is used for peripheral configuration control and receive/transmit data transfers initiated by the host CPU in the device.

[Section 12.5.2.4.11.1.3, Transfers Through the Data Port \(DATA\)](#), and [Section 12.5.2.4.11.1.4, Transfers Through the Configuration Bus \(CFG\)](#), discuss how to perform transfers through the data port (DATA) and the configuration port (CFG), respectively.

A device CPU and DMA usages are discussed in [Section 12.5.2.4.11.1.5, Using the device CPUs for MCASP Servicing](#), and [Section 12.5.2.4.11.1.6, Using the DMA for MCASP Servicing](#), respectively.

MCASP DATA port allows DMAs to access the MCASP transmit buffer more efficiently on the CBASS0, using burst transfers. The physical addresses to access these registers are listed in *DMA Registers*.

#### **12.5.2.4.11.1 Data Ready Status and Event/Interrupt Generation**

##### **12.5.2.4.11.1.1 Transmit Data Ready**

The transmit data ready flag - XDATA in the MCASP\_XSTAT register reflects the data ready status of XRBUF<sub>n</sub> buffers for all of the active slot transmitting serializers. The XDATA flag is set whenever data is transferred from a transmitting serializer buffer - XRBUF<sub>n</sub> to its corresponding XRSR<sub>n</sub> shift register. Thus, the XDATA bit indicates the global event that some of the serializers data buffer - XRBUF<sub>n</sub> is emptied and ready to accept new data from the host (CPU or DMA). The transmit data ready event is individually indicated per serializer in its corresponding control register MCASP\_SRCTL<sub>n</sub>[4] XRDY status bit. When this bit is set to 0b1, it notifies to host that this serializer Tx buffer must be serviced (written). When MCASP\_XBUF<sub>n</sub> is written to by the host, the MCASP\_SRCTL<sub>n</sub>[4] XRDY is deasserted to 0b0. As XDATA global flag is an OR-event of all active serializers XRDY flags, it indicates to software the moment, when write service operation has to be initiated by the MCASP host (XDATA=0b1). The XRDY flags have to be sequentially scanned by user software to determine which serializer MCASP\_XBUF<sub>n</sub> register has to be currently written. Once all requested MCASP\_XBUF<sub>n</sub> are written, the serializers control XRDY flags are cleared to 0b0. As a consequence, XDATA flag is deasserted to 0b0, to indicate to SW that write operation is completed for all serializers.

The global XDATA flag can be cleared when the MCASP\_XSTAT[5] XDATA bit is written to 0b1, or once MCASP\_XBUF<sub>n</sub> registers of all the serializers, that have previously raised their XRDY flags, are written with corresponding active slot data by the host.

Whenever XDATA is set, the XINT event is automatically generated on MCASP[0-2]\_XMIT\_DMA\_EVT line (if enabled in the MCASP\_XEVTCTL register) to notify the DMA of the MCASP\_XBUF<sub>n</sub> empty status. An interrupt - MCASP[0-2]\_XMIT\_INTR\_PEND can be also generated if the XDATA interrupt is enabled in the MCASP\_XINTCTL register (for details, see [Section 12.5.2.4.13.1, Transmit Data Ready Interrupt](#)).

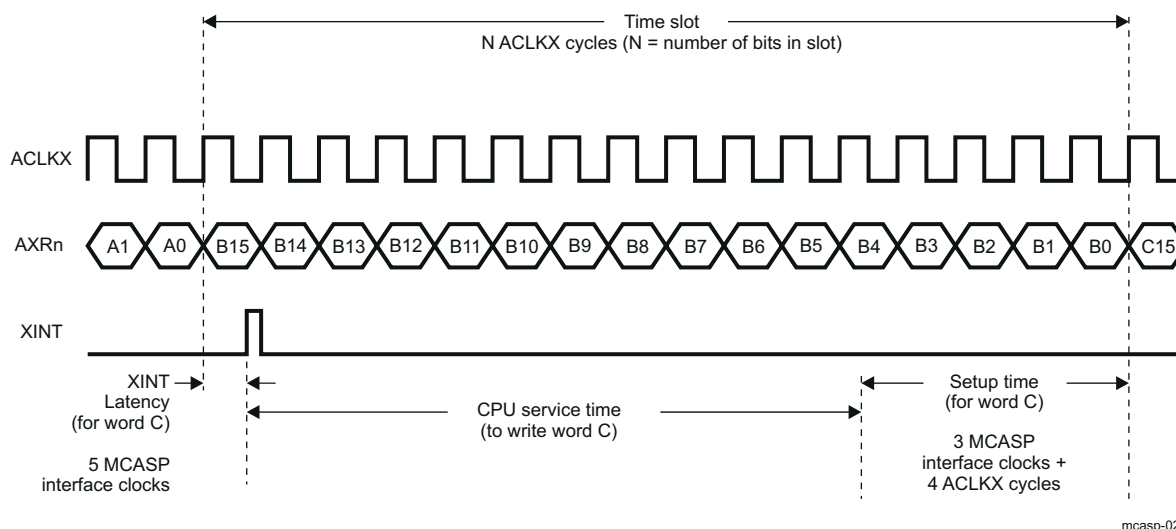
For DMA requests, the MCASP does not require that MCASP\_XSTAT register be read between DMA events. This means that, even if MCASP\_XSTAT register already has the XDATA flag set to 1 from a previous request, the next transfer triggers another DMA request.

Because the serializer acts in lockstep, only one DMA event is generated to indicate that the transmit serializer is ready to be written to with new data.

[Figure 12-506](#) shows the timing details of when XINT is generated at the MCASP boundary. In this example, as soon as the last bit (A0) of word A is transmitted, the MCASP sets the XDATA flag and generates an XINT event. However, it takes up to five MCASP interface clocks (XINT latency) before XINT is active at the MCASP boundary. Upon XINT, the CPU can begin servicing the MCASP by writing word C into the MCASP\_XBUF<sub>n</sub> (service time). The CPU must write word C into the MCASP\_XBUF<sub>n</sub> within the setup time required by the MCASP (setup time) (n = 0 to 15).

The maximum service time (see [Figure 12-506](#)) can be calculated as:

*Service Time = Time Slot – XINT Latency – Setup Time*



**Figure 12-506. Service Time Upon Transmit DMA Event (XINT)**

#### 12.5.2.4.11.1.2 Receive Data Ready

Similarly, the receive data ready flag - RDATA in the MCASP\_RSTAT register reflects the data ready status of XRBUFFn buffers for all of the active slot receiving serializers. The RDATA flag is set whenever data is transferred from a receiving serializer shift register XRSRn to its corresponding XRBUFFn data buffer. Thus, the RDATA bit indicates the global event that some of the receivers data buffer - RXBUFFn already contains received data (this is, a buffer is full) and is ready to transfer it to the host. The receive data ready event is individually indicated per serializer in its corresponding control register MCASP\_SRCTLn [5] RRDY status bit. When this bit is set to 0b1, it notifies to host that this serializer Rx buffer must be serviced (read). When MCASP\_RBUFFn register is read from the host, the MCASP\_SRCTLn [5] RRDY bit is deasserted to 0b0. As RDATA global flag is an OR-event of all active serializers RRDY flags, it indicates to software the moment, when read service operation has to be initiated by the MCASP host (RDATA = 0b1). The RRDY flags have to be sequentially scanned by user software to determine which serializer MCASP\_RBUFFn register has to be currently read. Once all requested MCASP\_RBUFFn registers are read, the serializers control RRDY flags are cleared to 0b0. As a consequence, RDATA flag is deasserted to 0b0, to indicate to SW that read operation is completed for all serializers.

The global RDATA flag can be cleared when the MCASP\_RSTAT[5] RDATA bit is written to 0b1, or once MCASP\_RBUFFn registers of all the serializers, that have previously raised their RRDY flags, are read by the host.

Whenever RDATA flag is set, the RINT event is automatically generated on MCASP[0-2]\_REC\_DMA\_EVT line (if enabled in the MCASP\_PIDTCTL register) to notify the DMA of the MCASP\_RBUFFn full status. An interrupt - MCASP[0-2]\_REC\_INTR\_PEND can be also generated if the RDATA interrupt is enabled in the MCASP\_RINTCTL register (for details, see [Section 12.5.2.4.13.1, Receive Data Ready Interrupt](#)).

[Figure 12-507](#) shows the timing details of when RINT event is generated at the MCASP boundary. In this example, as soon as the last bit (bit A0) of Word A is received, the MCASP sets the RDATA flag and generates an RINT event. However, it takes up to five MCASP interface clocks (RINT Latency) before RINT is active at the MCASP boundary. Upon RINT, the CPU can begin servicing the MCASP by reading Word A from the MCASP\_RBUFFn (service time). The CPU must read Word A from the MCASP\_RBUFFn register no later than the setup time required by the MCASP (Setup Time) (n = 0 to 15).

The maximum service time (see [Figure 12-507](#)) can be calculated as:

$$\text{Service Time} = \text{Time Slot} - \text{RINT Latency} - \text{Setup Time}$$

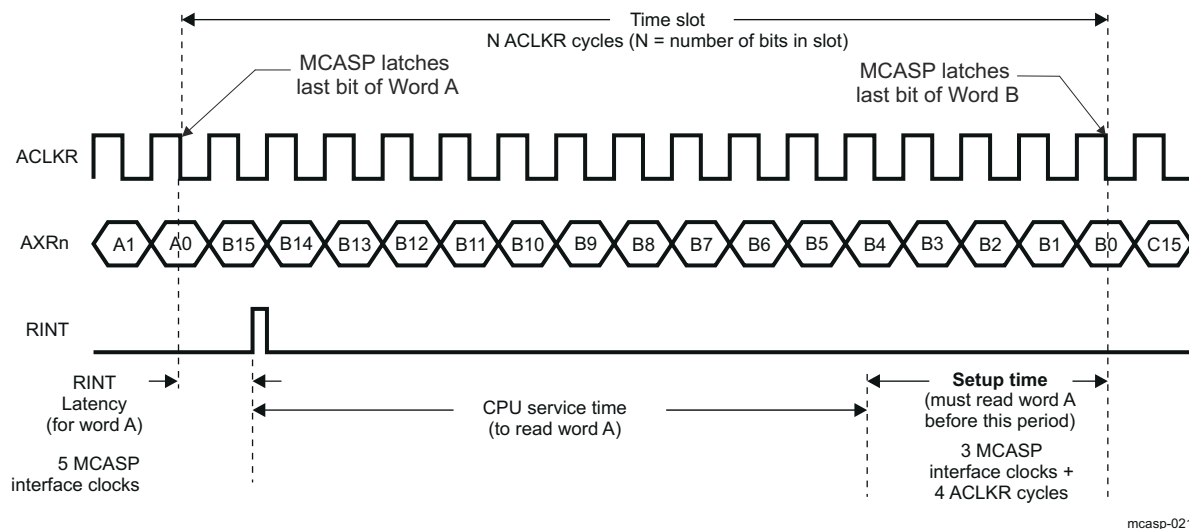


Figure 12-507. CPU Service Time Upon Receive Event (RINT)

#### 12.5.2.4.11.3 Transfers Through the Data Port (DATA)

##### CAUTION

To perform internal transfers through the DATA port, clear the XBUSEL/RBUSEL bit to 0b0 in the MCASP\_XFMT/MCASP\_RFMT register, respectively. Failure to do so may result in software malfunction.

In a typical MCASP transfer scenario, the DMA Controller write accesses the XRBUFn transmit buffer through the MCASP data port (DATA) on CBASS0 Interconnect. CPU hosts can access both XRBUFn transmit and receive data buffers on their corresponding DATA port address via DATA port corresponding address. To perform transfers through the DATA port, simply have the DMA Controller write the MCASP Tx buffer through Interconnect DATA port location. Refer to *DMA Registers*. Although the transfer is passed through an integrated AFIFO transmit/receive buffer, the host (DMA or CPU) must follow the described below procedure to access the data buffers of each serializer, regardless the AFIFO is enabled or disabled. The AFIFO operation is described in [Section 12.5.2.4.12](#).

For accesses through the DATA port, the DMA/CPU services all the serializers through accessing only a single address. In addition, as can be seen in *DMA Registers*, the same physical DATA port address is used regardless of a read or write access is performed. The MCASP automatically cycles through the active slot transmitting/receiving serializers, internally generating the appropriate offsets.

##### Note

DATA port allows the DMA/CPU to automatically access only the data buffers. There is no way for DMA/CPU to access the MCASP configuration registers addressing their corresponding MCASP DATA port.

For transmit operations through the DATA port, the host must always write to the same transmit buffer DATA port address (which is same than the receive buffer DATA port address) to service all of the active slot transmitting serializers. Regardless of MCASP serializer 0 being configured inactive or active, the user software must always configure the destination address to match the DATA port location of TXBUF buffer (See *DMA Registers*).

In addition, the DMA/CPU must write the buffers of all transmitting serializers in incremental (although not necessarily consecutive) order. For example, if only serializers 1 and 3 are set up as active transmitters, to the same transmit buffer DATA port address twice - first data for serializer 1 and second data for serializer 3

upon each transmit data ready event. This exact servicing order must be followed so that data appears in the appropriate serializers.

#### Note

For write transfers through MCASP DATA port it is preferable to use DMA on corresponding Interconnect. This is because DMAs initiated traffic gets better advantage of the burst transfers supported by DATA port.

For receive operations through the DATA port, the DMA/CPU must always read from the same receive buffer DATA port address (which is same than the transmit buffer DATA port address) to service all of the active slot receiving serializers. Regardless of MCASP serializer 0 being configured inactive or active, the user software must always configure the DMA/CPU source address to match the DATA port location of RXBUF buffer (See *DMA Registers*).

In addition, reads from the receive buffer for all active slot receiving serializers through the Rx DATA port return data in incremental (although not necessarily consecutive) order. For example, if serializers 0, 1 and 3 are set up as active receivers, the CPU should read from the same receive buffer DATA port address three times to obtain data for serializers 0, 1 and 3 in this exact order, upon each receive data ready event.

#### Note

To service a serializer for a transmit or receive operation through the MCASP DATA port, the initiator always writes (preferably DMA) and reads from the same address (refer to *DMA Registers*), respectively.

#### Note

When transmitting through the DATA port, the DMA/CPU must write data (at the same address) to each serializer configured as *active* (active slot selected in MCASP\_XTDM) and *transmit* (Tx enabled in MCASP\_SRCTLn) within each time slot. Failure to do so results in a buffer underrun condition (see [Section 12.5.2.4.16.1, Buffer Underrun Error - Transmitter](#)). Similarly, when DMA/CPU receives, data must be read from each serializer configured as *active* (active slot selected in MCASP\_RTDM) and *receive* (Rx enabled in MCASP\_SRCTLn) within each time slot. Failure to do so results in a buffer overrun condition (see [Section 12.5.2.4.16.2, Buffer Overrun Error - Receiver](#)) (n = 0 to 15).

#### 12.5.2.4.11.1.4 Transfers Through the Configuration Bus (CFG)

#### CAUTION

To perform internal transfers through the configuration bus, set the XBUSEL/RBUSEL bit to 1 in the MCASP\_XFMT/MCASP\_RFMT registers, respectively. Failure to do so may result in software malfunction.

#### Note

MCASP[0-2] whose data ports are accessible directly via CBASS0 do not support FIFO/constant addressing modes. Incrementing transfers must be used instead.

In this method, the CPU accesses the XRBUFn transmit or receive buffer through corresponding configuration bus (CFG) address.

The exact XRBUFn transmit/receive buffer physical address for any particular serializer is determined by adding the transmit/receive buffer alias register offset for that particular serializer to the base address of MCASP CFG port actual for CBASS0 accesses. The XRBUFn buffer of the n-th serializer configured as a transmitter is aliased



- MCASP\_XBUF<sub>n</sub> in the CFG port address space. For example, the XBUF2 transmit buffer is mapped as the MCASP\_XBUF2 register. Similarly, the XBUF<sub>n</sub> buffer of the n-th serializer configured as a receiver is aliased - MCASP\_RBUF<sub>n</sub> in the CFG port address space. For example, the XBUF3 receive buffer is mapped as the MCASP\_RBUF3 register.

Accessing the XBUF through the DATA port (see [Section 12.5.2.4.11.1.3](#)) is different than CFG port accesses because the DATA port access demands the same physical address, regardless of transfer direction or current channel index, while accessing through the peripheral configuration port - CFG, the CPU must provide the exact MCASP\_XBUF<sub>n</sub> or MCASP\_RBUF<sub>n</sub> address upon accessing n-th serializer TX or RX buffer, respectively. For more details about MCASP\_XBUF<sub>n</sub> and MCASP\_RBUF<sub>n</sub> addresses corresponding to MCASP CFG port, see *CFG Registers* (n = 0 to 15).

#### 12.5.2.4.11.1.5 Using a Device CPU for MCASP Servicing

The device CPUs can be used to service the MCASP transmit channels through interrupts (upon MCASP[0-2]\_XMIT\_INTR\_PEND and MCASP[0-2]\_REC\_INTR\_PEND interrupts). Because these interrupt events are connected to device COMPUTE\_CLUSTER0, PRU\_ICSSG0/1, MAIN2MCU\_LVL\_INTRTR0, R5FSS0/1\_INTRTR0, C66SS0/1\_INTRTR0, R5FSS0/1 modules, they could be software mapped to input interrupt lines of any device CPU. Another way to service the transmit and receive channels, a polling of the XDATA bit in the MCASP\_XSTAT register and RDATA bit in the MCASP\_RSTAT register can be performed by device CPUs, respectively. As discussed in [Section 12.5.2.4.11.1.3, Transfers Through the Data Port \(DATA\)](#), and [Section 12.5.2.4.11.1.4, Transfers Through the Configuration Bus \(CFG\)](#), the device CPUs can access MCASP XBUF serializer buffer through their corresponding DATA and CFG port locations.

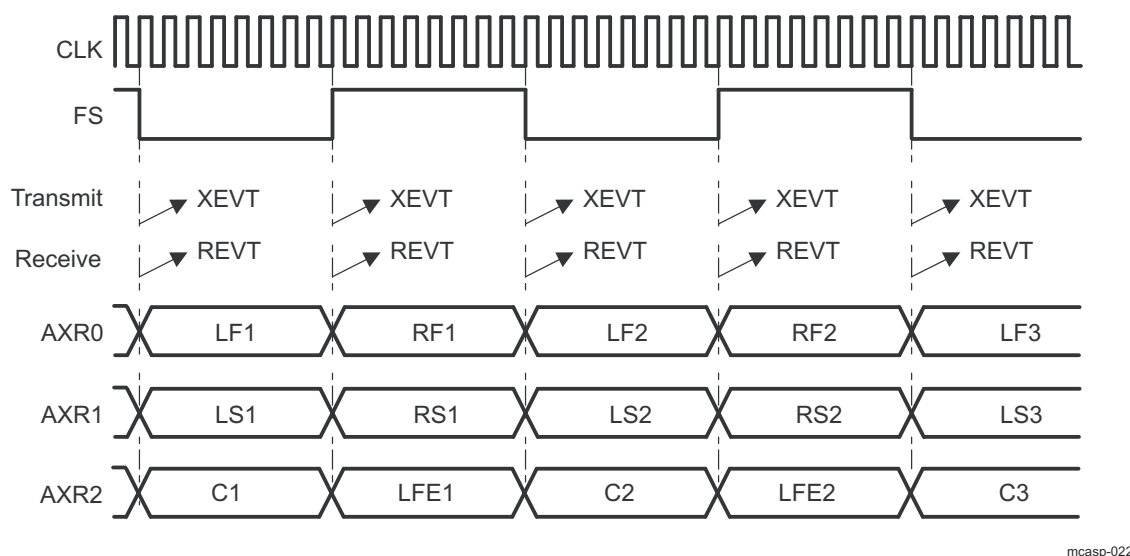
To use the device CPUs to service the MCASP through interrupts, the XDATA/RDATA bit must be enabled in the respective MCASP\_XINTCTL/MCASP\_RINTCTL registers, to generate interrupts MCASP[0-2]\_XMIT\_INTR\_PEND/MCASP[0-2]\_REC\_INTR\_PEND to the device CPUs upon data ready

#### 12.5.2.4.11.1.6 Using the DMA for MCASP Servicing

##### Note

The associated static Transfer Request (TR) operations of PDMA0, located in front of MCASP, must be configured to match the MCASP configuration. For more information, refer to *DMA Controllers*.

The typical scenario is to use the DMA to service the MCASP transmit and receive logic through the DATA port. The transfer passes through integrated AFIFO transmit/receive buffer. If AFIFO is enabled, DMA requests are collected and fed to a device DMA controller (see [Figure 12-497](#)). The data transfer is managed by the AFIFO according to generated transmit and receive events in the MCASP and data is fed to transmit buffers and fetched from receive buffers as described in [Section 12.5.2.4.12, MCASP Audio FIFO \(AFIFO\)](#). The generation of transmit and receive request is described below. After generation of transmit/receive DMA events from MCASP module, these events are collected in AFIFO and on specific AFIFO conditions described in [Section 12.5.2.4.12, MCASP Audio FIFO \(AFIFO\)](#) the requests (transmit or receive) are forwarded to a DMA controller via MCASP[0-2]\_XMIT\_DMA\_EVT and MCASP[0-2]\_REC\_DMA\_EVT outputs. If the AFIFO is disabled (default state) it is transparent for the MCASP module and all request are directly sent to the DMA controller.



**Figure 12-508. DMA Transmit and Receive Event in an Audio Example – One Event**

In transmit mode, the DMA event - XINT (MCASP[0-2]\_XMIT\_DMA\_EVT output), which is triggered upon each XDATA transition from 0 to 1, is used to service the MCASP TXBUF<sub>n</sub> transmit buffers. In receive mode, the DMA event RINT (MCASP[0-2]\_REC\_DMA\_EVT output) which is triggered upon each RDATA transition from 0 to 1, is used to service the MCASP RXBUF<sub>n</sub> receive buffers.

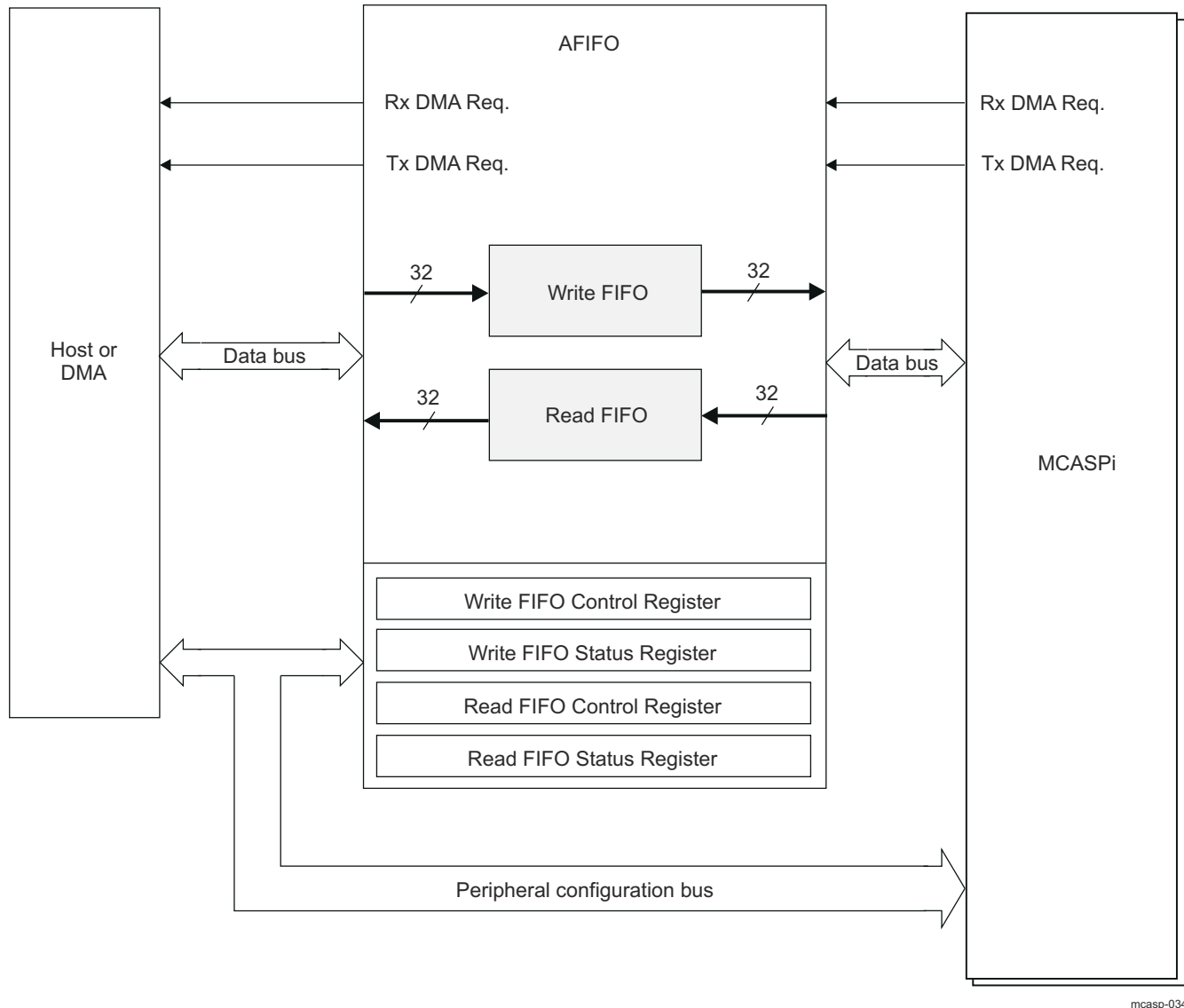
Figure 12-508 is an example of an audio system with six audio channels (LF, RF, LS, RS, C and LFE) transmitted or received through the MCASP signals - AXR0, AXR1 and AXR2. It shows the points at which events XINT/RINT are triggered.

In Figure 12-508, a Tx DMA event XINT is triggered on each time slot. In the example, XINT is triggered for each of the transmit audio channel time slot (time slot for channels LF, LS, and C; and time slot for channels RF, RS, LFE). Transmit DMA events are generated automatically upon transmit data ready, provided that DMA TX requests generation is enabled in the MCASP\_XEVTCTL register. Similarly, Rx DMA event RINT is triggered for each of the receive audio channel time slot. Receive DMA events are generated automatically upon receive data ready, provided that DMA RX requests generation is enabled in the MCASP\_PIDTCTL register.

#### 12.5.2.4.12 MCASP Audio FIFO (AFIFO)

The AFIFO contains two FIFOs: one Read FIFO (RFIFO), and one Write FIFO (WFIFO). The RFIFO and the WFIFO are the same size: 64 32-bit Words. To ensure backward compatibility with existing software, both the Read and Write FIFOs are disabled by default. See Figure 12-509 for a high-level block diagram of the AFIFO. The AFIFO may be enabled/disabled and configured via the MCASP\_WFIFOCTL and MCASP\_RFIFOCTL registers. Note that if the Read or Write FIFO is to be enabled, it must be enabled prior to initializing the receive/transmit section of the MCASP.





**Figure 12-509. MCASP Audio FIFO (AFIFO) Block Diagram**

#### 12.5.2.4.12.1 AFIFO Data Transmission

When the Write FIFO is disabled, transmit DMA requests pass through directly from the MCASP to the host/DMA controller. Whether the WFIFO is enabled or disabled, the MCASP generates transmit DMA requests as needed; the AFIFO is “invisible” to the MCASP. When the Write FIFO is enabled, transmit DMA requests from the MCASP are sent to the AFIFO, which in turn generates transmit DMA requests to the host/DMA controller. If the Write FIFO is enabled, upon a transmit DMA request from the MCASP, the WFIFO writes WNUMDMA 32-bit words to the MCASP if and when there are at least WNUMDMA words in the Write FIFO. If there are not, the WFIFO waits until this condition has been satisfied. At that point, it writes WNUMDMA words to the MCASP (see description for the MCASP\_WFIFOCTL[7-0] WNUMDMA). If the host CPU writes to the Write FIFO, independent of a transmit DMA request, the WFIFO will accept host writes until full. After this point, excess data will be discarded. Note that when the WFIFO is first enabled, it will immediately issue a transmit DMA request to the host. This is because it begins in an empty state, and is therefore ready to accept data.

##### 12.5.2.4.12.1.1 Transmit DMA Event Pacer

The AFIFO may be configured to delay making a transmit DMA request to the host until the Write FIFO has enough space for a specified number of words. In this situation, the number of transmit DMA requests to the

host or DMA controller is reduced. If the Write FIFO has space to accept WNUMEVT 32-bit words, it generates a transmit DMA request to the host and then waits for a response. Once WNUMEVT words have been written to the FIFO, it checks again to see if there is space for WNUMEVT 32-bit words. If there is space, it generates another transmit DMA request to the host, and so on. In this fashion, the Write FIFO will attempt to stay filled. Note that if transmit DMA event pacing is desired, MCASP\_WFIFOCTL[15-8] WNUMEVT bit field should be set to a non-zero integer multiple of the value in MCASP\_WFIFOCTL[7-0] WNUMDMA bit field. If transmit DMA event pacing is not desired, then the value in MCASP\_WFIFOCTL[15-8] WNUMEVT bit field should be set equal to the value in MCASP\_WFIFOCTL[7-0] WNUMDMA bit field.

#### 12.5.2.4.12.2 AFIFO Data Reception

When the Read FIFO is disabled, receive DMA requests pass through directly from MCASP to the host/DMA controller. Whether the RFIFO is enabled or disabled, the MCASP generates receive DMA requests as needed; the AFIFO is “invisible” to the MCASP. When the Read FIFO is enabled, receive DMA requests from the MCASP are sent to the AFIFO, which in turn generates receive DMA requests to the host/DMA controller. If the Read FIFO is enabled and the MCASP makes a receive DMA request, the RFIFO reads RNUMDMA 32-bit words from the MCASP, if and when the RFIFO has space for RNUMDMA words. If it does not, the RFIFO waits until this condition has been satisfied; at that point, it reads RNUMDMA words from the MCASP (see description for the MCASP\_RFIFOCTL[7-0] RNUMDMA). If the host CPU reads the Read FIFO, independent of a receive DMA request, and the RFIFO at that time contains less than RNUMEVT words, those words will be read correctly, emptying the FIFO.

#### 12.5.2.4.12.1 Receive DMA Event Pacer

The AFIFO may be configured to delay making a receive DMA request to the host until the Read FIFO contains a specified number of words. In this situation, the number of receive DMA requests to the host or DMA controller is reduced. If the Read FIFO contains at least RNUMEVT 32-bit words, it generates a receive DMA request to the host and then waits for a response. Once RNUMEVT 32-bit words have been read from the RFIFO, the RFIFO checks again to see if it contains at least another RNUMEVT words. If it does, it generates another receive DMA request to the host, and so on. In this fashion, the Read FIFO will attempt to stay empty. Note that if receive DMA event pacing is desired, MCASP\_RFIFOCTL[15-8] RNUMEVT bit field should be set to a non-zero integer multiple of the value in MCASP\_RFIFOCTL[7-0] RNUMDMA bit field. If receive DMA event pacing is not desired, then the value in MCASP\_RFIFOCTL[15-8] RNUMEVT bit field should be set equal to the value in MCASP\_RFIFOCTL[7-0] RNUMDMA bit field.

#### 12.5.2.4.12.3 Arbitration Between Transmit and Receive DMA Requests

If both the WFIFO and the RFIFO are enabled and a transmit DMA request and receive DMA request occur simultaneously, priority is given to the transmit DMA request. Once a transfer is in progress, it is allowed to complete. If only the WFIFO is enabled and a transmit DMA request and receive DMA request occur simultaneously, priority is given to the transmit DMA request. Once a transfer is in progress, it is allowed to complete. If only the RFIFO is enabled and a transmit DMA request and receive DMA request occur simultaneously, priority is given to the receive DMA request. Once a transfer is in progress, it is allowed to complete.

#### 12.5.2.4.13 MCASP Events and Interrupt Requests

[Table 12-501](#) lists all the transmit event flags. [Table 12-502](#) lists all the Receive event flags. Source of each of these TX/RX events can be a TX/RX channel from any MCASP serializer configured as transmitter or receiver respectively.

**Table 12-501. TX Events**

Event Mask <sup>(1)</sup>	Event Flag	Map to <sup>(1)</sup>	Description
MCASP_XINTCTL[0] XUNDRN	MCASP_XSTAT[0] XUNDRN	MCASP[0-2]_XMIT_INTR_PEND	Transmit buffer underrun
MCASP_XINTCTL[1] XSYNCERR	MCASP_XSTAT[1] XSYNCERR	MCASP[0-2]_XMIT_INTR_PEND	Unexpected transmit frame sync
MCASP_XINTCTL[2] XCKFAIL	MCASP_XSTAT[2] XCKFAIL	MCASP[0-2]_XMIT_INTR_PEND	Transmit clock failure

**Table 12-501. TX Events (continued)**

Event Mask <sup>(1)</sup>	Event Flag	Map to <sup>(1)</sup>	Description
MCASP_XINTCTL[3] XDMAERR	MCASP_XSTAT[7] XDMAERR	MCASP[0-2]_XMIT_INTR_PEND	DATA port transmit error
MCASP_XINTCTL[4] XLAST	MCASP_XSTAT[4] XLAST	MCASP[0-2]_XMIT_INTR_PEND	Transmit last slot interrupt
MCASP_XINTCTL[5] XDATA	MCASP_XSTAT[5] XDATA	MCASP[0-2]_XMIT_INTR_PEND	Transmit data-ready interrupt
MCASP_XINTCTL[7] XSTAFRM	MCASP_XSTAT[6] XSTAFRM	MCASP[0-2]_XMIT_INTR_PEND	Transmit start of frame interrupt
n.a.	MCASP_XSTAT[8] XERR	n.a.	OR-event of all Tx-error events: (XDMAERR   XCKFAIL   XUNDRN   XSYNCERR ). It is cleared ONLY when all error flags are cleared
n.a.	MCASP_XSTAT[3] XTDM SLOT	n.a.	Qualifies the current TDM slot as an odd or an even slot.

(1) Every MCASP module generates separate interrupt event.

**Table 12-502. RX Events**

Event Mask <sup>(1)</sup>	Event Flag	Map to <sup>(1)</sup>	Description
MCASP_RINTCTL[0] ROVRN	MCASP_RSTAT[0] ROVRN	MCASP[0-2]_REC_INTR_PEND	Receive buffer overrun
MCASP_RINTCTL[1] RSYNCERR	MCASP_RSTAT[1] RSYNCERR	MCASP[0-2]_REC_INTR_PEND	Unexpected receive frame sync
MCASP_RINTCTL[2] RCKFAIL	MCASP_RSTAT[2] RCKFAIL	MCASP[0-2]_REC_INTR_PEND	Receive clock failure
MCASP_RINTCTL[3] RDMAERR	MCASP_RSTAT[7] RDMAERR	MCASP[0-2]_REC_INTR_PEND	DATA port receive error
MCASP_RINTCTL[4] RLAST	MCASP_RSTAT[4] RLAST	MCASP[0-2]_REC_INTR_PEND	Receive last slot
MCASP_RINTCTL[5] RDATA	MCASP_RSTAT[5] RDATA	MCASP[0-2]_REC_INTR_PEND	Receive data-ready
MCASP_RINTCTL[7] RSTAFRM	MCASP_RSTAT[6] RSTAFRM	MCASP[0-2]_REC_INTR_PEND	Receive start of frame
n.a.	MCASP_RSTAT[8] RERR	n.a.	OR-event of all Rx-error events: (RDMAERR   RCKFAIL   ROVRN   RSYNCERR ). RERR event is cleared once all error flags are cleared.
n.a.	MCASP_RSTAT[3] RTDM SLOT	n.a.	Qualifies the current TDM slot as an odd or an even slot.

(1) Every MCASP module generates separate interrupt event.

Software has to read the MCASP\_XSTAT/MCASP\_RSTAT register to determine which event occurs at a global level for MCASP Tx/Rx logic. In addition user software has to scan the XRDY/RRDY read-only flags in the MCASP\_SRCTLn registers to determine which active serializer is the actual source of the event.

A Tx interrupt line (MCASP[0-2]\_XMIT\_INTR\_PEND) is asserted (active high) when one of the MCASP\_XSTAT notified events occurs, provided that it is enabled in its corresponding MCASP\_XINTCTL bit. Similarly, a Rx interrupt line (MCASP[0-2]\_REC\_INTR\_PEND) is asserted (active high) when one of MCASP\_RSTAT notified events occurs, provided that it is enabled in its corresponding MCASP\_RINTCTL bit. See also [Section 12.5.2.4.13.4, Multiple Interrupts](#) and the [Section 12.5.2.4.11.1, Data Ready Status and Event/Interrupt Generation](#) (n = 0 to 15).

#### 12.5.2.4.13.1 Transmit Data Ready Event and Interrupt

The transmit data-ready interrupt (XDATA) is generated if the MCASP\_XSTAT[5] XDATA bit is 1 and MCASP\_XINTCTL[5] XDATA bit is enabled. The [Section 12.5.2.4.11.1, Data Ready Status and Event/Interrupt Generation](#), provides details on when XDATA is set in the MCASP\_XSTAT register.

A transmit-start-of-frame interrupt (XSTAFRM) is triggered by the recognition of a transmit frame sync.

A transmit-last-slot interrupt (XLAST) is a qualified version of the data-ready interrupt (XDATA). It has the same behavior than the data-ready interrupt, but is further qualified by having the data requested belonging to the last slot (the slot that just ended is the next-to-last TDM slot, the current slot is the last slot).

#### 12.5.2.4.13.2 Receive Data Ready Event and Interrupt

The receive data-ready interrupt (RDATA) is generated if the MCASP\_RSTAT[5] RDATA bit is 1 and MCASP\_RINTCTL[5] RDATA bit is enabled. The [Section 12.5.2.4.11.1, Data Ready Status and Event/Interrupt Generation](#), provides details on when the MCASP\_RSTAT[5] RDATA bit is set.

A receiver start of frame (RSTAFRM) interrupt is triggered by the recognition of a receiver frame sync.

A receiver last slot (RLAST) interrupt is a qualified version of the data ready interrupt (RDATA). It has the same behavior as the data ready interrupt, but is further qualified by having the data in the buffer come from the last TDM time slot (the slot that just ended was last TDM slot).

#### 12.5.2.4.13.3 Error Interrupt

Upon detection, the following error conditions generate interrupt flags:

In the transmit status register (MCASP\_XSTAT):

- Transmit underrun (XUNDRN)
- Unexpected transmit frame sync (XSYNCERR)
- Transmit clock failure (XCKFAIL)
- Transmit DATA port error (XDMAERR)

Each interrupt source also has a corresponding enable bit in the transmit interrupt control register (MCASP\_XINTCTL). If the enable bit is set, an interrupt is requested when the interrupt flag is set in MCASP\_XSTAT. If the enable bit is not set, no interrupt request is generated. However, the interrupt flag may be polled.

In the receive status register (MCASP\_RSTAT) :

- Receiver overrun (ROVRN)
- Unexpected receive frame sync (RSYNCERR)
- Receive clock failure (RCKFAIL)
- Receive DATA port error (RDMAERR)

Each interrupt source also has a corresponding enable bit in the receive interrupt control register (MCASP\_RINTCTL). If the enable bit is set, an interrupt is requested when the interrupt flag is set in MCASP\_RSTAT. If the enable bit is not set, no interrupt request is generated. However, the interrupt flag may be polled.

#### 12.5.2.4.13.4 Multiple Interrupts

This only applies to interrupts and not to DMA requests. The following terms are defined:

- **Active Interrupt Request:** a flag in MCASP\_XSTAT is set and the interrupt is enabled in MCASP\_XINTCTL.
- **Outstanding Interrupt Request:** An interrupt request has been issued on one of the MCASP transmit interrupt port, but that request has not yet been serviced.
- **Serviced:** The CPUs write to MCASP\_XSTAT to clear one or more of the active interrupt request flags.

The first interrupt request to become active for the serializer with the interrupt flag set in MCASP\_XSTAT/MCASP\_RSTAT and the interrupt enabled in MCASP\_XINTCTL/MCASP\_RINTCTL generates a request on the MCASP transmit or receive interrupt port.

If more than one interrupt request becomes active in the same cycle, a single interrupt request is generated on the MCASP transmit or receive interrupt port. Subsequent interrupt requests that become active while the first interrupt request is outstanding do not immediately generate a new request pulse on the MCASP transmit or receive interrupt port.

The interrupt is serviced with the CPU writing to MCASP\_XSTAT/MCASP\_RSTAT. If any interrupt requests are active after the write, a new request is generated on the MCASP transmit or receive interrupt port.

One outstanding interrupt request is allowed on each port, so a transmit and a receive interrupt request may both be outstanding at the same time.

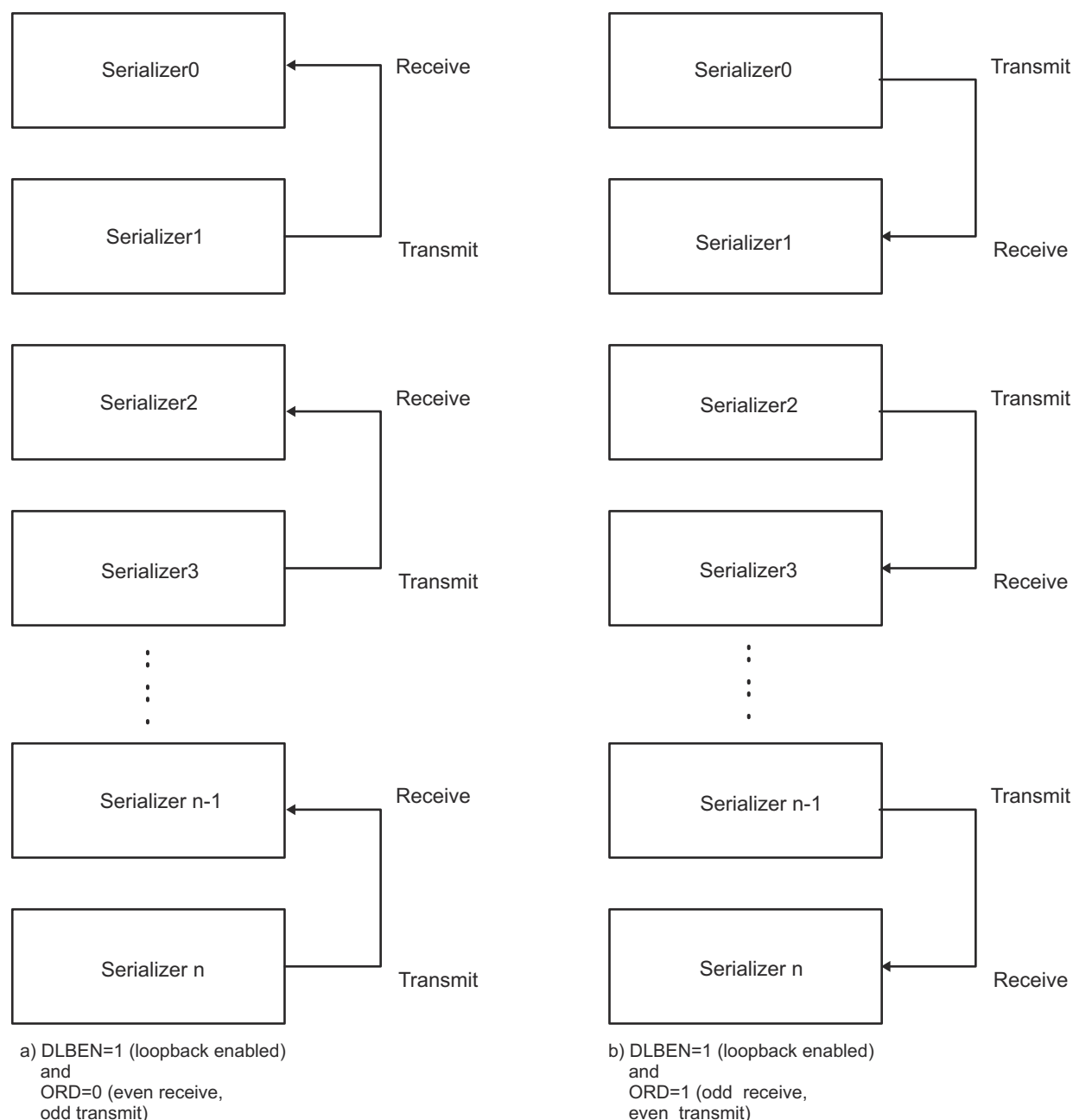
#### **12.5.2.4.14 MCASP DMA Requests**

The MCASP can generate one DMA request to the DMA controller to transmit (MCASP[0-2]\_XMIT\_DMA\_EVT) or receive (MCASP[0-2]\_REC\_DMA\_EVT) data. A DMA request to transmit data is generated if the MCASP\_XEVTCTL[0] XDATDMA bit is cleared. A DMA request to receive data is generated if the MCASP\_PIDTCTL[0] RDATDMA bit is cleared.

#### **12.5.2.4.15 MCASP Loopback Modes**

The MCASP features a digital loopback mode (DLB) that allows loopback test transfers in TDM mode between MCASP transmitters and receivers within the same device. In loopback mode, the output of a transmit serializer is connected internally to the input of a receive serializer. Therefore, a receiver data can be checked against a transmitter data to ensure that the MCASP settings are correct. Digital loopback mode applies to TDM mode only (2 to 32 slots in a frame). It does not apply to DIT mode (XMOD = 0x180) or burst mode (XMOD = 0).

[Figure 12-510](#) shows the basic logical connection of the serializers in loopback mode.



mcasp-023

**Figure 12-510. MCASP Serializers Operation in Loopback Mode**

Two types of loopback connections are possible, selected by the ORD bit in the digital loopback control register - MCASP\_DLBCCTL as follows:

- ORD = 0: Outputs of odd serializers are connected to inputs of even serializers. If this mode is selected, the odd serializers must be configured as transmitters and even serializers as receivers.
- ORD = 1: Outputs of even serializers are connected to inputs of odd serializers. If this mode is selected, the even serializers must be configured as transmitters and odd serializers as receivers.

User can choose in software (the MCASP\_DLBCCTL[4] IOLBEN bit) between a MCASP module internal loopback and a device I/O level loopback.

When a **MCASP internal loopback** is selected (MCASP\_DLBCTL[4] IOLBEN = 0b0 ), it is NOT necessary to configure MCASP\_PFUNC and MCASP\_PDIR registers for MCASP pin settings. Nevertheless, data can be optionally made externally visible at the I/O pin of the transmit serializer, if the pin is configured as a MCASP output pin by setting the corresponding MCASP\_PFUNC bit to 0 (this is, to function as MCASP, not GPIO) and MCASP\_PDIR bit to 1 (output).

When a **device I/O level loopback** is selected (MCASP\_DLBCTL[4] IOLBEN = 0b1 ), the MCASP\_PFUNC and MCASP\_PDIR registers must be configured with the appropriate settings for all AXRn pins, according to ORD bit configuration.

In case of device I/O loopback, the connectivity is externally applied between device pads (this is, reaching device I/O buffers ).

When in loopback mode, the transmit clock and frame sync are used by both the transmit and receive sections of the MCASP. The transmit and receive sections operate synchronously. This is achieved by setting the MCASP\_DLBCTL[3-2] MODE bit field to 0x1 and the MCASP\_ACLKXCTL[6] ASYNC bit to 0b0.

#### 12.5.2.4.15.1 Loopback Mode Configurations

This is a summary of the settings required for digital loopback mode for TDM format :

- The MCASP\_DLBCTL[0] DLBEN bit must be set to 0b1 to enable a loopback mode. It must be kept at 0b0 during normal MCASP operation.
- The MCASP\_DLBCTL[4] IOLBEN bit must be set to select between internal (MCASP local) loopback mode or device I/O level loopback mode.
- The MCASP\_DLBCTL[3-2] MODE bit field must be set to 0x1 for both the transmit and receive sections to use the transmit clock and frame sync generator.
- The MCASP\_DLBCTL[1] ORD bit must be programmed appropriately to select odd or even serializers to be transmitters or receivers.
- The corresponding serializers must be configured accordingly.
- The MCASP\_ACLKXCTL[6] ASYNC bit must be cleared to 0b0 to ensure synchronous transmit and receive operations.
- The MCASP\_AFSRCTL[15-7] RMOD bit field and MCASP\_AFSXCTL[15-7] XMOD bit field must be set within range (0x2 - 0x20) to indicate TDM mode.

#### Note

Loopback mode does not apply to DIT or burst mode, because MCASP receivers do NOT natively support DIR - reception.

#### 12.5.2.4.16 MCASP Error Reporting

The MCASP includes error-checking capability for the serial protocol and data underrun. In addition, the MCASP includes a timer that continually measures the high-frequency master clock every 32 AHCLKX clock cycles. The value of the timer can be read to get a measurement of the clock frequency and has a minimum and maximum range setting that can set an error flag if the master clock goes out of a specified range.

When one or more errors (software selectable) are detected, an interrupt can be generated if desired, based on one or more error sources.

##### 12.5.2.4.16.1 Buffer Underrun Error -Transmitter

A buffer underrun occurs when a serializer is instructed by the transmit state-machine to transfer data from XRBUFFn buffer to XRSRn shift register, but the corresponding (MCASP\_XBUFFn ) register has not yet been written with new data since the last time the transfer occurred. When this occurs, the transmit state-machine sets the XUNDRN flag.

An underrun is checked only once per time slot. The MCASP\_XSTAT[0] XUNDRN flag is set when an underrun condition occurs. Once set, the XUNDRN flag remains set until the host explicitly writes 1 to the XUNDRN bit to clear it (n = 0 to 15).



In DIT mode, a pair of BMC zeros is shifted out when an underrun occurs (four bit times at 128 bfs). By shifting out a pair of zeros, a clock can be recovered on the receiver. To recover, reset the MCASP and restart with the proper initialization.

In TDM mode, during an underrun case, a long stream of zeros are shifted out causing the DACs to mute. To recover, reset the MCASP and start again with the proper initialization.

#### 12.5.2.4.16.2 Buffer Overrun Error-Receiver

A buffer overrun occurs when a serializer is instructed to transfer data from XRSRn shift register to XRBUFn receiver buffer, but the corresponding MCASP\_RBUFn register has not yet been read since the last time the transfer occurred. When this occurs, the receiver state machine sets the overrun flag - ROVRN. However, the individual serializer writes over the data in the XRBUFn buffer register (destroying the previous sample) and continues shifting (n = 0 to 15).

An overrun is checked only once per time slot. The MCASP\_RSTAT[0] ROVRN flag is set when an overrun condition occurs. It is possible that an overrun occurs on one time slot but then the host catches up and does not cause an overrun on the following time slots. However, once the ROVRN flag is set, it remains set until the host explicitly writes a 1 to the ROVRN bit to clear the ROVRN bit.

#### 12.5.2.4.16.3 DATA Port Error - Transmitter

A transmit DATA port error, as indicated by the MCASP\_XSTAT[7] XDMAERR bit, occurs when the DMA or device CPU writes more words to the DATA port of the MCASP than it should.

The MCASP\_XSTAT[7] XDMAERR = 0b1 indicates that the DMA or device CPU wrote too many words to the MCASP DATA port for a given transmit DMA event. Writing too few words results in a transmit underrun error setting the MCASP\_XSTAT[0] XUNDRN bit.

While XDMAERR occurs infrequently, an occurrence indicates a serious loss of synchronization between the MCASP and the DMA or device CPU. The MCASP transmitter and the DMA must be reinitialized to resynchronize them.

#### 12.5.2.4.16.4 DATA Port Error - Receiver

A receive DATA port error, as indicated by the MCASP\_RSTAT[7] RDMAERR bit, occurs when the DMA or device CPU reads more words from the DATA port of the MCASP than it should.

The MCASP\_RSTAT[7] RDMAERR bit indicates that the DMA or device CPU read too many words from the MCASP DATA port for a given receive RINT event. Reading too few words results in a receiver overrun error setting the MCASP\_RSTAT[0] ROVRN bit.

While RDMAERR occurs infrequently, an occurrence indicates a serious loss of synchronization between the MCASP and the DMA or device CPU. The MCASP receiver and the DMA must be reinitialized to resynchronize them.

#### 12.5.2.4.16.5 Unexpected Frame Sync Error

An unexpected frame sync occurs in when:

- in burst mode and TDM mode, the next active edge of the frame sync occurs early such that the current slot will not be completed by the time the next slot is scheduled to begin.
- in TDM mode, an unexpected frame sync occurs also if the frame sync does NOT occur exactly during the correct bit clock (not a cycle earlier or later) and before slot 0.

When an unexpected frame sync occurs, there are two possible actions depending upon when the unexpected frame sync occurs:

1. **Early:** An early unexpected frame sync occurs when the MCASP is in the process of completing the current frame and a new frame sync is detected (not including overlap that occurs due to a 1 or 2 bit frame sync delay). When an early unexpected frame sync occurs:
  - Error event flag is set (XSYNCERR, if an unexpected transmit frame sync occurs; RSYNCERR, if an unexpected receive frame sync occurs).



- Current frame is not resynchronized. The number of bits in the current frame is completed. The next frame sync, which occurs after the current frame is completed, will be resynchronized.
- 2. **Late:** A late unexpected frame sync occurs when there is a gap or delay between the last bit of the previous frame and the first bit of the next frame. When a late unexpected frame sync occurs (as soon as the gap is detected):
  - Error event flag is set (XSYNCERR, if an unexpected transmit frame sync occurs; RSYNCERR, if an unexpected receive frame sync occurs).
  - Resynchronization occurs upon the arrival of the next frame sync.

Late frame sync is detected the same way in burst mode and TDM mode. However, in burst mode, late frame sync is not meaningful and its interrupt enable should not be set.

#### 12.5.2.4.16.6 Clock Failure Detection

##### 12.5.2.4.16.6.1 Clock Failure Check Startup

It is initially expected of the clock failure circuits to generate an error until at least one measurement is taken. Therefore, the clock failure interrupts, clock switch, and mute functions should not be enabled immediately, but only after a specific startup procedure.

To start the transmit clock failure check procedure:

1. Configure the transmit clock failure detect logic (XMIN, XMAX, XPS) in the transmit clock check control register (MCASP\_XCLKCHK).
2. Clear the transmit clock failure flag (XCKFAIL) in the transmit status register (MCASP\_XSTAT).
3. Wait until the first measurement is taken (> 32 AHCLKX clock periods).
4. Verify that no clock failure is detected.
5. Repeat Step 2 through Step 4 until the clock is running and is no longer issuing clock failure errors.
6. After the transmit clock is measured and falls within the acceptable range, the following can be enabled:
  - a. The transmit clock failure interrupt enable bit (XCKFAIL) in the transmitter interrupt control register (MCASP\_XINTCTL)

To start the receive clock failure check procedure:

1. Configure receive clock failure detect logic (RMIN, RMAX, RPS) in the receive clock check control register (MCASP\_RCLKCHK).
2. Clear receive clock failure flag (RCKFAIL) in the receive status register (MCASP\_RSTAT).
3. Wait until first measurement is taken (> 32 AHCLKR clock periods).
4. Verify no clock failure is detected.
5. Repeat steps 2–4 until clock is running and is no longer issuing clock failure errors.
6. After the receive clock is measured and falls within the acceptable range, the following may be enabled:
  - a. the receive clock failure (RCKFAIL) interrupt enable bit in the receive interrupt control register (MCASP\_RINTCTL)

##### 12.5.2.4.16.6.2 Transmit Clock Failure Check and Recovery

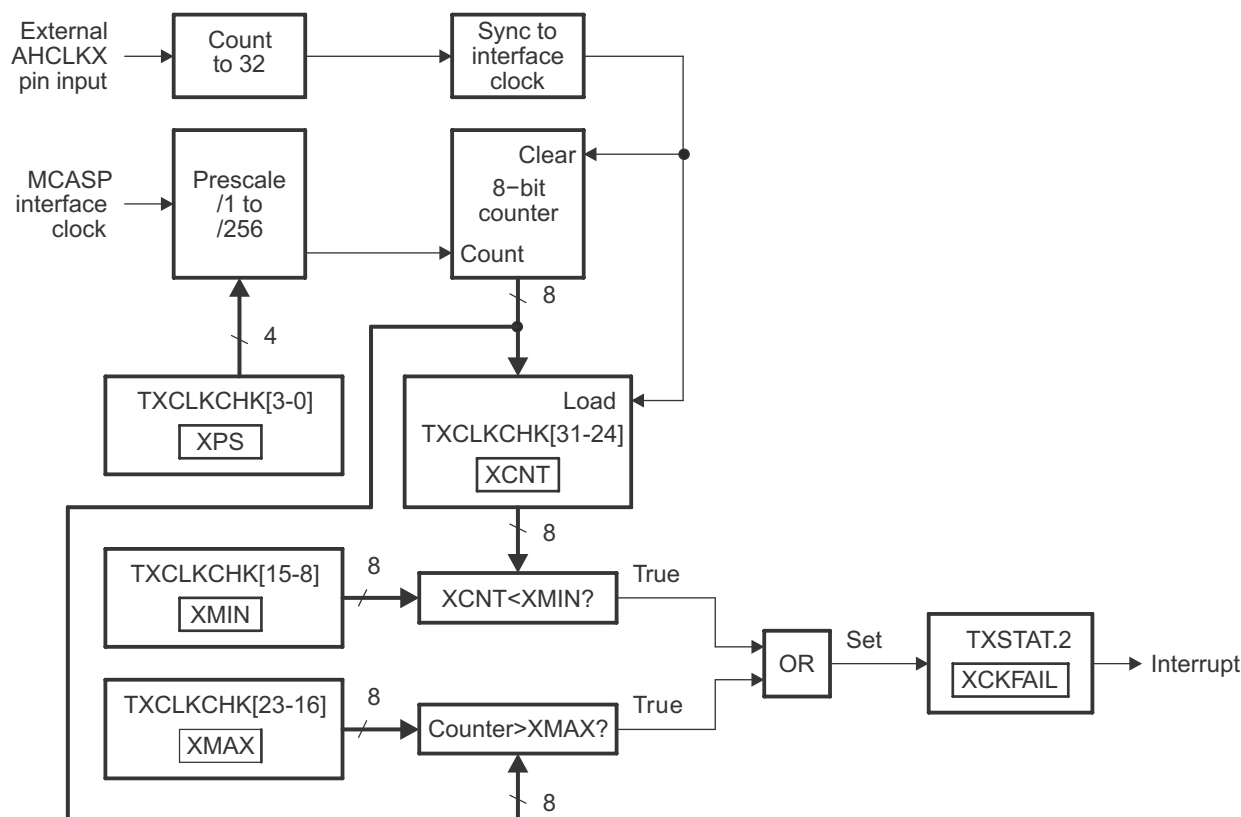
The transmit clock failure check circuit (see [Figure 12-511](#)) works off the internal MCASP interface clock and the external high-frequency serial clock (AHCLKX). It continually counts the number of interface clocks for every 32 high-rate serial clock (AHCLKX) periods, and stores the count in XCNT of the transmit clock check control register (MCASP\_XCLKCHK) every 32 high-rate serial clock cycles.

The logic compares the count against a user-defined minimum allowable boundary (XMIN), and automatically flags an interrupt (the MCASP\_XSTAT[2] XCKFAIL bit) when an out-of-range condition occurs. An out-of-range minimum condition occurs when the count is less than XMIN. The logic continually compares the current count (from the running interface clock counter) to the maximum allowable boundary (XMAX). This is so that if the external clock completely stops, the counter value is not copied to XCNT. An out-of-range maximum condition occurs when the count is greater than XMAX. The XMIN and XMAX fields are 8-bit unsigned values, and the comparison is performed using unsigned arithmetic.

An out-of-range count may indicate that an unstable clock was detected or that the audio source has changed and a new sample rate is being used.

For the transmit clock failure check circuit to operate correctly, the high-frequency serial clock divider must be taken out of reset.

If a clock failure is detected, the MCASP\_XSTAT[2] XCKFAIL transmit clock failure flag is set. This causes an interrupt if the MCASP\_XINTCTL[2] XCKFAIL transmit clock failure interrupt enable bit is set.



mcasps-024

**Figure 12-511. Transmit Clock Failure Detection Circuit Block Diagram**

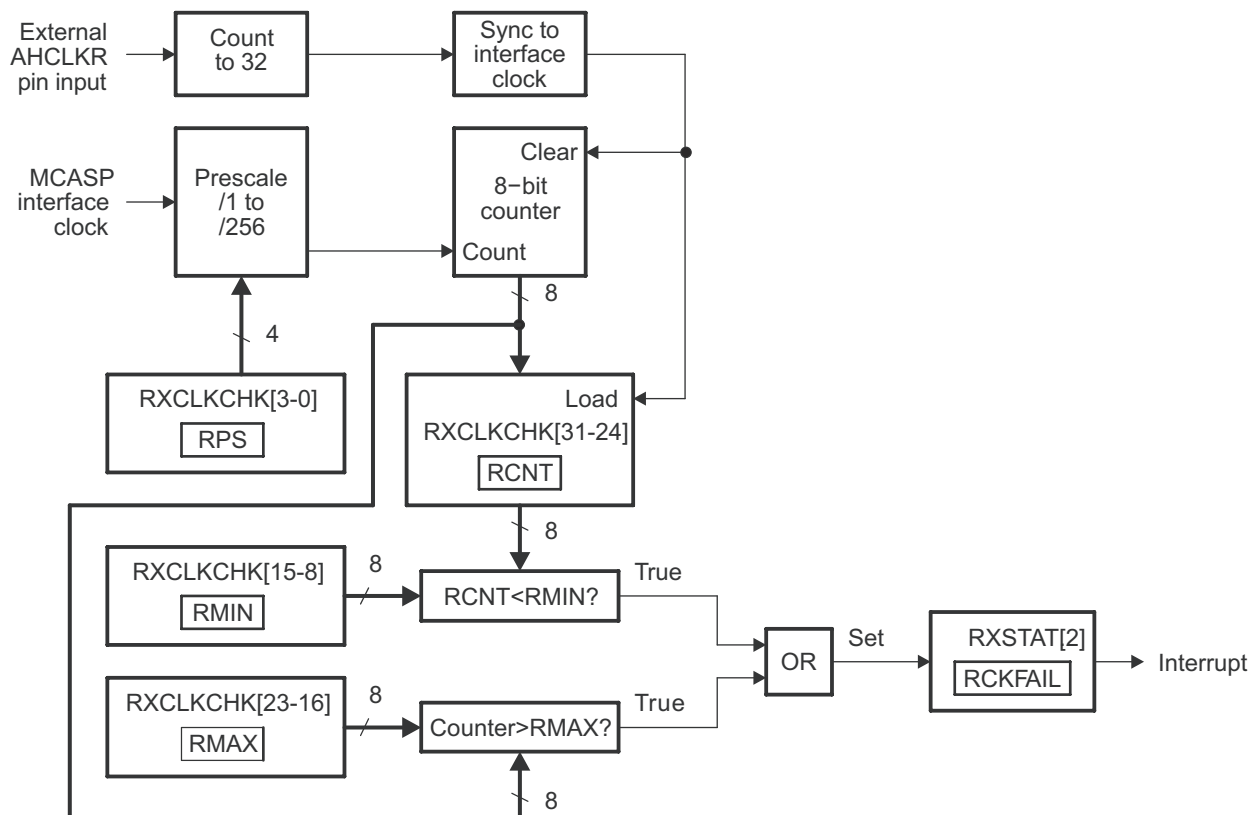
#### 12.5.2.4.16.6.3 Receive Clock Failure Check and Recovery

The receive clock failure check circuit (see [Figure 12-512](#)) works off both the internal MCASP interface clock and the external high-frequency serial clock (AHCLKR). It continually counts the number of interface clocks for every 32 high rate serial clock (AHCLKR) periods, and stores the count in RCNT of the receive clock check control register (MCASP\_RCLKCHK) every 32 high rate serial clock cycles.

The logic compares the count against a user-defined minimum allowable boundary (RMIN) and automatically flags an event (the MCASP\_RSTAT[2] RCKFAIL bit) when an out-of-range condition occurs. An out-of-range minimum condition occurs when the count is smaller than RMIN. The logic continually compares the current count (from the running interface clock counter) against the maximum allowable boundary (RMAX). This is in case the external clock completely stops, so that the counter value is not copied to RCNT. An out-of-range maximum condition occurs when the count is greater than RMAX. Note that the RMIN and RMAX fields are 8-bit unsigned values, and the comparison is performed using unsigned arithmetic.

An out-of-range count may indicate either that an unstable clock was detected or that the audio source has changed and a new sample rate is being used.

In order for the receive clock failure check circuit to operate correctly, the high-frequency serial clock divider must be taken out of reset.



mcasp-025

**Figure 12-512. Receive Clock Failure Detection Circuit Block Diagram**

## 12.5.2.5 MCASP Programming Guide

This section describes the low-level hardware programming sequences for the configuration and use of the MCASP module.

### 12.5.2.5.1 MCASP Global Initialization

#### 12.5.2.5.1.1 Surrounding Modules Global Initialization

This section identifies the requirements for initializing the surrounding modules when the MCASP module is used for the first time after a device reset. This initialization of surrounding modules is based on the integration and environment of the MCASP (for more information, see *MCASP Integration*, and *MCASP Environment* ).

[Table 12-503](#) describes the global initialization of surrounding modules.

**Table 12-503. Global Initialization of Surrounding Modules**

Surrounding Modules	Comments
LPSC3	Module reset must be enabled. For more information about the module configuration, see <i>Reset</i> .
PLLCTRL0	PLLCTRL0 configuration must be done to enable the clocks to the MCASP modules, see <i>Clocking</i> .
PLL4	PLL4 configuration must be done to enable the clocks to the MCASP modules, see <i>Clocking</i> .
PLL2	PLL2 configuration must be done to enable the clocks to the MCASP modules, see <i>Clocking</i> .
ATL	ATL configuration must be done to enable the clocks to the MCASP modules, see <i>Audio Tracking Logic (ATL)</i> .
COMPUTE_CLUSTER0	Device INTCs must be configured to enable the interrupt request generation. For information about enabling COMPUTE_CLUSTER0 interrupts, see <i>Interrupts</i> .
MAIN2MCU_LVL_INTRTR0	Device INTCs must be configured to enable the interrupt request generation. For information about enabling MAIN2MCU_LVL_INTRTR0 interrupts, see <i>Interrupts</i> .
R5FSS0_CORE0/1	Device INTCs must be configured to enable the interrupt request generation. For information about enabling R5FSS0_CORE0/1 interrupts, see <i>Interrupts</i> .
PDMA0_MCASP_G0	PDMA0_MCASP_G0 controllers configuration must be done to enable the module PDMA0_MCASP_G0 channel request, see <i>PDMA Controller</i> .
Interconnects	For information about the CBASS0 interconnects configuration, see <i>System Interconnect</i> .

#### Note

NAVSS configuration (UMDA channel set-up) must be done in order to move data from/to MCASP via the PDMA Controller. For more information, see *Navigator Subsystem (NAVSS)*.

#### Note

The COMPUTE\_CLUSTER0, MAIN2MCU\_LVL\_INTRTR0, R5FSS0\_CORE0/1, and the PDMA0\_MCASP\_G0 configurations are required when the interrupt and DMA-based communication modes are used. Further initialization of the selected interrupt and DMA controllers of the host CPU must be done for full functionality of the MCASP DMA and interrupt lines.

### 12.5.2.5.1.2 MCASP Global Initialization

#### 12.5.2.5.1.2.1 Main Sequence – MCASP Global Initialization for DIT-Transmission

The procedure in [Table 12-504](#) initializes the MCASP serializers transmitters to operate in DIT-mode (S/PDIF-transmission protocol) after a power-on reset (POR).

#### CAUTION

Before performing MCASP global initialization, If external clock ACLKR is used, it must be running already for proper synchronization of the MCASP\_GBLCTL register.

**Table 12-504. MCASP Transmitters Global Initialization for DIT-Mode Operation**

Step	Register/Bit Field/Programming Model	Value
1. Apply software reset to different MCASP components.	MCASP_GBLCTL[12-8]	0x00
2. Poll the bits to ensure the active reset value (0x00) is successfully latched into the register.	MCASP_GBLCTL[12-8]	=0x00
3. Configure the local power management.	MCASP_PWRIDLESYSCONFIG[1-0] IDLE_MODE	0x1
4. Configure the transmit format unit.	See <a href="#">Section 12.5.2.5.1.2.1.1</a> .	
5. Configure the transmit frame sync generator.	See <a href="#">Section 12.5.2.5.1.2.1.2</a> .	
6. Configure the transmit clock generator.	See <a href="#">Section 12.5.2.5.1.2.1.3</a> .	
7. Configure the TDM sequencer—set all slots active.	MCASP_XTDM[31-0] XTDMs	0xFFFF FFFF
8. Configure the desired n-th serializer (n=0 to 3) for transmit mode operation. <sup>(3)</sup>	MCASP_SRCTLn [1-0] SRMOD; n = 0 to 15	0x1
9. Configure the MCASP pins functionality.	See <a href="#">Section 12.5.2.5.1.2.1.4</a> .	
10. Enable the MCASP DIT - transmission mode.	MCASP_DITCTL[0] DITEN	0x1 <sup>(2)</sup>
11. Configure DIT-specific subframe fields.	See <a href="#">Table 12-509</a> .	
12. Release from reset state the divider that outputs the AHCLKX clock. <sup>(1)</sup>	MCASP_GBLCTL[9] XHCLKRST	0x1
13. Poll the bit to ensure that it is successfully latched in the register.	MCASP_GBLCTL[9] XHCLKRST	=0x1
14. Release from reset state the divider that outputs the ACLKX clock. <sup>(1)</sup>	MCASP_GBLCTL[8] XCLKRST	0x1
15. Poll the bit to ensure that it is successfully latched in the register.	MCASP_GBLCTL[8] XCLKRST	=0x1

(1) During reset state the local MCASP internal clock dividers maintain a 1:1 ratio at their outputs. The values stored in the MCASP\_AHCLKXCTL and MCASP\_ACLKXCTL registers are ignored; hence, the transmission clock does not stop during the reset state of the dividers.

(2) This globally configures all active transmitters to operate in DIT-mode.

(3) For an unused serializer n, write MCASP\_SRCTLn [1-0] SRMOD = 0x0 to disable it (n = 0 to 15).

#### 12.5.2.5.1.2.1.1 Subsequence – Transmit Format Unit Configuration for DIT-Transmission

The procedure in [Table 12-505](#) configures the transmit frame format unit of the MCASP module for a DIT-transmission.

#### Note

- The first transmit data bit always has a 0-bit delay.
- The bitstream is always transmitted in least-significant-bit (LSB)-first order.
- Pad value for extra bits in a certain slot is always 0.

**Table 12-505. Transmit Format Unit Configuration for DIT-Transmission**

Step	Register/Bit Field/Programming Model	Value
Configure the slot size to 32 bits.	MCASP_XFMT[7-4] XSSZ	0xF
<b>IF:</b> the data to transmit is left- aligned	Software test condition	
Set data mask in the range 0xFFFF FF00 – 0xFFFF 0000.	MCASP_XMASK[31-0] XMASK	0x- <sup>(1)</sup>
Rotate data right by a multiple-of-4- bit positions.	MCASP_XFMT[2-0] XROT	0x- <sup>(1)</sup>
<b>ELSE</b>		
Set data mask in the range 0x00FF FFFF– 0x0000 FFFF.	MCASP_XMASK[31-0] XMASK	0x- <sup>(1)</sup>
Rotate data right by 0-bit positions.	MCASP_XFMT[2-0] XROT	0x0
<b>ENDIF</b>		

**Table 12-505. Transmit Format Unit Configuration for DIT-Transmission (continued)**

Step	Register/Bit Field/Programming Model	Value
Select to write data to active serializers transmit buffers using peripheral (CFG) or DATA port	MCASP_XFMT[3] XBUSEL	0x-

(1) Refer to [Section 12.5.2.4.5.1](#), *Transmit Format Unit* and [Section 12.5.2.4.5.1.2](#), *DIT-Mode Transmission Data Alignment Settings*.

#### 12.5.2.5.1.2.1.2 Subsequence – Transmit Frame Synchronization Generator Configuration for DIT-Transmission

The procedure in [Table 12-506](#) configures the transmit frame synchronization generator of the MCASP module.

#### Note

The frame synchronization signal is always rising-edge active and always has a single-bit width.

**Table 12-506. Transmit Frame-Synchronization Generator Configuration for DIT-Transmission**

Step	Register/Bit Field/Programming Model	Value
Select 384-slot size block.	MCASP_AFSXCTL[15-7] XMOD	0x180
Select internally-generated transmit frame sync.	MCASP_AFSXCTL[1] FSXM	0x1

#### 12.5.2.5.1.2.1.3 Subsequence – Transmit Clock Generator Configuration for DIT-Transmission

#### Note

By default, the ACLKX and AHCLKX clocks are generated only from the MCASP internal clock source.

The procedure in [Table 12-507](#) configures the transmit clock generator of the MCASP module.

**Table 12-507. Transmit Clock Generator Configuration in DIT-Mode**

Step	Register/Bit Field/Programming Model	Value
Set the divisor for the internally generated high frequency clock– AHCLKX.	MCASP_AHCLKXCTL[11-0] HCLKXDIV	0x-
Set the divisor for the internally generated transmission clock– ACLKX.	MCASP_ACLKXCTL[4-0] CLKXDIV	0x-
Configure the transmit clock failure detect logic.	See <a href="#">Section 12.5.2.4.16.6.1</a> , <i>Clock Failure Check Startup</i> .	

#### 12.5.2.5.1.2.1.4 Subsequence - MCASP Pins Functional Configuration

The procedure in [Table 12-508](#) configures the MCASP pins for MCASP functionality.

**Table 12-508. MCASP Pins Functional Configuration**

Step	Register/Bit Field/Programming Model	Value
Configure module different pins to have MCASP functionality.	MCASP_PFUNC[31-0]	0x0
Configure the MCASP pins as outputs:		
AFSX	MCASP_PDIR[28] AFSX;	0x1
AHCLKX	MCASP_PDIR[27] AHCLKX;	0x1
ACLKX	MCASP_PDIR[26] ACLKX;	0x1
Desired i-th MCASP data pin AXRi is configured as an output for DIT-transmission.	MCASP_PDIR [i] AXRi	0x1

#### 12.5.2.5.1.2.1.5 Subsequence – DIT-specific Subframe Fields Configuration

The procedure in [Table 12-509](#) configures the DIT-specific subframe fields as part of the S/PDIF format data.

**Table 12-509. DIT-Specific Subframe Fields Configuration**

Step	Register/Bit Field/Programming Model	Value
Configure the valid bit value for odd time slots.	MCASP_DITCTL[3] VB	0x-

**Table 12-509. DIT-Specific Subframe Fields Configuration (continued)**

Step	Register/Bit Field/Programming Model	Value
Configure the valid bit value for even time slots.	MCASP_DITCTL[2] VA	0x-
Configure the user data bit for each subframe A and B in a 384-slot S/PDIF block.	MCASP_DITUDRAi[31-0] DITUDRAi, where i = 0 to 5 MCASP_DITUDRBi[31-0] DITUDRBi, where i = 0 to 5	0x- 0x-
Configure the channel status bit for each subframe A and B in a 384-slot S/PDIF block.	MCASP_DITCSRai[31-0], where i = 0 to 5 MCASP_DITCSRBi[31-0], where i = 0 to 5	0x- 0x-

#### 12.5.2.5.1.2.2 Main Sequence – MCASP Global Initialization for TDM-Reception

The procedure in [Table 12-510](#) initializes a MCASP serializer n receiver(s) to operate in TDM-mode (the only mode supported by MCASP receivers) after a power-on reset (POR). This is used for I2S (2-slot TDM) and other TDM-based audio protocols reception.

#### CAUTION

Before performing MCASP global initialization, If external clock ACLKR is used, it must be running already for proper synchronization of the MCASP\_GBLCTL register.

#### Note

The MCASP receivers support only TDM-frames (including 384-TDM frames) reception. DIT-frames reception (this is, S/PDIF stream) can be implemented indirectly via an external DIR-chip converter with DIT-input and TDM (I2S)-compatible output connected to device MCASP receiver input (TDM-only compatible).

**Table 12-510. MCASP Receivers Global Initialization for TDM-Mode Operation**

Step	Register/Bit Field/Programming Model	Value
1. Apply software reset to different MCASP receive components.	MCASP_GBLCTL[4-0]	0x00
2. Poll the bits to ensure the active reset value (0x00) is successfully latched into the register.	MCASP_GBLCTL[4-0]	=0x00
3. Configure the local power management.	MCASP_PWRIDLESYSCONFIG[1-0] IDLE_MODE	0x1
4. Configure the receive format unit.	See <a href="#">Section 12.5.2.5.1.2.2.1</a> .	
5. Configure the receive frame sync generator.	See <a href="#">Section 12.5.2.5.1.2.2.2</a> .	
6. Configure the receive clock generator.	See <a href="#">Section 12.5.2.5.1.2.2.3</a> .	
7. Program all bits - RTDMSk (where k = 0 to 31) according to the time slot characteristics desired (positions of active versus inactive slots within a frame).	MCASP_RTDM[k] RTDMSk, where k = 0 to 31	0x-
8. Configure the desired n-th serializer for receive mode operation. <sup>(4)</sup>	MCASP_SRCTLn [1-0] SRMOD; n = 0 to 15	0x2
9. Configure the MCASP pins functionality.	See <a href="#">Section 12.5.2.5.1.2.2.4</a> .	
10. Optional: Configure a MCASP Rx channel for a loopback operation (TDM mode only) in MCASP_DLBCTL [31-0].	See <a href="#">Section 12.5.2.4.15.1</a> , <i>Loopback Mode Configurations</i> .	0x- <sup>(5)</sup>
11. Release from reset state the divider that outputs the AHCLKR clock. <sup>(1)</sup> See also <sup>(2)</sup> .	MCASP_GBLCTL[1] RHCLKRST	0x1
12. Poll the bit to ensure that it is successfully latched in the register. See also <sup>(2)</sup> .	MCASP_GBLCTL[1] RHCLKRST	=0x1
13. Release from reset state the divider that outputs the ACLKR clock. <sup>(1)</sup> See also <sup>(3)</sup> .	MCASP_GBLCTL[0] RCLKRST	0x1



**Table 12-510. MCASP Receivers Global Initialization for TDM-Mode Operation (continued)**

Step	Register/Bit Field/Programming Model	Value
14. Poll the bit to ensure that it is successfully latched in the register. See also <sup>(3)</sup> .	MCASP_GBLCTL[0] RCLKRST	=0x1

- (1) During reset state the local MCASP internal clock dividers maintain a 1:1 ratio at their outputs. The values stored in the MCASP\_AHCLKRCTL and MCASP\_ACLKRCTL registers are ignored; hence, the reception clock does not stop during the reset state of the dividers.
- (2) This step is necessary even if external high-frequency serial clocks are used.
- (3) This step can be skipped if external serial clocks are used and they are running.
- (4) For an unused serializer n, write MCASP\_SRCTLn [1-0] SRMOD = 0x0 to disable it (n = 0 to 15).
- (5) In this case the receiver clock and frame sync are derived from the MCASP transmitter logic, so MCASP\_ACLKXCTL[6] ASYNC must be set to 0b0. Neither MCASP internal receiver clock and frame sync generators, nor external clock and frame sync source are used.

#### 12.5.2.5.1.2.2.1 Subsequence – Receive Format Unit Configuration in TDM Mode

The procedure in [Table 12-511](#) configures the receive frame format unit of the MCASP module for TDM slots reception.

**Table 12-511. Receive Format Unit Configuration for TDM-Reception**

Step	Register/Bit Field/Programming Model	Value
Configure the desired TDM-slot size	MCASP_RFMT[7-4] RSSZ	0x- <sup>(1)</sup>
Set data mask out value (0x0000 0000 - 0xFFFF FFFF).	MCASP_RMASK[31-0] RMASK	0x- <sup>(2)</sup>
Select a padding value for masked-out bits.	MCASP_RFMT[14-13] RPAD	0x-
Specify position (0x0-0x1F) of the bit in corresponding register MCASP_RBUF <sub>n</sub> which value to be used as a pad value in case MCASP_RFMT[14-13] RPAD = 0x2. (n = 0 to 15)	MCASP_RFMT[12-8] RPBIT	0x-
Rotate data right by a multiple of 4- bit positions.	MCASP_RFMT[2-0] RROT	0x- <sup>(3)</sup>
Received stream bit order (LSB- or MSB-first ). Must be set to 0x1 for an I2S stream reception (MSB-first).	MCASP_RFMT[15] RRVRS	0x- <sup>(3)</sup>
Specify a delay between frame sync and first bit of data in number of bits. Must be set to 0x1 for an I2S stream reception.	MCASP_RFMT[17-16] RDATDLY	0x-
Select to read data from active serializers receive buffers using peripheral (CFG) or DATA port	MCASP_RFMT[3] RBUSEL	0x-

- (1) Refer to [Section 12.5.2.4.5.2, Receive Format Unit](#), regarding options for received TDM-slot sizes.
- (2) For more details on Rx masking value, refer to [Section 12.5.2.4.5.2.1, TDM - Mode Reception Data Alignment Settings](#)
- (3) For more details on rotation and received TDM stream bit order, refer to [Section 12.5.2.4.5.2.1, TDM - Mode Reception Data Alignment Settings](#) and [Table 12-498, MCASP RFU Settings](#).

#### 12.5.2.5.1.2.2.2 Subsequence – Receive Frame Synchronization Generator Configuration in TDM Mode

The procedure in [Table 12-512](#) configures the transmit frame synchronization generator of the MCASP module.

#### Note

The same bit - MCASP\_ACLKXCTL[6] ASYNC which is used to determine if MCASP receivers and transmitters work synchronously on the same clock, is also used to define if receiver frame sync is derived from the transmit frame sync generator, or generated independently in the receiver (either internally or externally sourced). Hence, the settings in below table [Table 12-512](#) have no effect, if MCASP\_ACLKXCTL[6] ASYNC = 0.

**Table 12-512. Receive Frame-Synchronization Generator Configuration for TDM-Reception**

Step	Register/Bit Field/Programming Model	Value
Select number of TDM slots per frame. Must be set to 0x2, in case of an I2S-reception. For more details on frame-sync generator, refer to <a href="#">Section 12.5.2.4.2.3</a> .	MCASP_AFSRCTL[15-7] RMOD	0x- <sup>(1)</sup>



**Table 12-512. Receive Frame-Synchronization Generator Configuration for TDM-Reception (continued)**

Step	Register/Bit Field/Programming Model	Value
Choose the receive frame sync width -single bit/single word. For more details on frame-sync generator, refer to <a href="#">Section 12.5.2.4.2.3</a> .	MCASP_AFSRCTL[4] FRWID	0x-
Select start of received frame sync polarity - rising / falling edge. For more details on frame-sync generator, refer to <a href="#">Section 12.5.2.4.2.3</a> .	MCASP_AFSRCTL[0] FSRP	0x-
<b>IF</b> receive frame sync - FS is internally generated	Software test condition	
Select internally- generated receive frame sync. For more details on frame-sync generator, refer to <a href="#">Section 12.5.2.4.2.3</a> .	MCASP_AFSRCTL[1] FSRM	0b1
If MCASP receiver is required to output internally generated frame, AFSR pin must be set as an output in step 9 of the sequence documented in the <a href="#">Table 12-510</a> . This must not be done in current step because the frame control register - MCASP_AFSXCTL must be appropriately configured prior to AFSR pin outputting a frame to an external device.	MCASP_PDIR[31] AFSR	0b1
<b>ELSE</b>		
Select externally- generated receive frame sync. For more details on frame-sync generator, refer to <a href="#">Section 12.5.2.4.2.3</a> .	MCASP_AFSRCTL[1] FSRM	0b0
Setup the AFSR pin as input (device level: MCASPi_AFSR)	MCASP_PDIR[31] AFSR	0b0
<b>ENDIF</b>		
To generate MCASP receive frame sync in receiver logic, select an asynchronous frame sync.	MCASP_ACLKXCTL[6] ASYNC	0b1

(1) Must be set to 0x180 in case of 384-TDM slot frame reception from a DIR component I2S-output. For more details on TDM-frame settings, refer to *MCASP Integration*.

#### 12.5.2.5.1.2.2.3 Subsequence – Receive Clock Generator Configuration

The procedure in [Table 12-513](#) configures the receive clock generator of the MCASP module.

#### Note

The settings in below table [Table 12-513](#) have no effect, if MCASP\_ACLKXCTL[6] ASYNC = 0 (this is, receive clock is sourced from the inverted version of the transmit clock). For example, such is the case when MCASP loopback mode is used.

**Table 12-513. Receive Clock Generator Configuration**

Step	Register/Bit Field/Programming Model	Value
To use the MCASP receive clock generator, select an asynchronous receiver clock schema (ASYNC = 1). Otherwise an inverted version of transmit clock XCLK is used (receiver synchronized with transmitter).	MCASP_ACLKXCTL[6] ASYNC	0b1
<b>IF</b> receive clock - RCLK is internally generated	Software test condition	
The high-speed receive clock - AHCLKR is internally generated based on AUXCLK		
Select an internally-generated high-frequency clock.	MCASP_AHCLKRCTL[15] HCLKRM	0b1
Select the internal high-speed clock source polarity: non-inverted or inverted.	MCASP_AHCLKRCTL[14] HCLKRP	0x-
Set the divisor for the internally generated high-frequency clock – AHCLKR in range (1 - 4096).	MCASP_AHCLKRCTL[11-0] HCLKRDIV	0x-

**Table 12-513. Receive Clock Generator Configuration (continued)**

Step	Register/Bit Field/Programming Model	Value
Select an internally-generated receive clock.	MCASP_ACLKRCTL[5] CLKRM	0b1
Receiver samples on rising/falling edge. Select Rx sampling on the rising edge if transmitter shifts out on falling edge, and vice versa.	MCASP_ACLKRCTL[7] CLKRP	0x-
Set the divisor for the internally generated receive clock– ACLKR in range (1 - 32).	MCASP_ACLKRCTL[4-0] CLKRDIV	0x-
Optional: If MCASP receiver is required to output internally generated clock, ACLKR pin must be set as an output in step 9 of the sequence documented in the <a href="#">Table 12-510</a> . This must not be done in current step because the clock control register - MCASP_ACLKRCTL must be appropriately configured prior to ACLKR pin outputting a receive clock to an external device.	MCASP_PDIR[29] ACLKR	0b1
<b>ELSE</b>		
Select an externally-generated receive clock. Note that in this case the AHCLKR signal path and the CLKRDIV divider are NOT used.	MCASP_ACLKRCTL[5] CLKRM	0b0
Receiver samples on rising/falling edge. Select Rx sampling on the rising edge if transmitter shifts out on falling edge, and vice versa.	MCASP_ACLKRCTL[7] CLKRP	0x-
Setup an input direction for the ACLKR pin	MCASP_PDIR[29] ACLKR	0b0
<b>ENDIF</b>		
Configure the transmit clock failure detect logic.	See <a href="#">Section 12.5.2.4.16.6.1</a> , <i>Clock Failure Check Startup</i> .	

#### 12.5.2.5.1.2.2.4 Subsequence—MCASP Receiver Pins Functional Configuration

The procedure in [Table 12-514](#) configures the MCASP pins for MCASP functionality.

**Table 12-514. MCASP Receiver Pins Functional Configuration**

Step	Register/Bit Field/Programming Model	Value
Configure module different pins to have MCASP functionality.	MCASP_PFUNC[31-0]	0x0
Configure the MCASP pins direction:	MCASP_PDIR[31] AFSR;	0x-(1)
AFSR	MCASP_PDIR[29] ACLKR;	0x-(2)
ACLRK	MCASP_PDIR[n] AXRn;	0x0
Desired n-th MCASP data pin AXRn is configured as an input for receiving.		

(1) See [Table 12-512](#).

(2) For more details on MCASP clock configurations, refer to [Table 12-513](#).

#### 12.5.2.5.1.2.3 Main Sequence – MCASP Global Initialization for TDM -Transmission

The procedure in [Table 12-515](#) initializes a MCASP serializer n transmitter(s) to operate in TDM-mode after a power-on reset (POR). This is used for I2S (2-slot TDM) and other TDM-based audio protocols transmission.

#### CAUTION

Before performing MCASP global initialization, If external clock ACLKR is used, it must be running already for proper synchronization of the MCASP\_GBLCTL register.

**Table 12-515. MCASP Transmitters Global Initialization for TDM-Mode Operation**

Step	Register/Bit Field/Programming Model	Value
1. Apply software reset to different MCASP transmit components.	MCASP_GBLCTL[12-8]	0x00

**Table 12-515. MCASP Transmitters Global Initialization for TDM-Mode Operation (continued)**

Step	Register/Bit Field/Programming Model	Value
2. Poll the bits to ensure the active reset value (0x00) is successfully latched into the register.	MCASP_GBLCTL[12-8]	=0x00
3. Configure the local power management.	MCASP_PWRIDLESYSCONFIG[1-0] IDLE_MODE	0x1
4. Configure the transmit format unit.	See <a href="#">Section 12.5.2.5.1.2.3.1</a> .	
5. Configure the transmit frame sync generator.	See <a href="#">Section 12.5.2.5.1.2.3.2</a> .	
6. Configure the transmit clock generator.	See <a href="#">Section 12.5.2.5.1.2.3.3</a> .	
7. Program all bits - XTDMSk, where k = 0 to 31, according to the time slot characteristics desired (positions of active versus inactive slots within a frame).	MCASP_XTDM[k] XTDMSk, where k = 0 to 31 <sup>(4)</sup>	0x-
8. Configure the desired n-th serializer for transmit mode operation. <sup>(3)</sup>	MCASP_SRCTLn[1-0] SRMOD; n = 0 to 15	0x1
9. Setup all active transmitters to operate in TDM mode.	MCASP_DITCTL[0] DITEN	0x0 <sup>(2)</sup>
10. Configure the MCASP pins functionality.	See <a href="#">Section 12.5.2.5.1.2.3.4</a> .	
11. Optional: Configure a MCASP Tx channel for loopback operation (TDM mode only) in MCASP_DLBCTL [31-0].	See <a href="#">Section 12.5.2.4.15.1</a> , <i>Loopback Mode Configurations</i> .	0x-
12. Release from reset state the divider that outputs the AHCLKR clock. See <sup>(1)</sup>	MCASP_GBLCTL[9] XHCLKRST	0x1
13. Poll the bit to ensure that it is successfully latched in the register.	MCASP_GBLCTL[9] XHCLKRST	=0x1
14. Release from reset state the divider that outputs the ACLKR clock. See <sup>(1)</sup>	MCASP_GBLCTL[8] XCLKRST	0x1
15. Poll the bit to ensure that it is successfully latched in the register.	MCASP_GBLCTL[8] XCLKRST	=0x1

- (1) During reset state the local MCASP internal clock dividers maintain a 1:1 ratio at their outputs. The values stored in the MCASP\_AHCLKXCTL and MCASP\_ACLKXCTL registers are ignored; hence, the transmission clock does not stop during the reset state of the dividers.
- (2) All active transmit channels operate either in TDM mode or in DIT mode depending on DITEN value. There is no option to choose Tx Mode between DIT and TDM separately per serializer transmitter.
- (3) For an unused serializer n, write MCASP\_SRCTLn [1-0] SRMOD = 0x0 to disable it (n = 0 to 15).
- (4) Appropriately program in the MCASP\_SRCTLn[3-2] DISMOD bit field, the desired level (high-impedance state, 0, or 1) at AXRn output, during time of inactive slots. Note, that this setting does NOT apply when all slots are programmed to be active within a frame (in particular DIT-mode) (n = 0 to 15).

#### 12.5.2.5.1.2.3.1 Subsequence – Transmit Format Unit Configuration in TDM Mode

The procedure in [Table 12-516](#) configures the transmit frame format unit of the MCASP module for TDM slots transmission.

**Table 12-516. Transmit Format Unit Configuration for TDM-Transmission**

Step	Register/Bit Field/Programming Model	Value
Configure the desired TDM-slot size	MCASP_XFMT[7-4] XSSZ	0x- <sup>(1)</sup>
Set data mask out value (0x0000 0000 - 0xFFFF FFFF).	MCASP_XMASK[31-0] XMASK	0x- <sup>(2)</sup>
Select a padding value for masked-out bits.	MCASP_XFMT[14-13] XPAD	0x-
Specify position (0x0-0x1F) of the bit in corresponding register MCASP_XBUFn which value to be used as a pad value in case MCASP_XFMT[14-13] XPAD = 0x2 (n = 0 to 15).	MCASP_XFMT[12-8] XPBIT	0x-
Rotate data right by a multiple of 4- bit positions.	MCASP_XFMT[2-0] XROT	0x- <sup>(3)</sup>
transmitted stream bit order (LSB- or MSB-first). Must be set to 0x1 for an I2S stream transmission (MSB-first).	MCASP_XFMT[15] XRVRS	0x- <sup>(3)</sup>
Specify a delay between frame sync and first bit of data in number of bits. Must be set to 0x1 for an I2S stream transmission.	MCASP_XFMT[17-16] XDATDLY	0x-

**Table 12-516. Transmit Format Unit Configuration for TDM-Transmission (continued)**

Step	Register/Bit Field/Programming Model	Value
Select to write data to active serializers transmit buffers using peripheral (CFG) or DATA port	MCASP_XFMT[3] XBUSEL	0x-

- (1) Refer to [Section 12.5.2.4.5.1, Transmit Format Unit](#), regarding options for transmitted TDM-slot sizes.
- (2) For more details on Tx masking value, refer to [Section 12.5.2.4.5.1.1, TDM - Mode Transmission Data Alignment Settings](#)
- (3) For more details on rotation and transmitd TDM stream bit order, refer to [Section 12.5.2.4.5.1.1, TDM - Mode Transmission Data Alignment Settings](#) and [Table 12-496, MCASP TFU TDM Mode Settings](#).

#### 12.5.2.5.1.2.3.2 Subsequence – Transmit Frame Synchronization Generator Configuration in TDM Mode

The procedure in [Table 12-517](#) configures the transmit frame synchronization generator of the MCASP module.

**Table 12-517. Transmit Frame-Synchronization Generator Configuration for TDM-Transmission**

Step	Register/Bit Field/Programming Model	Value
Select number of TDM slots per frame (2 - 32). Must be set to 0x2, in case of an I2S-transmission. For more details on frame-sync generator, refer to <a href="#">Section 12.5.2.4.2.3, Frame-Sync Generator</a> .	MCASP_AFSXCTL[15-7] XMOD	0x-
Choose the transmit frame sync width -single bit/single word. For more details on frame-sync generator, refer to <a href="#">Section 12.5.2.4.2.3, Frame-Sync Generator</a> .	MCASP_AFSXCTL[4] FXWID	0x-
Select start of transmit frame sync polarity - rising / falling edge. For more details on frame-sync generator, refer to <a href="#">Section 12.5.2.4.2.3, Frame-Sync Generator</a> .	MCASP_AFSXCTL[0] FSXP	0x-
<b>IF</b> transmit frame sync - FS is internally generated	Software test condition	
Select internally- generated transmit frame sync. For more details on frame-sync generator, refer to <a href="#">Section 12.5.2.4.2.3, Frame-Sync Generator</a> .	MCASP_AFSXCTL[1] FSXM	0b1
If MCASP transmitter is required to output internally generated frame, AFSX pin must be set as an output in step 10 of the sequence documented in the <a href="#">Table 12-515</a> . This must NOT be done in current step because the frame control register - MCASP_AFSXCTL must be appropriately configured prior to AFSX pin outputting a frame sync to an external device.	MCASP_PDIR[28] AFSX	0b1
<b>ELSE</b>		
Select externally- generated transmit frame sync. For more details on frame-sync generator, refer to <a href="#">Section 12.5.2.4.2.3, Frame-Sync Generator</a> .	MCASP_AFSXCTL[1] FSXM	0b0
Setup the AFSX pin as input	MCASP_PDIR[28] AFSX	0b0

#### 12.5.2.5.1.2.3.3 Subsequence – Transmit Clock Generator Configuration for TDM Cases

The procedure in [Table 12-518](#) configures the transmit clock generator of the MCASP module.

**Table 12-518. Transmit Clock Generator Configuration for TDM Cases**

Step	Register/Bit Field/Programming Model	Value
<b>IF</b> transmit clock - XCLK is internally generated	Software test condition	
<b>IF</b> high-speed transmit clock - AHCLKX is internally generated based on AUXCLK	Software test condition	
Select an internally-generated high-frequency clock.	MCASP_AHCLKXCTL[15] HCLKXM	0b1
Select the high-frequency clock source polarity: non-inverted or inverted.	MCASP_AHCLKXCTL[14] HCLKXP	0x-
Set the divisor for the internally generated high-frequency clock – AHCLKX in range (1 - 4096).	MCASP_AHCLKXCTL[11-0] HCLKXDIV	0x-

**Table 12-518. Transmit Clock Generator Configuration for TDM Cases (continued)**

Step	Register/Bit Field/Programming Model	Value
Optional: If MCASP transmitter is required to output internally generated high-frequency clock, AHCLKX pin must be set as an output in step 10 of the sequence documented in the <a href="#">Table 12-515</a> . This must NOT be done in current step because the clock control register - MCASP_AHCLKXCTL must be appropriately configured prior to AHCLKX pin outputting a high-speed clock to an external device.	MCASP_PDIR[27] AHCLKX	0b1
<b>ELSE</b>		
Select an externally-generated high frequency clock (HCLKXDIV divider can not be used).	MCASP_AHCLKXCTL[15] HCLKXM	0b0
Select the high-speed transmit clock source polarity: non-inverted or inverted.	MCASP_AHCLKXCTL[14] HCLKXP	0x-
Setup an input direction for the AHCLKX pin	MCASP_PDIR[27] AHCLKX	0b0
<b>ENDIF</b>		
Select an internally-generated transmit clock.	MCASP_AHCLKXCTL[5] CLKXM	0b1
Transmitter samples on rising/falling edge. Select Tx shifting out data on the rising edge if receiver samples on falling edge, and vice versa.	MCASP_AHCLKXCTL[7] CLKXP	0x-
Set the divisor for the internally generated transmit clock– ACLKX in range (1 - 32).	MCASP_AHCLKXCTL[4-0] CLKXDIV	0x-
Optional: If MCASP transmitter is required to output internally generated clock, ACLKX pin) must be set as an output in step 10 of the sequence documented in the <a href="#">Table 12-515</a> . This must NOT be done in current step because the clock control register - MCASP_AHCLKXCTL must be appropriately configured prior to ACLKX pin outputting a transmit clock to an external device.	MCASP_PDIR[26] ACLKX	0b1
<b>ELSE</b>		
Select an externally-generated transmit clock. Note that in this case the AHCLKX signal path and the CLKXDIV divider are NOT used.	MCASP_AHCLKXCTL[5] CLKXM	0b0
Transmitter samples on rising/falling edge. Select Tx shifting out data on the rising edge if receiver samples on falling edge, and vice versa.	MCASP_AHCLKXCTL[7] CLKXP	0x-
Setup an input direction for the ACLKX pin	MCASP_PDIR[26] ACLKX	0b0
<b>ENDIF</b>		
Configure the transmit clock failure detect logic.	See <a href="#">Section 12.5.2.4.16.6.1</a> , <i>Clock Failure Check Startup</i> .	

#### 12.5.2.5.1.2.3.4 Subsequence—MCASP Transmit Pins Functional Configuration

The procedure in [Table 12-519](#) configures the MCASP pins for MCASP functionality.

**Table 12-519. MCASP Transmit Pins Functional Configuration**

Step	Register/Bit Field/Programming Model	Value
Configure module different pins to have MCASP functionality.	MCASP_PFUNC[31-0]	0x0
Configure the MCASP pins direction:	MCASP_PDIR[28] AFSR;	0x-(1)
AFSX	MCASP_PDIR[27] AHCLKR;	0x-(2)
AHCLKX	MCASP_PDIR[26] ACLKR;	0x-(2)
ACLKX	MCASP_PDIR[n] AXRn	0x1
Desired n-th MCASP data pin AXRn is configured as an output for transmission.		

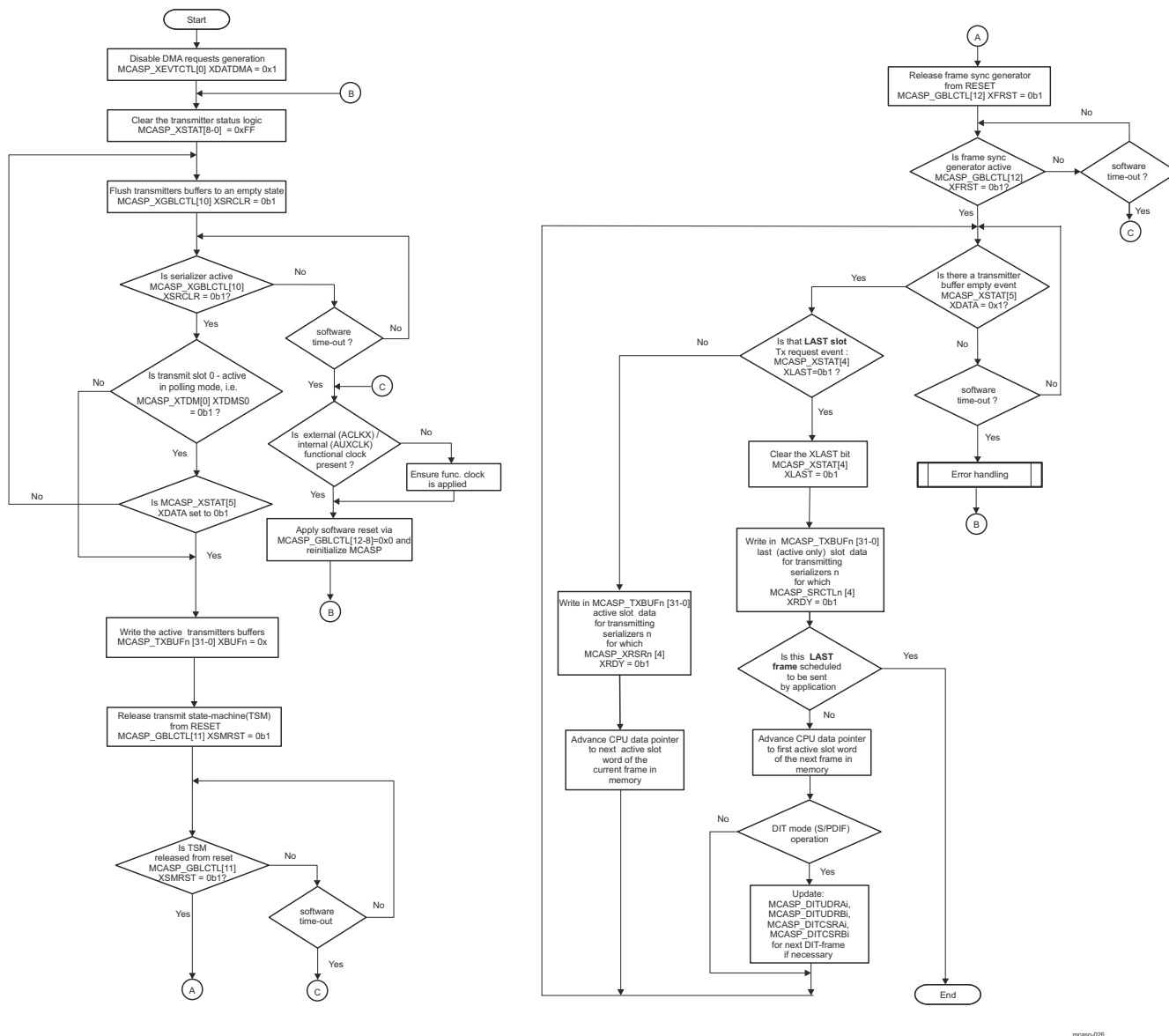
(1) See [Table 12-517](#).

### 12.5.2.5.2 MCASP Operational Modes Configuration

#### 12.5.2.5.2.1 MCASP Transmission Modes

##### 12.5.2.5.2.1.1 Main Sequence – MCASP DIT-/TDM- Polling Transmission Method

Figure 12-513 shows the MCASP DIT-/TDM- polling method.



**Figure 12-513. MCASP DIT-/TDM- Transmission Polling Method**

These registers are for MCASP DIT-/TDM- transmission polling method: MCASP\_XEVTCTL, MCASP\_XSTAT, MCASP\_GBLCTL, MCASP\_XTDM, MCASP\_XBUFn, MCASP\_SRCTLn, MCASP\_DITUDRAi, MCASP\_DITUDRbi, MCASP\_DITCSRai, MCASP\_DITCSRbi (n = 0 to 15 and i = 0 to 5).

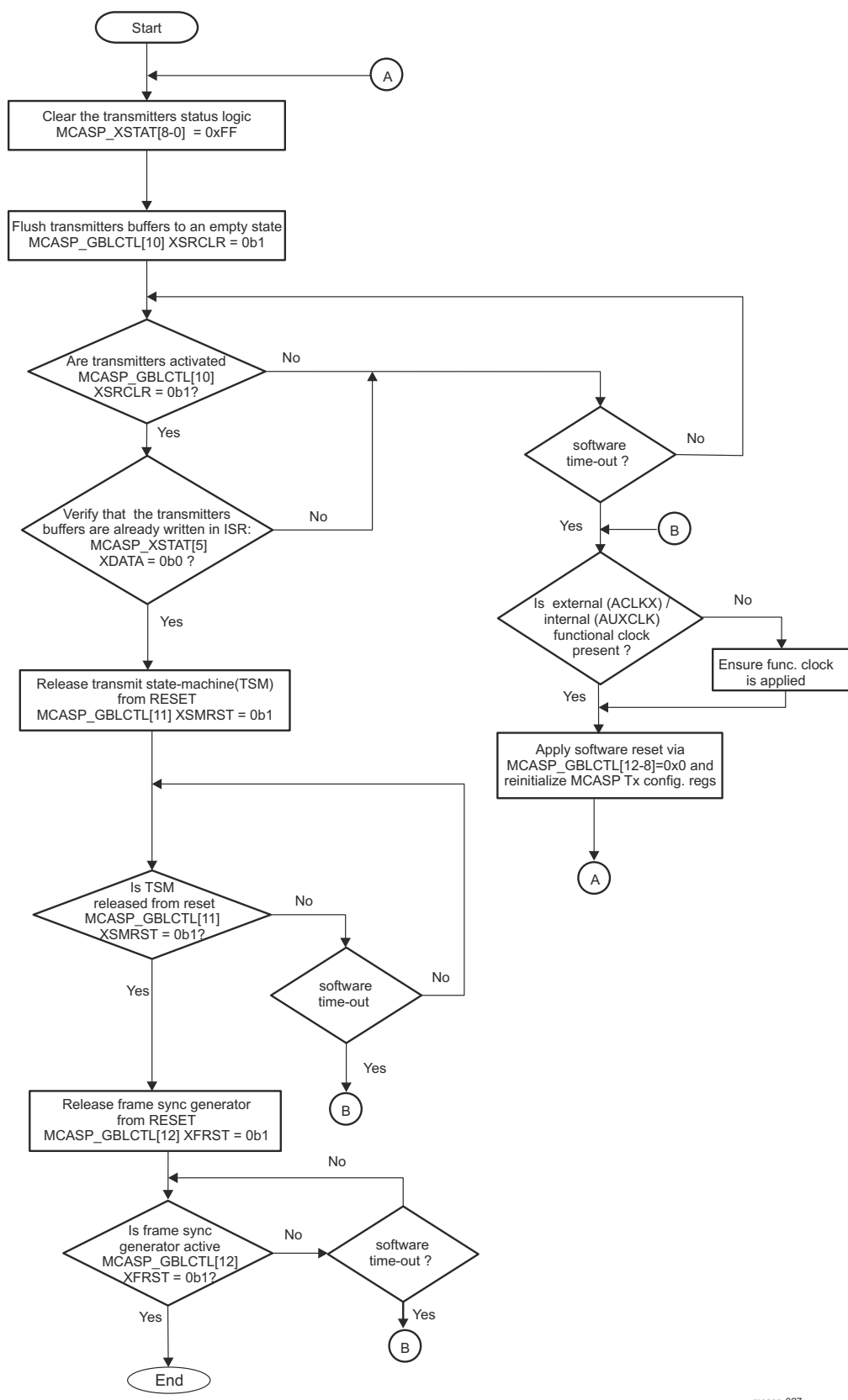
Table 12-520 summarizes the subprocess call for the DIT-/TDM- transmission polling mode.

**Table 12-520. Subprocess Call Summary for Main Sequence – MCASP DIT-/TDM- Transmission Polling Method**

Subprocess Name	Cross-Reference
Error handling	Figure 12-519

**12.5.2.5.2.1.2 Main Sequence – MCASP DIT- /TDM - Interrupt Transmission Method**

Figure 12-514 shows the initial setup for interrupt-based transmission.



mcasep-027

**Figure 12-514. Subsequence – DIT-/TDM- Transmission Startup Procedure**



Table 12-521 shows the configuration of the MCASP using an interrupt method for DIT-/TDM- transmission.

**Table 12-521. MCASP DIT-/TDM- Interrupt Transmission Model**

Step	Register/Bit Field/Programming Model	Value
Disable Tx DMA requests generation.	MCASP_XEVTCTL[0] XDATDMA	0x1
Enable the data ready event transmit interrupt.	MCASP_XINTCTL[5] XDATA	0x1
Optional: Enable the transmit error event interrupts.	MCASP_XINTCTL[2] XCKFAIL	0x1
	MCASP_XINTCTL[1] XSYNCERR	0x1
	MCASP_XINTCTL[0] XUNDRN	0x1
Optional: Enable the start of frame interrupt.	MCASP_XINTCTL[7] XSTAFRM	0x1
Optional: Enable the last slot data interrupt (useful for DIT user data/ channel status next S/PDIF frame info update).	MCASP_XINTCTL[4] XLAST	0x1
IF write transfer is through the MCASP DATA port (MCASP_XFMT[3] XBUSEL is set to 0b0).	Software test condition (setting is done in step4 of the MCASP Transmitters Global Initialization - see MCASP Transmitters Global Initialization for DIT-Mode Operation)	
Enable the DATA port error based interrupt.	MCASP_XINTCTL[3] XDMAERR	0x1
<b>ELSE</b>		
Disable the DATA port error based interrupt.	MCASP_XINTCTL[3] XDMAERR	0x0
<b>ENDIF</b>		
DIT/TDM - Transmission Startup Procedure	See Figure 12-514.	

These registers are for MCASP DIT-/TDM- transmission startup procedure: MCASP\_GBLCTL, MCASP\_XSTAT.

#### 12.5.2.5.2.1.3 Main Sequence –MCASP DIT- /TDM - Mode DMA Transmission Method

Table 12-522 shows the configuration of the ↓ using the DMA method for transmission. Possible interrupt error event servicing is also considered. shows the initial setup for DMA - based transmission.

#### Note

Because of the DATA port burst access capability with the DMA method, it is strongly recommended that DMA transfers are initiated through the MCASP DATA port.

**Table 12-522. MCASP DMA Transmission Model with Interrupt Events Servicing**

Step	Register/Bit Field/Programming Model	Value
<b>Recommended:</b> Select DATA port to access the transmit buffers.	MCASP_XFMT[3] XBUSEL	0x0
Enable the Tx DMA requests generation.	MCASP_XEVTCTL[0] XDATDMA	0x0
Enable the Tx DMA error event, because of MCASP DATA port usage.	MCASP_XINTCTL[3] XDMAERR	0x1
Optional: Enable the transmit error event interrupts.	MCASP_XINTCTL[2] XCKFAIL	0x1
	MCASP_XINTCTL[1] XSYNCERR	0x1
	MCASP_XINTCTL[0] XUNDRN	0x1
Optional: Enable the start of frame interrupt.	MCASP_XINTCTL[7] XSTAFRM	0x1
Optional: Enable the last slot data interrupt.	MCASP_XINTCTL[4] XLAST	0x1
Disable the data ready event transmit interrupt, as DMA is used to service this request.	MCASP_XINTCTL[5] XDATA	0x0
DMA startup transmission procedure. This procedure is identical than the one shown in Figure 12-514. The only difference is that DMA automatically services all the XINT events raised by the MCASP, and no CPU data processing intervention is required. The CPU is involved only in error handling shown in Figure 12-517.	See Figure 12-514.	

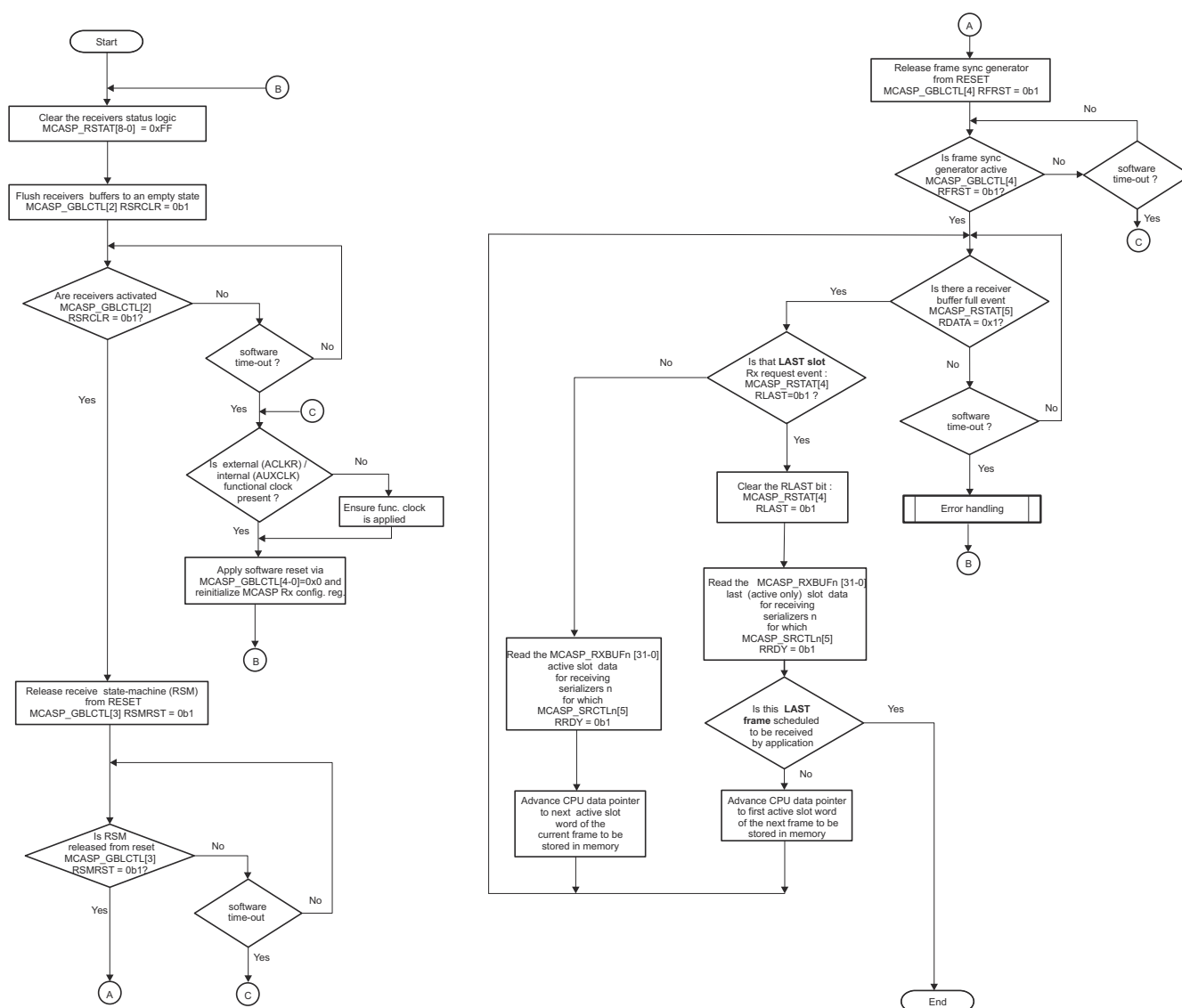
### 12.5.2.5.2.2 MCASP Reception Modes

#### 12.5.2.5.2.2.1 Main Sequence – MCASP Polling Reception Method

Figure 12-515 shows the MCASP polling reception method.

#### Note

The MCASP polling reception model considers the device CPUs as the accessor of audio data from the MCASP receive buffers.



mcaspp-028

**Figure 12-515. MCASP Polling Reception Method**

These registers are for MCASP reception polling method: MCASP\_RSTAT, MCASP\_XGBLCTL, MCASP\_RBUFn, MCASP\_SRCTLn (n = 0 to 15).

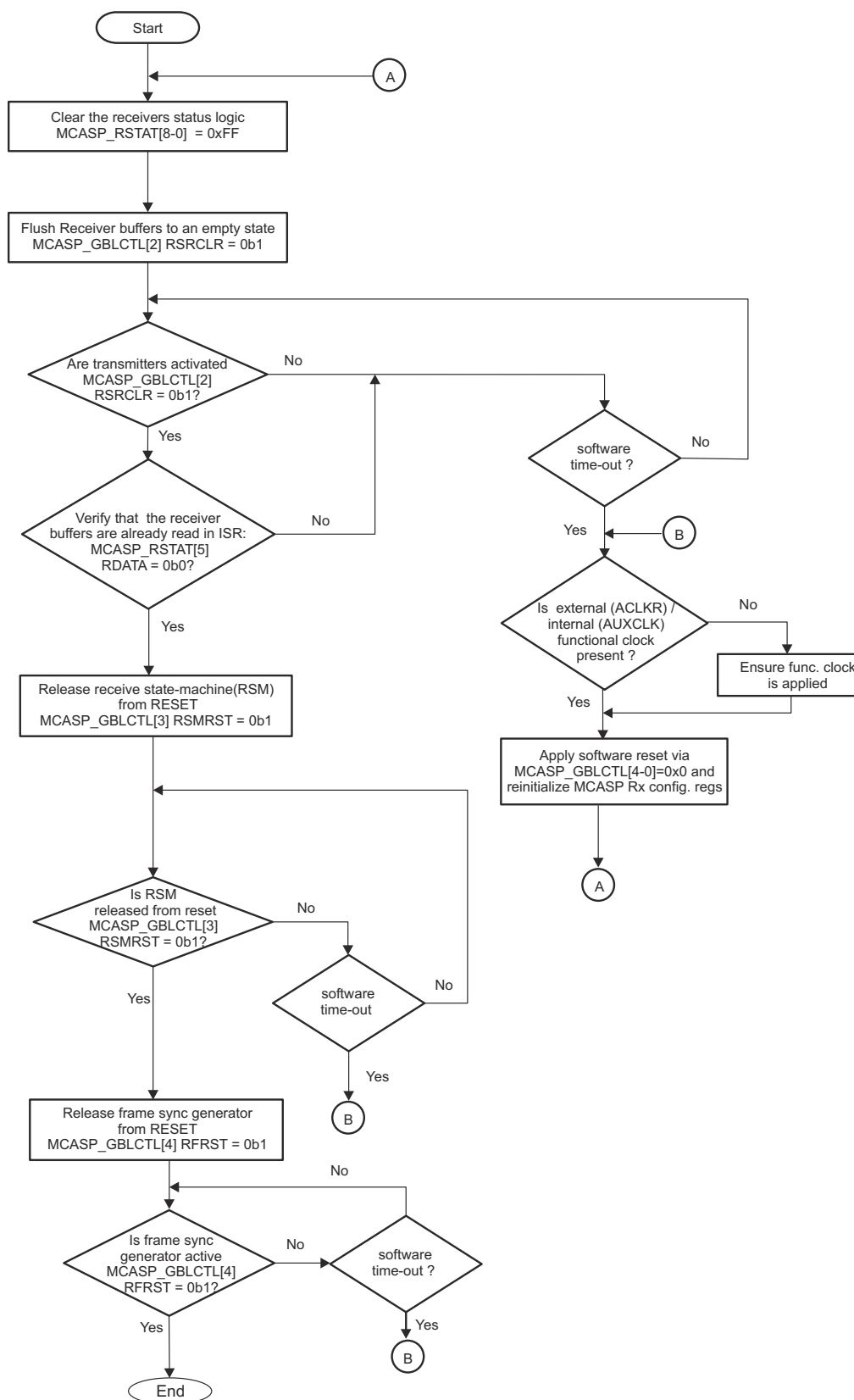
[Table 12-523](#) summarizes the subprocess call for the polling mode.

**Table 12-523. Subprocess Call Summary for Main Sequence – MCASP Reception Polling Method**

Subprocess Name	Cross-Reference
Error handling	<a href="#">Figure 12-520</a>

#### **12.5.2.5.2.2 Main Sequence – MCASP TDM - Interrupt Reception Method**

[Figure 12-516](#) shows the initial setup for interrupt-based reception.



mcasp-032

**Figure 12-516. Subsequence – TDM - Reception Startup Procedure**

Table 12-524 shows the configuration of the MCASP using an interrupt method for TDM- reception.

**Table 12-524. MCASP TDM- Interrupt Reception Model**

Step	Register/Bit Field/Programming Model	Value
Disable Rx DMA requests generation.	MCASP_PIDTCTL[0] RDATDMA	0x1
Enable the data ready event receive interrupt.	MCASP_RINTCTL[5] RDATA	0x1
Optional: Enable the receive error event interrupts.	MCASP_RINTCTL[2] RCKFAIL	0x1
	MCASP_RINTCTL[1] RSYNCERR	0x1
	MCASP_RINTCTL[0] ROVRN	0x1
Optional: Enable the start of frame interrupt.	MCASP_RINTCTL[7] RSTAFRM	0x1
Optional: Enable the last slot data interrupt	MCASP_RINTCTL[4] RLAST	0x1
<b>IF</b> read transfer is through the MCASP DATA port (MCASP_RFMT[3] RBUSEL is set to 0b0).		
Software test condition (setting is done in step4 of the <i>MCASP Receivers Global Initialization for TDM-Mode Operation</i> - see <i>MCASP Receivers Global Initialization for TDM-Mode Operation</i> )		
Enable the DATA port error based interrupt.	MCASP_RINTCTL[3] RDMAERR	0x1
<b>ELSE</b>		
Disable the DATA port error based interrupt.	MCASP_RINTCTL[3] RDMAERR	0x0
<b>ENDIF</b>		
TDM - Transmission Startup Procedure	See <a href="#">Figure 12-516</a> .	

These registers are for MCASP TDM- interrupt reception model: MCASP\_XGBLCTL, MCASP\_RSTAT.

#### 12.5.2.5.2.2.3 Main Sequence – MCASP TDM - Mode DMA Reception Method

Table 12-525 shows the configuration of the MCASP using the DMA method for reception. Possible interrupt error event servicing is also considered. shows the initial setup for DMA - based transmission.

#### Note

Because of the DATA port burst access capability with the DMA method, it is strongly recommended that DMA transfers are initiated through the MCASP DATA port.

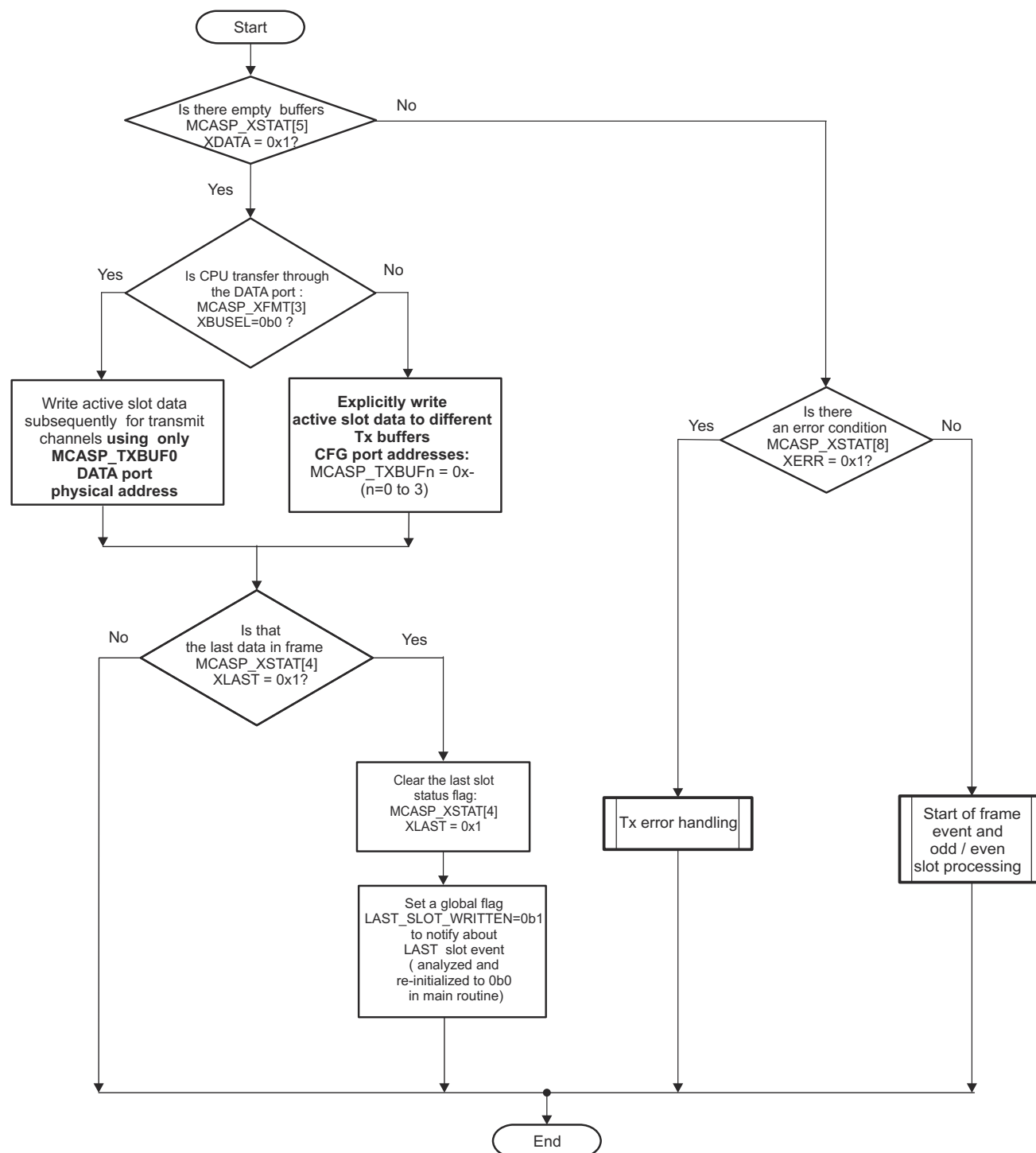
**Table 12-525. MCASP DMA Reception Model with Interrupt Events Servicing**

Step	Register/Bit Field/Programming Model	Value
<b>Recommended:</b> Select DATA port to access the transmit buffers.	MCASP_RFMT[3] RBUSEL	0x0
Enable the Rx DMA requests generation.	MCASP_PIDTCTL[0] RDATDMA	0x0
Enable the Rx DMA error event, because of MCASP DATA port usage.	MCASP_RINTCTL[3] RDMAERR	0x1
Optional: Enable the receive error event interrupts.	MCASP_RINTCTL[2] RCKFAIL	0x1
	MCASP_RINTCTL[1] RSYNCERR	0x1
	MCASP_RINTCTL[0] ROVRN	0x1
Optional: Enable the start of frame interrupt.	MCASP_RINTCTL[7] RSTAFRM	0x1
Optional: Enable the last slot data interrupt.	MCASP_RINTCTL[4] RLAST	0x1
Disable the data ready event receive interrupt, as DMA is used to service this request.	MCASP_RINTCTL[5] RDATA	0x0
DMA startup reception procedure. This procedure is identical than the one shown in <a href="#">Figure 12-516</a> . The only difference is that DMA automatically services all the RINT events raised by the MCASP, and no CPU data processing intervention is required. The CPU is involved only in error handling shown in <a href="#">Figure 12-518</a> .	See <a href="#">Figure 12-516</a> .	

### 12.5.2.5.2.3 MCASP Event Servicing

#### 12.5.2.5.2.3.1 MCASP DIT-/TDM- Transmit Interrupt Events Servicing

Figure 12-517 shows the flow of DIT-/TDM- mode transmit interrupt events servicing for the MCASP module.

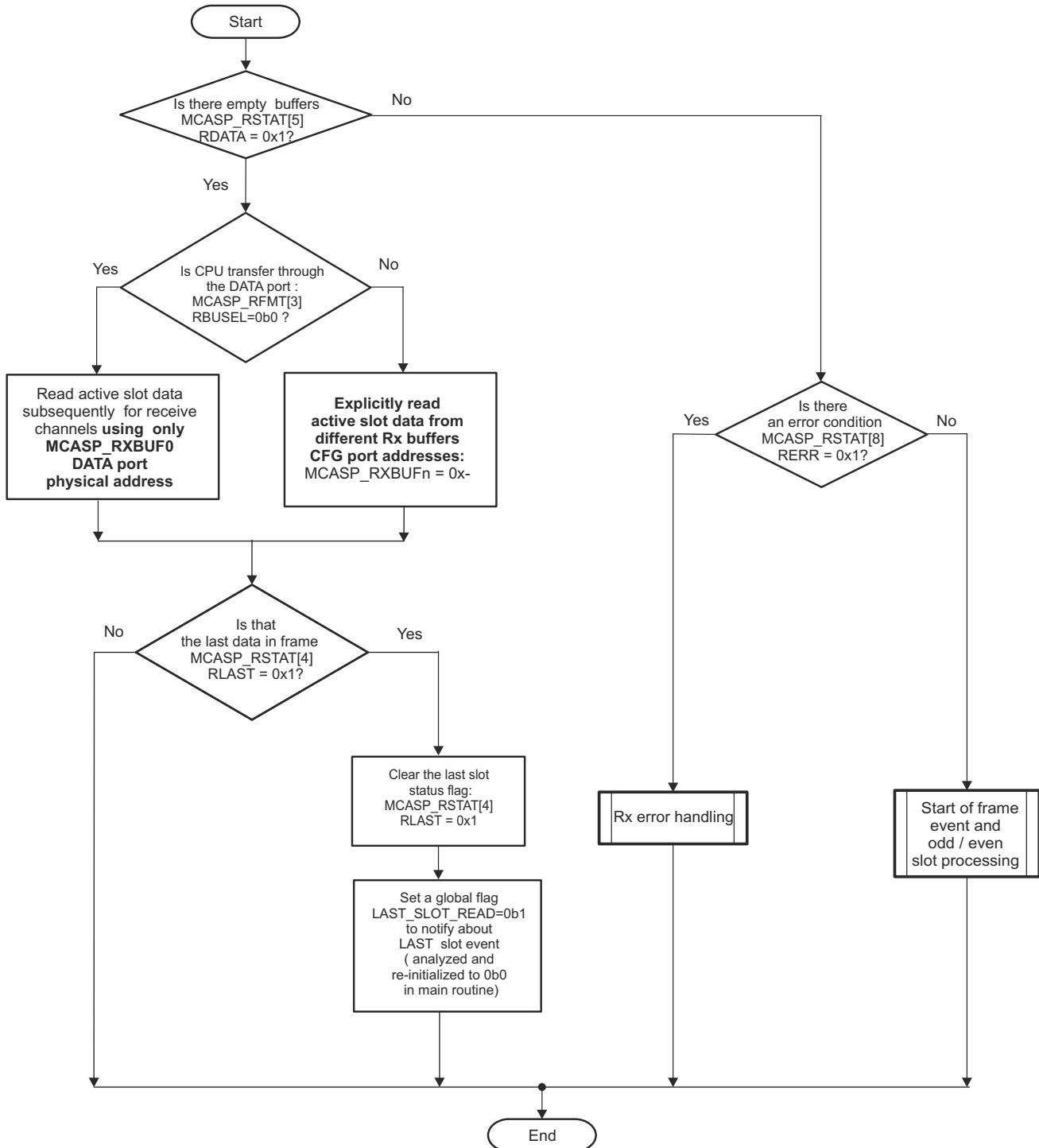


mcasp-029

**Figure 12-517. MCASP Transmit Interrupt Events Servicing**

### 12.5.2.5.2.3.2 MCASP TDM- Receive Interrupt Events Servicing

Figure 12-518 shows the flow of DIT-/TDM- mode transmit interrupt events servicing for the MCASP module.



mcasp-033

**Figure 12-518. MCASP Receive Interrupt Events Servicing**

These registers are for MCASP receive interrupt events servicing: MCASP\_RSTAT, MCASP\_RBUF<sub>n</sub>, MCASP\_RFMT (n = 0 to 15).

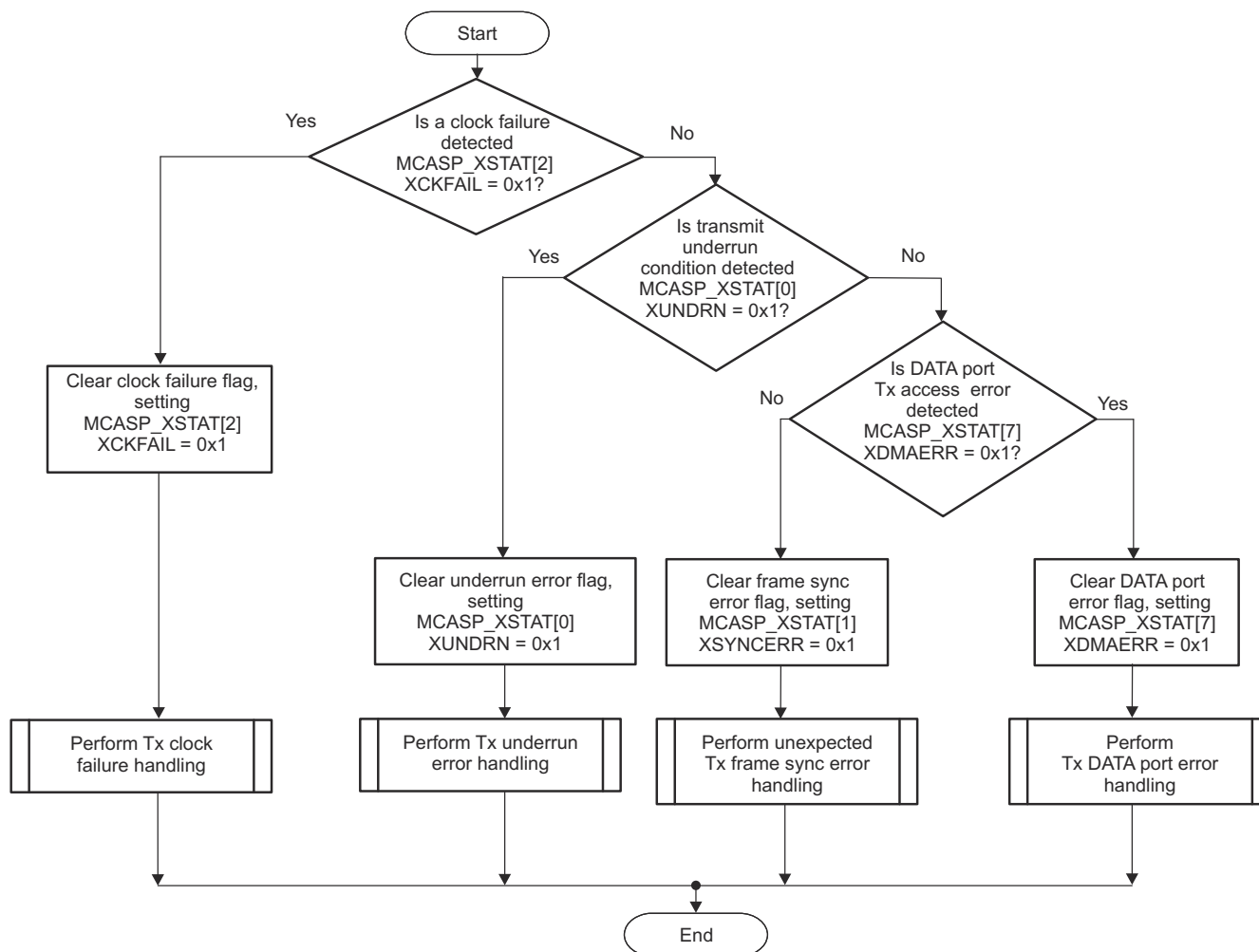
Table 12-526 lists the subprocess call summary for receive interrupt events servicing.

**Table 12-526. Subprocess Call Summary for Receive Interrupt Events Servicing**

Subprocess Name	Cross-Reference
MCASP receive error handling	Figure 12-520
Start of frame handling	Section 12.5.2.4.13.2, <i>Receive Data Ready Event and Interrupt</i>

### 12.5.2.5.2.3.3 Subsequence – MCASP DIT-/TDM -Modes Transmit Error Handling

Figure 12-519 shows the transmit error handling schema for the MCASP, which can be implemented as part of the Tx interrupt service routine or as part of the Tx polling sequence.



mcasp-030

**Figure 12-519. MCASP Transmit Error Handling**

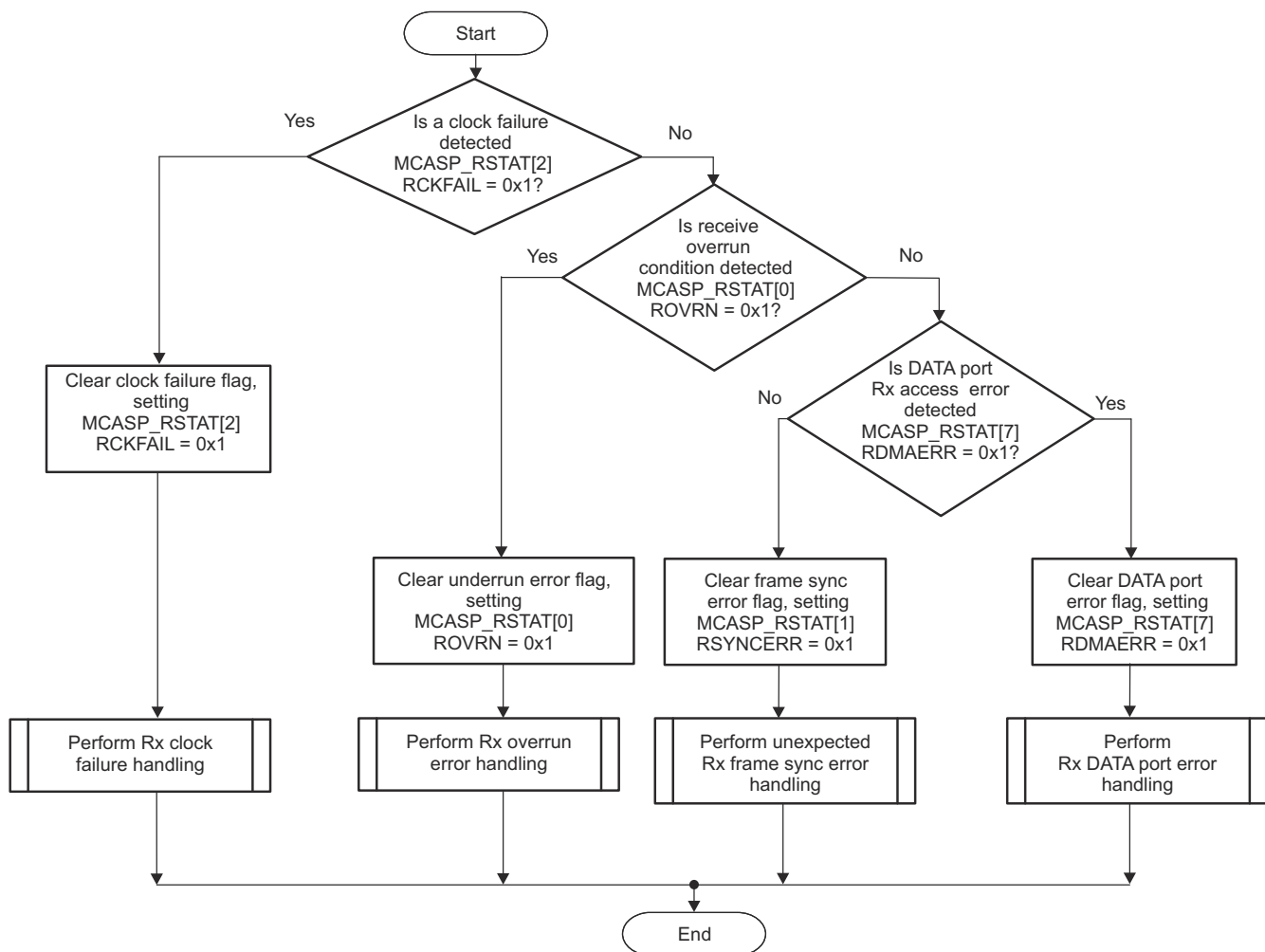


### Note

- For more information about transmit clock failure handling, see [Section 12.5.2.4.16.6.2, Transmit Clock Failure Check and Recovery](#).
- For more information about transmit buffer underrun handling, see [Section 12.5.2.4.16.1, Buffer Underrun Error - Transmitter](#).
- For more information about DATA port Tx error handling, see [Section 12.5.2.4.16.3, DATA Port Error - Transmitter](#).
- For more information about unexpected Tx frame sync error handling, see [Section 12.5.2.4.16.5, Unexpected Frame Sync Error](#).

#### 12.5.2.5.2.3.4 Subsequence – MCASP Receive Error Handling

Figure 12-520 shows the receive error handling schema for the MCASP, which can ONLY be implemented as part of the Rx polling sequence.



mcasp-031

**Figure 12-520. MCASP Receive Error Handling**

This register is for MCASP receive error handling: MCASP\_RSTAT.

---

**Note**

- For more information about receive clock failure handling, see [Section 12.5.2.4.16.6.3](#), *Receive Clock Failure Check and Recovery*.
  - For more information about receive buffer overrun handling, see [Section 12.5.2.4.16.2](#), *Buffer Overrun Error - Receiver*.
  - For more information about DATA port Rx error handling, see [Section 12.5.2.4.16.4](#), *DATA Port Error - Receiver*.
  - For more information about unexpected Rx frame sync error handling, see [Section 12.5.2.4.16.5](#), *Unexpected Frame Sync Error*.
-

## 12.6 Display Subsystem (DSS) and Peripherals

This section describes the Display Subsystem (DSS) in the device, integrated together with display peripherals, such as the MIPI Display Serial Interface (DSI) transmitter host controller with associated MIPI DSI (Physical Layer) D-PHY module (DPHY\_TX), and the Embedded Display Port (eDP) transmitter with associated SerDes and auxiliary complex I/O (eDP AUXPHY) modules.

### 12.6.1 DSS Overview

The DSS is a flexible composition-enabled display subsystem, that supports multiple high resolution display outputs. It consists of a Display Controller (DISPC) and a Frame Buffer Decompression Core (FBDC). The DISPC supports a multi-layer blending and transparency for each of its display outputs. The DISPC also supports a write-back pipeline with scaling to enable memory-to-memory composition and/or to capture a display output for Ethernet video encoding. The DISPC supports gamma correction and programmable color control in both source and destination pipelines for enhanced color space control and video output quality. The DISPC video outputs connect to various display interface controllers, integrated at the SoC level, to drive specific displays in the end application. The DSS also includes freeze frame detection and data signature check capabilities for each display output path.

The MIPI DSI v1.3.1 Controller (DSITX) implements the stream arbitration and low-level protocol layer functionalities required by MIPI DSI 1.3 standard. It supports up to 4 x 2.5 Gbps D-PHY data lanes in a single-link configuration and handles the byte lane mapping per use case (1, 2, 3, or 4-lanes). The accompanying DSI (Physical Layer) D-PHY module (DPHY\_TX) provides the video output interfacing by implementing a four-lane MIPI D-PHY transmitter.

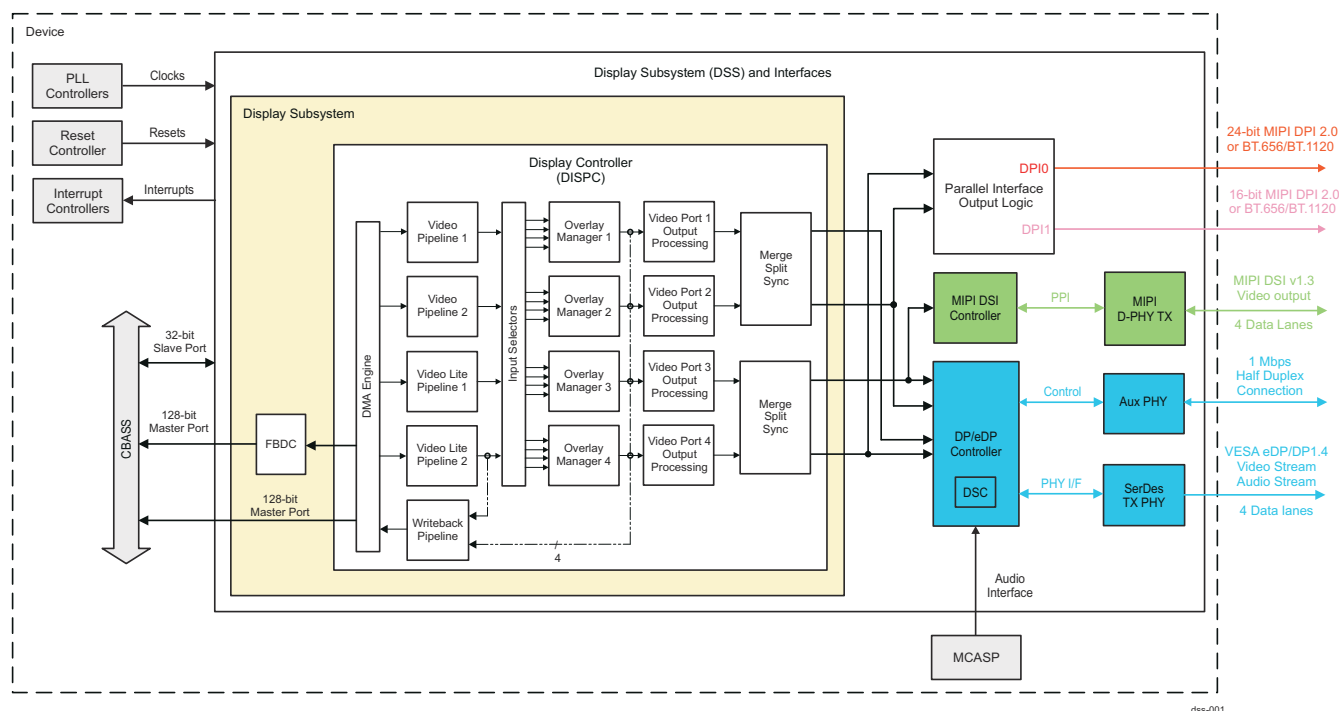
The VESA DP1.4/eDP1.4 Compliant Transmitter Host Controller (EDP) can output up to 4 video streams (through Multiple Stream Transport / MST) and one audio stream through the 4-lane accompanying SerDes module. It provides up to 25.92 Gbps of application bandwidth. An additional auxiliary PHY (AUXPHY) module implements a doubly-terminated differential pair required for 1 Mbps data rates over a long (15m) cable. This is a half duplex, Manchester II encoded connection, used by the EDP to configure the link. The AUX channel is used to access the Display Port Configuration Data (DPCD) memory area in the sink device and to implement VESA Enhanced Extended Display Identification Data (EDID) function.

[Table 12-527](#) shows the DSS and display peripherals allocation across device domains.

**Table 12-527. DSS and Display Peripherals Allocation Across Device Domains**

Module Instance	Domain		
	WKUP	MCU	MAIN
DSS0 (DISPC0 and FBDC)	-	-	✓
DSI0 with shared DPHY_TX0	-	-	✓
EDP0 with AUXPHY0	-	-	✓

[Figure 12-521](#) shows an overview of the DSS module and connections to the display peripherals.



**Figure 12-521. DSS and Display Peripherals Overview**

### 12.6.1.1 DSS Features

The Display Controller (DISPC) supports:

- An embedded DMA Controller with the following features:
  - Support for 1D-only DMA transfers
  - Support for 48b addressable memory space
  - Support for memory fragmentation through external PAT (Page Address Translation) table at SoC level
  - Integrated shared buffer management for pipelines within the same DMA controller group
    - Support for up to 8K-pixels wide frame buffer for 8/16/32/64 bits per pixel (total buffer space requirement not exceeding the total DMA buffer size of the DISPC)
  - Programmable DMA request management
    - Programmable buffer thresholds for request priority
    - Bandwidth limiter on write requests (insertion on idle cycles between requests)
    - Self-refresh using the DMA buffers
    - Arbitration between normal/low priority pipelines
  - Support for source image flip along X and Y-axis
  - Support for secure access to firewall protected frame buffer in DDR memory
- Two input display processing Video Pipelines, each supporting:
  - Input RGB source pixel formats:
    - Bitmap-1, Bitmap-2, Bitmap-4, Bitmap-8
    - ARGB16-4444, ABGR16-4444, RGBA16-4444, RGB16-565, BGR16-565, ARGB16-1555, ABGR16-1555
    - RGB16-565/BGR16-565 with a separate A8 plane
    - RGB24-888, BGR24-888
    - ARGB32-8888, ABGR32-8888, BGRA32-8888, RGBA32-8888, ARGB32-2101010, ABGR32-2101010
    - ARGB64-16161616, RGBA64-16161616

- Additionally, equivalent RGBx, xRGB, xBGR, and BGRx pixel formats defined, considering that A component of RGBA, ARGB, ABGR, BGRA pixel formats is ignored by HW (for example, ARGB → xRGB); ("A" can be ignored by not selecting Alpha pixel)
- Pre-multiplied ARGB/RGBA formats
- Input YUV source pixel formats:
  - Packed: YUV422-UYYVY, YUV422-YUV2
  - 2-plane: YUV420-NV12, YUV420-NV21, YUV422-NV12, YUV422-NV21
  - 8-bit per component support for all YUV formats
  - 10-bit IMG pack-format and 12-bit TI fully packed format support for 420/422 (internally processed as 10-bit component data)
  - 16-bit unpacked source format (internally processed as 10-bit component data)
- Programmable poly-phase filter (scaler):
  - Independent horizontal and vertical re-sampling: up-sampling (up to x16) and down-sampling (down to x0.25)
  - Maximum input width of 2048 pixels using ARGB pixels and 5-tap, 4096 pixels using YUV pixels and 5-tap, and 4096 pixels using ARGB pixels and 3-tap. No limitation on the input height.
  - Alpha blending factor is re-scaled like the R, G and B color components
  - Implementation with 16 phases with symmetrical coefficients
- Programmable color space conversion from YUV422/YUV420 (chroma upsampled to YUV444 using the scaler) into ARGB48-12121212
- Programmable VC1 range mapping
- Programmable Brightness/Contrast/Hue/Saturation
- Programmable Gamma Correction LUT
- Luma Key generation
- 10-bit processing pipeline
- Two input display processing Video Lite Pipelines, each supporting:
  - Input RGB source pixel formats:
    - Bitmap-1, Bitmap-2, Bitmap-4, Bitmap-8
    - ARGB16-4444, ABGR16-4444, RGBA16-4444, RGB16-565, BGR16-565, ARGB16-1555, ABGR16-1555
    - RGB16-565/BGR16-565 with a separate A8 plane
    - RGB24-888, BGR24-888
    - ARGB32-8888, ABGR32-8888, BGRA32-8888, RGBA32-8888, ARGB32-2101010, ABGR32-2101010
    - ARGB64-16161616, RGBA64-16161616
    - Additionally, equivalent RGBx, xRGB, xBGR, and BGRx pixel formats defined, considering that A component of RGBA, ARGB, ABGR, BGRA pixel formats is ignored by HW (for example, ARGB → xRGB); ("A" can be ignored by not selecting Alpha pixel)
    - Pre-multiplied ARGB/RGBA formats
  - Input YUV source pixel formats:
    - Packed: YUV422-UYYVY, YUV422-YUV2
    - 2-plane: YUV420-NV12, YUV420-NV21, YUV422-NV12, YUV422-NV21
    - 8-bit per component support for all YUV formats
    - 10-bit IMG pack-format and 12-bit TI fully packed format support for 420/422 (internally processed as 10-bit component data)
    - 16-bit unpacked source format (internally processed as 10-bit component data)
  - YUV420 to YUV422 chroma upsampling using an average filter
  - YUV422 to YUV444 chroma upsampling using a 4-tap filter based on Catmull-Rom algorithm
  - Programmable color space conversion from YUV422/YUV420 into ARGB48-12121212
  - Programmable VC1 range mapping
  - Programmable Brightness/Contrast/Hue/Saturation
  - Programmable Gamma Correction LUT
  - Luma Key generation

- 10-bit processing pipeline
- One Write-back (WB) pipeline, supporting:
  - Destination RGB pixel formats:
    - ARGB16-4444, ABGR16-4444, RGBA16-4444, RGB16-565, BGR16-565, ARGB16-1555, ABGR16-1555
    - RGB24-888 (no support for BGR24-888)
    - RGB16-565/BGR16-565 with a separate A8 plane
    - ARGB32-8888, ABGR32-8888, BGRA32-8888, RGBA32-8888, ARGB32-2101010, ABGR32-2101010
    - ARGB64-16161616, RGBA64-16161616
  - Destination YUV pixel formats:
    - Packed: YUV422-UYVY, YUV422-YUV2
    - 2-plane: YUV420-NV12, YUV420-NV21, YUV422-NV12, YUV422-NV21
    - Only 8-bit output support for YUV formats
  - Programmable poly-phase filter (scaler):
    - Independent horizontal and vertical re-sampling: up-sampling (up to x16) and down-sampling (down to x0.25). When the output format of the WB pipeline includes a format change RGB/YUV422 → YUV420, the maximum downscaling provided by the WB scaler is x0.5.
    - Maximum input width of 2048 pixels using ARGB pixels and 5-tap, 4096 pixels using YUV pixels and 5-tap, and 4096 pixels using ARGB pixels and 3-tap. No limitation on the input height.
    - Alpha blending factor is re-scaled like the R, G and B color components
    - Implementation with 16 phases with symmetrical coefficients
  - Operation modes:
    - Output capture mode (to save one of the display outputs)
    - Memory-to-memory (M2M) mode (to save a M2M composition/conversion operation result)
- Four Overlay Managers (OVR), each supporting:
  - Input pixel format: ARGB48-12121212
  - Output pixel format: ARGB48-12121212 ("A" component data is only used for the Write-back path)
  - Overlay of the input pipelines (fully mapped to all input pipelines)
  - Up to 4 input layers blending, plus background layer
  - Transparency color key (source and destination)
  - Alpha blending support: Embedded pixel alpha (ARGB and RGBA), global pixel, and combination of global pixel and pixel alpha
  - Z-order programmable (full flexibility)
  - Color bar test pattern insertion
  - Any overlay output can be selected to drive the Write-back pipeline
- Four Video Port (VP) display outputs, each supporting:
  - 36-bit per pixel on the RGB output interface (12-bits per component)
  - Independent programmable timing generator, supporting up to 600 MHz pixel clock video formats (actual support limited by the maximum pixel clock provided to DSS at SoC level)
  - Independent programmable 10-bit gamma correction
  - Independent programmable multiple cycles output format on 8/9/12/16-bit interface (TDM)
  - Selection between RGB and YUV422 output pixel format (YUV422 only available when BT.656/BT.1120 output is enabled)
  - Configurable VP output mode: progressive mode only on RGB output, or progressive/interlaced mode on BT.656/BT.1120 output
- Internal data check diagnostic features:
  - Supports up to 4 programmable (position/size) check regions on the DISPC video port display outputs
  - Support for 1 check region on each input video pipeline output
  - MISR (Multiple Input Signature Register) used on each check region to perform data correctness check and/or freeze frame detection
- Local power management features:

- Low power saving modes
- Capability to associate all buffers with a single pipeline for a display self-refresh
- System interconnect ports:
  - Two 128-bit VBUSM master interfaces for data read/write
  - One 32-bit VBUSP slave interfaces for configuration

The MIPI Display Serial Interface (DSI) transmitter host controller supports:

- Compliance with MIPI DSI v1.3.1 protocol specification and previous specifications
- Compliance with MIPI Stereoscopic Display Formats (MIPI SDF v1.0) specification
- Video and command operational modes
- Both burst and non-burst modes for video mode data transmission (with either sync pulses or sync events)
- Up to 4 virtual channels via command mode
  - Supports data interleaving support for one synchronous stream (video mode) from the display controller, and up to three interleaved asynchronous streams (command mode) from the interconnect concurrently
  - Supports data interleaving for up to four interleaved asynchronous streams (virtual channels in command mode) from the interconnect
- Bi-directional communication and escape mode (only on Lane-0)
- Pixel clock rate range 25-330 MHz (10-bit/component, 4MPix @60fps performance). Actual performance is limited by the maximum pixel clock provided to the DSI controller at SoC level.
- Programmable display resolutions with maximum clock rate not exceeding the total available bandwidth over 4MPix @60fps (10-bit/component RGB)
- 16/18/24/30/36-bit RGB input data formats for video mode
- RGB16, RGB18 packed, and RGB24 input data formats for command mode
- All generic data types defined by MIPI
- Display Command Set (DCS): transparent to the protocol engine, no decoding and interpretation of the information from and to the peripheral
- ECC on the APB interface
- Data splitter for 2-, 3-, or 4-data lane configuration
- Connection to a single MIPI D-PHY complex I/O through an 8-bit Protocol Peripheral Interface (PPI)
- Bus contention recovery
- Video mode pattern generator: color bar pattern image and D-PHY BET testing pattern
- APB slave interface with 32-bit data and address for configuration

The MIPI DSI (Physical Layer) D-PHY module supports:

- Compliance with MIPI D-PHY 1.2 physical layer interface specification and features
- 1, 2 or 4 data lanes, in addition to clock signaling
- Maximum data rate up to 2.5 Gbps per data lane
- Protocol Peripheral Interface (PPI)
- HS continuous and burst mode
- LP (Low-Power), ULPM (Ultra-Lower Power Mode), and Shutdown modes
- Forward direction and reverse direction escape modes (only on Lane-0)
- Automatic termination control in both high-speed and low-power modes
- DSI D-PHY IO pad signals work at/with electrical specification specified by requested standards
- Single 32-bit VBUSP slave interface

The Embedded Display Port (eDP) transmitter host controller supports:

- Compliance with VESA Display Port (DP) 1.3 (with 1.4 DSC/FEC support) specification
- Compliance with VESA embedded Display Port (eDP) 1.4 specification
- Static configuration of either DP or eDP mode
- Link rates up to HBR3 (maximum application bandwidth up to 25.92Gbps)
- Pixel clock rate range 25-600 MHz (10-bit/component, 8MPix@60fps performance).
- Transceiver AUX CH (1 MHz Manchester II coding mode) for access of DPCD and EDID
- 8, 10, and 12 bpc (bits per component), in RGB/YCbCr444 colorimetry formats (CEA-861 compliant) and YCbCr422 (via simple decimation)



- Data splitter for 1-, 2-, or 4-data lane configuration
- SST (Single Stream Transport)
- MST (Multiple Stream Transport):
  - Up to 4 video and up to 1 audio sources
  - Support for up to 25 Gbps throughput (equivalent to approximately 4K + 2xFHD streams) use case
  - HDCP support on one video source
  - DSC (Display Stream Compression) support on one 4K or 2x2.5K streams
- HDCP 1.4 and HDCP 2.2 with True Random Number Generator (TRNG with 8 FRO)
- DSC (Display Stream Compression) encoded stream data transport via an embedded DSC core:
  - Supports VESA DSC 1.1 compliant video compression at 2x~3x compression ratios
  - Supports all DSC 1.1 mandatory encoding mechanisms (MMAP, BP, MPP and ICH)
  - Supports configurable maximum display resolution up to 8Mpix @60fps including (but not limited to) 4K@60 (4096x2160), and up to 8K wide-aspect-ratio resolution displays
  - Supports two hard encoder slices (peak pixel ratio <= 340MP/s on each slice)
  - Supports 8 and 10 bits per video component
  - Supports RGB/YCbCr4:4:4 video input format (Native Encoding) only
  - Supports Dual pixel streams or Split panel input streams
  - Supports dual or single transport link
- Forward Error Correction (FEC) encoder with/without DSC enabled in DP mode
- Various eDP specific features:
  - eDP DPCD registers
  - Full and Fast Link Training
  - (Regional) backlights and multi-touch command over AUX channel
  - Alternate Scrambler Reset
- Audio transport features:
  - I2S (LPCM/IEC60948/619376) audio input stream
  - Audio transport of uncompressed multichannel (up to eight channels) via SDP (Secondary-data Packet)
- Metadata transport via MSA (Main Stream Attribute) packet or via SDP
- Display Port transmitter functionalities:
  - Scrambler
  - 8/10-bit Encoder (in the DPTX core)
  - Inter Lane Skew Insertion
  - Training Pattern Generation – TPS1,2,3,4 PRBS7 and 80-bit custom training pattern generation (bypassing the scrambler and encoder)
- PAPB Interface:
  - The primary APB slave port controls the HD Display controller from the host processor
  - During the boot, the primary APB slave port enables access to the I-MEM and D-MEM, and to the full address space for debugging purposes
  - After the boot, the primary APB slave port enables direct register access to designated HW modules and enables communication with uCPU through a mailbox channel
- SAPB Interface
  - For configuring the embedded HDCP, this bus is considered as a secured APB to carry secured commands over the mailbox channel
- Internal diagnostic features:
  - ECC on the critical memories
  - Parity check on the configuration interface
  - Encoder self-check diagnostics support in the DSC core
  - Corrected/Uncorrected error interrupt generation
  - Injection of ECC and parity errors

The DP (Physical Layer) SERDES and Aux PHY modules support:

- DP1.3 HBR3 and eDP1.4a HBR3 throughput



- 1, 2, or 4 lanes at 1.62 Gbps, 2.7 Gbps, 5.4 Gbps, and 8.1 Gbps per lane
- Additional link rates (2.16, 2.43, 3.24, 4.32 Gbps) per lane in eDP mode
- Reduced differential voltage swing (0.2/0.25/0.30/0.35/0.40/0.45) in eDP mode
- Hot Plug Detect (HPD) for connection detection and interrupt from sink
- Integrated Low Jitter, Fixed Bandwidth PLL
- 1 Mbps AUX PHY for link training, DPCD register access, HDCP authentication and EDID access.

#### 12.6.1.2 DSS Not Supported Features

The DISPC does not support the following features:

- 2D tiled buffer access
- On the fly rotation (90/270-degree)
- Sending ancillary data on the video port outputs during VBLANK period
- YUV422 planar formats (8-bit and 10-bit) for video pipelines and write-back (encoder/decoder compatibility)

The DSI controller does not support the following features:

- Audio data transmission
- Optional sub-link
- VESA DSC (Display Stream Compression)
- 12-bit wide input component width
- SDI video mode

The eDP transmitter does not support the following features:

- Multi-SST Operation (MSO) for segmented display panel support in eDP mode
- Type-C DP Alt Mode
- FAUX (Fast AUX)(720MBps AUX)
- SPDIF Audio Interface
- Advanced Low Power Management (ALPM)(Pending proposal in VESA to remove the requirement of ALPM in eDP)
- MCCS Support (other mode available to access needed registers in Sink device)
- PSR-1 mode (Full Panel Refresh Mode)
- Specific DP 1.4 standard features:
  - SDP splitting and chaining (DP 1.4 specification, chapter 2.2.5.12)
  - New SDP formats introduced in DP 1.4 specification
  - Video Stream Configuration Extension (VESA SDP), (DP 1.4 specification, chapter 2.2.5.10)
  - CEA SDP (DP 1.4 specification, chapter 2.2.5.11)
  - New DP 1.4 audio formats:
    - 3D LPCM audio (up to 32 audio channels) (DP 1.4 specification, chapter A.4.5, A.4.6)
    - 1-bit audio (DP 1.4 specification, chapter A.4.3)
    - Direct Stream Transfer (DST) audio (DP 1.4 specification, chapter A.4.7)
    - HBR 8-channel audio up to 1536 kHz
    - Changes to the Audio\_Stream SDP header (DP 1.4 specification, chapter 2.2.5.3.2)
  - New 3DChannelCount, OneBit\_DST\_Double\_Rate fields
  - New supported values for the Coding Type field
  - 16-bit data format
- Specific DSC core features:
  - VESA DSC 1.2 updated requirements (no support for YCbCr 422 or 420 native coding)
  - De-rasterization buffer
  - 12-bit wide input component width

## 12.6.2 DSS Environment

This section describes the interfaces handled by the Display Subsystem.

The DSS is capable of driving multiple displays in parallel, through a combination of interfaces:

- Two DPI parallel interfaces, directly delivered on SoC pads from DISPC Video Port (VP) outputs
- One MIPI Display Serial Interface (DSI)
- One Embedded DisplayPort (DP/eDP) interface

The pixel format mapping between the DISPC VP outputs and DPI, eDP, and DSI display peripherals varies depending on the interface used. The pixel bit color mapping for the DISPC VP outputs is as shown in [Figure 12-522](#).

Color Format	Pixel Width	Pixel Interface Mapping																																			
		35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RGB565	16-bit																																				
RGB666	18-bit																																				
RGB888	24-bit																																				
RGB101010	30-bit																																				
RGB121212	36-bit																																				

dss-007

**Figure 12-522. DSS Pixel Mapping on the DISPC Video Port Outputs**

The pixel format support at a DISPC VP output when connected to each of the DPI, eDP, and DSI peripherals is as listed in [Table 12-528](#).

**Table 12-528. DSS Pixel Format Interoperability between DISPC VP Output and DPI, eDP, and DSI**

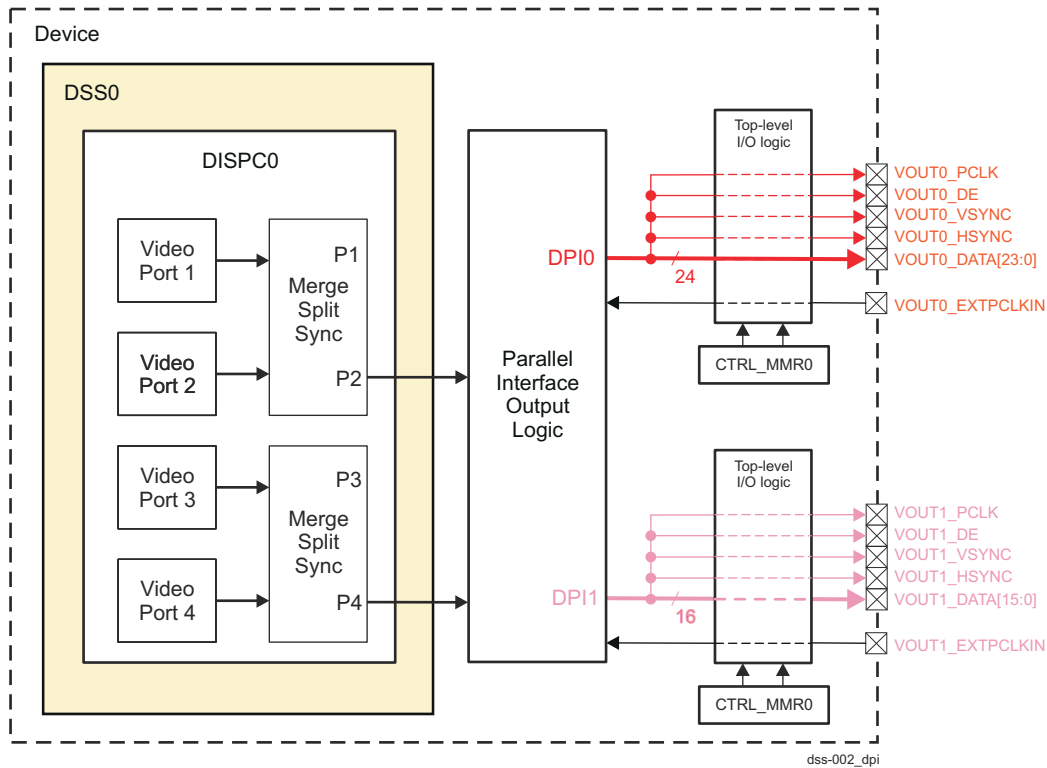
Pixel Format	DISPC VP Output Pixel Format	DPI Parallel Out	DSI		eDP
	DSS0_VP_CONTROL [10-8] DATALINES Register Field Value		DSI-DPI Interface	DSI-SDI Interface	
RGB565	0x1	Supported	Supported	Supported	Not supported
RGB666	0x2	Supported	Supported	Supported	Not supported
RGB888	0x3	Supported	Supported	Supported	Not supported
RGB101010	0x4	Not supported	Supported	Not supported	Supported
RGB121212	0x5	Not supported	Supported	Not supported	Supported

### Note

The DISPC VP outputs are connected to the display peripherals through the Merge-Split-Sync (MSS) block. For more details, see [Section 12.6.4.11.9, DISPC VP Merge-Split-Sync \(MSS\) Module](#).

### 12.6.2.1 DISPC Environment

[Figure 12-523](#) shows the DSS DPI parallel interface signals. The output data bus of the DISPC video ports is 36 bits wide. The DSS DPI parallel interface uses up to the 24 LSB bits [23:0] of the data bus.



**Figure 12-523. DSS DPI Parallel Interface Signals**

Each DISPC video port in [Figure 12-523](#) can be connected to any DPI output via the parallel interface output logic.

The [3-0] DPI\_0\_CONN and [7-4] DPI\_1\_CONN fields in the [DSS0\\_COMMON\\_DISPC\\_CONNECTIONS](#) register define the DISPC VP connections to DPI0 and DPI1 outputs, respectively.

The DISPC video ports output the required data and control signals to device pads to support one of the following display interface modes:

- Parallel MIPI DPI 2.0 (Digital Pixel Interface): RGB 16/18/24-bit output with separate sync signals.
- BT.656/BT.1120 interface: YUV422 output (8/10-bit modes) with embedded syncs.
- BT.601 interface: YUV422 output with discrete syncs. For more details, see [Section 12.6.2.1.4, VSYNC/HSYNC/DE Signal Export to SoC Boundary](#).

[Table 12-529](#) describes the DSS DPI parallel interface I/O signals.

**Table 12-529. DSS DPI Parallel Interface I/O Signals**

Module Pin	Device Level Signal	I/O <sup>(1)</sup>	Description
<b>DPI0 Interface</b>			
DSS_DPI0_DATA[23:0]	VOUT0_DATA[23:0]	O	Pixel data output. RGB data for MIPI DPI 2.0 interface. YUV data for BT.656/BT.1120 interfaces.
DSS_DPI0_VSYNC	VOUT0_VSYNC	O	Vertical synchronization. The frame synchronization pulse (vsync) toggles after all the lines in a frame are transmitted and a programmable number of line clock cycles has elapsed at the beginning and the end of each frame.
DSS_DPI0_HSYNC	VOUT0_HSYNC	O	Horizontal synchronization. The line synchronization pulse (hsync) toggles after all pixels in a line are transmitted and a programmable number of pixel clock wait-states has elapsed at the beginning and the end of each line.

**Table 12-529. DSS DPI Parallel Interface I/O Signals (continued)**

Module Pin	Device Level Signal	I/O <sup>(1)</sup>	Description
DSS_DPI0_PCLK	VOUT0_PCLK	O	Pixel clock output.
DSS_DPI0_DE	VOUT0_DE	O	Pixel data output-enable signal to indicate when data must be latched using the pixel clock.
DSS_DPI0_EXTPCLKIN	VOUT0_EXTPCLKIN	I	Pixel clock input from external source.
-	VOUT0_VP0_DE	O	Alternative pixel data output-enable signal from DISPC VP1. <sup>(2)</sup>
-	VOUT0_VP0_HSYNC	O	Alternative horizontal synchronization signal from DISPC VP1. <sup>(2)</sup>
-	VOUT0_VP0_VSYNC	O	Alternative vertical synchronization signal from DISPC VP1. <sup>(2)</sup>
-	VOUT0_VP2_DE	O	Alternative pixel data output-enable signal from DISPC VP3. <sup>(2)</sup>
-	VOUT0_VP2_HSYNC	O	Alternative horizontal synchronization signal from DISPC VP3. <sup>(2)</sup>
-	VOUT0_VP2_VSYNC	O	Alternative vertical synchronization signal from DISPC VP3. <sup>(2)</sup>
<b>DPI1 Interface</b>			
DSS_DPI1_DATA[15:0]	VOUT1_DATA[15:0]	O	Pixel data output. RGB data for MIPI DPI 2.0 interface. YUV data for BT.656/BT.1120 interfaces.
DSS_DPI1_VSYNC	VOUT1_VSYNC	O	Vertical synchronization. Vertical synchronization. The frame synchronization pulse (vsync) toggles after all the lines in a frame are transmitted and a programmable number of line clock cycles has elapsed at the beginning and the end of each frame.
DSS_DPI1_HSYNC	VOUT1_HSYNC	O	Horizontal synchronization. Horizontal synchronization. The line synchronization pulse (hsync) toggles after all pixels in a line are transmitted and a programmable number of pixel clock wait-states has elapsed at the beginning and the end of each line.
DSS_DPI1_PCLK	VOUT1_PCLK	O	Pixel clock output.
DSS_DPI1_DE	VOUT1_DE	O	Pixel data output-enable signal to indicate when data must be latched using the pixel clock.
DSS_DPI1_EXTPCLKIN	VOUT1_EXTPCLKIN	I	Pixel clock input from external source.
-	VOUT1_VP0_DE	O	Alternative pixel data output-enable signal from DISPC VP1. <sup>(2)</sup>
-	VOUT1_VP0_HSYNC	O	Alternative horizontal synchronization signal from DISPC VP1. <sup>(2)</sup>
-	VOUT1_VP0_VSYNC	O	Alternative vertical synchronization signal from DISPC VP1. <sup>(2)</sup>
<b>FSYNC Interface<sup>(2)</sup></b>			
-	DSS_FSYNC0	O	Video Output Frame Sync 0 (from DISPC VP1 VSYNC).
-	DSS_FSYNC1	O	Video Output Frame Sync 1 (from DISPC VP2 VSYNC).
-	DSS_FSYNC2	O	Video Output Frame Sync 2 (from DISPC VP3 VSYNC).
-	DSS_FSYNC3	O	Video Output Frame Sync 3 (from DISPC VP4 VSYNC).

(1) I = Input; O = Output

(2) For more details on the usage of the alternative sync signals, see [Section 12.6.2.1.4, VSYNC/HSYNC/DE Signal Export to SoC Boundary](#).

### Note

For more information about device level signals, see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Data Manual.

The effective output pixel clock rate for interleaved data formats (that is, BT.656 output mode or TDM (Time Division Multiplexed) output mode) will be either 1/2 or 1/3 of the maximum pixel clock rate, respectively, depending on the interleaving ratio.

#### 12.6.2.1.1 RGB Data Output

This section describes the pixel data bus for RGB formats and shows timing diagrams of transactions and synchronizations.

For the active matrix display type, one pixel per pixel clock is displayed. The diagrams represent the configuration of assertion of the data on the rising edge of the pixel clock. It is possible to program the interface timings to output the data on the falling edge of the pixel clock.

Figure 12-524 through Figure 12-527 show the interface to 12-, 16-, 18-, and 24-bit RGB active matrix displays. Each vertical line represents one output pixel. The width of the data bus can be configured through DSS0\_VP\_CONTROL[10-8] DATALINES register bitfield.

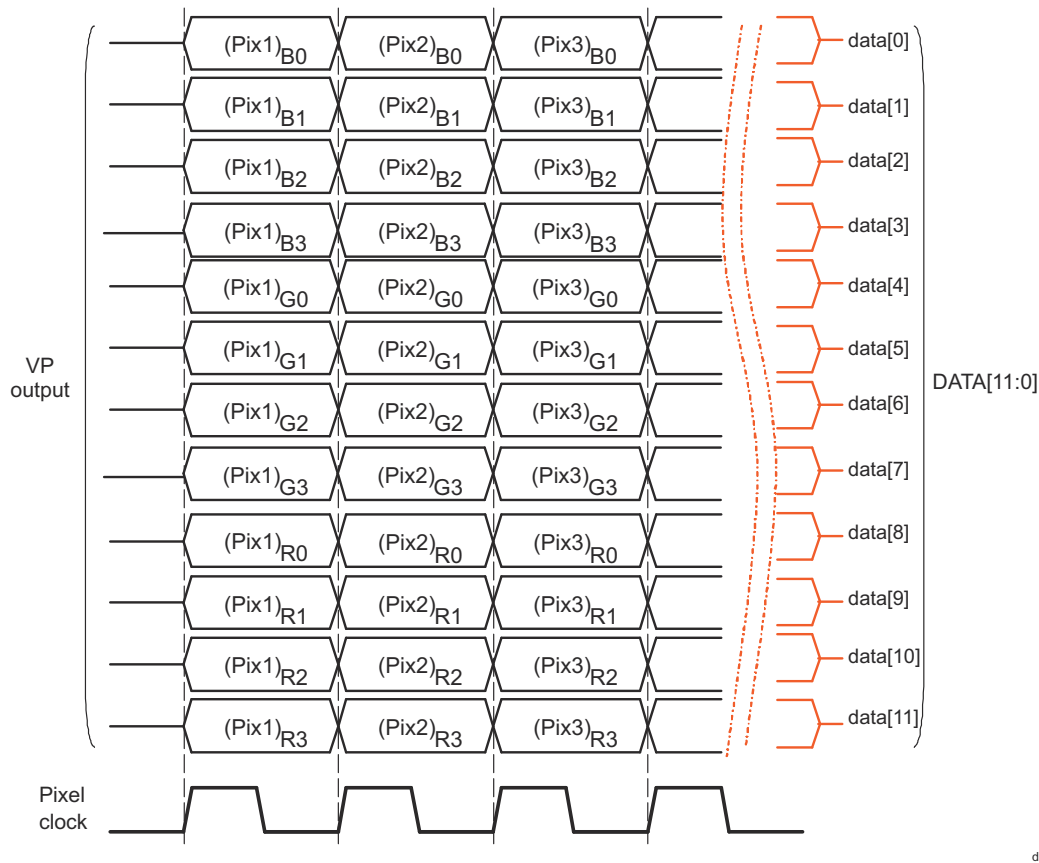
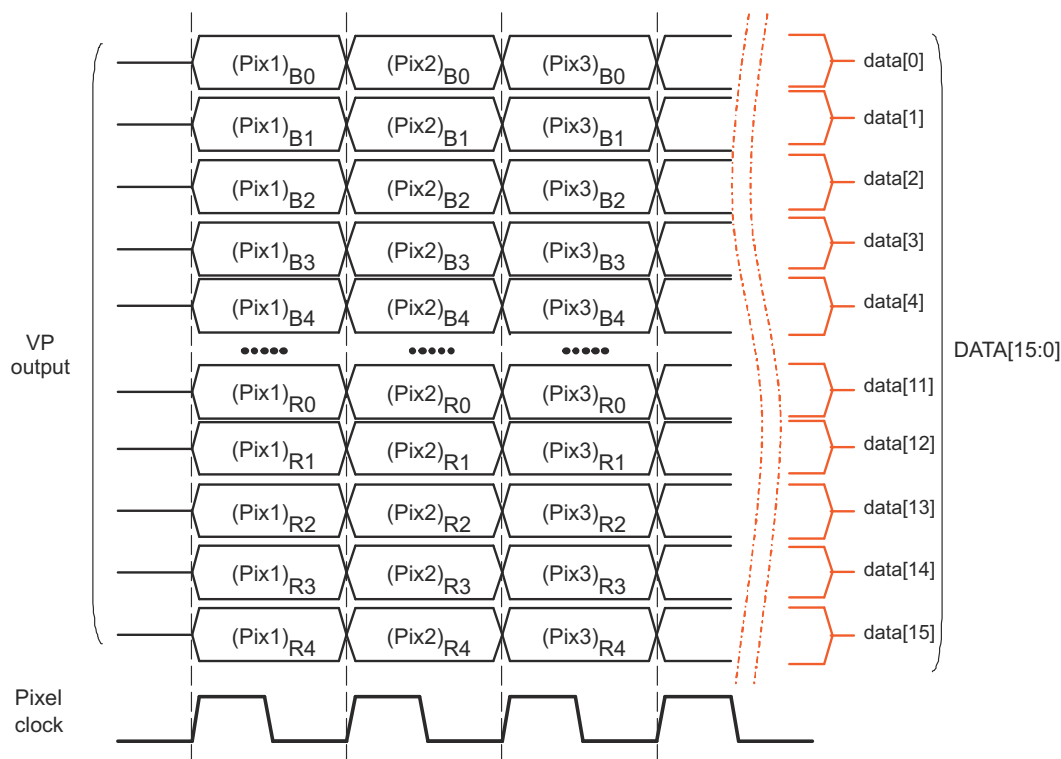
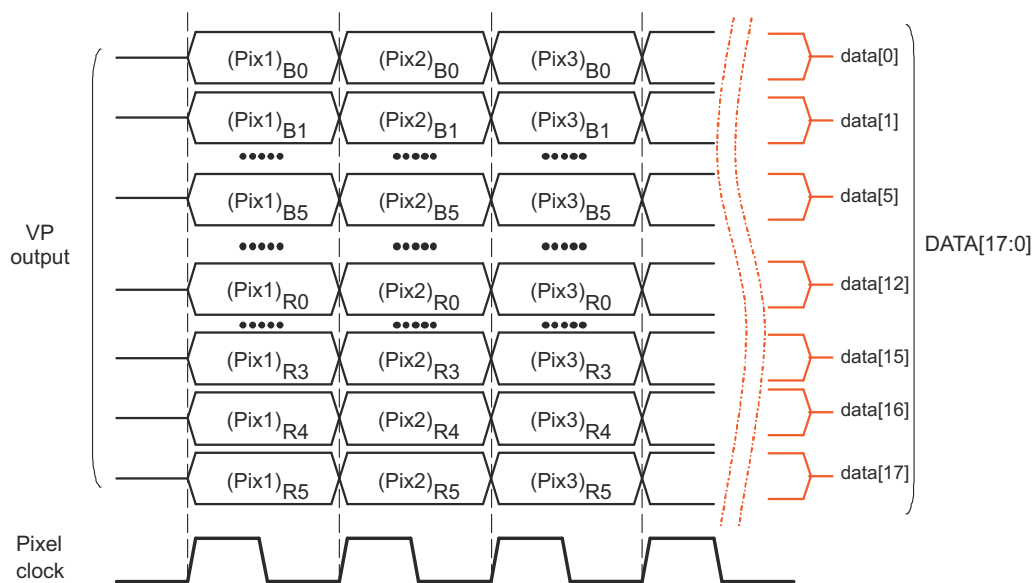


Figure 12-524. DISPC Video Port Pixel Data - 12-bit RGB Active Matrix



dispc-051

**Figure 12-525. DISPC Video Port Pixel Data - 16-bit RGB Active Matrix**



dispc-052

**Figure 12-526. DISPC Video Port Pixel Data - 18-bit RGB Active Matrix**

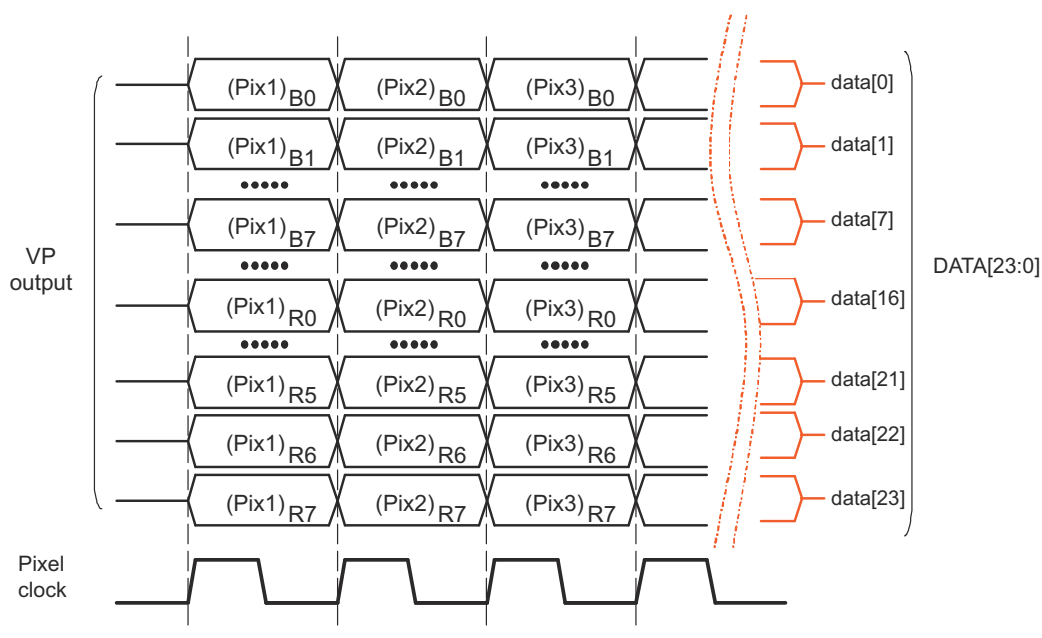


Figure 12-527. DISPC Video Port Pixel Data - 24-bit RGB Active Matrix

#### 12.6.2.1.2 YUV Data Output (BT.656/BT.1120)

Figure 12-528 shows the signal mapping on the DATA[23:0] output data bus for the BT.656 mode. Bits [9-0] are dedicated for BT.656 mode (10-bit). In BT.656 mode however, for compatibility with existing 8-bit interfaces, the two LSBs are ignored and only bits [9-2] are effectively used.

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Unused														BT.656									

Figure 12-528. DISPC Video Port Data Mapping for BT.656 Mode

Figure 12-529 shows the signal mapping on a DATA[23:0] output data bus for the BT.1120 mode. Bits [19-10] (CbCr) and [9-0] (Y) are used in 20-bit mode. Bits [19-12] (CbCr) and [9-2] (Y) are used in 16-bit mode (YCbCr422).

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Unused				BT.1120 (CbCr, 20-bit mode)										BT.1120 (Y, 20-bit mode)									

Figure 12-529. DISPC Video Port Data Mapping for BT.1120 Mode

The DISPC VP outputs support both interlace and progressive content in BT.656/BT.1120 modes. For more information on timings configuration, see [Section 12.6.4.11.8, DISPC VP Timing Generator and Display Panel Settings](#).

#### Note

In progressive BT.656/BT.1120 mode the maximum output resolution will be limited, as it requires two pixel clock cycles to send out one pixel.

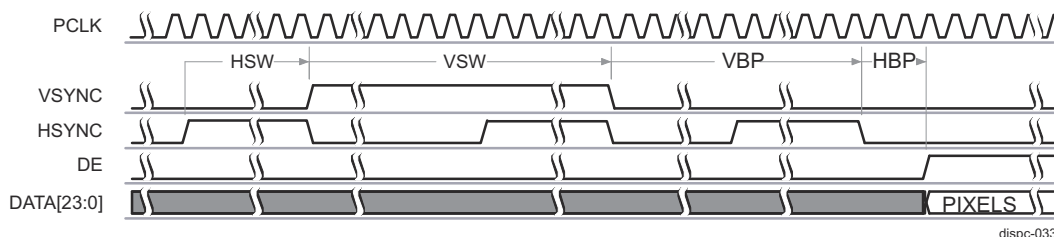
### 12.6.2.1.3 Display Timing Diagrams

Figure 12-530 through Figure 12-533 show examples with timing diagrams of synchronization signals and pixel clocks for active matrix panels. The DISPC video ports directly drive these signals, which are related to the programmable fields listed in Table 12-530. For more information, see also Section 12.6.4.11.8, *DISPC VP Timing Generator and Display Panel Settings*.

**Table 12-530. DISPC Video Port Register Fields for Active Matrix Display**

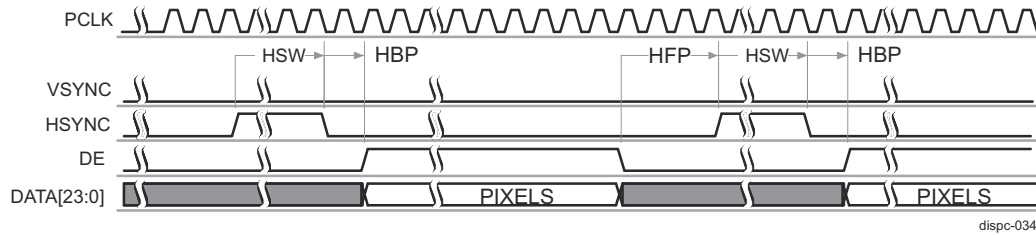
Name	Register	Description
PPL	DSS0_VP_SIZE_SCREEN[13-0] PPL value + 1	Pixels per line
LPP	DSS0_VP_SIZE_SCREEN[29-16] LPP value + 1	Lines per panel
HBP	DSS0_VP_TIMING_H[31-20] HBP value + 1	Horizontal back porch
HFP	DSS0_VP_TIMING_H[19-8] HFP value + 1	Horizontal front porch
HSW	DSS0_VP_TIMING_H[7-0] HSW value + 1	Horizontal synchronization pulse width
VBP	DSS0_VP_TIMING_V[31-20] VBP value	Vertical back porch
VFP	DSS0_VP_TIMING_V[19-8] VFP value	Vertical front porch
VSW	DSS0_VP_TIMING_V[7-0] VSW value + 1	Vertical synchronization pulse width
ALIGN	DSS0_VP_POL_FREQ[18] ALIGN	Alignment between HSYNC and VSYNC assertion
ONOFF	DSS0_VP_POL_FREQ[17] ONOFF	HSYNC and VSYNC pixel clock control
RF	DSS0_VP_POL_FREQ[16] RF	HSYNC and VSYNC pixel clock edge control
IEO	DSS0_VP_POL_FREQ[15] IEO	Invert output enable
IPC	DSS0_VP_POL_FREQ[14] IPC	Invert PCLK
IHS	DSS0_VP_POL_FREQ[13] IHS	Invert HSYNC
IVS	DSS0_VP_POL_FREQ[12] IVS	Invert VSYNC

- Active matrix timing configuration 1:
  - DSS0\_VP\_POL\_FREQ[17] ONOFF = 0
  - DSS0\_VP\_POL\_FREQ[16] RF = 0
  - The HSYNC and VSYNC signals are driven on the opposite edge of PCLK from the pixel data.
  - DSS0\_VP\_POL\_FREQ[15] IEO = 0
  - The DE signal is active high.
  - DSS0\_VP\_POL\_FREQ[14] IPC = 0
  - The pixel data are driven on the rising edge of PCLK.
  - DSS0\_VP\_POL\_FREQ[13] IHS = 0
  - The HSYNC signal is active high.
  - DSS0\_VP\_POL\_FREQ[12] IVS = 0
  - The VSYNC signal is active high.
  - DSS0\_VP\_POL\_FREQ[18] ALIGN = 0
  - The VSYNC and HSYNC assertion is not aligned.

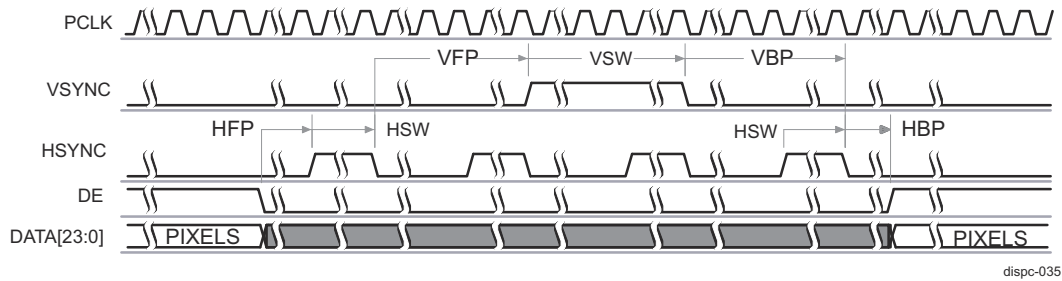


**Figure 12-530. DISPC Display Timing Diagram of Configuration 1 (Start of Frame)**

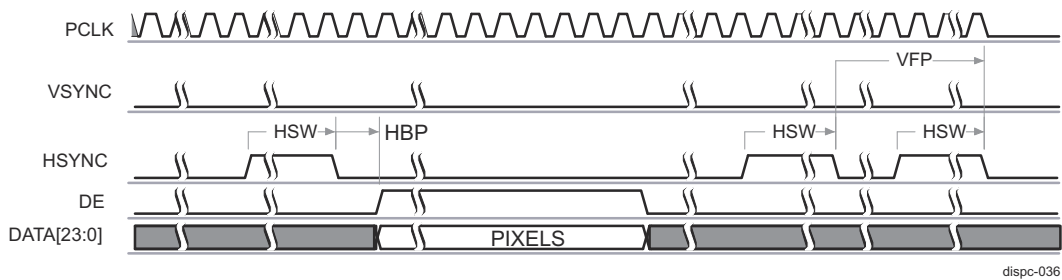




**Figure 12-531. DISPC Display Timing Diagram of Configuration 1 (Between Lines)**



**Figure 12-532. DISPC Display Timing Diagram of Configuration 1 (Between Frames)**



**Figure 12-533. DISPC Display Timing Diagram of Configuration 1 (End of Frame)**

- Active matrix timing configuration 2:
  - DSS0\_VP\_POL\_FREQ[17] ONOFF = 1
  - DSS0\_VP\_POL\_FREQ[16] RF = 1

The HSYNC and VSYNC signals are driven on the rising edge of PCLK.

- DSS0\_VP\_POL\_FREQ[15] IEO = 1

The DE signal is active low.

- DSS0\_VP\_POL\_FREQ[14] IPC = 1

The pixel data is driven on the falling edge of PCLK.

- DSS0\_VP\_POL\_FREQ[13] IHS = 1

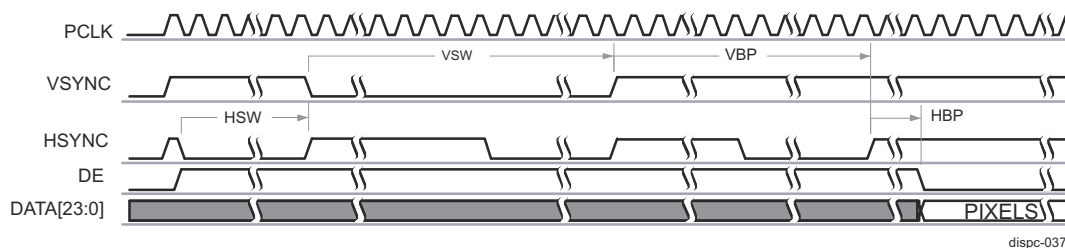
The HSYNC signal is active low.

- DSS0\_VP\_POL\_FREQ[12] IVS = 1

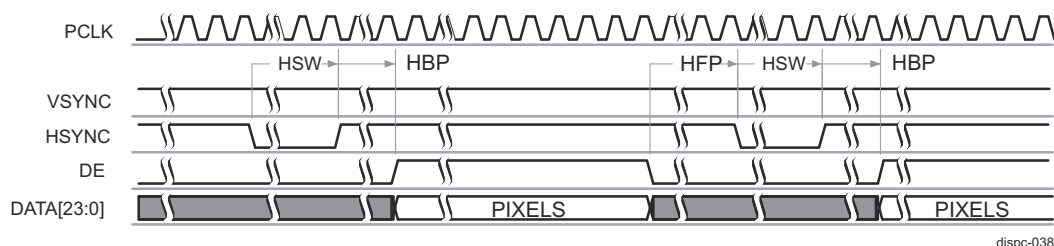
The VSYNC signal is active low.

- DSS0\_VP\_POL\_FREQ[18] ALIGN = 0

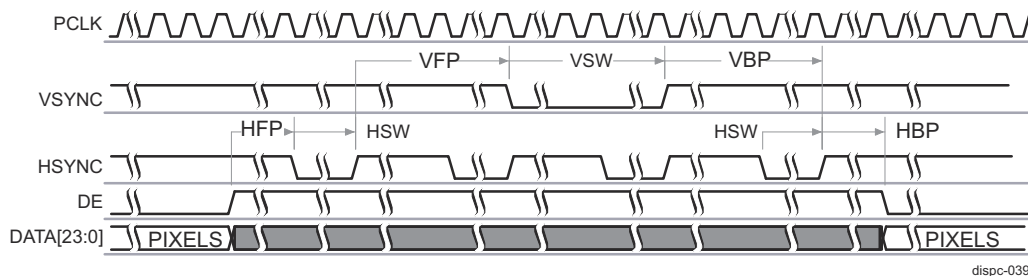
The VSYNC and HSYNC assertion is not aligned.



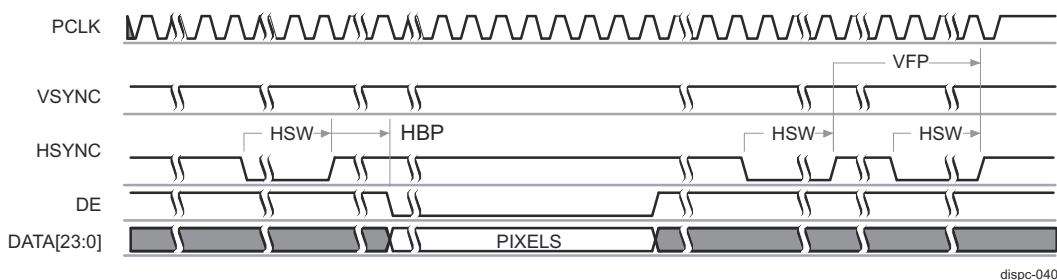
**Figure 12-534. DISPC Display Timing Diagram of Configuration 2 (Start of Frame)**



**Figure 12-535. DISPC Display Timing Diagram of Configuration 2 (Between Lines)**



**Figure 12-536. DISPC Display Timing Diagram of Configuration 2 (Between Frames)**



**Figure 12-537. DISPC Display Timing Diagram of Configuration 2 (End of Frame)**

- Active matrix timing configuration 3:
  - DSS0\_VP\_POL\_FREQ[17] ONOFF = 1
  - DSS0\_VP\_POL\_FREQ[16] RF = 1

The HSYNC and VSYNC signals are driven on the rising edge of PCLK.

- DSS0\_VP\_POL\_FREQ[15] IEO = 0

The DE signal is active high.

- DSS0\_VP\_POL\_FREQ[14] IPC = 0

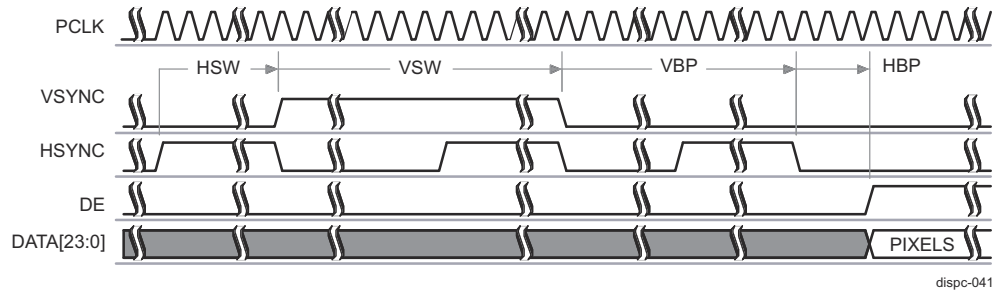
The pixel data are driven on the rising edge of PCLK.

- DSS0\_VP\_POL\_FREQ[13] IHS = 0

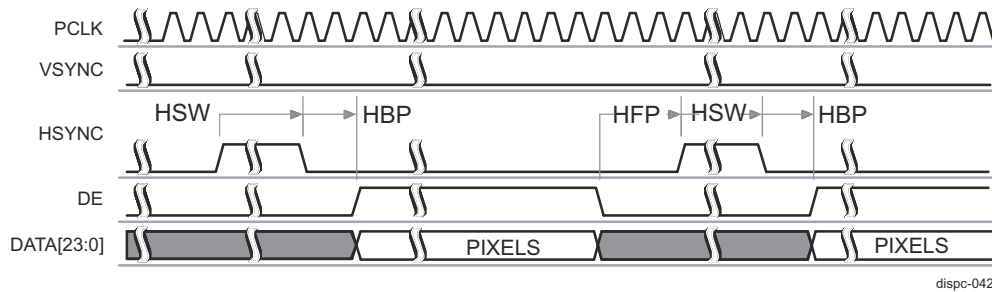
The HSYNC signal is active high.

- DSS0\_VP\_POL\_FREQ[12] IVS = 0  
The VSYNC signal is active high.
- DSS0\_VP\_POL\_FREQ[18] ALIGN = 0  
The VSYNC and HSYNC assertion is not aligned.

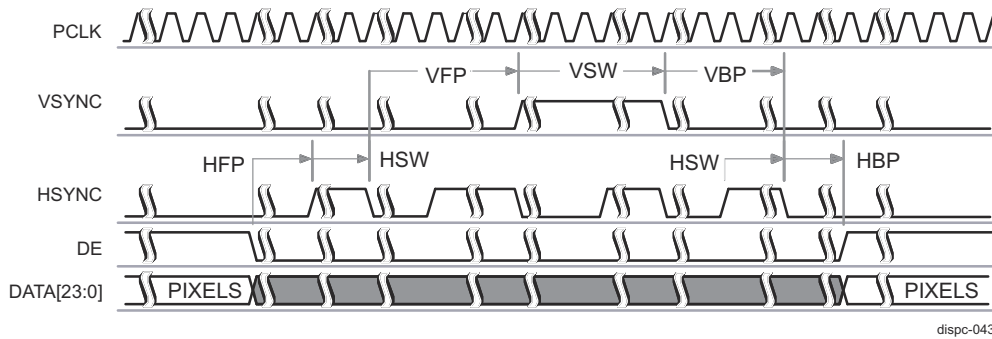
The VSYNC and HSYNC assertion is not aligned.



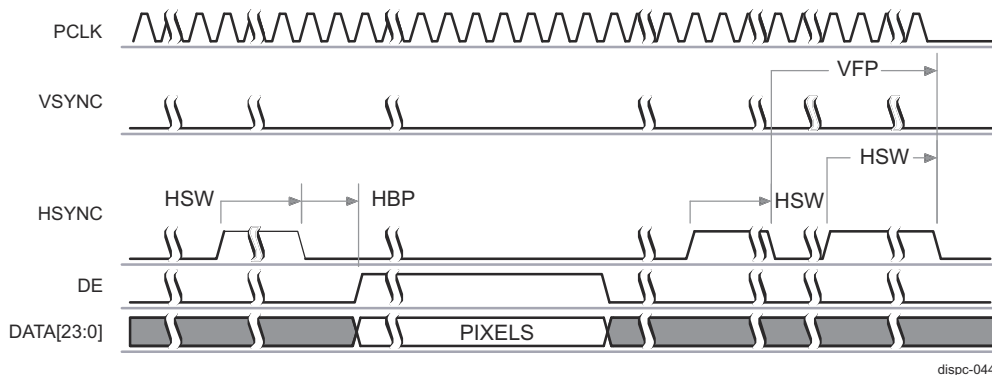
**Figure 12-538. DISPC Display Timing Diagram of Configuration 3 (Start of Frame)**



**Figure 12-539. DISPC Display Timing Diagram of Configuration 3 (Between Lines)**



**Figure 12-540. DISPC Display Timing Diagram of Configuration 3 (Between Frames)**



**Figure 12-541. DISPC Display Timing Diagram of Configuration 3 (End of Frame)**

#### 12.6.2.1.4 VSYNC/HSYNC/DE Signal Export to SoC Boundary

The DSS exports the VSYNC, HSYNC and DE signals of all DISPC video port outputs to the SoC boundary. This feature is supplementary to the signals already exported out on the parallel DPI interfaces (DPI0/DPI1). This allows the support of the following use cases:

- BT.601 interface (YUV output with discrete syncs)
- FSYNC support for external camera sensor

The necessary singnal muxing is achieved at SoC level (via I/O pin multiplexing).

#### BT.601 Standard Support

The DSS only supports YUV output with embedded syncs (BT.656 and BT.1120 standards). YUV output with discrete sync (BT.601 standard) is not supported natively on the DISPC VP outputs.

At SoC level, it is possible to achieve BT.601 operation on a DPI parallel out (DPI0/DPI1) by combining two DISPC VP outputs as follows:

- Enable one DISPC VP to output YUV422 data in BT.656 mode on either DPI0 or DPI1 output.
- Use an alternative DISPC VP, that is synchronized and programmed (in RGB TDM mode), to output the discrete sync signals on the same DPI output. This can be achieved by configuring the I/O pin multiplexing at SoC level.
- The following restrictions apply:
  - If DPI0 is used to output YUV422 data, then the VSYNC/HSYNC/DE signals of either VP1 or VP3 can be output on DPI0.
  - If DPI1 is used to output YUV422 data, then the VSYNC/HSYNC/DE signals of only VP1 can be output on DPI1.

#### FSYNC Support for External Camera Sensor

An external camera sensors can use the VSYNC output from DSS as a synchronization input (FSYNC input). To make this connection, the VSYNC signals of all DISPC video ports are made available at SoC boundary.

Generally, in the cases where the DISPC VP is connected to DSI or eDP, the VSYNC information is "lost" as a separate signal at SoC boundary, although this VSYNC is still available at DISPC module boundary. For such cases, the VSYNCs are exported to SoC level in parallel to the DSI/eDP connection.

#### 12.6.2.2 DSI Environment

[Figure 12-542](#) shows the DSI interface signals.

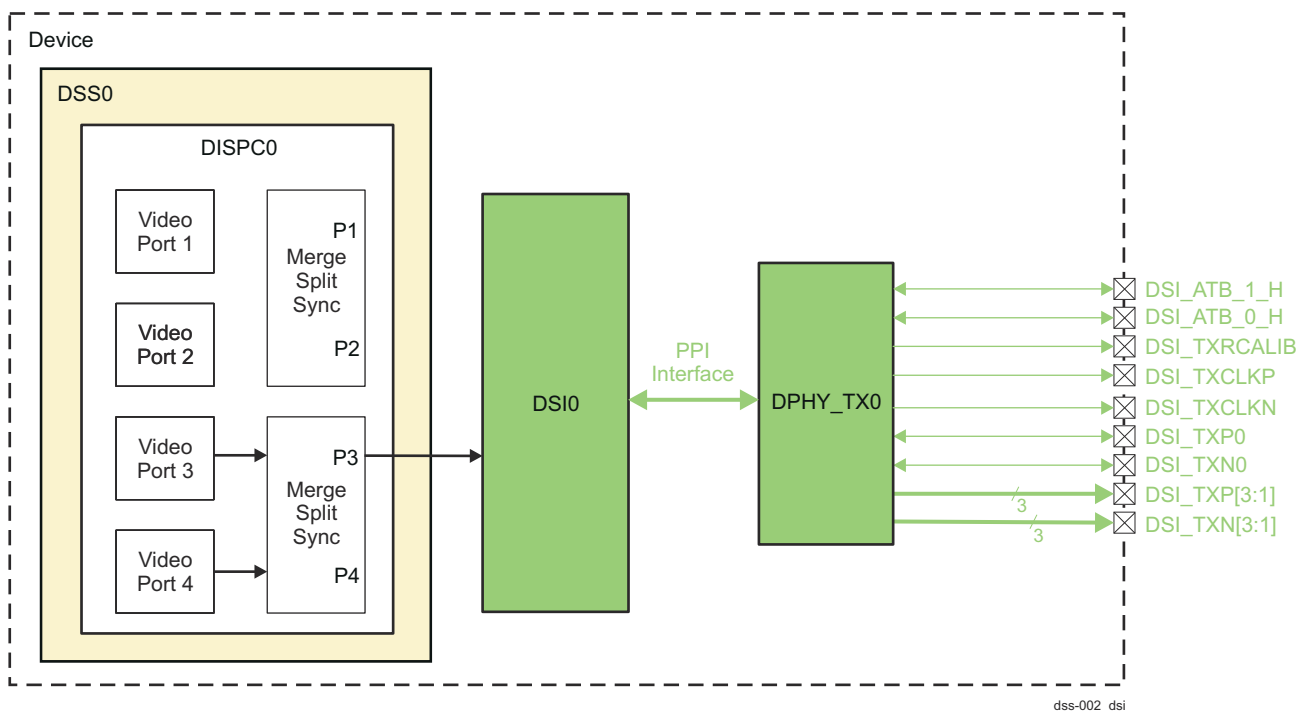

**Figure 12-542. DSI Interface Signals**

Table 12-531 describes the DSI I/O signals.

**Table 12-531. DSI Interface I/O Signals**

Device Level Signal	I/O <sup>(1)</sup>	Description
DSI_TXN0	I/O	External differential I/O Lane 0
DSI_TXP0	I/O	
DSI_TXN1	O	External differential output Lane 1
DSI_TXP1	O	
DSI_TXN2	O	External differential output Lane 2
DSI_TXP2	O	
DSI_TXN3	O	External differential output Lane 3
DSI_TXP3	O	
DSI_TXCLKN	O	External differential clock Lane
DSI_TXCLKP	O	
DSI_TXRCALIB	A	Pin for external calibration resistor
DSI_ATB_0_H	I/O	Analog test bus
DSI_ATB_1_H	I/O	

(1) I = Input; O = Output; I/O = Bidirectional; A = Analog

#### Note

For more information about device level signals, see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Data Manual.

### 12.6.2.3 EDP Environment

Figure 12-543 shows the EDP interface signals.

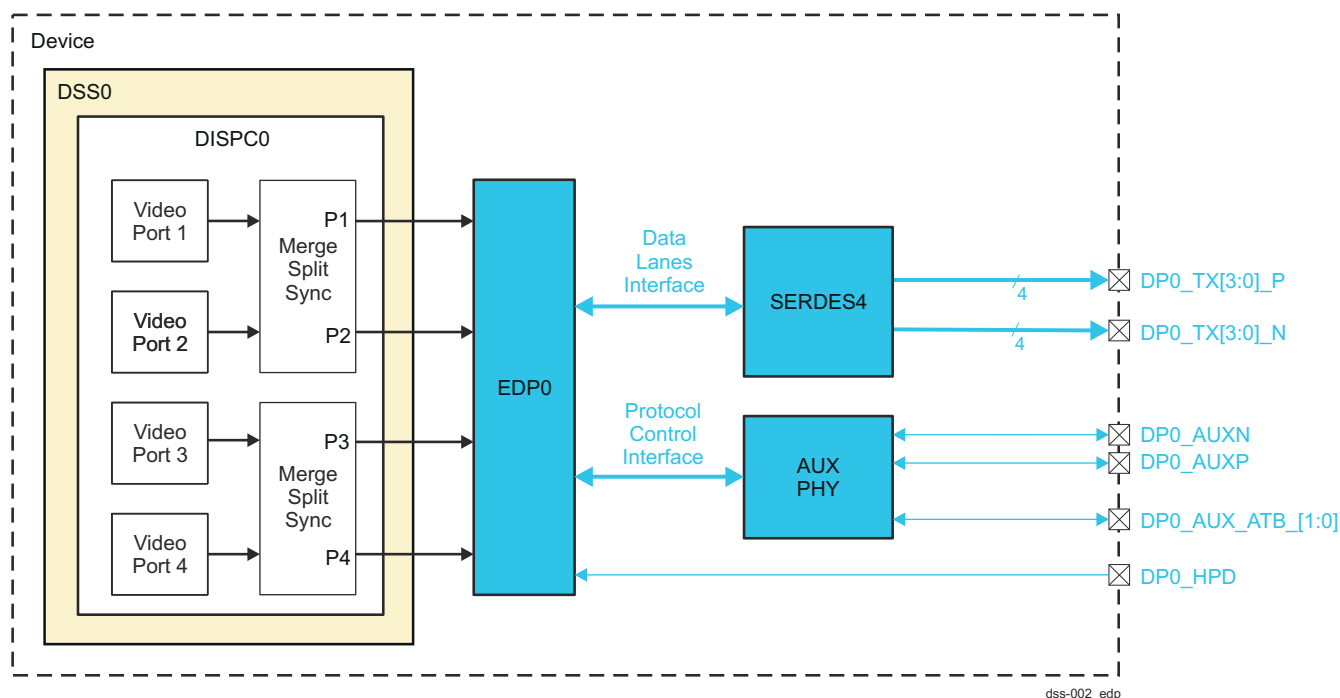

**Figure 12-543. DP/eDP Interface Signals**

Table 12-532 describes the EDP I/O signals.

**Table 12-532. DP/eDP I/O Signals**

Device Level Signal	I/O <sup>(1)</sup>	Description
<b>EDP TX PHY (SERDES)</b>		
DP0_TX0_N	O	External differential output Lane 0
DP0_TX0_P	O	
DP0_TX1_N	O	External differential output Lane 1
DP0_TX1_P	O	
DP0_TX2_N	O	External differential output Lane 2
DP0_TX2_P	O	
DP0_TX3_N	O	External differential output Lane 3
DP0_TX3_P	O	
<b>EDP AUX PHY</b>		
DP0_AUXN	I/O	Auxiliary channel differential transceiver
DP0_AUXP	I/O	
DP0_AUX_ATB_0	I/O	Analog test bus
DP0_AUX_ATB_1	I/O	
<b>EDP</b>		
DP0_HPD	I	Hot plug detect input

(1) I = Input; O = Output; I/O = Bidirectional

### Note

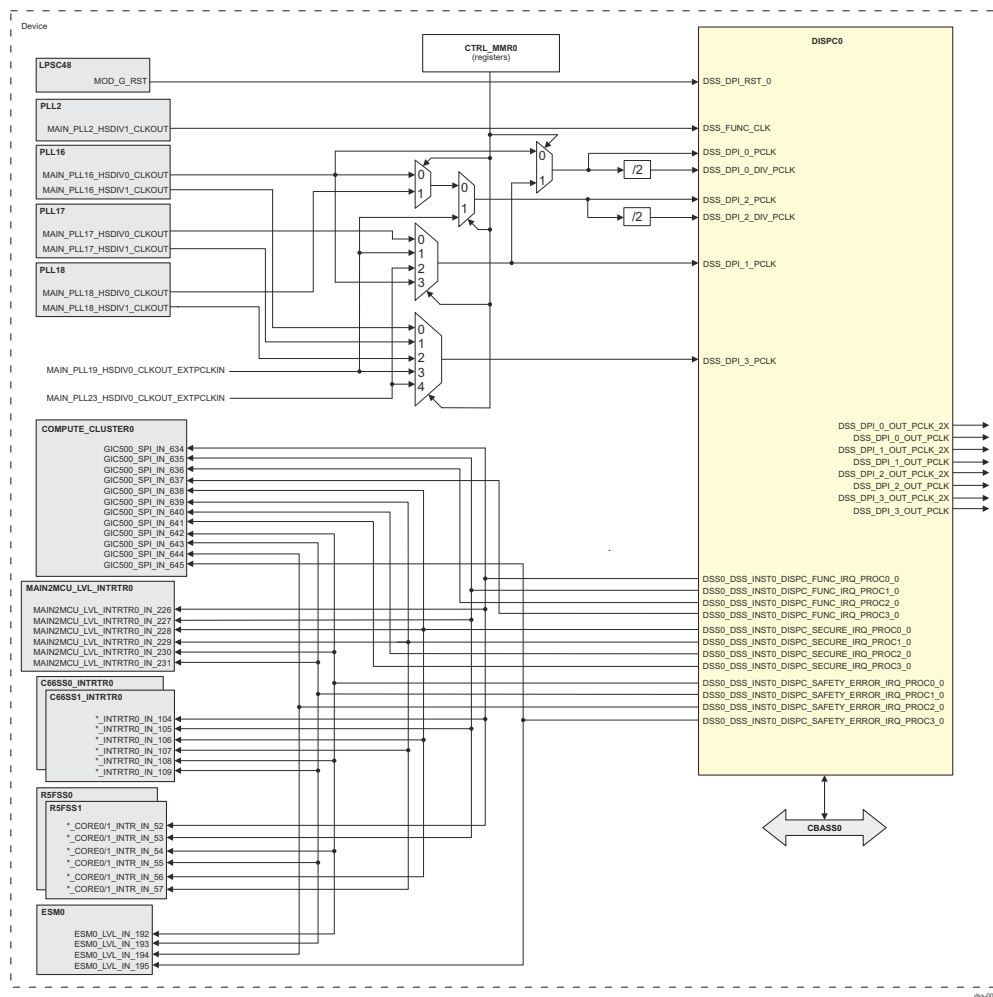
For more information about device level signals, see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Data Manual.

## 12.6.3 DSS Integration

This section describes the DSS integration in the device, including information about clocks, resets, and hardware requests.

### 12.6.3.1 DISPC Integration

There is one DISPC module integrated in the device MAIN domain. [Figure 12-544](#) shows the integration of DISPC0.



**Figure 12-544. DISPC Integration**

Table 12-533 through Table 12-535 summarize the integration of DISPC in the device MAIN domain.

**Table 12-533. DISPC Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
DISPC0	PSC0	PD2	LPSC48	CBASS0

**Table 12-534. DISPC Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
DISPC0	DSS_FUNC_CLK	MAIN_PLL2_HSDIV1_CLKOUT	PLL2	DISPC0 functional clock.
	DSS_DPI_0_PCLK	MAIN_PLL16_HSDIV0_CLKOUT or DSS_DPI_1_PCLK source clock	PLL16 or the selected DSS_DPI_1_PCLK source clock	DISPC0 peripheral pixel clocks for VP1. The selection of the source signal (see <a href="#">Figure 12-544, DISPC Integration</a> ) can be done via the CTRLMMR_DSS_DISPC0_CLKSEL3[2:0] DPI3_PCLK register field in device Control Module.
	DSS_DPI_0_DIV_PCLK	MAIN_PLL16_HSDIV0_CLKOUT / 2 or DSS_DPI_1_PCLK source clock / 2		
	DSS_DPI_1_PCLK	MAIN_PLL16_HSDIV0_CLKOUT	PLL16	DISPC0 peripheral pixel clock for VP2.
		MAIN_PLL19_HSDIV0_CLKOUT_EXTCLKIN	See (1)	The selection of the source signal (see <a href="#">Figure 12-544, DISPC Integration</a> ) can be done via the CTRLMMR_DSS_DISPC0_CLKSEL1[1:0] DPI1_PCLK register field in device Control Module.
		MAIN_PLL23_HSDIV0_CLKOUT_EXTCLKIN	See (1)	
		MAIN_PLL17_HSDIV0_CLKOUT	PLL17	
	DSS_DPI_2_PCLK	MAIN_PLL16_HSDIV0_CLKOUT	PLL16	DISPC0 peripheral pixel clocks for VP3.
		MAIN_PLL18_HSDIV0_CLKOUT	PLL18	The selection of the source signal (see <a href="#">Figure 12-544, DISPC Integration</a> ) can be done via the CTRLMMR_DSS_DISPC0_CLKSEL3[2:0] DPI3_PCLK and
		MAIN_PLL19_HSDIV0_CLKOUT_EXTCLKIN	See (1)	CTRLMMR_DSS_DISPC0_CLKSEL2[0] DPI2_PCLK register fields in device Control Module.
	DSS_DPI_2_DIV_PCLK	MAIN_PLL16_HSDIV0_CLKOUT / 2	PLL16	
		MAIN_PLL18_HSDIV0_CLKOUT / 2	PLL18	
	DSS_DPI_3_PCLK	MAIN_PLL19_HSDIV0_CLKOUT_EXTCLKIN / 2	See (1)	
		MAIN_PLL16_HSDIV1_CLKOUT	PLL16	DISPC0 peripheral pixel clock for VP4.
		MAIN_PLL17_HSDIV1_CLKOUT	PLL17	The selection of the source signal (see <a href="#">Figure 12-544, DISPC Integration</a> ) can be done via the CTRLMMR_DSS_DISPC0_CLKSEL3[2:0] DPI3_PCLK register field in device Control Module.
		MAIN_PLL18_HSDIV1_CLKOUT	PLL18	
		MAIN_PLL19_HSDIV0_CLKOUT_EXTCLKIN	See (1)	
		MAIN_PLL23_HSDIV0_CLKOUT_EXTCLKIN	See (1)	
Module Instance	Module Clock Output	Destination Clock Signal	Destination	Description



**Table 12-534. DISPC Clocks and Resets (continued)**

DISPC0	DSS_DPI_0_OUT_PCLK_2X	-	-	DISPC0 output pixel clocks to display peripherals. For more details on connectivity, see <i>DSI Integration</i> and <i>EDP Integration</i> .
	DSS_DPI_0_OUT_PCLK	-	-	
	DSS_DPI_1_OUT_PCLK_2X	-	-	
	DSS_DPI_1_OUT_PCLK	-	-	
	DSS_DPI_2_OUT_PCLK_2X	-	-	
	DSS_DPI_2_OUT_PCLK	-	-	
	DSS_DPI_3_OUT_PCLK_2X	-	-	
	DSS_DPI_3_OUT_PCLK	-	-	

Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
DISPC0	DSS_DPI_RST_0	MOD_G_RST	LPSC48	DISPC0 reset

- (1) For more details on the MAIN\_PLL19\_HSDIV0\_CLKOUT\_EXTCLKIN and MAIN\_PLL23\_HSDIV0\_CLKOUT\_EXTCLKIN clocks generation, see Section *MAIN Domain PLLs Overview* within *Device Configuration*.

**Table 12-535. DISPC Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DISPC0	DSS0_DSS_INST0_DISPC_FUNC_IRQ_PROC0_0	GIC500_SPI_IN_634	COMPUTE_CLUSTER0	DISPC0 functional interrupt request	Level
		C66SS0_INTRTR0_IN_10_4	C66SS0_INTRTR0		Level
		C66SS1_INTRTR0_IN_10_4	C66SS1_INTRTR0		Level
		MAIN2MCU_LVL_INTRTR0_IN_226	MAIN2MCU_LVL_INTRTR0		Level
		R5FSS0_CORE0_INTR_I_N_52	R5FSS0_CORE0		Level
		R5FSS0_CORE1_INTR_I_N_52	R5FSS0_CORE1		Level
		R5FSS1_CORE0_INTR_I_N_52	R5FSS1_CORE0		Level
		R5FSS1_CORE1_INTR_I_N_52	R5FSS1_CORE1		Level

**Table 12-535. DISPC Hardware Requests (continued)**

DSS0_DSS_INST0_DISPC_FUNC_IRQ_PROC1_0	GIC500_SPI_IN_635	COMPUTE_CLUSTER0	DISPC0 functional interrupt request	Level
	C66SS0_INTRTR0_IN_10	C66SS0_INTRTR0		Level
	C66SS1_INTRTR0_IN_10	C66SS1_INTRTR0		Level
	MAIN2MCU_LVL_INTRTR0_IN_227	MAIN2MCU_LVL_INTRTR0		Level
	R5FSS0_CORE0_INTR_I_N_53	R5FSS0_CORE0		Level
	R5FSS0_CORE1_INTR_I_N_53	R5FSS0_CORE1		Level
	R5FSS1_CORE0_INTR_I_N_53	R5FSS1_CORE0		Level
	R5FSS1_CORE1_INTR_I_N_53	R5FSS1_CORE1		Level
DSS0_DSS_INST0_DISPC_FUNC_IRQ_PROC2_0	GIC500_SPI_IN_636	COMPUTE_CLUSTER0	DISPC0 functional interrupt request	Level
DSS0_DSS_INST0_DISPC_FUNC_IRQ_PROC3_0	GIC500_SPI_IN_637	COMPUTE_CLUSTER0	DISPC0 functional interrupt request	Level
DSS0_DSS_INST0_DISPC_SAFETY_ERROR_IRQ_PROC0_0	GIC500_SPI_IN_642	COMPUTE_CLUSTER0	DISPC0 internal diagnostic error interrupt request	Level
	ESM0_LVL_IN_192	ESM0		Level
	C66SS0_INTRTR0_IN_10	C66SS0_INTRTR0		Level
	C66SS1_INTRTR0_IN_10	C66SS1_INTRTR0		Level
	MAIN2MCU_LVL_INTRTR0_IN_230	MAIN2MCU_LVL_INTRTR0		Level
	R5FSS0_CORE0_INTR_I_N_54	R5FSS0_CORE0		Level
	R5FSS0_CORE1_INTR_I_N_54	R5FSS0_CORE1		Level
	R5FSS1_CORE0_INTR_I_N_54	R5FSS1_CORE0		Level
	R5FSS1_CORE1_INTR_I_N_54	R5FSS1_CORE1		Level

**Table 12-535. DISPC Hardware Requests (continued)**

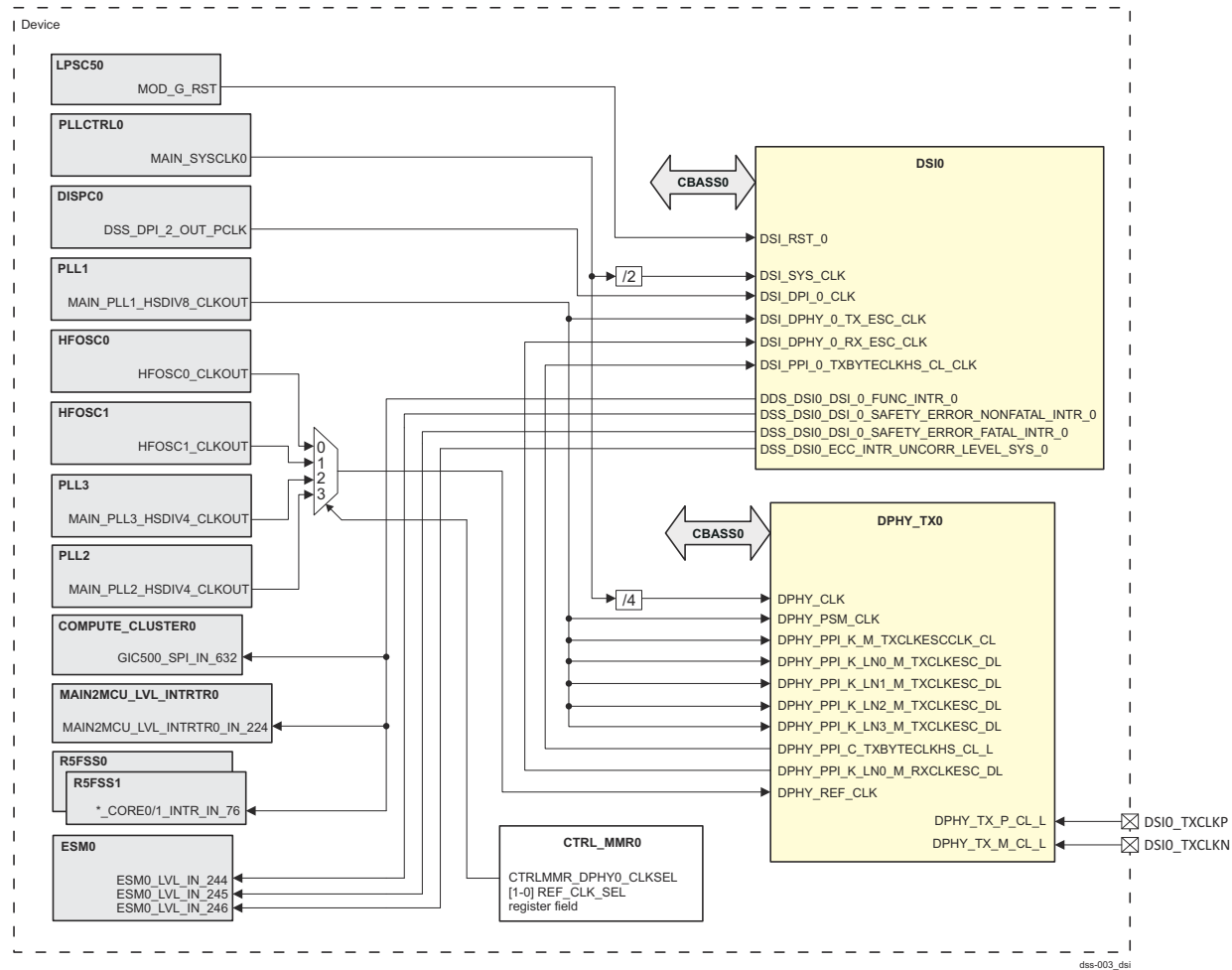
DSS0_DSS_INST0_DISPC_SAFETY_ERROR_IRQ_PROC1_0	GIC500_SPI_IN_643	COMPUTE_CLUSTER0	DISPC0 internal diagnostic error interrupt request	Level
	ESM0_LVL_IN_193	ESM0		Level
	C66SS0_INTRTR0_IN_10_9	C66SS0_INTRTR0		Level
	C66SS1_INTRTR0_IN_10_9	C66SS1_INTRTR0		Level
	MAIN2MCU_LVL_INTRTR0_IN_231	MAIN2MCU_LVL_INTRTR0		Level
	R5FSS0_CORE0_INTR_I_N_55	R5FSS0_CORE0		Level
	R5FSS0_CORE1_INTR_I_N_55	R5FSS0_CORE1		Level
	R5FSS1_CORE0_INTR_I_N_55	R5FSS1_CORE0		Level
DSS0_DSS_INST0_DISPC_SAFETY_ERROR_IRQ_PROC2_0	GIC500_SPI_IN_644	COMPUTE_CLUSTER0	DISPC0 internal diagnostic error interrupt request	Level
	ESM0_LVL_IN_194	ESM0		Level
DSS0_DSS_INST0_DISPC_SAFETY_ERROR_IRQ_PROC3_0	GIC500_SPI_IN_645	COMPUTE_CLUSTER0	DISPC0 internal diagnostic error interrupt request	Level
	ESM0_LVL_IN_195	ESM0		Level
DSS0_DSS_INST0_DISPC_SECURE_IRQ_PROCO_0	GIC500_SPI_IN_638	COMPUTE_CLUSTER0	DISPC0 secure interrupt request	Level
	C66SS0_INTRTR0_IN_10_6	C66SS0_INTRTR0		Level
	C66SS1_INTRTR0_IN_10_6	C66SS1_INTRTR0		Level
	MAIN2MCU_LVL_INTRTR0_IN_228	MAIN2MCU_LVL_INTRTR0		Level
	R5FSS0_CORE0_INTR_I_N_56	R5FSS0_CORE0		Level
	R5FSS0_CORE1_INTR_I_N_56	R5FSS0_CORE1		Level
	R5FSS1_CORE0_INTR_I_N_56	R5FSS1_CORE0		Level
	R5FSS1_CORE1_INTR_I_N_56	R5FSS1_CORE1		Level

**Table 12-535. DISPC Hardware Requests (continued)**

DSS0_DSS_INST0_DISPC_SECURE_IRQ_PRO C1_0	GIC500_SPI_IN_639	COMPUTE_CLUSTER0	DISPC0 secure interrupt request	Level
	C66SS0_INTRTR0_IN_10	C66SS0_INTRTR0		Level
	C66SS1_INTRTR0_IN_10	C66SS1_INTRTR0		Level
	MAIN2MCU_LVL_INTRTR0_IN_229	MAIN2MCU_LVL_INTRTR0		Level
	R5FSS0_CORE0_INTR_I_N_57	R5FSS0_CORE0		Level
	R5FSS0_CORE1_INTR_I_N_57	R5FSS0_CORE1		Level
	R5FSS1_CORE0_INTR_I_N_57	R5FSS1_CORE0		Level
	R5FSS1_CORE1_INTR_I_N_57	R5FSS1_CORE1		Level
DSS0_DSS_INST0_DISPC_SECURE_IRQ_PRO C2_0	GIC500_SPI_IN_640	COMPUTE_CLUSTER0	DISPC0 secure interrupt request	Level
DSS0_DSS_INST0_DISPC_SECURE_IRQ_PRO C3_0	GIC500_SPI_IN_641	COMPUTE_CLUSTER0	DISPC0 secure interrupt request	Level

### 12.6.3.2 DSI Integration

There is one DSI module integrated in the device MAIN domain. [Figure 12-545](#) shows the integration of DSI0.



**Figure 12-545. DSI Integration**

Table 12-536 through summarize the integration of DSI in the device MAIN domain.

**Table 12-536. DSI Integration Attributes**

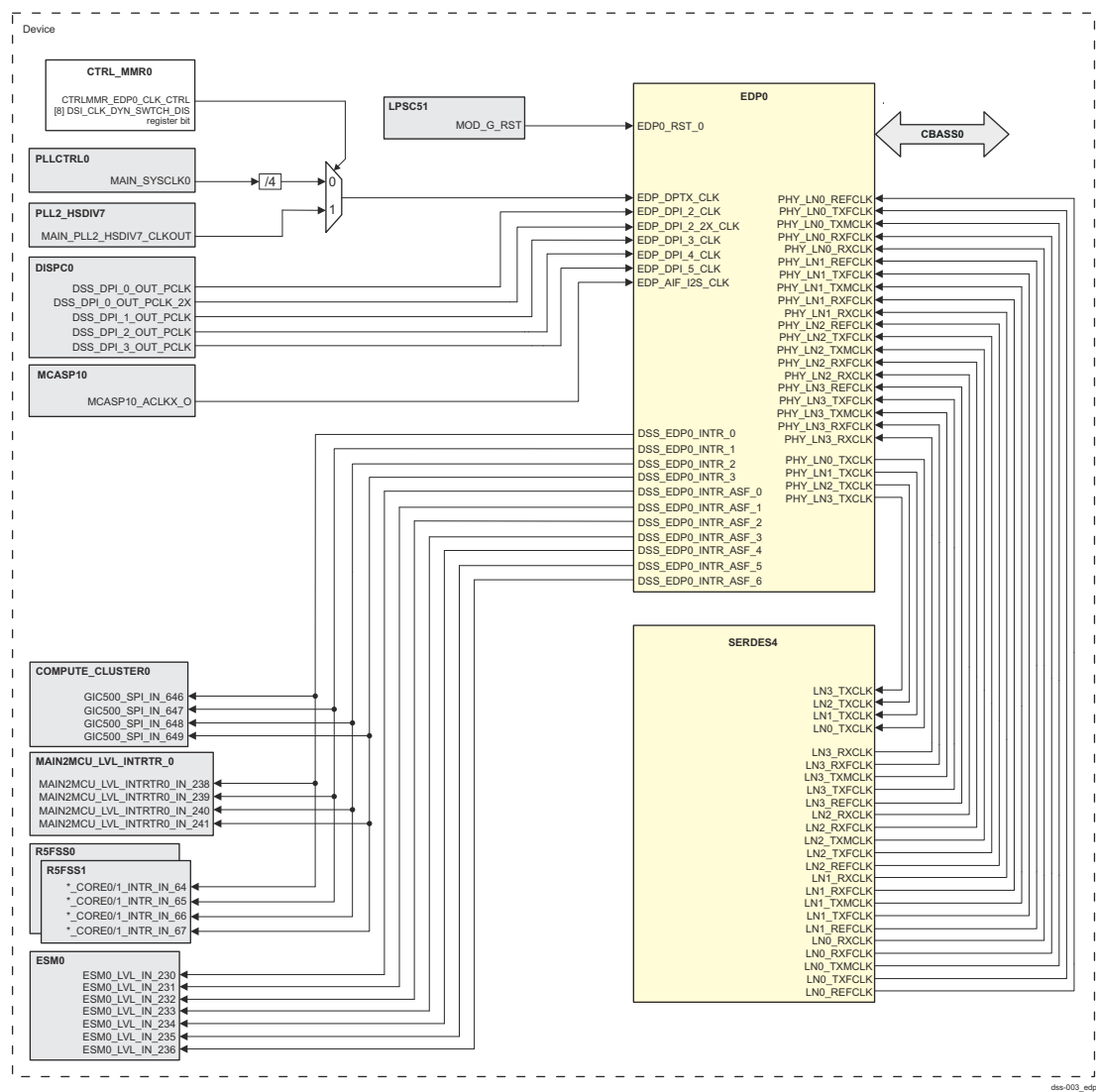
Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
DSI0	PSC0	PD2	LPSC50	CBASS0

**Table 12-537. DSI Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
DSI0	DSI_SYS_CLK	MAIN_SYSCLK0/2	PLLCTRL0	DSI0 system clock.
	DSI_DPI_0_CLK	DSS_DPI_2_OUT_PCLK	DISPC0	DSI0 DPI clock.
	DSI_DPHY_0_TX_ESC_CLK	DPHY_PPI_K_LN0_M_TXCLKESC_DL	DSI0_DPHY_TX0	DSI0 DPHY clock.
		MAIN_PLL1_HSDIV8_CLKOUT	PLL1	DSI0 clock.
	DSI_DPHY_0_RX_ESC_CLK	DPHY_PPI_K_LN0_M_RXCLKESC_DL	DSI0_DPHY_TX0	DSI0 DPHY clock.
	DSI_PPI_0_TXBYTECLKHS_CL_CLK	DPHY_PPI_C_TXBYTECLKHS_CL_L	DSI0_DPHY_TX0	
DPHY_TX0	DPHY_CLK	MAIN_SYSCLK0/4	PLLCTRL0	DPHY_TX0 clocks.
	DPHY_PSM_CLK	MAIN_PLL1_HSDIV8_CLKOUT	PLL1	
	DPHY_PPI_K_M_TXCLKESCCLK_CL	MAIN_PLL1_HSDIV8_CLKOUT	PLL1	
	DPHY_PPI_K_LN0_M_TXCLKESC_DL	MAIN_PLL1_HSDIV8_CLKOUT	PLL1	
		DSI_DPHY_0_TX_ESC_CLK	DSI0	
	DPHY_PPI_K_LN1_M_TXCLKESC_DL	MAIN_PLL1_HSDIV8_CLKOUT	PLL1	
	DPHY_PPI_K_LN2_M_TXCLKESC_DL	MAIN_PLL1_HSDIV8_CLKOUT	PLL1	
	DPHY_PPI_K_LN3_M_TXCLKESC_DL	MAIN_PLL1_HSDIV8_CLKOUT	PLL1	
	DPHY_REF_CLK	HFOSC0_CLKOUT	HFOSC0	DPHY_TX0 reference clock.
		HFOSC1_CLKOUT	HFOSC1	The selection of the source clock (see <i>DSI0 Integration</i> ) is done via the
		MAIN_PLL3_HSDIV4_CLKOUT	PLL3	CTRLMMR_DPHY0_CLKSEL[1-0] REF_CLK_SEL register field
		MAIN_PLL2_HSDIV4_CLKOUT	PLL2	the device Control Module.
	DPHY_TX_P_CL_L	DSI0_TXCLKP	I/O pin	DSI0_DPHY_TX0 input pin clock.
	DPHY_TX_M_CL_L	DSI0_TXCLKN	I/O pin	DSI0_DPHY_TX0 input pin clock.
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
DSI0	DSI_RST_0	MOD_G_RST	LPSC50	DSI0 reset.

### 12.6.3.3 EDP Integration

There is one EDP module integrated in the device MAIN domain. [Figure 12-546](#) shows the integration of EDP0.



**Figure 12-546. EDP Integration**

Table 12-538 through Table 12-540 summarize the integration of EDP0 in the device MAIN domain.

**Table 12-538. EDP Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
EDP0 (with AUXPHY0)	PSC0	PD2	LPSC51	CBASS0

**Table 12-539. EDP Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
EDP0	EDP_DPTX_CLK	MAIN_SYSCCLK0 / 4	PLLCTRL0	EDP0 main clock.
		MAIN_PLL2_HSDIV7_CLKOUT	PLL2_HSDV17	To enable a reset isolation clock selection for the EDP0, the source clock can be dynamically switched to the MAIN_PLL2_HSDIV7_CLKOUT on a warm reset event, by configuring the CTRLMMR_EDP0_CLK_CTRL[8] DSI_CLK_DYN_SWITCH_DIS register bit in the device Control Module.
	EDP_DPI_2_CLK	DSS_DPI_0_OUT_PCLK	DISPC0	EDP0 video stream pixel clocks.
	EDP_DPI_2_2X_CLK	DSS_DPI_0_OUT_PCLK_2X		
	EDP_DPI_3_CLK	DSS_DPI_1_OUT_PCLK		
	EDP_DPI_4_CLK	DSS_DPI_2_OUT_PCLK		
	EDP_DPI_5_CLK	DSS_DPI_3_OUT_PCLK		
	EDP_AIF_I2S_CLK	MCASP10_ACLKX_O	MCASP10	EDP0 audio stream clock.
	PHY_LN0_REFCLK	IP1_LN0_REFCLK	SERDES4	EDP0 lane input clocks.
	PHY_LN0_TXFCLK	IP1_LN0_TXFCLK		
	PHY_LN0_TXMCLK	IP1_LN0_TXMCLK		
	PHY_LN0_RXFCLK	IP1_LN0_RXFCLK		
	PHY_LN0_RXCLK	IP1_LN0_RXCLK		
	PHY_LN1_REFCLK	IP1_LN1_REFCLK		
	PHY_LN1_TXFCLK	IP1_LN1_TXFCLK		
	PHY_LN1_TXMCLK	IP1_LN1_TXMCLK		
	PHY_LN1_RXFCLK	IP1_LN1_RXFCLK		
	PHY_LN1_RXCLK	IP1_LN1_RXCLK		
	PHY_LN2_REFCLK	IP1_LN2_REFCLK		
	PHY_LN2_TXFCLK	IP1_LN2_TXFCLK		



**Table 12-539. EDP Clocks and Resets (continued)**

	PHY_LN2_TXMCLK	IP1_LN2_TXMCLK		
	PHY_LN2_RXFCLK	IP1_LN2_RXFCLK		
	PHY_LN2_RXCLK	IP1_LN2_RXCLK		
	PHY_LN3_REFCLK	IP1_LN3_REFCLK		
	PHY_LN3_TXFCLK	IP1_LN3_TXFCLK		
	PHY_LN3_TXMCLK	IP1_LN3_TXMCLK		
	PHY_LN3_RXFCLK	IP1_LN3_RXFCLK		
	PHY_LN3_RXCLK	IP1_LN3_RXCLK		
SERDES4	IP1_LN0_TXCLK	PHY_LN0_TXCLK	EDP0	SERDES4 TX return clocks.
	IP1_LN1_TXCLK	PHY_LN1_TXCLK		
	IP1_LN2_TXCLK	PHY_LN2_TXCLK		
	IP1_LN3_TXCLK	PHY_LN3_TXCLK		
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
EDP0	EDP_RST_0	MOD_G_RST	LPSC51	EDP0 reset.

**Table 12-540. EDP Hardware Requests**

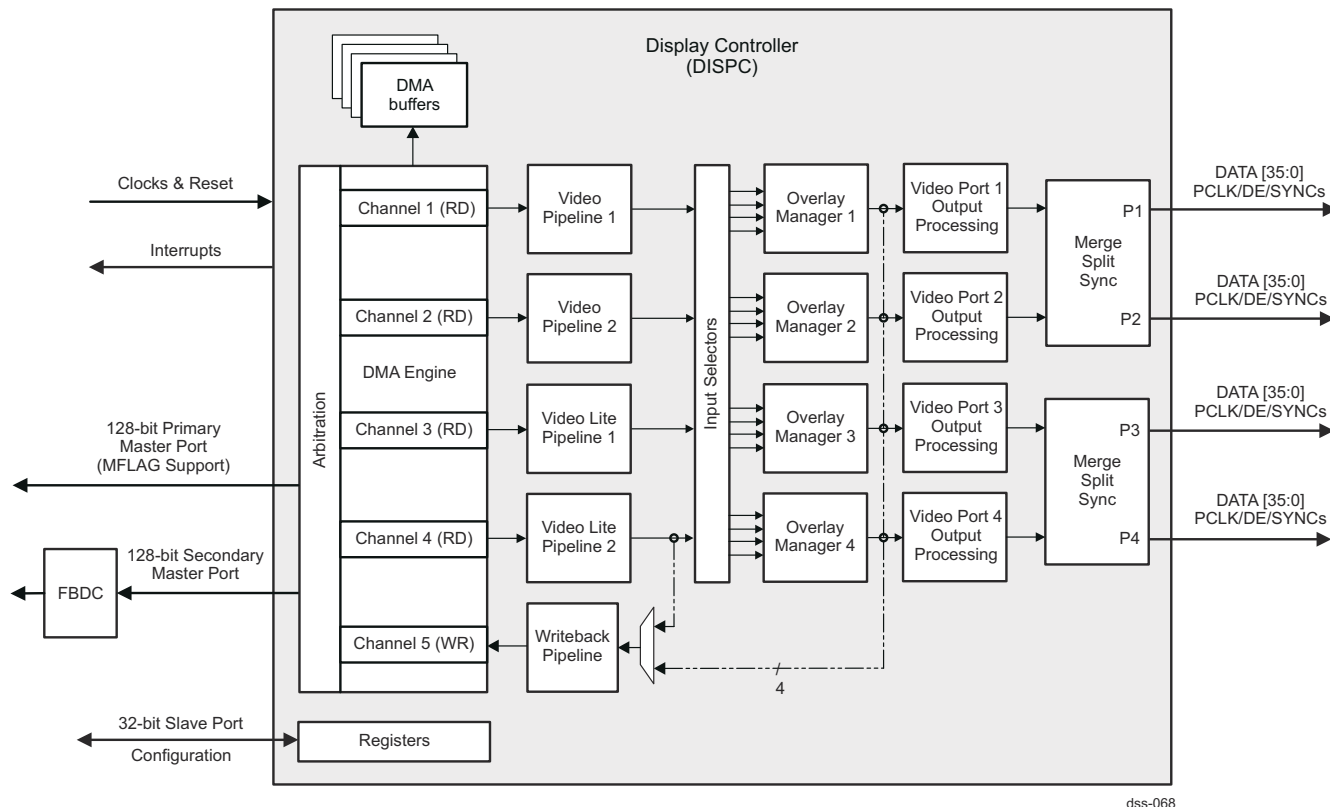
<b>Interrupt Requests</b>					
<b>Module Instance</b>	<b>Module Interrupt Signal</b>	<b>Destination Interrupt Input</b>	<b>Destination</b>	<b>Description</b>	<b>Type</b>

EDP0	DSS_EDP0_INTR_0	GIC500_SPI_IN_646	COMPUTE_CLUSTER0	EDP0 interrupt request.	Level
		R5FSS0_CORE0_INTR_IN_64	R5FSS0_CORE0		Level
		R5FSS0_CORE1_INTR_IN_64	R5FSS0_CORE1		Level
		R5FSS1_CORE0_INTR_IN_64	R5FSS1_CORE0		Level
		R5FSS1_CORE1_INTR_IN_64	R5FSS1_CORE1		Level
		MAIN2MCU_LVL_INTRTR0_IN_238			MAIN2MCU_LVL_INTRTR0
	DSS_EDP0_INTR_1	GIC500_SPI_IN_647			COMPUTE_CLUSTER0
R5FSS0_CORE0_INTR_IN_65		R5FSS0_CORE0			Level
R5FSS0_CORE1_INTR_IN_65		R5FSS0_CORE1	Level		
R5FSS1_CORE0_INTR_IN_65		R5FSS1_CORE0	Level		
R5FSS1_CORE1_INTR_IN_65		R5FSS1_CORE1	Level		
MAIN2MCU_LVL_INTRTR0_IN_239		MAIN2MCU_LVL_INTRTR0	Level		
		DSS_EDP0_INTR_2	GIC500_SPI_IN_648	COMPUTE_CLUSTER0	
			R5FSS0_CORE0_INTR_IN_66	R5FSS0_CORE0	
		R5FSS0_CORE1_INTR_IN_66	R5FSS0_CORE1	Level	
		R5FSS1_CORE0_INTR_IN_66	R5FSS1_CORE0	Level	
R5FSS1_CORE1_INTR_IN_66		R5FSS1_CORE1	Level		
MAIN2MCU_LVL_INTRTR0_IN_240	MAIN2MCU_LVL_INTRTR0		Level		
DSS_EDP0_INTR_3	GIC500_SPI_IN_649		COMPUTE_CLUSTER0	EDP0 interrupt request.	
		R5FSS0_CORE0_INTR_IN_67	R5FSS0_CORE0		

## 12.6.4 Display Subsystem Controller (DISPC) with Frame Buffer Decompression Core (FBDC)

### 12.6.4.1 DISPC Overview

Figure 12-547 is a simplified block diagram of DISPC.



**Figure 12-547. DISPC Architecture Overview**

The DISPC integrated within DSS is capable of fetching pixel data from the device system memory through its dual master ports, performing various pixel processing, and then providing the processed pixels to an external display panel. The internal DMA engine tightly coupled to the DISPC is used for the pixel data transfer from system memory (frame buffer). The DISPC DMA engine is in charge of scheduling the memory requests. Several processes are configurable in order to manage the video pipeline features (color space conversion, up-sampling, down-sampling) and overlay features. The internal timing generator logic generates the video port output signals based on VESA DMT and CEA-861 standards. The DISPC video port outputs can be connected to display panels either directly (for MIPI DPI 2.0 or BT.656/BT.1120 support), or through either MIPI DSI or DisplayPort (DP/eDP) interfaces.

#### Note

The DISPC does not support any tiled frame buffer. The DISPC has no internal capability to support rotation of the frame buffer. The support for a compressed frame buffer is provided via the Frame Buffer Decompression (FBDC) module connected to the DISPC secondary master port.

### Note

The display resolution is programmable and can be any width in the range [1:4096] pixels. The following limitations apply, related to the type of display or the processing done:

- Active Matrix screen + dithering forces a width multiple of 2 pixels
- Active Matrix + TDM may force a width multiple of 2 pixels

The display buffers in the system memory must consist of contiguous pixels.

#### 12.6.4.2 DISPC Clocks

The DISPC has one clock domain for its internal logic and separate domains for each video port output.

The DISPC functional clock (DSS\_FUNC\_CLK) serves as the internal logic clock and also acts as the interface clock for the DISPC master and slave ports to system interconnect. There is no internal divider on this clock.

The DISPC pixel clocks (DSS\_DPI\_p\_PCLK and DSS\_DPI\_p\_DIV\_PCLK, where  $p = 0$  to 3) serve as the clocks for the output display interface. The DSS\_DPI\_p\_PCLK clock is the 2x version of the DSS\_DPI\_p\_DIV\_PCLK clock. There are no internal dividers on the pixel clocks.

The relationship between the outgoing pixel clock and the input pixel clocks is as shown in [Section 12.6.4.11.9.1, MSS Clocking Scheme](#).

The frequency of the display controller logic clock (DSS\_FUNC\_CLK) has to be greater than the frequency of the DSS\_DPI\_p\_PCLK clocks, in order to get the DISPC internal logic to function properly. The frequency of the DSS\_DPI\_p\_PCLK and DSS\_DPI\_p\_DIV\_PCLK clocks depend on the required output display resolution and frame rate. For information on the maximum supported frequency ratings, see the device-specific Datasheet.

The DSS\_FUNC\_CLK is asynchronous to DSS\_DPI\_p\_PCLK and DSS\_DPI\_p\_DIV\_PCLK clocks. They can be generated by different sources.

The DSS\_DPI\_p\_PCLK and DSS\_DPI\_p\_DIV\_PCLK clocks are synchronous to one another (for the same value of  $p$ ).

The DSS0\_COMMON\_DSS\_SYSCONFIG[0] AUTOCLKGATING register bit is set by default to allow the auto-gating of the interface and functional clocks. The AUTOCLKGATING bit can be reset to disable the auto-gating of the clocks, if required.

The DISPC provides also a clock-gating control on sub-module level, via configuration of the appropriate DSS0\_COMMON\_DISPC\_CLKGATING\_DISABLE register fields.

#### 12.6.4.3 DISPC Resets

DISPC receives a single hardware reset signal. For more information, see *DSS Integration*.

To perform a software reset on the DISPC, set the DSS0\_COMMON\_DSS\_SYSCONFIG[1] SOFTRESET bit to 0x1. The DSS0\_COMMON\_DSS\_SYSSTATUS[0] DISPC\_FUNC\_RESETDONE bit indicates that the software reset is complete (for the DISPC internal logic) when its value is 0x1. When the software reset completes, the DSS0\_COMMON\_DSS\_SYSCONFIG[1] SOFTRESET bit is automatically reset. Software must ensure that the software reset completes before performing DISPC operations.

The completion of the software reset for the video ports logic is indicated in the DSS0\_COMMON\_DSS\_SYSSTATUS[3-1] DISPC\_VP\_RESETDONE register bit-field.

#### 12.6.4.4 DISPC Power Management

DISPC supports a power management protocol with the device Power Sleep Controller (PSC).

The (software) sequence, while performing a *clkstop\_req* to DISPC, is as follows:

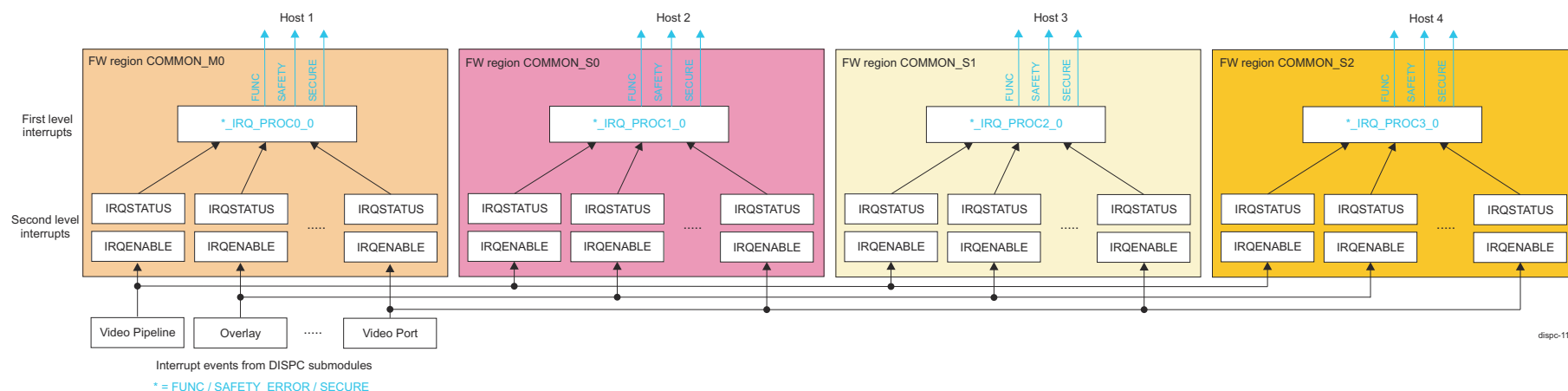
1. Disable DISPC by programming the [0] ENABLE bit of DSS0\_VP\_CONTROL register to 0.
2. DISPC hardware completes the output of the current frame. Then hardware sets the DSS0\_COMMON\_DSS\_SYSSTATUS[9] DISPC\_IDLE\_STATUS register bit to 1.

3. Poll the DSS\_IDLE\_STATUS bit to ensure that DISPC is in idle mode.
4. PSC initiates *clkstop\_req* to DISPC.
5. DISPC hardware acknowledges with *clkstop\_ack* immediately.

### 12.6.4.5 DISPC Interrupt Requests

The DISPC supports twelve interrupt output lines. The DISPC interrupt events are classified into functional, internal diagnostic error, and secure interrupt sub-categories, and then mapped to the respective sub-category interrupt line. Multiple sets of IRQ aggregation registers and IRQ generators enable fully independent monitoring and control of the interrupt events by up to 4 processor hosts at SoC level, as shown in [Figure 12-548](#). For more details on the mapping of the interrupt lines to SoC-level resources, see *DISPC Integration*.

- The 4 functional interrupt (FUNC\_IRQ) lines are DISPC\_FUNC\_IRQ\_PROC0\_0 through DISPC\_FUNC\_IRQ\_PROC3\_0.
- The 4 internal diagnostic error interrupt (SAFETY\_ERROR\_IRQ) lines are DISPC\_SAFETY\_ERROR\_IRQ\_PROC0\_0 through DISPC\_SAFETY\_ERROR\_IRQ\_PROC3\_0.
- The 4 secure interrupt (SECURE\_IRQ) lines are DISPC\_SECURE\_IRQ\_PROC0\_0 through DISPC\_SECURE\_IRQ\_PROC3\_0.



**Figure 12-548. DISPC Interrupts Generation**

Each of the interrupt signals indicates that one or more interrupt events are detected by the hardware. Each event is independently maskable for each interrupt output.

There are two level of interrupt events, as shown in [Figure 12-548](#). The first level is used to indicate common events and is also source for the second level of interrupts. The second level of interrupt events consists of status and enable interrupt registers for each video pipeline and each video port.

[Table 12-541](#) describes the first level of interrupt events with associated mask and status register fields. Each interrupt event is captured in an interrupt status register. Equivalent DSS0\_COMMON\_DISPC\_IRQSTATUS\_RAW register exist, which is updated even if interrupts are not enabled. This allows software to get access to updated status for all interrupt events.

**Table 12-541. DISPC Interrupts - First Level**

Interrupt Name	Set Interrupt Enable Register DSS0_COMMON_DISPC_IRQENABLE_SET	Clear Interrupt Enable Register DSS0_COMMON_DISPC_IRQENABLE_CLR	Interrupt Status Register DSS0_COMMON_DISPC_IRQSTATUS	Description
VID1_IRQ	[4] SET_VID_IRQ	[4] CLR_VID_IRQ	[4] VID_IRQ	At least one event of the VID1 pipeline interrupt events has occurred. See <a href="#">Table 12-542</a> for more details.
VIDL1_IRQ	[5] SET_VID_IRQ	[5] CLR_VID_IRQ	[5] VID_IRQ	At least one event of the VIDL1 pipeline interrupt events has occurred. See <a href="#">Table 12-543</a> for more details.
VID2_IRQ	[6] SET_VID_IRQ	[6] CLR_VID_IRQ	[6] VID_IRQ	At least one event of the VID2 pipeline interrupt events has occurred.
VIDL2_IRQ	[7] SET_VID_IRQ	[7] CLR_VID_IRQ	[7] VID_IRQ	At least one event of the VIDL2 pipeline interrupt events has occurred.
VP1_IRQ	[0] SET_VP_IRQ	[0] CLR_VP_IRQ	[0] VP_IRQ	At least one event of the VP1 interrupt events has occurred. See <a href="#">Table 12-545</a> for more details.
VP2_IRQ	[1] SET_VP_IRQ	[1] CLR_VP_IRQ	[1] VP_IRQ	At least one event of the VP2 interrupt events has occurred. See <a href="#">Table 12-546</a> for more details.
VP3_IRQ	[2] SET_VP_IRQ	[2] CLR_VP_IRQ	[2] VP_IRQ	At least one event of the VP3 interrupt events has occurred.
VP4_IRQ	[3] SET_VP_IRQ	[3] CLR_VP_IRQ	[3] VP_IRQ	At least one event of the VP4 interrupt events has occurred.
WB_IRQ	[14] SET_WB_IRQ	[14] CLR_WB_IRQ	[14] WB_IRQ	At least one event of the WB pipeline interrupt events has occurred.

[Table 12-542](#) describes the second level of interrupts for VID1 and VID2 pipelines with associated mask and status register bits.

**Table 12-542. DISPC Interrupts - Second Level - VID1 and VID2 Pipeline**

Interrupt Name	VID1 Interrupt Mask DSS0_COMMON_VID_IRQENABLE_0	VID1 Interrupt Status DSS0_COMMON_VID_IRQSTATUS_0	VID2 Interrupt Mask DSS0_COMMON_VID_IRQENABLE_2	VID2 Interrupt Status DSS0_COMMON_VID_IRQSTATUS_2	Description
VIDBUFFERUNDERFLOW_IRQ	[0] VIDBUFFERUNDERFLOW_EN	[0] VIDBUFFERUNDERFLOW_IRQ	[0] VIDBUFFERUNDERFLOW_EN	[0] VIDBUFFERUNDERFLOW_IRQ	Video DMA buffer underflow (FUNC_IRQ): The input video DMA buffer goes underflow. This does not necessarily mean that the buffer is empty (out of order refill), but simply that the required pixel is not in yet.

**Table 12-542. DISPC Interrupts - Second Level - VID1 and VID2 Pipeline (continued)**

Interrupt Name	VID1 Interrupt Mask DSS0_COMMON_VID_IRQEN ABLE_0	VID1 Interrupt Status DSS0_COMMON_VID_IRQST ATUS_0	VID2 Interrupt Mask DSS0_COMMON_VID_IRQEN ABLE_2	VID2 Interrupt Status DSS0_COMMON_VID_IRQST ATUS_2	Description
VIDENDWINDOW_IRQ	[1] VIDENDWINDOW_EN	[1] VIDENDWINDOW_IRQ	[1] VIDENDWINDOW_EN	[1] VIDENDWINDOW_IRQ	End of the video window (FUNC_IRQ): The DMA engine has fetched all the data from memory for the video for the current frame.
VIDSAFETYREGION_IRQ	[2] SAFETYREGION_EN	[2] SAFETYREGION_IRQ	[2] SAFETYREGION_EN	[2] SAFETYREGION_IRQ	Video output MISR signature mismatch OR Video output freeze frame detect (SAFETY_ERROR_IRQ). The MISR signature generated does not match the expected signature OR the Video output frame freeze detection has triggered.
FBDC_CORRUPTTILE_IRQ	[3] FBDC_CORRUPTTILE_EN	[3] FBDC_CORRUPTTILE_IRQ	[3] FBDC_CORRUPTTILE_EN	[3] FBDC_CORRUPTTILE_IRQ	Corrupt tile is detected (FUNC_IRQ).
FBDC_ILLEGALTILEREQ_IRQ	[4] FBDC_ILLEGALTILEREQ_EN	[4] FBDC_ILLEGALTILEREQ_IRQ	[4] FBDC_ILLEGALTILEREQ_EN	[4] FBDC_ILLEGALTILEREQ_IRQ	Illegal tile request is detected (FUNC_IRQ).

Table 12-543 describes the second level of interrupts for VIDL1 and VIDL2 pipelines with associated mask and status register bits.

**Table 12-543. DISPC Interrupts - Second Level - VIDL1 and VIDL2 Pipelines**

Interrupt Name	VIDL1 Interrupt Mask DSS0_COMMON_VID_IRQEN ABLE_1	VIDL1 Interrupt Status DSS0_COMMON_VID_IRQST ATUS_1	VIDL2 Interrupt Mask DSS0_COMMON_VID_IRQEN ABLE_3	VIDL2 Interrupt Status DSS0_COMMON_VID_IRQST ATUS_3	Description
VIDBUFFERUNDERFLOW_IRQ	[0] VIDBUFFERUNDERFLOW_EN	[0] VIDBUFFERUNDERFLOW_IRQ	[0] VIDBUFFERUNDERFLOW_EN	[0] VIDBUFFERUNDERFLOW_IRQ	Video DMA buffer underflow (FUNC_IRQ): The input video DMA buffer goes underflow. This does not necessary means that the buffer is empty (out of order refill), but simply that the required pixel is not in yet.



**Table 12-543. DISPC Interrupts - Second Level - VIDL1 and VIDL2 Pipelines (continued)**

Interrupt Name	VIDL1 Interrupt Mask DSS0_COMMON_VID_IRQEN ABLE_1	VIDL1 Interrupt Status DSS0_COMMON_VID_IRQST ATUS_1	VIDL2 Interrupt Mask DSS0_COMMON_VID_IRQEN ABLE_3	VIDL2 Interrupt Status DSS0_COMMON_VID_IRQST ATUS_3	Description
VIDENDWINDOW_IRQ	[1] VIDENDWINDOW_EN	[1] VIDENDWINDOW_IRQ	[1] VIDENDWINDOW_EN	[1] VIDENDWINDOW_IRQ	End of the video window (FUNC_IRQ): The DMA engine has fetched all the data from memory for the video for the current frame.
VIDSAFETYREGION_IRQ	[2] SAFETYREGION_EN	[2] SAFETYREGION_IRQ	[2] SAFETYREGION_EN	[2] SAFETYREGION_IRQ	Video output MISR signature mismatch OR Video output freeze frame detect (SAFETY_ERROR_IRQ). The MISR signature generated does not match the expected signature OR the Video output frame freeze detection has triggered.
FBDC_CORRUPTTILE_IRQ	[3] FBDC_CORRUPTTILE_EN	[3] FBDC_CORRUPTTILE_IRQ	[3] FBDC_CORRUPTTILE_EN	[3] FBDC_CORRUPTTILE_IRQ	Corrupt tile is detected (FUNC_IRQ).
FBDC_ILLEGALTILEREQ_IRQ	[4] FBDC_ILLEGALTILEREQ_EN	[4] FBDC_ILLEGALTILEREQ_IRQ	[4] FBDC_ILLEGALTILEREQ_EN	[4] FBDC_ILLEGALTILEREQ_IRQ	Illegal tile request is detected (FUNC_IRQ).

Table 12-544 describes the second level of interrupts for the WB pipeline with associated mask and status register bits.

**Table 12-544. DISPC Interrupts - Second Level - WB Pipeline**

Interrupt Name	WB Interrupt Mask DSS0_COMMON_WB_IRQENABLE	WB Interrupt Status DSS0_COMMON_WB_IRQSTATUS	Description
BUFFEROVERFLOW_IRQ	[0] WBBUFFEROVERFLOW_EN	[0] WBBUFFEROVERFLOW_IRQ	Write-back DMA buffer Overflow (FUNC_IRQ): The output Write-back DMA buffer goes overflow. It cannot occur when write-back channel is used in memory to memory transfer mode but only in capture mode. In capture mode the timings are defined by the timer associated with the output. In memory-to-memory mode, there is a timing constraint.

**Table 12-544. DISPC Interrupts - Second Level - WB Pipeline (continued)**

Interrupt Name	WB Interrupt Mask DSS0_COMMON_WB_IRQENABLE	WB Interrupt Status DSS0_COMMON_WB_IRQSTATUS	Description
UNCOMPLETEERROR_IRQ	[1] WBUNCOMPLETEERROR_EN	[1] WBUNCOMPLETEERROR_IRQ	Write-back un-complete error (FUNC_IRQ): The WB pipeline is reset before all data of the frame currently written back are output to the interconnect interface.
FRAMEDONE_IRQ	[2] WBFRAMEDONE_EN	[2] WBFRAMEDONE_IRQ	Write-back Frame Done (FUNC_IRQ): The WB frame done in memory-to-memory mode of operation.
SECURITYVIOLATION_IRQ	[3] SECURITYVIOLATION_EN	[3] SECURITYVIOLATION_IRQ	Security Violation for WB output (SECURE_IRQ): A security violation (for example, a secure VID pipeline connected to a non-secure WB channel) has occurred.
SYNC_IRQ	[4] WBSYNC_EN	[4] WBSYNC_IRQ	Write-back sync (FUNC_IRQ): A configuration is copied from shadow registers to work for WB for next frame.

Table 12-545 describes the second level of interrupts for VP1 and VP2 outputs with associated mask and status register bits.

**Table 12-545. DISPC Interrupts - Second Level - VP1 and VP2 Outputs**

Interrupt Name	VP1 Interrupt Mask DSS0_COMMON_VP_IRQENABLE_0	VP1 Interrupt Status DSS0_COMMON_VP_IRQSTATUS_0	VP2 Interrupt Mask DSS0_COMMON_VP_IRQENABLE_1	VP2 Interrupt Status DSS0_COMMON_VP_IRQSTATUS_1	Description
FRAMEDONE_IRQ	[0] VPFRAMEDONE_EN	[0] VPFRAMEDONE_IRQ	[0] VPFRAMEDONE_EN	[0] VPFRAMEDONE_IRQ	Frame done for VP output (FUNC_IRQ). After disabling the VP output of the DISPC, the interrupt is set when the active frame related to the VP has completed.
VSYSNCR_IRQ	[1] VPVSYSNCR_EN	[1] VPVSYSNCR_IRQ	[1] VPVSYSNCR_EN	[1] VPVSYSNCR_IRQ	VSYSNCR for VP output (FUNC_IRQ): VSYSNCR interrupt for the VP has occurred at the end of the frame.

**Table 12-545. DISPC Interrupts - Second Level - VP1 and VP2 Outputs (continued)**

Interrupt Name	VP1 Interrupt Mask DSS0_COMMON_VP_IRQEN ABLE_0	VP1 Interrupt Status DSS0_COMMON_VP_IRQST ATUS_0	VP2 Interrupt Mask DSS0_COMMON_VP_IRQEN ABLE_1	VP2 Interrupt Status DSS0_COMMON_VP_IRQST ATUS_1	Description
VSYNC_ODD_IRQ	[2] VPVSYNC_ODD_EN	[2] VPVSYNC_ODD_IRQ	[2] VPVSYNC_ODD_EN	[2] VPVSYNC_ODD_IRQ	VSYNC for odd field (FUNC_IRQ). VSYNC_ODD interrupt has occurred at the end of the frame (EVSynch received and the field polarity is odd).
PROGRAMMEDLINENUMBER_IRQ	[3] VPPROGRAMMEDLINENUMBER_EN	[3] VPPROGRAMMEDLINENUMBER_IRQ	[3] VPPROGRAMMEDLINENUMBER_EN	[3] VPPROGRAMMEDLINENUMBER_IRQ	Programmed line number (FUNC_IRQ). The VP has reached the user-programmed line number.
SYNCCLOST_IRQ	[4] VPSYNCCLOST_EN	[4] VPSYNCCLOST_IRQ	[4] VPSYNCCLOST_EN	[4] VPSYNCCLOST_IRQ	Synchronization lost on VP output (FUNC_IRQ): Occurs when VSYNC width/ front or back porches are not wide enough to load the pipeline with data (VP output).
ACBIASCOUNTSTATUS_IRQ	[5] ACBIASCOUNTSTATUS_EN	[5] ACBIASCOUNTSTATUS_IRQ	[5] ACBIASCOUNTSTATUS_EN	[5] ACBIASCOUNTSTATUS_IRQ	ACBIASCOUNTSTATUS for VP output (FUNC_IRQ): AC BIAS transition counter has decremented to zero. Refer to the DSS0_VP_POL_FREQ[11-8] ACBI and [7-0] ACB register field descriptions.
VPSAFETYREGION_IRQ	[9-6] SAFETYREGION_EN Bit [9] = Internal Diagnostic Region 3 Bit [8] = Internal Diagnostic Region 2 Bit [7] = Internal Diagnostic Region 1 Bit [6] = Internal Diagnostic Region 0	[9-6] SAFETYREGION_IRQ Bit [9] = Internal Diagnostic Region 3 Bit [8] = Internal Diagnostic Region 2 Bit [7] = Internal Diagnostic Region 1 Bit [6] = Internal Diagnostic Region 0	[9-6] SAFETYREGION_EN Bit [9] = Internal Diagnostic Region 3 Bit [8] = Internal Diagnostic Region 2 Bit [7] = Internal Diagnostic Region 1 Bit [6] = Internal Diagnostic Region 0	[9-6] SAFETYREGION_IRQ Bit [9] = Internal Diagnostic Region 3 Bit [8] = Internal Diagnostic Region 2 Bit [7] = Internal Diagnostic Region 1 Bit [6] = Internal Diagnostic Region 0	VP output MISR signature mismatch, or VP output freeze frame detect (SAFETY_ERROR_IRQ). The MISR signature generated does not match the expected signature, or the VP output frame freeze detection has triggered.
SECURITYVIOLATION_IRQ	[10] SECURITYVIOLATION_EN	[10] SECURITYVIOLATION_IRQ	[10] SECURITYVIOLATION_EN	[10] SECURITYVIOLATION_IRQ	Security violation for VP output (SECURE_IRQ). A security violation (for example, a secure video pipeline connected to a non-secure VP/OVR) has occurred.

**Table 12-545. DISPC Interrupts - Second Level - VP1 and VP2 Outputs (continued)**

Interrupt Name	VP1 Interrupt Mask DSS0_COMMON_VP_IRQEN ABLE_0	VP1 Interrupt Status DSS0_COMMON_VP_IRQST ATUS_0	VP2 Interrupt Mask DSS0_COMMON_VP_IRQEN ABLE_1	VP2 Interrupt Status DSS0_COMMON_VP_IRQST ATUS_1	Description
VPSYNC_IRQ	[11] VPSYNC_EN	[11] VPSYNC_IRQ	[11] VPSYNC_EN	[11] VPSYNC_IRQ	<p>Sync for VP output (FUNC_IRQ). Shadow to work copy of registers associated with VP1 has occurred. DSS0_VP_CONTROL[5] GOBIT register bit is cleared. This interrupt can trigger under the following cases:</p> <ol style="list-style-type: none"> <li>1. NORMAL Operation: OVRn is connected to VPn. VPSYNC indicates sync of VPn.</li> <li>2. Overlay Cascade: OVRn is connected to VPk. VPSYNC indicates sync of VPk.</li> <li>3. Overlay M2M: OVRn is connected to WB. VPSYNC indicates sync event (WB start) of WB.</li> </ol>
VPSAFETYREGION1_IRQ	[16-13] SAFETYREGION1_EN Bit [16] = Internal Diagnostic Region 7 Bit [15] = Internal Diagnostic Region 6 Bit [14] = Internal Diagnostic Region 5 Bit [13] = Internal Diagnostic Region 4	[16-13] SAFETYREGION1_IRQ Bit [16] = Internal Diagnostic Region 7 Bit [15] = Internal Diagnostic Region 6 Bit [14] = Internal Diagnostic Region 5 Bit [13] = Internal Diagnostic Region 4	[16-13] SAFETYREGION1_EN Bit [16] = Internal Diagnostic Region 7 Bit [15] = Internal Diagnostic Region 6 Bit [14] = Internal Diagnostic Region 5 Bit [13] = Internal Diagnostic Region 4	[16-13] SAFETYREGION1_IRQ Bit [16] = Internal Diagnostic Region 7 Bit [15] = Internal Diagnostic Region 6 Bit [14] = Internal Diagnostic Region 5 Bit [13] = Internal Diagnostic Region 4	<p>VP output MISR signature mismatch, or VP output freeze frame detect (SAFETY_ERROR_IRQ). The MISR signature generated does not match the expected signature, or the VP output frame freeze detection has triggered.</p>

Table 12-546 describes the second level of interrupts for VP3 and VP4 outputs with associated mask and status register bits.

**Table 12-546. DISPC Interrupts - Second Level - VP3 and VP4 Outputs**

Interrupt Name	VP3 Interrupt Mask DSS0_COMMON_VP_IRQEN ABLE_2	VP3 Interrupt Status DSS0_COMMON_VP_IRQST ATUS_2	VP4 Interrupt Mask DSS0_COMMON_VP_IRQEN ABLE_3	VP4 Interrupt Status DSS0_COMMON_VP_IRQST ATUS_3	Description
FRAMEDONE_IRQ	[0] VPFRAMEDONE_EN	[0] VPFRAMEDONE_IRQ	[0] VPFRAMEDONE_EN	[0] VPFRAMEDONE_IRQ	Frame done for VP output (FUNC_IRQ). After disabling the VP output of the DISPC, the interrupt is set when the active frame related to the VP has completed.
VSYNC_IRQ	[1] VPVSYNC_EN	[1] VPVSYNC_IRQ	[1] VPVSYNC_EN	[1] VPVSYNC_IRQ	VSYNC for VP output (FUNC_IRQ): VSYNC interrupt for the VP has occurred at the end of the frame.
VSYNC_ODD_IRQ	[2] VPVSYNC_ODD_EN	[2] VPVSYNC_ODD_IRQ	[2] VPVSYNC_ODD_EN	[2] VPVSYNC_ODD_IRQ	VSYNC for odd field (FUNC_IRQ). VSYNC_ODD interrupt has occurred at the end of the frame (EVSynch received and the field polarity is odd).
PROGRAMMEDLINENUMBER_IRQ	[3] VPPROGRAMMEDLINENUMBER_EN	[3] VPPROGRAMMEDLINENUMBER_IRQ	[3] VPPROGRAMMEDLINENUMBER_EN	[3] VPPROGRAMMEDLINENUMBER_IRQ	Programmed line number (FUNC_IRQ). The VP has reached the user-programmed line number.
SYNCLOST_IRQ	[4] VPSYNCLIST_EN	[4] VPSYNCLIST_IRQ	[4] VPSYNCLIST_EN	[4] VPSYNCLIST_IRQ	Synchronization lost on VP output (FUNC_IRQ): Occurs when VSYNC width/front or back porches are not wide enough to load the pipeline with data (VP output).
ACBIASCOUNTSTATUS_IRQ	[5] ACBIASCOUNTSTATUS_EN	[5] ACBIASCOUNTSTATUS_IRQ	[5] ACBIASCOUNTSTATUS_EN	[5] ACBIASCOUNTSTATUS_IRQ	ACBIASCOUNTSTATUS for VP output (FUNC_IRQ): AC BIAS transition counter has decremented to zero. Refer to the DSS0_VP_POL_FREQ[11-8] ACBI and [7-0] ACB register field descriptions.

**Table 12-546. DISPC Interrupts - Second Level - VP3 and VP4 Outputs (continued)**

Interrupt Name	VP3 Interrupt Mask DSS0_COMMON_VP_IRQEN ABLE_2	VP3 Interrupt Status DSS0_COMMON_VP_IRQST ATUS_2	VP4 Interrupt Mask DSS0_COMMON_VP_IRQEN ABLE_3	VP4 Interrupt Status DSS0_COMMON_VP_IRQST ATUS_3	Description
VPSAFETYREGION_IRQ	[9-6] SAFETYREGION_EN Bit [9] = Internal Diagnostic Region 3 Bit [8] = Internal Diagnostic Region 2 Bit [7] = Internal Diagnostic Region 1 Bit [6] = Internal Diagnostic Region 0	[9-6] SAFETYREGION_IRQ Bit [9] = Internal Diagnostic Region 3 Bit [8] = Internal Diagnostic Region 2 Bit [7] = Internal Diagnostic Region 1 Bit [6] = Internal Diagnostic Region 0	[9-6] SAFETYREGION_EN Bit [9] = Internal Diagnostic Region 3 Bit [8] = Internal Diagnostic Region 2 Bit [7] = Internal Diagnostic Region 1 Bit [6] = Internal Diagnostic Region 0	[9-6] SAFETYREGION_IRQ Bit [9] = Internal Diagnostic Region 3 Bit [8] = Internal Diagnostic Region 2 Bit [7] = Internal Diagnostic Region 1 Bit [6] = Internal Diagnostic Region 0	VP output MISR signature mismatch, or VP output freeze frame detect (SAFETY_ERROR_IRQ). The MISR signature generated does not match the expected signature, or the VP output frame freeze detection has triggered.
SECURITYVIOLATION_IRQ	[10] SECURITYVIOLATION_EN	[10] SECURITYVIOLATION_IRQ	[10] SECURITYVIOLATION_EN	[10] SECURITYVIOLATION_IRQ	Security violation for VP output (SECURE_IRQ). A security violation (for example, a secure video pipeline connected to a non-secure VP/OVR) has occurred.
VPSYNC_IRQ	[11] VPSYNC_EN	[11] VPSYNC_IRQ	[11] VPSYNC_EN	[11] VPSYNC_IRQ	Sync for VP output (FUNC_IRQ). Shadow to work copy of registers associated with VP1 has occurred. DSS0_VP_CONTROL[5] GOBIT register bit is cleared. This interrupt can trigger under the following cases: <ol style="list-style-type: none"> <li>1. NORMAL Operation: OVRn is connected to VPn. VPSYNC indicates sync of VPn.</li> <li>2. Overlay Cascade: OVRn is connected to VPk. VPSYNC indicates sync of VPk.</li> <li>3. Overlay M2M: OVRn is connected to WB. VPSYNC indicates sync event (WB start) of WB.</li> </ol>

**Table 12-546. DISPC Interrupts - Second Level - VP3 and VP4 Outputs (continued)**

Interrupt Name	VP3 Interrupt Mask DSS0_COMMON_VP_IRQEN ABLE_2	VP3 Interrupt Status DSS0_COMMON_VP_IRQST ATUS_2	VP4 Interrupt Mask DSS0_COMMON_VP_IRQEN ABLE_3	VP4 Interrupt Status DSS0_COMMON_VP_IRQST ATUS_3	Description
VPSAFETYREGION1_IRQ	[16-13] SAFETYREGION1_EN Bit [16] = Internal Diagnostic Region 7 Bit [15] = Internal Diagnostic Region 6 Bit [14] = Internal Diagnostic Region 5 Bit [13] = Internal Diagnostic Region 4	[16-13] SAFETYREGION1_IRQ Bit [16] = Internal Diagnostic Region 7 Bit [15] = Internal Diagnostic Region 6 Bit [14] = Internal Diagnostic Region 5 Bit [13] = Internal Diagnostic Region 4	[16-13] SAFETYREGION1_EN Bit [16] = Internal Diagnostic Region 7 Bit [15] = Internal Diagnostic Region 6 Bit [14] = Internal Diagnostic Region 5 Bit [13] = Internal Diagnostic Region 4	[16-13] SAFETYREGION1_IRQ Bit [16] = Internal Diagnostic Region 7 Bit [15] = Internal Diagnostic Region 6 Bit [14] = Internal Diagnostic Region 5 Bit [13] = Internal Diagnostic Region 4	VP output MISR signature mismatch, or VP output freeze frame detect (SAFETY_ERROR_IRQ). The MISR signature generated does not match the expected signature, or the VP output frame freeze detection has triggered.

#### 12.6.4.6 DISPC DMA Controller

The DISPC DMA controller is based on a 5-channel DMA engine that:

- Requests and supplies data from system memory to the VID and VIDL pipelines (up to 4 input layers) through the SoC system interconnect, based on the configuration of the pipeline settings.
- Stores the composed frames back into system memory by using the WB pipeline.
- Is fully programmable and fetches pixel data via 128-bit requests using 1D burst.

The DISPC DMA engine has an additional (secondary) 128-bit master port. By default, all transactions from every channel go through the primary 128-bit master port. If compression is enabled for a particular channel (by using the FBDC module), then all the transactions of that channel go through the secondary master port. It is also possible to force transactions from a particular channel to the secondary master port (even without compression) by setting the DSS0\_VID\_ATTRIBUTES2[25] MPORTSEL register bit to 0x1. This is possible only when compression through the FBDC is completely disabled (that is, none of the read channels are fetching compressed data).

Each pipeline has a dedicated DMA buffer and channel with independent settings. If a pipeline is disabled (that is, not used), then its DMA buffer can be assigned to another pipeline by configuring the DSS0\_COMMON\_DISPC\_GLOBAL\_BUFFER register. For example, unused buffers for a VIDL pipeline can be used by a VID pipeline.

Each DMA channel supports a total of 8 line buffers, each of which can store 2048 32-bit pixels.

- The size of each DMA buffer associated to the VID, VIDL and WB pipelines is 8x512x128-bit.

The DMA engine fetches encoded pixels from the system memory only when the video layer is enabled (a valid configuration has been programmed for the video layer), that is, when the video window is present and the video pipeline is active.

##### 12.6.4.6.1 DISPC DMA Addressing and Bursts

For each line to be fetched, the DMA engine address generator:

- Calculates the pixel address
- Aligns the address
- Determines byte enable pattern
- Determines 1D burst length

The meaning of the address bits is as follows:

- Bits [31-0]: system (DDR) memory address (pixel address)
- Bits [47-32]: 32 bits + address extension

The address extension bits are defined by the programmable parameter DSS0\_VID\_BA\_EXT\_0/DSS0\_VID\_BA\_EXT\_1 and DSS0\_VID\_BA\_UV\_EXT\_0/DSS0\_VID\_BA\_UV\_EXT\_1 registers, optionally used to extend the BA memory addressing to 48-bit addressed external memory space (that is, to extend DISPC address space into 4GB+ space).

The DDR scan pixel addresses are generated by the DMA engine in order to read data from the system memory. The base address defines the start address of the first pixel, and then the address is incremented based on the number of pixels per line, offset between two consecutive lines and number of lines. The DSS0\_VID\_ROW\_INC register allow the access to a frame using 1D bursts (but as a two-dimensional block) by adding a fixed address offset at the end of a line. The ROW\_INC can also be used to skip lines from the input frame.

The byte address of each pixel in the frame buffer located in the system memory is determined by:

Pixel address = base\_address + x × (bpp/8) + y × (width × (bpp/8) + increment), where:

- "base\_address" corresponds to the base address defined by:
  - DSS0\_VID\_BA\_0/DSS0\_VID\_BA\_1[31-0] BA register bit-fields for all formats, and Y frame buffer in case of YUV-NV12 format. Optional extension bits in DSS0\_VID\_BA\_EXT\_0/DSS0\_VID\_BA\_EXT\_1 registers.



- DSS0\_VID\_BA\_UV\_0/DSS0\_VID\_BA\_UV\_1[31-0] BA register bit-fields for UV frame buffer in case of YUV-NV12 format. Optional extension bits in DSS0\_VID\_BA\_UV\_EXT\_0/DSS0\_VID\_BA\_UV\_EXT\_1 registers.
- "bpp" corresponds to the number of bits per pixel defined by the DSS0\_VID\_ATTRIBUTES[6-1] FORMAT register bit-field.
- "width" corresponds to the number of pixels per line defined by the DSS0\_VID\_PICTURE\_SIZE[11-0] MEMSIZEX + 1 register bit-field.
- "increment" corresponds to the number of bytes to skip between two contiguous lines defined by the DSS0\_VID\_ROW\_INC [31-0] ROWINC – 1 register bit-field.
- "x" corresponds to the pixel position on the x-axis.
- "y" corresponds to the pixel position on the y-axis.

In cases where the DMA controller is doing a flip/mirror (see [Section 12.6.4.6.4, DISPC Flip/Mirror Support](#)) or fetching a compressed frame buffer (see [DISPC Compressed Data Format Support](#)), the above equation for pixel byte address needs to be modified according to those features.

### Note

Since the base address is aligned on pixel size boundary the horizontal resolution is one pixel. In case of YUV422 formats, the resolution is 4 bytes (2 pixels). In case of RGB24 packed format the resolution is 4 pixels. In case of Y frame buffer (YUV-NV12 format) the resolution is one byte. The vertical resolution is one line.

In case of YUV422 non-planar format, the number of pixels per line shall be a multiple of 2 pixels and the size of a pixel shall be considered as 2 bytes. In case of YUV planar format (YUV420-NV12, YUV420-NV21, YUV422-NV12, YUV422-NV21), the Y buffer shall be considered as an 8-bit frame buffer, and the CbCr shall be considered as a 16-bit frame buffer. The pixel size is 1 byte and 2 bytes respectively for Y and CbCr.

For two-plane pixel formats (YUV420-NV12, YUV420-NV21, YUV422-NV12, YUV422-NV21, RGB565-A8), the pixel values are defined in two separate buffers (Y and UV buffers). The first buffer consists of Y values (8 bits for each Y sample) or 16-bit RGB value. The second buffer consists of CbCr values (16 bits for each pair of CbCr samples) or 8-bit Alpha value. The base address, the number of bytes to skip between pixels and between lines of each buffer are defined by separate registers:

- The DSS0\_VID\_BA\_0/DSS0\_VID\_BA\_1, DSS0\_VID\_PIXEL\_INC and DSS0\_VID\_ROW\_INC registers define the values to use for the first buffer
- The DSS0\_VID\_BA\_UV\_0/DSS0\_VID\_BA\_UV\_1 and DSS0\_VID\_ROW\_INC\_UV define the values to use for the second buffer
- Internal to the DMA controller, the two planes use the same shared memory space, which is allocated to the relevant pipeline which has the two-plane pixel format set

In case of interlaced mode, DSS0\_VID\_BA\_0 and DSS0\_VID\_BA\_UV\_0 registers define the base address of the even field, and DSS0\_VID\_BA\_1 and DSS0\_VID\_BA\_UV\_1 registers define the base address of the odd field.

[Table 12-547](#) summarizes the register settings for a simple access of a picture in the system memory.

**Table 12-547. DISPC Register Settings for Accessing Image in Internal Memory**

Pipeline Registers	Value
DSS0_VID_BA_0 <sup>(1)</sup> and DSS0_VID_BA_1 <sup>(2)</sup> / DSS0_WB_BA_0 and DSS0_WB_BA_1	The physical base address (PBA) of image in the memory for all formats and Y buffer.
DSS0_VID_BA_EXT_0 and DSS0_VID_BA_EXT_1 / DSS0_WB_BA_EXT_0 and DSS0_WB_BA_EXT_1	Address extension bits of PBA for all formats and Y buffer.
DSS0_VID_BA_UV_0 <sup>(1)</sup> and DSS0_VID_BA_UV_1 <sup>(2)</sup> / DSS0_WB_BA_UV_0 and DSS0_WB_BA_UV_1	The physical base address (PBA) of UV buffers image in the memory.

**Table 12-547. DISPC Register Settings for Accessing Image in Internal Memory (continued)**

Pipeline Registers	Value
DSS0_VID_BA_UV_EXT_0 and DSS0_VID_BA_UV_EXT_1 / DSS0_WB_BA_UV_EXT_0 and DSS0_WB_BA_UV_EXT_1	Address extension bits of PBA for UV buffers.
DSS0_VID_PIXEL_INC / DSS0_WB_PIXEL_INC	1 or other in pixel incremental value.
DSS0_VID_ROW_INC / DSS0_WB_ROW_INC	1 or other in row incremental value. Used for Y buffer.
DSS0_VID_ROW_INC_UV / DSS0_WB_ROW_INC_UV	1 or other in row incremental value. Used for UV buffer.

(1) The BA\_0 and BA\_UV\_0 registers define the base address of even field, in case of interlaced mode.

(2) The BA\_1 and BA\_UV\_1 registers define the base address of odd field, in case of interlaced mode.

An interconnect request (128 bits) corresponds to one or several pixels, depending on the bits per pixel. Therefore, the DMA engine determines the appropriate burst sequence to optimize the fetching of each new line. The DMA engine must prevent a single burst from crossing two lines. The DMA engine supports only 1D burst. 1D burst is used, if the fetch data is linear in memory. The size of the burst can be one of the following values:

- 1x128-bit, 2x128-bit, 4x128-bit: Only during start of the row and end of the row, if there is a misaligned address
- 8x128-bit: Steady state burst size
- 16x128-bit: Only while accessing a compressed frame buffer

The DMA controller supports the start address of a row of pixels to be aligned to any arbitrary boundary in memory (with the restriction that a 32-bit format is aligned to 32-bit boundary, a 16-bit format is aligned to a 16-bit boundary, a 8-bit format is aligned to a 8-bit boundary, etc.). However, it operates most efficiently when the start address of a row is aligned to the max burst boundary of 8x128-bit. For all other cases of unaligned rows the DMA will need to go through non-optimal transactions at the start and end of each row, till it can issue a max burst of 8x128.

While accessing a compressed frame buffer (through the FBDC module), the base-address needs to be aligned to 256 bytes. In this case, the DMA will always issue a burst of 16x128-bit.

#### 12.6.4.6.2 DISPC Read DMA Buffers

The read DMA buffers are used by the VID and VIDL pipelines.

When the vertical front porch (VFP) period starts after the last horizontal front porch (HFP) of the last line, the DMA buffers are flushed according to the video port output associated with the particular video pipeline. The DMA engine restarts fetching new frame data from the memory through the DISPC master port. Enabling or disabling the DISPC also flushes the DMA buffers.

Programmable high and low thresholds, independent for each DMA buffer, are used by the DMA engine to start and stop requesting data through the master port.

- When low threshold (set in the DSS0\_VID\_BUF\_THRESHOLD [15:0] BUFLOWTHRESHOLD register bit-field) is reached, the DMA engine starts a request on the device interconnect to fill the DMA buffer.
- When high threshold (set in the DSS0\_VID\_BUF\_THRESHOLD [31:16] BUFHIGHTHRESHOLD register bit-field) is reached, the DMA engine stops requesting encoded pixels.

### Note

The configuration of thresholds for optimal performance can be defined using the DSS0\_VID\_BUF\_SIZE\_STATUS[15-0] BUFSIZE register field value, as follows:

- For high threshold: BUFSIZE (in number of 128-bit words) – 1
- For low threshold: BUFSIZE (in number of 128-bit words) – burst size (in number of 128-bit words)

The following limitations for BUFLOWTHRESHOLD values must be also considered:

- If the scaler in VID pipeline is disabled, BUFLOWTHRESHOLD can be programmed as low as interconnect latency and pixel output rate allow it.
- If the scaler in VID pipeline is enabled, BUFLOWTHRESHOLD must be programmed to guarantee that at least four full lines can be stored.

To avoid underflow at the beginning of a frame and have sufficient encoded pixel data to start some processing, a preloading of the DMA buffer is configurable between a fixed value of bytes and the high threshold value. When the preload value is reached, the associated channel will start pulling pixels out of the DMA buffer. To enable the preload based on the value entered in the DSS0\_VID\_PRELOAD[11-0] PRELOAD register bit-field, the DSS0\_VID\_ATTRIBUTES[19] BUFPRELOAD register bit must be set to 0x0.

The vertical blanking between two frames must be long enough to allow fetching the number of pixels defined by the DSS0\_VID\_PRELOAD register and preloading the whole video pipeline. If the value set in the preload register is greater than some overflow conditions detected by the hardware, then data will start to be read from the video DMA buffer before the preload value is reached. If SYNCLOST\_IRQ event occurs the video buffer needs to be increased (buffer merge). Preload value must be greater or equal to low threshold, and smaller or equal to high threshold value.

### Note

When self-refresh mode is selected (which means that the data in the DMA buffer are used for multiple frames) the DMA buffers are not flushed at the end of each frame. Each DMA buffer has an independent control for selecting the self-refresh mode. For more information, see [Section 12.6.4.6.10, DISPC DMA Ultra-Low Power Mode](#).

#### 12.6.4.6.3 DISPC Write DMA Buffer

The write DMA buffer is used by the WB pipeline.

Two modes for filling the DMA buffer are supported by the WB channel, selectable through the DSS0\_WB\_ATTRIBUTES[19] WRITEBACKMODE bit:

- Capture mode, WRITEBACKMODE bit set to 0: One of the overlay outputs connected to an external interface is captured at the same time the data are sent on the output. The WB timings are controlled by the VP timings.
- Memory-to-memory mode, WRITEBACKMODE bit set to 1: One of the overlay outputs or one of the pipelines is captured to perform a memory-to-memory transfer, with some processing by the DISPC (rescaling, overlaying, color space conversion, etc.).

**In capture mode:** Transfer is synchronous (that is, tightly coupled) to one of the display outputs. In this mode, the write back buffer starts receiving data after DSS0\_WB\_ATTRIBUTES[0] ENABLE register bit has been set and the associated output vertical sync has triggered the start of the transfer (VFP). In addition, the WB ENABLE bit shall be set prior to the enable of the captured channel output in order to capture the first frame.

**In memory-to-memory mode:** The data transfer is not synchronous to any of the display outputs. In this mode, the write back buffer starts receiving data after the DSS0\_WB\_ATTRIBUTES[0] ENABLE has been set. The WB ENABLE bit also is triggering the update of the shadow register in the WB pipeline.

Programmable high and low thresholds are used by the DMA engine to start and stop sending data to the system interconnect.

- When high threshold (set in the DSS0\_WB\_BUF\_THRESHOLD[31:16] BUFHIGHTHRESHOLD bit field) is reached, and there are enough data for at least one burst of pixels in the DMA buffer ready for transfer, the DMA engine generates a request to the arbitration logic. The size of the burst is defined by the user.
- When low threshold (set in the DSS0\_WB\_BUF\_THRESHOLD[15:0] BUFLOWTHRESHOLD bit field) is reached, the DMA engine stops sending encoded pixels.

At the end of the frame, to completely drain the DMA buffer, some smaller bursts (even single requests) may have to be issued. To limit the number of interconnect requests from the DISPC (that is, to limit the throughput of the write-back channel to the memory), a number of IDLE cycles between requests can be inserted. IDLE cycles can be inserted only when WB is used in memory-to-memory mode. It is ignored when WB is in capture mode.

The number of IDLE cycles between requests can be activated and determined by:

- Setting the DSS0\_WB\_ATTRIBUTES[27] IDLESIZE bit to 0x0 (default value) and entering the number of idles between requests in the DSS0\_WB\_ATTRIBUTES[31:28] IDLENUMBER bit field. Idle numbers vary from 0 to 15.
- Setting the DSS0\_WB\_ATTRIBUTES[27] IDLESIZE bit to 0x1, which considers the size of the burst (the DSS0\_WB\_ATTRIBUTES[15:14] BURSTSIZE bit field) to determine the number of IDLE cycles.
  - If BURSTSIZE = 0x0, then the number of IDLE cycles equals IDLENUMBER (0 to 15).
  - If BURSTSIZE = 0x1, then the number of IDLE cycles equals IDLENUMBER × 4 (0 to 60).
  - If BURSTSIZE = 0x2, then the number of IDLE cycles equals IDLENUMBER × 8 (0 to 120).

#### 12.6.4.6.4 DISPC Flip/Mirror Support

The DISPC supports on-the-fly source image flip along the x/y-axis for 8-bit/component formats (ARGB or YUV) to create a mirror effect on the source data. The DMA engine reads the source frame from right to left by requesting burst transfers for each line in negative address increments while performing each burst transfer as a linear incremental burst. The read data is repacked and stored in the line buffer incrementally - effectively storing the frame as a flipped image for the processing pipeline. The configuration of the flip/mirror operation is explained in [Table 12-548](#).

**Table 12-548. DISPC Flip/Mirror Configuration**

Flip Direction	FLIP Bit	Base Address	Byte Increment
Along Y-axis (vertical mirror)	1	Start of window	1
Along X-axis (horizontal mirror)	0	Start of last line of the window	-(2*(width of the line in bytes))
Along both X-axis and Y-axis (horizontal and vertical mirror)	1	Start of last line of the window	-(2*(width of the line in bytes))

In [Table 12-548](#):

- The FLIP bit is located in DSS0\_VID\_ATTRIBUTES register.
- The base address of the video buffer must be configured as explained in [Section 12.6.4.6.1, DISPC DMA Addressing and Bursts](#).
- The the number of bytes to increment at the end of the row in the video buffer can be configured through DSS0\_VID\_ROW\_INC and DSS0\_VID\_ROW\_INC\_UV registers. For more information, see [Section 12.6.4.6.1, DISPC DMA Addressing and Bursts](#), and [Section 12.6.4.6.5, DISPC DMA Predecimation](#).

The flip/mirror feature is not supported for compressed data formats (when the DMA engine receives data from the FBDC module).

#### 12.6.4.6.5 DISPC DMA Predecimation

The predecimation process consists of downscaling an image by fetching only the necessary pixels out of the memory. Vertical and horizontal predecimation are possible:

- Vertical predecimation: The picture stored in memory can be predecimated vertically by skipping lines. Burst mode is used to fetch the data when skipping lines. Only the lines that will be used by the DISPC are fetched from memory; the other lines are skipped. The DMA engine sends requests only for the useful lines using 1D

burst. The base address indicates the first valid pixel to fetch from memory. The number of lines to skip is set in the ROW\_INC and ROW\_INC\_UV registers (see [Section 12.6.4.6.1, DISPC DMA Addressing and Bursts](#)).

- Horizontal predecimation: When fetching data from memory, it is possible to skip 1 out of 2 pixels, up to 1 of out 2047 pixels, by setting the PIXEL\_INC register (see [Section 12.6.4.6.1, DISPC DMA Addressing and Bursts](#)) to the number of pixels to skip (n), multiplied by the size of a pixel (in bytes), +1. The condition to generate a burst is that there is at least one useful pixel per 128-bit OCP request. Therefore, when the pixels are 16/32-bit, the maximum number of pixels that can be skipped is 8/4. If  $\text{PixelWidthInBytes} + \text{BytesToSkip}$  is greater than 16, then the programmed burst is changed into single request.

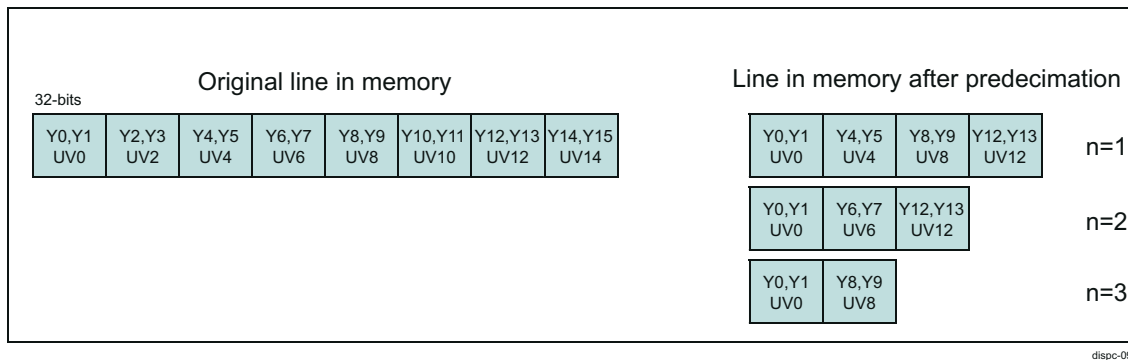
No decimation is supported when the input format is 1, 2, 4, or 8-bit BITMAP.

For RGB and YUV420 data formats, each pixel data container in memory holds 1 pixel. Thus, when configuring the PIXEL\_INC register, the value of n equals the number of pixels to skip:

- For RGB format, one pixel data container = 32 bits = 1 pixel
- For YUV format:
  - One Y pixel data container = 8 bits = 1 pixel
  - One UV pixel data container = 16 bits = 1 pixel

For YUV422 format, each 32-bit pixel data container holds the Luma components for 2 pixels, and the Chrominance component of 1 pixel (see [Figure 12-549](#)). Therefore, for the valid values of the PIXELINC bit field in the case of the following YUV422 format, caution must be taken because n equals the number of pixel data containers to skip, and not the number of pixels:

- For n = 1, PIXELINC = 5
- For n = 2, PIXELINC = 9
- For n = 3, PIXELINC = 13
- For n = 4, PIXELINC = 17, etc.

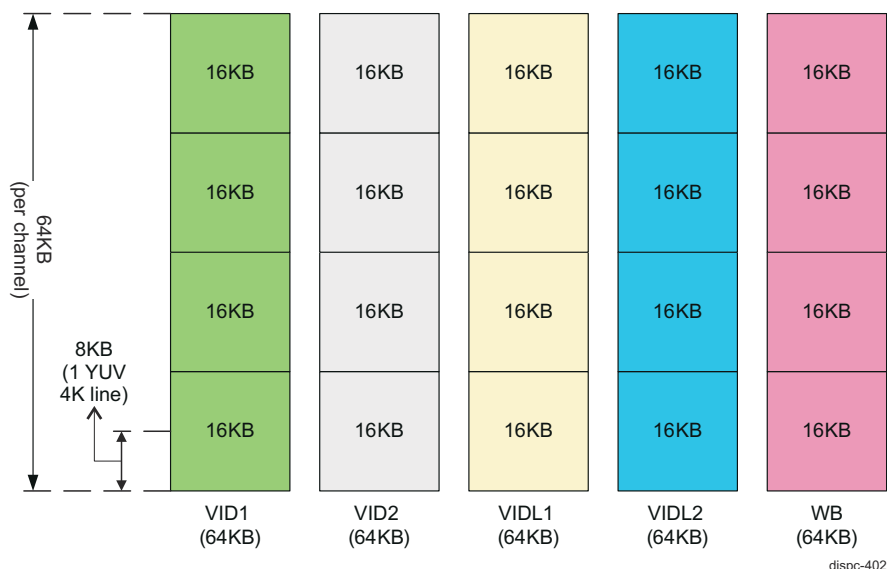


**Figure 12-549. DISPC YUV422 Predecimation**

#### 12.6.4.6.6 DISPC DMA Buffer Sharing

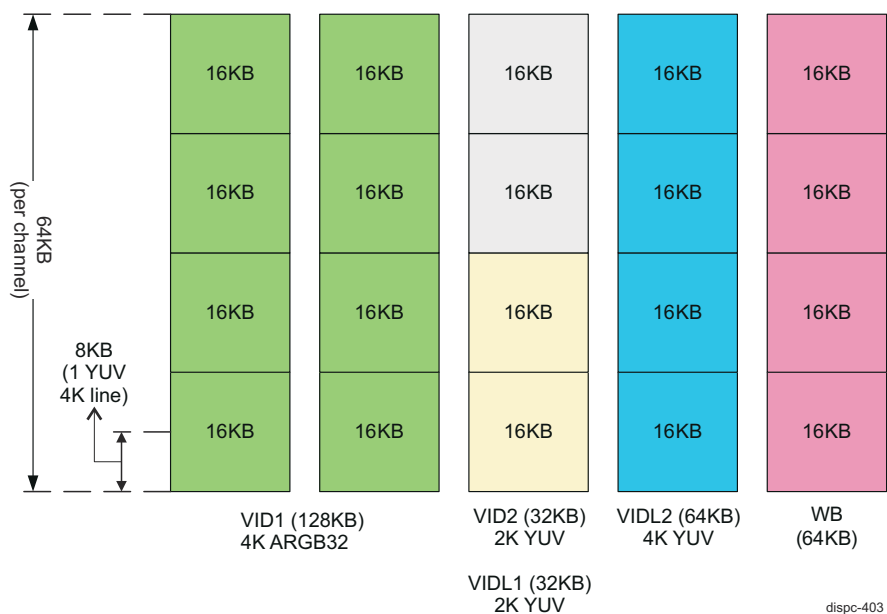
The DISPC DMA controller supports dynamic re-partition of the memory among the different channels.

At reset, each channel in DMA is allocated a uniform buffer size of 64KB. This 64KB is further partitioned into four 16KB blocks as shown in [Figure 12-550](#). A 64KB buffer is enough to support ping-pong mode for resolutions up to 4K for YUV420/YUV422 formats and up to 2K for ARGB32 formats.



**Figure 12-550. DISPC DMA Channel Memory Allocation at Reset**

To support resolutions larger than 4K YUV or 2K ARGB (with ping-pong configuration), the default buffer size of 64KB per read channel is not sufficient. In such cases, the DISPC allows dynamic repartition of the allocated buffers across the different read channels. In this way each read channel can have a variable buffer size allocated to it, starting from a minimum of 16KB to a maximum of 256KB, in 16KB increments. The WB channel buffers cannot be re-allocated and remain fixed. This is the case even when WB channel is disabled. A particular configuration (for a specific example use-case) is as shown in [Figure 12-551](#).



**Figure 12-551. DISPC Dynamic Buffer Repartition between DMA Read Channels**

When the size of the buffer is changed, the thresholds shall be re-programmed by the user to reflect the new DMA buffer configuration.

The activity on a read channel is controlled by the timing of the Overlay/VP to which the channel is connected (or to the WB, if the channel is connected to WB in M2M mode). Due to this, the DISPC HW ensures that any



change in a channel buffer setting is synchronized with regard to the Overlay/VP or WB (M2M mode) to which the channel is connected. In addition, the software also must follow the following programming sequences:

- Initialization sequence (before enabling any pipe or VP or WB):
  1. Allocate the total available memory among the different threads (configurable through the DSS0\_COMMON\_GLOBAL\_DMA\_THREADSIZE[19-0] VP#nTHREADSIZESIZE and [24-20] WBTHREADSIZESIZE register bit-fields).
  2. Wait for the DISPC HW to sync to the programmed THREADSIZESIZE (read the value from DSS0\_COMMON\_GLOBAL\_DMA\_THREADSIZESIZESTATUS[19-0] VP#nTHREADSIZESIZE and [24-20] WBTHREADSIZESIZE register bit-fields).
  3. Allocate a buffer for each of the (active) read channels (configurable through the DSS0\_VID\_DMA\_BUFSIZE[4-0] BUFSIZE register field). Software needs to ensure that the total size allocated to all the channels in a thread does not surpass the total size allocated to the thread.
  4. Enable the read channels as well as the VP output.
- Intra thread re-allocation sequence. Within the thread the re-allocation always happens at the next VSYNC. There is no handshake needed on the part of software.
  1. Allocate the new buffer size (configurable through the DSS0\_VID\_DMA\_BUFSIZE[4-0] BUFSIZE register field).
- Inter thread re-allocation sequence. The synchronization events (VSYNC) between two threads will be different. Therefore, any re-allocation between two threads needs to go through a software managed handshake sequence as follows:
  1. Free the required amount of memory from Thread-#n (configurable through the DSS0\_COMMON\_GLOBAL\_DMA\_THREADSIZESIZE register).
  2. Wait till the synchronization event of Thread-#n (achieved by looking at the status reflected in the DSS0\_COMMON\_GLOBAL\_DMA\_THREADSIZESIZESTATUS register).
  3. Allocate the freed buffer space to a different thread, Thread-#m (configurable through the DSS0\_COMMON\_GLOBAL\_DMA\_THREADSIZESIZE register).
  4. Further allocate this space to the different read channels of Thread-#m (configurable through the DSS0\_COMMON\_GLOBAL\_DMA\_THREADSIZESIZESTATUS register).
  5. Further allocate the new buffer size to the read channels in Thread-#m (configurable through DSS0\_VID\_DMA\_BUFSIZE register).

#### Note

In a case where a channel is connected to multiple Overlay/VP (multi-cast use-case), then the channel belongs to the thread corresponding to the first (lowest numbered) Overlay/VP to which it is connected.

#### 12.6.4.6.7 DISPC DMA MFLAG Mechanism

The MFLAG mechanism allows a dynamic increase of the priority of DISPC real-time traffic, when required, based on the fullness of the DISPC DMA read buffers.

The MFLAG mechanism is used when fullness of the DMA buffers is critical (close to underflow). The mechanism is implemented for all DMA buffers of the video pipelines.

Programmable buffer thresholds (forming hysteresis) are used to configure when a local MFLAG signal is generated. The 1-bit MFLAG signal is generated on DISPC master port in order to inform the system that the outstanding requests from DISPC shall be considered with higher priority in order to get faster interconnect responses. The MFLAG signal is asynchronous to any ongoing interconnect transaction.

Each pipeline maintains its own MFLAG bit. The MFLAG bit is asserted depending on the fullness of the DMA buffers associated with the pipeline and depending on the thresholds programmed by software. All MFLAG signals are OR-ed together to generate the single 1-bit MFLAG for the master port connected to the DISPC DMA engine.

The threshold for video pipelines corresponds to the fullness of the associated DMA buffer, and is defined by two threshold parameters:

- HT\_MFLAG: High threshold.
  - For read access from video pipelines, when the pipeline buffer reaches the programmed value, the associated local MFLAG signal goes low (deasserted).
  - This threshold can be programmed in the DSS0\_VID\_MFLAG\_THRESHOLD [31-16] HT\_MFLAG register field.
- LT\_MFLAG: Low threshold.
  - For read access from video pipelines, when the pipeline buffer reaches the programmed value, the associated local MFLAG signal goes high (asserted).
  - This threshold can be programmed in the DSS0\_VID\_MFLAG\_THRESHOLD [15-0] LT\_MFLAG register field.

Summary of the MFLAG value, based on DMA read buffer fullness:

- If DMA read buffer fullness < LT\_MFLAG, then MFLAG signal = 1
- If LT\_MFLAG < DMA read buffer fullness < HT\_MFLAG, then MFLAG signal = 1
- If DMA read buffer fullness > HT\_MFLAG, then MFLAG signal = 0

Similarly, the MFLAG is set based on fullness and the transition history on the out bound writeback pipeline. There is no pre-fetch state for out bound DMA controller. The MFLAG is set when the buffer fullness rises above the HT\_MFLAG bit-field value. The MFLAG is cleared only when the buffer fullness falls back below the LT\_MFLAG bit-field value. In between these two fullness states, the MFLAG bit keeps the previous value.

By default, the MFLAG mechanism is disabled (DSS0\_COMMON\_DISPC\_GLOBAL\_MFLAG\_ATTRIBUTE[1-0] MFLAG\_CTRL register field = 0x0), and the MFLAG signal is low (de-asserted). The arbitration scheme for the pipelines is the same as described in [Section 12.6.4.6.9, DISPC DMA Arbitration](#). That is, round-robin either between high-priority pipelines, or between normal-priority pipelines (if all pipelines are of normal priority).

When the MFLAG\_CTRL register field is set to 0x2, the MFLAG mechanism is enabled, and the MFLAG signal is dynamically set to 0 or 1, depending on DMA buffer fullness and programmed threshold levels, as explained previously in this section. In this case, the arbitration scheme for the pipelines is round-robin between those high-priority pipelines, which have asserted their local MFLAG signals. If there are no high-priority pipelines with their local MFLAG signals asserted, then the arbitration scheme is the same as described in [Section 12.6.4.6.9, DISPC DMA Arbitration](#).

The DSS0\_COMMON\_DISPC\_GLOBAL\_MFLAG\_ATTRIBUTE[6] MFLAG\_START bit defines the following additional rules for the MFLAG mechanism:

- If the MFLAG\_START bit is set to 0x0 (default value), then when the DMA buffer is empty at the beginning of the frame, the MFLAG signal of each pipeline is kept at 0 until PRELOAD is reached (for more information on preloading, see [Section 12.6.4.6.2, DISPC Read DMA Buffers](#)). Then, based on the setting of the DSS0\_COMMON\_DISPC\_GLOBAL\_MFLAG\_ATTRIBUTE[1-0] MFLAG\_CTRL register field, the MFLAG signal is generated and DISPC internal logic is arbitrating between pipeline requests.
- If the MFLAG\_START bit is set to 0x1, then even in the beginning of the frame when the DMA buffer is empty, the configuration of DSS0\_COMMON\_DISPC\_GLOBAL\_MFLAG\_ATTRIBUTE[1-0] MFLAG\_CTRL register field determines the generation of the MFLAG signal.

#### 12.6.4.6.8 DISPC DMA Priority Requests Control

The DSS0\_COMMON\_DSS\_CBA\_CFG register controls the priority level for DMA requests going out to the memory through the system interconnect.

As explained in [Section 12.6.4.6.7, DISPC DMA MFLAG Mechanism](#), the DISPC master port generates a 1-bit MFLAG output signal to raise the priority of all requests made on that port, if any of its DMA buffers runs critically low (determined by a set of user programmable threshold values for each buffer).

DSS uses the MFLAG signal from DISPC to set a 3-bit priority level output (*Mpriority*) for the master port to either a low or high value (configurable in DSS0\_COMMON\_DSS\_CBA\_CFG[2-0] PRI\_LO and [5-3] PRI\_HI register fields with optional values of 0~7) as follows:



- When MFLAG = 0, the PRI\_LO register field determines the value of the Mpriority output for the normal transactions.
- When MFLAG = 1, the PRI\_HI register field determines the value of the Mpriority output for the high-priority transactions.

This Mpriority output directly drives the respective input of the system interconnect port, which corresponds to the DISPC DMA master port.

#### CAUTION

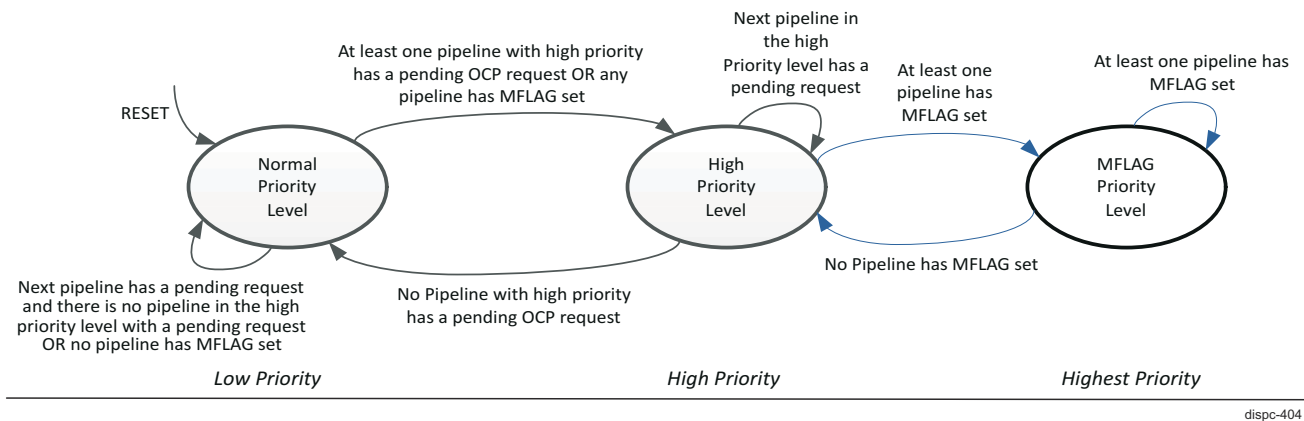
Upon a hardware reset, DSS0\_COMMON\_DSS\_CBA\_CFG[2-0] PRI\_LO and [5-3] PRI\_HI register fields are set by default to 4 and 1, respectively. Afterwards, these priority level register fields can only be modified by a secure host.

#### 12.6.4.6.9 DISPC DMA Arbitration

The read requests sent to the system interconnect are pipelined and arbitrated in a round-robin scheme. The default arbitration scheme can be modified by setting the priority attribute of each pipeline as defined in the DSS0\_VID\_ATTRIBUTES[23] ARBITRATION register bit.

By default, all pipelines have the same priority (normal priority), which means all pipeline requests are treated in a round-robin order manner. If one or more pipelines require a higher number of requests going to the system interconnect, its priority can be moved up to high priority. In this case, the high-priority pipeline is granted access before any pipeline in normal priority. If more than one active pipeline is in high priority, then the behavior is the same as all active pipelines in normal priority. Normal active pipelines are not treated until all high active pipelines are finished. The ARBITRATION bit cannot be modified during the entire frame.

In addition to the priority bit-field, the MFLAG mechanism can also result in a higher priority for a pipeline. An MFLAG priority level is set, for all pipelines for which MFLAG bit is set, which is defined as the highest priority level for arbitration. For more details on MFLAG mechanism, see [Section 12.6.4.6.7, DISPC DMA MFLAG Mechanism](#). [Figure 12-552](#) shows the transition between the different priority levels.



**Figure 12-552. DISPC DMA High/Low Priority Arbitration**

#### 12.6.4.6.10 DISPC DMA Ultra-Low Power Mode

In ultra-low power mode, the system interconnect is used to fill up the DMA buffers to store all the data required to display a full frame. Then, the system interconnect is not used anymore to fetch new pixels for the consequent frames. The data are fetched once into the DMA buffer and then the following frames re-use the DMA buffer to display on the screen.

The programming of the ultra-low power mode is independent for each pipeline and is achieved via the DSS0\_VID\_ATTRIBUTES[24] SELFREFRESH register bit. One pipeline may have all frame pixels into the DMA

buffer and other pipeline may have to refill the DMA buffers along the display scan, because the frame buffer is too big to be stored in the DMA buffer.

The DMA buffers can be merged in order to optimize the system interconnect OFF time. Each DMA buffer dedicated to one pipeline can be split into two buffers. The merge of the DMA buffers into a single one can be used at the same time to improve the Ultra-Low Power mode.

Two ultra-low power modes can be entered, manual or automatic mode:

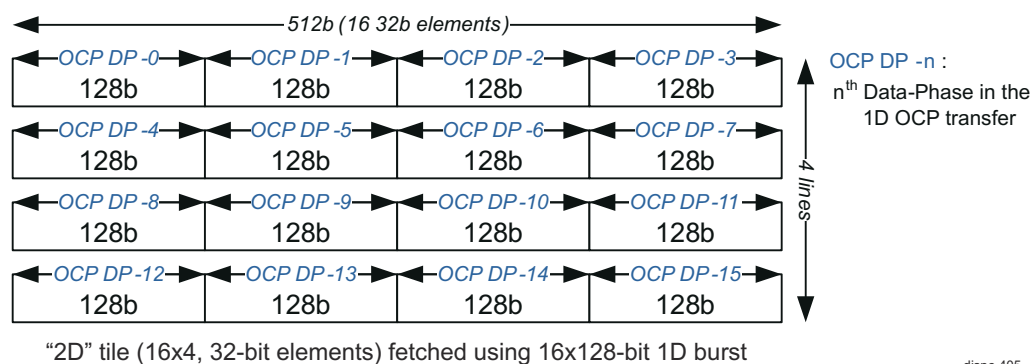
- **Manual self-refresh mode:** Starting self-refresh mode is done manually by setting the SELFREFRESH bit to 0x1 after capturing a frame in the DMA buffers. Self-refresh mode is stopped by setting the SELFREFRESH bit to 0x0. Software must first disable the SELFREFRESH bit during at least one frame in order to capture the data (by setting the GOBIT register bit of the video port the pipeline is associated with). Once the DSS0\_VP\_CONTROL[5] GOBIT bit has been set, the software must read the GOBIT bit to ensure that the frame has been loaded into the buffer, then it can set the SELFREFRESH bit to 1. Once SELFREFRESH is enabled, the fetch of data from the system memory is stopped for the following frames. The software needs to reset the SELFREFRESH bit in order to restart fetching data from system memory.
- **Automatic self-refresh mode:** By setting the DSS0\_VID\_ATTRIBUTES[17] SELFREFRESHAUTO bit to 0x1, the transition from disabled to enabled for self-refresh mode is controlled by hardware. This allows the software to reset the SELFREFRESH bit to "disabled", and then automatically after the fetch of the first frame the hardware switches back SELFREFRESH bit to "enabled". The SELFREFRESH must be disabled during at least one frame in order to capture the data, so software must reset the bit to 0 every time the data in the DMA buffer needs to be updated. The hardware reads the data inside the DMA buffer without accessing the interconnect and system memory during the frame, then modifies the SELFREFRESH bit to reflect the current state of the self-refresh mode by setting the bit to 0x1.

#### 12.6.4.6.11 DISPC Compressed Data Format Support

The DISPC is capable of reading a compressed frame buffer through the FBDC block. The FBDC is a Frame Buffer Decompression module that is compatible with the lossless compression module (FBC) in the GPU in the SoC. The FBDC performs the lossless decompression of the compressed images on a tile-by-tile basis. The size of the tiles is 16 pixels by 4 lines or 32 pixels by 2 lines. The FBDC is enabled by setting the DSS0\_VID\_FBDC\_ATTRIBUTES[0] ENABLE register bit.

##### 12.6.4.6.11.1 FBDC Tile Request

Since the compression/decompression is tile based, the DMA engine generates a 256-byte (16x128-bit) 1D burst request for each tile (a tile is a 16x4 or 32x2 2D structure of 32-bit elements) to FBDC which handles the DMA transfer and decompression of the requested tile. The de-compression phases are transparent to the DISPC hardware. Only ordering of the pixels from a returned tile is taken care by the DISPC DMA engine using 1D burst. The data returned from the FBDC, for a single tile over different OCP data-phases (DP), is as shown in Figure 12-553.



**Figure 12-553. DISPC FBDC Tile Request (16x4 Tile)**

---

**Note**

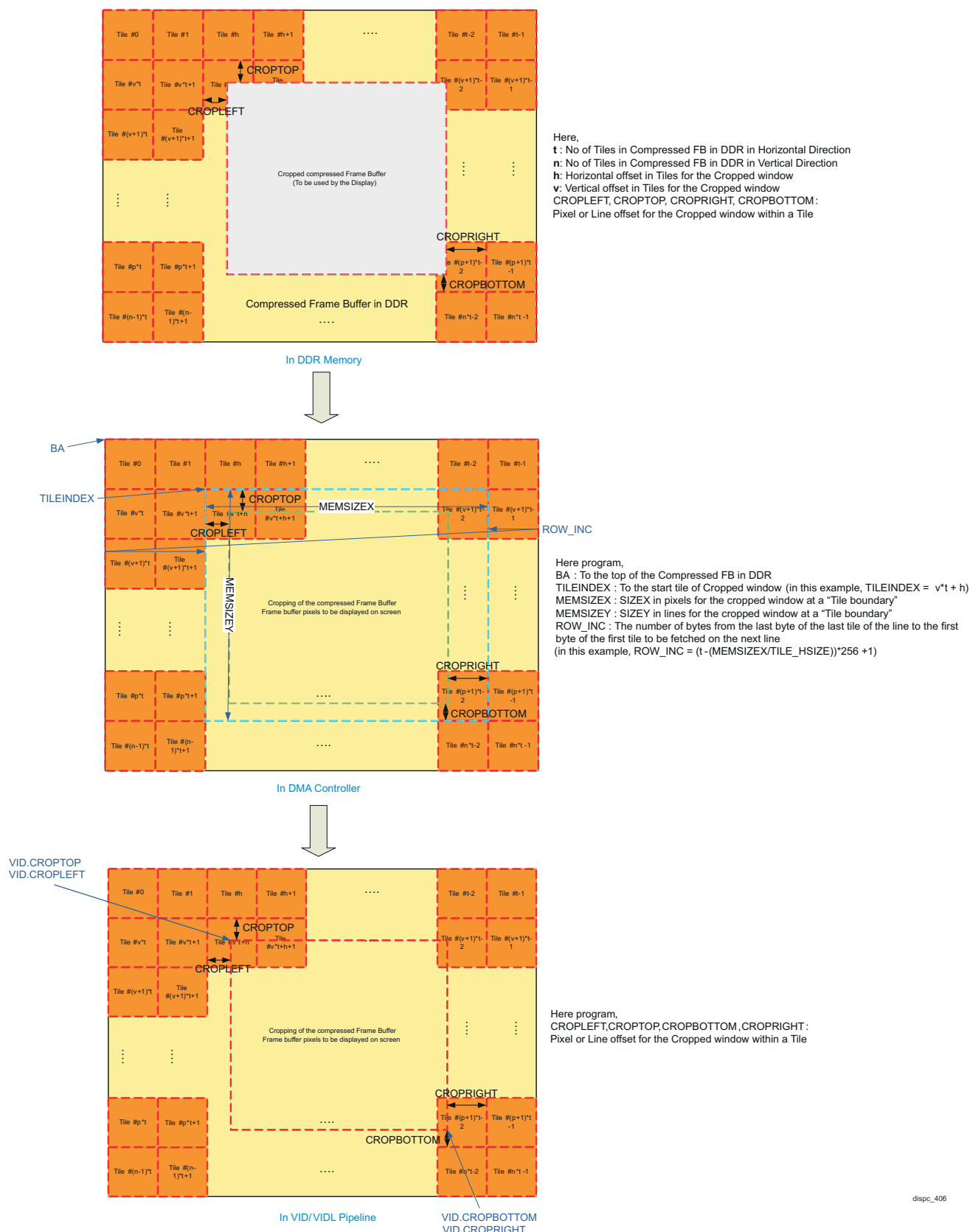
Due to the tile nature of the data, the following DMA features are not available with compressed data formats:

- Flip/Mirror support
  - ROW\_INC on a line basis (that is, ROW\_INC cannot be used to skip lines). It can only be specified in such a way as to skip entire tile rows.
- 

**12.6.4.6.11.2 FBDC Source Cropping**

The FBDC supports a source cropping of the compressed frame buffer. The source cropping is achieved as a two-step process.

- Cropping at a tile boundary is achieved by the DMA controller.
- Cropping within a tile (to a pixel or a line) is achieved by the VID/VIDL pipeline. That is a regular crop feature in the VID/VIDL pipeline which is useful for non-compressed frames as well.



dispc\_406

Figure 12-554. DISPC FBDC Example of a Frame Buffer Source Cropping (Decompression)

The necessary programming for cropping at a tile boundary within the DMA Controller (based on [Figure 12-554](#)) is as follows:

- BA must be programmed to the top of the compressed frame buffer in DDR.
- TILEINDEX must be programmed to the start tile of the cropped window in the DSS0\_VID\_TILE[22-0] TILEINDEX register field. In this example,  $TILEINDEX = v \cdot t + h$ .
- MEMSIZE\_X is the SIZE\_X in pixels for the cropped window at a "Tile boundary".
- MEMSIZE\_Y is the SIZE\_Y in lines for the cropped window at a "Tile boundary".
- ROW\_INC is the number of bytes from the last byte of the last tile of the line to the first byte of the first tile to be fetched on the next line. In this example,  $ROW\_INC = (t - (MEMSIZE\_X / TILE\_H\_SIZE)) \cdot 256 + 1$ , where  $TILE\_H\_SIZE = 16$  or  $32$  depending on whether 16x4 tile-type is used or a 32x2 tile-type is used. The tile type must be programmed in the DSS0\_VID\_FBDC\_ATTRIBUTES[9-8] TILETYPE register field.

For details on the register programming for the crop feature in the VID/VIDL pipeline, see [Section 12.6.4.8.9, DISPC VID Cropping Support](#).

#### 12.6.4.7 DISPC Pixel Data Formats

The DISPC pipelines support various types of memory formats, as listed in [Table 12-549](#).

For BITMAP formats the nibble mode (pixels in each byte are packed in reverse order) can be enabled by setting the DSS0\_VID\_ATTRIBUTES[10] NIBBLEMODE register bit to 0x1. The nibble mode is supported only for the VID/VIDL pipelines, but not for the WB pipeline.

The pixel data format can be selected by loading the corresponding value from [Table 12-549](#) in the DSS0\_VID\_ATTRIBUTES/DSS0\_WB\_ATTRIBUTES[6-1] FORMAT register bit-field.

**Table 12-549. DISPC Supported Pixel Data Formats**

FORMAT Register Field Value		Pixel Format <sup>(4)</sup>		Component Bit Depth
Alpha	Alpha-X	VID and VIDL Pipelines	WB Pipeline	
0x00	0x20	ARGB16-4444	ARGB16-4444	4
0x01	0x21	ABGR16-4444	ABGR16-4444	4
0x02	0x22	RGBA16-4444	RGBA16-4444	4
0x03	NA	RGB16-565	RGB16-565	5(R,B), 6(G)
0x04	NA	BGR16-565	BGR16-565	5(R,B), 6(G)
0x05	0x25	ARGB16-1555	ARGB16-1555	1(A), 5(R,G,B)
0x06	0x26	ABGR16-1555	ABGR16-1555	1(A), 5(R,G,B)
0x07	0x27	ARGB32-8888	ARGB32-8888	8
0x08	0x28	ABGR32-8888	ABGR32-8888	8
0x09	0x29	RGBA32-8888	RGBA32-8888	8
0x0A	0x2A	BGRA32-8888	BGRA32-8888	8
0x0B	NA	RGB24-888	RGB24-888	8
0x0C	NA	BGR24-888	BGR24-888	8
0x0E	0x2E	ARGB32-2101010	ARGB32-2101010	2(A), 10(R,G,B)
0x0F	0x2F	ABGR32-2101010	ABGR32-2101010	2(A), 10(R,G,B)
0x10	0x30	ARGB64-16161616	ARGB64-16161616	16
0x11	0x31	RGBA64-16161616	RGBA64-16161616	16
0x12	NA	BITMAP1	-	1
0x13	NA	BITMAP2	-	2
0x14	NA	BITMAP4	-	4
0x15	NA	BITMAP8	-	8
0x16	NA	RGB565A8 <sup>(1)</sup>	RGB565A8 <sup>(1)</sup>	5(R,B), 6(G), separate 8(A)

**Table 12-549. DISPC Supported Pixel Data Formats (continued)**

FORMAT Register Field Value		Pixel Format <sup>(4)</sup>		Component Bit Depth
0x17	NA	BGR565A8 <sup>(1)</sup>	BGR565A8 <sup>(1)</sup>	5(R,B), 6(G), separate 8(A)
Packed	Planar	Pixel Format	Pixel Format	Component Bit Depth <sup>(5)</sup>
0x3E	NA	YUV422-YUV2	YUV422-YUV2	8/10/12 <sup>(3)</sup>
0x3F	NA	YUV422-UYVY	YUV422-UYVY	8/10/12 <sup>(3)</sup>
NA	0x3C	YUV422-NV12	YUV422-NV12	8/10/12 <sup>(3)</sup>
NA	0x3D	YUV420-NV12	YUV420-NV12	8/10/12 <sup>(3)</sup>
NA	See <sup>(2)</sup>	YUV420-NV21 YUV422-NV21	YUV420-NV21 YUV422-NV21	8/10/12

- (1) The video and writeback pipelines support an optional 8-bit Alpha plane (separate buffer), if the pixel format of the source/destination buffer is either RGB16-565 or BGR16-565.
- (2) NV21 formats for YUV420/YUV422 are indirectly supported through chroma swapping during the color space conversion.
- (3) The 10-bit/12-bit versions of these YUV formats are also supported in both packed and unpacked (in 16-bit container) formats. For unpacked formats, both LSB or MSB alignments are supported. These configurations are set using a separate configuration registers: DSS0\_VID\_ATTRIBUTES2 and DSS0\_WB\_ATTRIBUTES2. The default is an 8-bit packed format support.
- (4) All RGB formats with alpha component include both pre/non-pre-multiplied data support. Also, alpha can be disabled to work as X (can be replaced with global alpha value).
- (5) The writeback path supports only 8-bit and 10-bit YUV data formats.

Figure 12-555 shows the pixel data memory organization for the bitmap pixel formats.

**BITMAP 1-bpp (0x12)**

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0
P31	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0

**BITMAP 2-bpp (0x13)**

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0																

**BITMAP 1-bpp (0x14)**

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0
Pixel 7		Pixel 6		Pixel 5		Pixel 4		Pixel 3		Pixel 2		Pixel 1		Pixel 0																	

**BITMAP 1-bpp (0x15)**

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0
Pixel 3				Pixel 2				Pixel 1				Pixel 0																			

For BITMAP P-1/2/4 bpp, the Nibble mode (pixels in each byte are packed in reverse order) is also supported.

**BITMAP 1-bpp (0x12) – Nibble Mode**

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0
P24	P25	P26	P27	P28	P29	P30	P31	P16	P17	P18	P19	P20	P21	P22	P23	P8	P9	P10	P11	P12	P13	P14	P15	P0	P1	P2	P3	P4	P5	P6	P7

**BITMAP 2-bpp (0x13) – Nibble Mode**

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0
P12	P13	P14	P15	P8	P9	P10	P11	P4	P5	P6	P7	P0	P1	P2	P3																

**BITMAP 4-bpp (0x14) – Nibble Mode**

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0
Pixel 6		Pixel 7		Pixel 4		Pixel 5		Pixel 2		Pixel 3		Pixel 0		Pixel 1																	

disp-301

**Figure 12-555. DISPC Bitmap Pixel Formats**

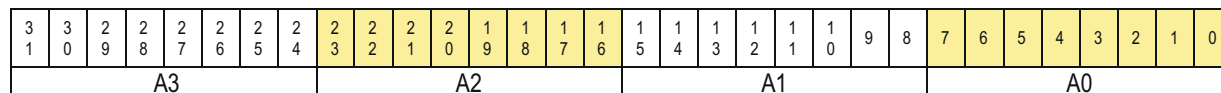
Figure 12-556 and Figure 12-557 show the pixel data memory organization for the RGB 16-bit pixel formats.

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0
B1				G1				R1				B0				G0				R0											

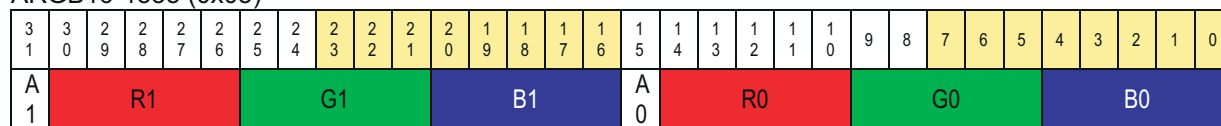
Copyright © 2024 Texas Instruments Incorporated



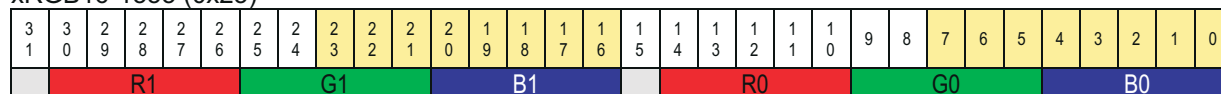
RGB565A8 (0x16) and BGR565A8 (0x17) are RGB16-565 and BGR16-565, respectively, with a separate Alpha-8bit plane.



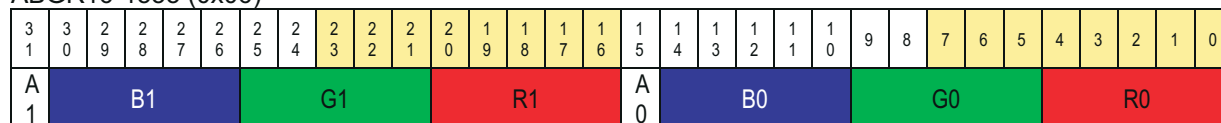
ARGB16-1555 (0x05)



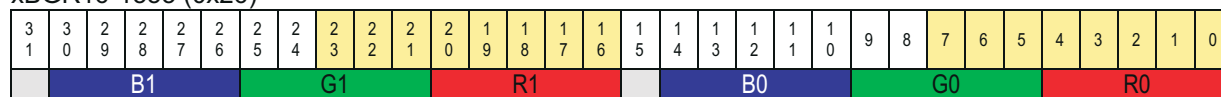
xRGB16-1555 (0x25)



ABGR16-1555 (0x06)



xBGR16-1555 (0x26)

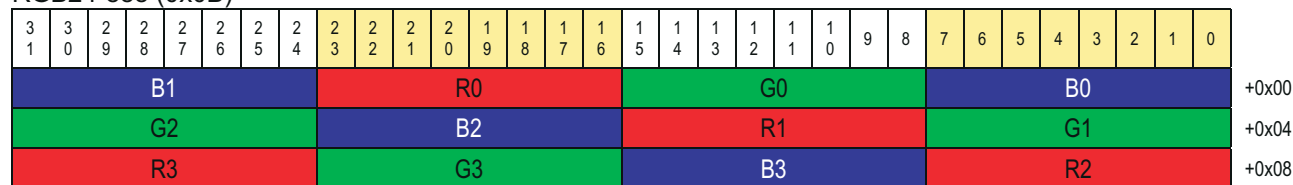


dispc-303

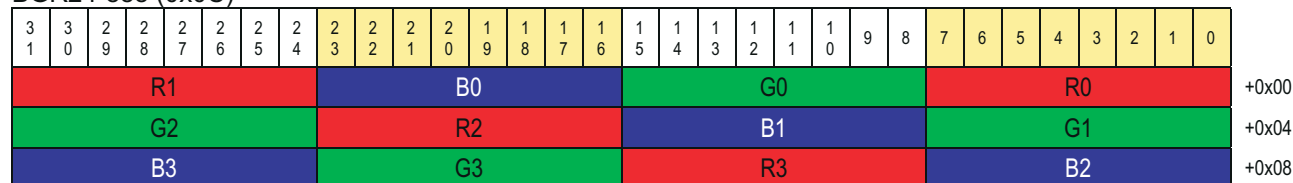
**Figure 12-557. DISPC RGB 16-bit Pixel Formats 2**

Figure 12-558 shows the pixel data memory organization for the RGB 24-bit pixel formats.

RGB24-888 (0x0B)



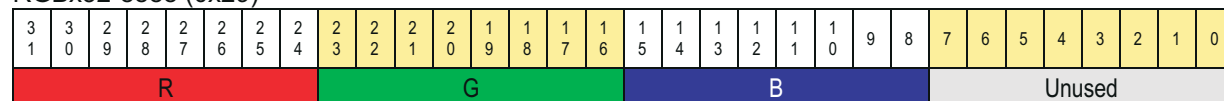
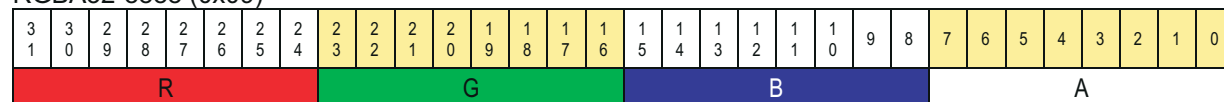
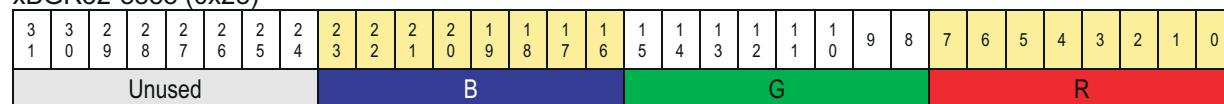
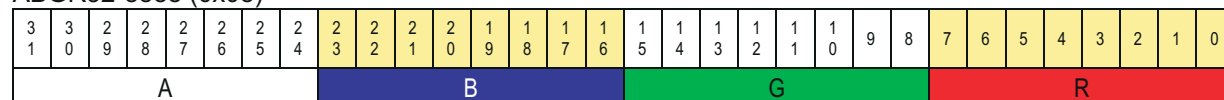
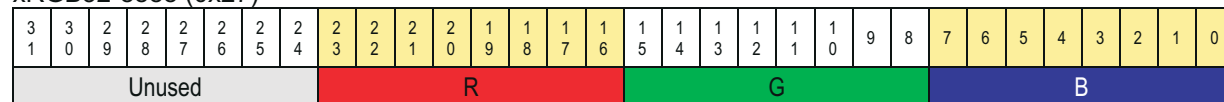
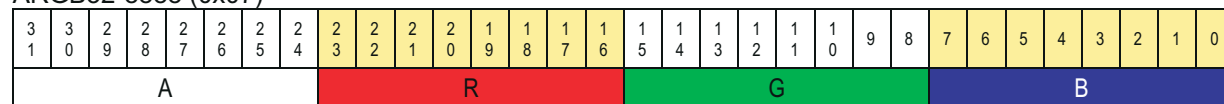
BGR24-888 (0x0C)



dispc-304

### Figure 12-558. DISPC RGB 24-bit Pixel Formats

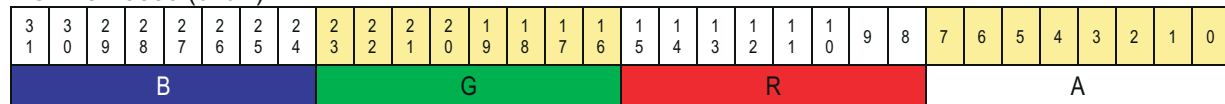
Figure 12-559 and Figure 12-560 show the pixel data memory organization for the RGB 32-bit pixel formats.



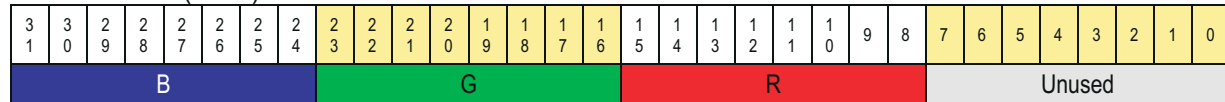
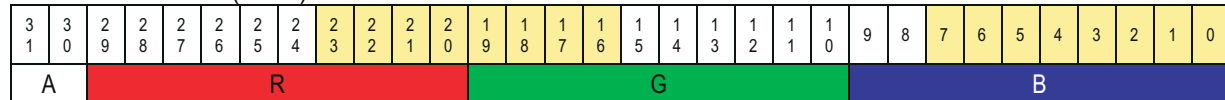
dispc-305

### Figure 12-559. DISPC RGB 32-bit Pixel Formats 1

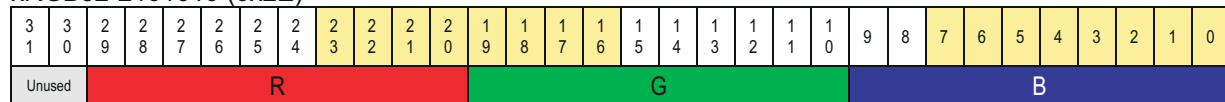
### BGRA32-8888 (0x0A)



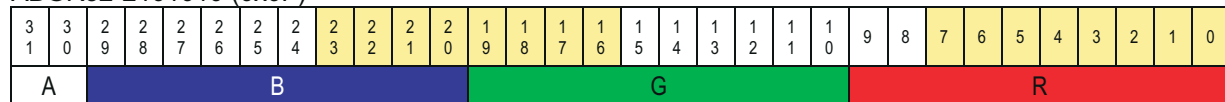
BGRx32-8888 (0x2A)

ARGB32-2101010 (0x0E)

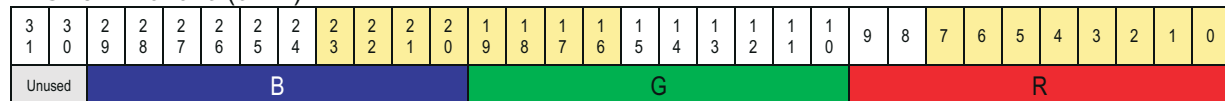
xRGB32-2101010 (0x2E)



ABGR32-2101010 (0x0F)



xBGR32-2101010 (0x2F)



dispc-306

### Figure 12-560. DISPC RGB 32-bit Pixel Formats 2

Figure 12-561 shows the pixel data memory organization for the RGB 64-bit pixel formats.

## dispc-307

dispc-308

Copyright © 2024 Texas Instruments Incorporated



Note: Each new line in memory must start at a 128-bit aligned address for this format

ATTRIBUTES2.YUV\_SIZE = 1 (10b)  
ATTRIBUTES2.YUV\_MODE = 0 (Packed)  
ATTRIBUTES2.YUV\_ALIGN = 0 (N/A)

### YUV422 (1-plane/Packed)

YUV2 4:2:2 – 10bit (0x3E)

3 1			3 0			2 9			2 8			2 7			2 6			2 5			2 4			2 3			2 2			2 1			2 0			1 9			1 8			1 7			1 6			1 5			1 4			1 3			1 2			1 1			1 0			9			8			7			6			5			4			3			2			1			0		
				Y1																Cb0																Y0																+0x0																																											
				Cb1																Y2																Cr0																+0x4																																											
				Y4																Cr1																Y3																+0x8																																											
				Cr2																Y5																Cb2																+0xC																																											

UYVY 4:2:2 – 10bit (0x3F)

CrV1 - 16x2 - 16x8 (3x9)																																
3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0	
		Cr0										Y0										Cb0										+0x0
		Y2										Cb1										Y1										+0x4
		Cb2										Y3										Cr1										+0x8
		Y5										Cr2										Y4										+0xC

YUV420/YUV422 (2-plane/Planar)

YUV 4:2:0 – NV12 (10-bit) (0x3D), YUV 4:2:2 – NV12 (10-bit) (0x3C)

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0	
		Y2										Y1										Y0										+0x0
		Y5										Y4										Y3										+0x4
		Y8										Y7										Y6										+0x0
		Y11										Y10										Y9										+0x4

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0	
		Cb1										Cr0										Cb0										+0x0
		Cr2										Cb2										Cr1										+0x4
		Cb4										Cr3										Cb3										+0x8
		Cr5										Cb5										Cr4										+0xC

dispc-310

### Figure 12-564. DISPC YUV 10-bit Pixel Packed Formats

Figure 12-565 and Figure 12-566 show the pixel data memory organization for the YUV 12-bit packed pixel formats, together with some specific register settings.

YUV 12-bit formats have the same component packing order as 8-bit formats except that the packing is done across a multiple 64-bit word with 4 MSB in each 64-bit word not used.

Note: Each new line in memory must start at a 128-bit aligned address for this format

ATTRIBUTES2.YUV SIZE = 2 (12b)

ATTRIBUTES2.YUV\_MODE = 0 (Packed)

ATTRIBUTES2.YUV\_ALIGN = 0 (N/A)

YUV422 (1-plane)

YUV2 4:2:2 – 12bit (0x3E)

COEFFICIENT MATRIX (COEF)																													
3	3	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	9	8	7	6	5	4	3	2	1	0	
Y1[7:0]								Cb0										Y0										+0x0	
								Y2										Cr0										Y1[11:8]	+0x4
Cr1[7:0]								Y3										Cb1											+0x8
								Cb2										Y4										Cr1[11:8]	+0xC
Y6[7:0]								Cr2										Y5											+0x10
								Y7										Cb3										Y6[11:8]	+0x14
Cb4[7:0]								Y8										Cr3											+018
								Cr4										Y9										Cb4[11:8]	+1C

UYVY 4:2:2 – 12bit (0x3F)

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	9	8	7	6	5	4	3	2	1	0
Cr0[7:0]								Y0								Cb0												+0x0		
		Cb1								Y1								Cr0[11:8]				+0x4								
Y3[7:0]						Cr1								Y2								+0x8								
		Y4								Cb2								Y3[11:8]				+0xC								
Cb3[7:0]						Y5								Cr2								+0x10								
		Cr3								Y6								Cb3[11:8]				+0x14								
Y8[7:0]						Cb4								Y7								+018								
		Y9								Cr4								Y8[11:8]				+1C								

dispc-311

**Figure 12-565. DISPC YUV 12-bit Packed Pixel Formats 1**

dispc-312



YUV 10-bit/12-bit unpacked formats have the same component packing order as 8-bit formats except that each component is stored in a 16-bit container (with MSB or LSB bits within the 16-bit container not used depending on the MSB/LSB alignment).

ATTRIBUTES2.YUV\_SIZE = 1 or 2 (10b or 12b)

ATTRIBUTES2.YUV\_MODE = 1 (Unpacked)

ATTRIBUTES2.YUV\_ALIGN = 0 or 1 (LSB or MSB aligned)

YUV422 (1-plane)

10-bit unpacked LSB aligned

YUV2 4:2:2 – 10bit (0x3E)

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0	
unused						Cb0										unused						Y0										+0x0
unused						Cr0										unused						Y1										+0x4

UYVY 4:2:2 – 10bit (0x3F)

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0	
unused						Y0										unused						Cb0										+0x0
unused						Y1										unused						Cr0										+0x4

10-bit unpacked MSB aligned

YUV2 4:2:2 – 10bit (0x3E)

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0
Cb0										unused						Y0						unused						+0x0			
Cr0										unused						Y1						unused						+0x4			

UYVY 4:2:2 – 10bit (0x3F)

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0
Y0										unused				Cb0								unused				+0x0					
Y1										unused				Cr0								unused				+0x4					

dispc-313

**Figure 12-567. DISPC YUV 10-bit Unpacked Pixel Formats 1**



YUV420/YUV422 (2-plane)

## 10-bit unpacked LSB aligned

YUV 4:2:0 – NV12 (10-bit) (0x3D), YUV 4:2:2 – NV12 (10-bit) (0x3C)

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0	
unused						Y1										unused						Y0										+0x0
unused						Y3										unused						Y2										+0x4

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0		
unused						Cr0										unused						Cb0										+0x0
unused						Cr1										unused						Cb1										+0x4

## 10-bit unpacked MSB aligned

YUV 4:2:0 – NV12 (10-bit) (0x3D), YUV 4:2:2 – NV12 (10-bit) (0x3C)

3	1	3	0	2	9	2	8	2	7	2	6	2	5	2	4	2	3	2	2	2	1	2	0	1	9	1	8	1	7	1	6	1	5	1	4	1	3	1	2	1	1	1	0	9	8	7	6	5	4	3	2	1	0
Y1										unused					Y0										unused					+0x0																							
Y3										unused					Y2										unused					+0x4																							

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0
Cr0										unused						Cb0						unused						+0x0			
Cr1										unused						Cb1						unused						+0x4			

## 12-bit unpacked LSB aligned

YUV 4:2:0 – NV12 (12-bit) (0x3D), YUV 4:2:2 – NV12 (12-bit) (0x3C)

3	1	3	0	2	9	2	8	2	7	2	6	2	5	2	4	2	3	2	2	2	1	2	0	1	9	1	8	1	7	1	6	1	5	1	4	1	3	1	2	1	1	1	0	9	8	7	6	5	4	3	2	1	0
unused				Y1																unused				Y0																+0x0													
unused				Y3																unused				Y2																+0x4													

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0	
unused			Cr0													unused			Cb0													+0x0
unused			Cr1													unused			Cb1													+0x4

## 12-bit unpacked MSB aligned

YUV 4:2:0 – NV12 (12-bit) (0x3D), YUV 4:2:2 – NV12 (12-bit) (0x3C)

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0
Y1												unused		Y0												unused		+0x0			
Y3												unused		Y2												unused		+0x4			

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0
Cr0												unused		Cb0												unused		+0x0			
Cr1												unused		Cb1												unused		+0x4			

**Figure 12-569. DISPC YUV 10-bit/12-bit Unpacked Pixel Formats 3**

### 12.6.4.8 DISPC Video Pipeline

The DISPC includes two types of input video pipelines:

- Video pipeline (VID1 and VID2)
- Video lite pipeline (VIDL1 and VIDL2)

The video pipeline (VID) consists of:

- VC-1 range mapping unit for YUV422/YUV420 input formats;
- Replication logic for ARGB input formats;
- One 256 x 24-bit entries Color Look-up Table (CLUT);
- Polyphase-filter based resizer unit (scaler) supporting chroma upsampling;
- Luma-key support;
- Color Space Conversion (CSC) unit (YUV to RGB), which can be used also for programmable BCHS (Brightness/Contrast/Hue/Saturation) control;

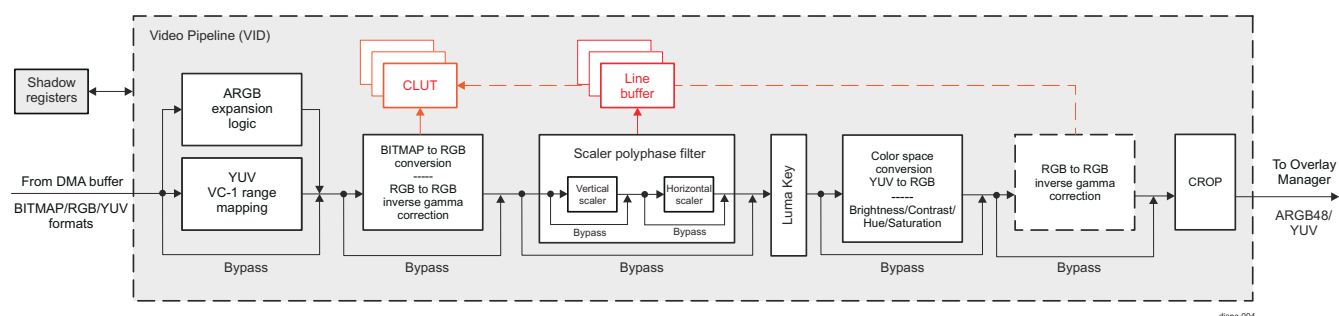
The video lite pipeline (VIDL) is identical to the video pipeline (VID), except for:

- Polyphase-filter based resizer (scaler) is not included;
- Chroma upsampling is done using dedicated YUV420-to-422 and YUV422-to-444 upsamplers (instead of using the scaler);

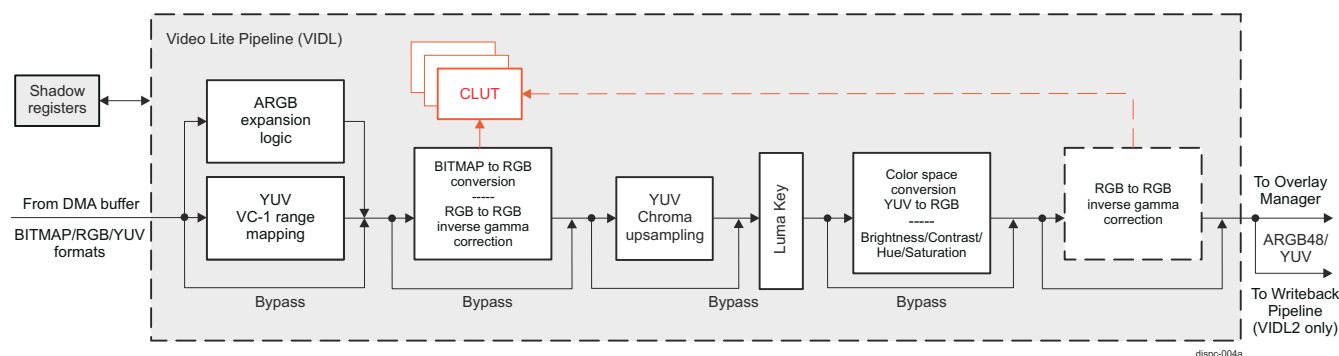
Each pipeline processing block can be independently bypassed.

#### Note

The DISPC input video pipelines, VID and VID1L, will be commonly referred to as *video pipeline (VID)* in the following sections. Any differences in their functionality will be highlighted.



**Figure 12-570. DISPC Video Pipeline Configuration**



**Figure 12-571. DISPC Video Lite Pipeline Configuration**

The input of the VID pipeline is connected to the video DMA buffer controller. The pixel output of the VID pipeline is connected to the overlay managers. The VID pipeline configuration supports various BITMAP, RGB (ARGB and RGBA), and YUV formats, as listed in [Table 12-549, DISPC Pixel Data Formats](#).

The 256-entry CLUT is either used to convert BITMAP (1, 2, 4, or 8-bit indexed formats) into RGB format, or for RGB to RGB inverse gamma correction. For a BITMAP format data, scaling is not supported. Scaling and color look-up table features are mutually exclusive. If the color look-up feature is enabled, then the video pipeline scaler has to be disabled.

For chroma sub-sampled YUV formats (YUV422 and YUV420-NV12/NV21):

- VID pipeline: the scaler unit upsamples the chrominance data using both vertical and horizontal polyphase filters;
- VIDL pipeline: dedicated YUV420 to YUV422 and YUV422 to YUV444 chroma upsamplers are used;

For ARGB source data with less than/equal to 10-bit component data size the replication logic (ARGB expansion) converts the data to ARGB48 by replicating the MSBs into the LSBs:

- When scaling is disabled (or no scaler supported), the resulting ARGB48 data is directly provided to the pipeline output;
- When vertical scaling is engaged, the resulting ARGB48 data is first truncated to ARGB8888, and then converted to ARGB10101010 (by MSBs replication into LSBs), before being fed to the vertical scaler input;
- When vertical scaling is disabled, but horizontal scaling is engaged, the resulting ARGB48 data is directly provided to the horizontal scaler input;

#### Note

There are limitations on using the inverse gamma (InvGamma) operation along with scaling. If the scaler is enabled, the InvGamma operation must always take place after the scaler in the data-path (as shown in [Figure 12-570](#)). This position of the InvGamma operation is controllable through the DSS0\_VID\_ATTRIBUTES2[29] GAMMAINVERSIONPOS register field.

All the memories (line buffers attached to the scaler, LUT and chroma upsamplers) are sized to support 10-bit per color component. This allows full 10-bit support inside the video pipeline.

The VID pipeline can be enabled by setting the DSS0\_VID\_ATTRIBUTES[0] ENABLE register bit. If the video pipeline is disabled, the video window does not exist on the screen and the whole video pipeline and its DMA are inactive. Prior to enabling the video layer a valid configuration has to be set by the user.

#### 12.6.4.8.1 DISPC VID Replication Logic

The replication logic (ARGB expansion) converts ARGB pixel formats into ARGB48 format by replicating the MSBs into the LSBs. The logic is always enabled.

[Table 12-550](#) shows how some of the ARGB formats supported by VID are remapped into ARGB48 by default.

**Table 12-550. DISPC VID Replication: ARGB Pixel Formats Remapping into ARGB48-12121212**

Formats	A[11:0]	R[11:0]	G[11:0]	B[11:0]
	MSB - LSB	MSB - LSB	MSB - LSB	MSB - LSB
xRGB12-4444	111111111111	R[3:0]R[3:0]R[3:0]	G[3:0]G[3:0]G[3:0]	B[3:0]B[3:0]B[3:0]
RGBx12-4444	111111111111	R[3:0]R[3:0]R[3:0]	G[3:0]G[3:0]G[3:0]	B[3:0]B[3:0]B[3:0]
RGB16-565	111111111111	R[4:0]R[4:0]R[4:3]	G[5:0]G[5:0]	B[4:0]B[4:0]B[4:3]
xRGB16-1555	111111111111	R[4:0]R[4:0]R[4:3]	G[4:0]G[4:0]G[4:3]	B[4:0]B[4:0]B[4:3]
ARGB16-1555	AAAAAAAAAA	R[4:0]R[4:0]R[4:3]	G[4:0]G[4:0]G[4:3]	B[4:0]B[4:0]B[4:3]
ARGB16-4444	A[3:0]A[3:0]A[3:0]	R[3:0]R[3:0]R[3:0]	G[3:0]G[3:0]G[3:0]	B[3:0]B[3:0]B[3:0]
RGBA16-4444	A[3:0]A[3:0]A[3:0]	R[3:0]R[3:0]R[3:0]	G[3:0]G[3:0]G[3:0]	B[3:0]B[3:0]B[3:0]
ARGB32-8888	A[7:0]A[7:4]	R[7:0]R[7:4]	G[7:0]G[7:4]	B[7:0]B[7:4]

#### 12.6.4.8.2 DISPC VID VC-1 Range Mapping Unit

The VC-1 range mapping unit is used when the video frame picture is decoded using a VC-1 codec by the video accelerator. It remaps the Y, Cb, and Cr components to the full range (from the reduced data range) before

displaying. The unit is used primarily for YUV420-NV12 (NV21) pixel format, but also can be applied to YUV422 pixel formats (YUV2 and UYVY).

The VC-1 range mapping unit is enabled by setting the DSS0\_VID\_ATTRIBUTES2[0] VC1ENABLE bit to 0x1. The VC-1 range mapping must be enabled only for 8 bits/component YUV input data.

The DSS0\_VID\_ATTRIBUTES2[3:1] VC1\_RANGE\_Y and [6:4]VC1\_RANGE\_CBCR register bitfields are two 3-bit values programmed by the user.

The equations for the mapping process are:

$$Y_{out} = \text{CLIP}((((Y_{int} - 128) \times (\text{VC1\_RANGE\_Y} + 9) + 4) / 8) + 128)$$

$$C_b = \text{CLIP}((((C_b - 128) \times (\text{VC1\_RANGE\_CBCR} + 9) + 4) / 8) + 128)$$

$$C_r = \text{CLIP}((((C_r - 128) \times (\text{VC1\_RANGE\_CBCR} + 9) + 4) / 8) + 128)$$

#### Note

The input and output pixel values are unsigned (Y, Cr, and Cb).

The function CLIP() clips to 0 or 255 when minimum or maximum, respectively, is reached. Otherwise, the resulting output remains identical.

#### 12.6.4.8.3 DISPC VID Color Look-Up Table (CLUT)

The video pipeline supports a look up table to perform either of the following operations:

- Conversions of BITMAP formats (1, 2, 4, or 8-bit indexed) into RGB format (CLUT mode), or
- Inverse gamma correction on non-linear RGB source data (gamma mode)

The look-up table consists of 3 separate 1024 x 10-bit memories and is indexed either by the source BITMAP data or by R/G/B component data. The table is loaded through direct register access by writing to the CLUT registers.

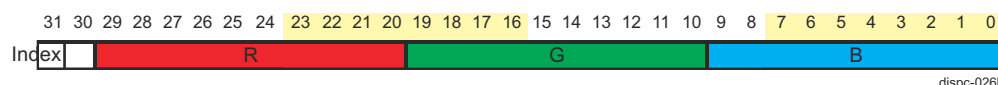
The sequence to load the table is:

1. SW writes (only writes are supported) 32-bit values for both color and monochrome mode using single access, or burst access in linear increment burst mode, into the DSS0\_VID\_CLUT\_0 through DSS0\_VID\_CLUT\_15 registers. The LSB 30 bits [29:0] are used for the value and the MSB 1 bit [31] is used to reset the index of the table for the color mode, while the LSB 8 bits are used for the value and the MSB 1 bit to reset the index of the table for monochrome mode. [Figure 12-572](#) describes the format of one of the palette value in the memory.
2. Loop to Step 1, if there is a new access to the CLUT registers. Software can access other registers between two accesses to the CLUT registers.

SW needs to ensure that there is no visible effect when modifying the table, since it is not under hardware control.

The usage of the look-up table in CLUT mode is activated when a BITMAP format is selected in the DSS0\_VID\_ATTRIBUTES[6:1] FORMAT register bit-field.

The usage of the look-up table for RGB inverse gamma correction can be enabled by setting DSS0\_VID\_ATTRIBUTES[30] GAMMAINVERSION register bit.



**Figure 12-572. DISPC VID CLUT/Gamma Data Memory Organization**

### Note

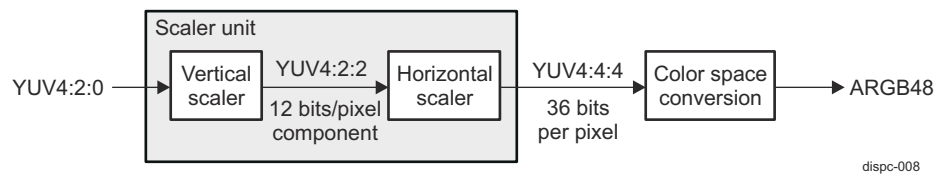
Scaling and color look-up table features are mutually exclusive. If color look-up feature is enabled, then video pipeline scaler has to be disabled.

#### 12.6.4.8.4 DISPC VID Chrominance Resampling

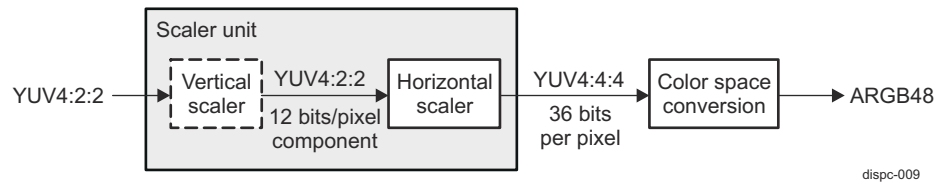
##### 12.6.4.8.4.1 Chrominance Resampling for VID Pipeline

The chrominance resampling from 8-bpc or 10-bpc to 12-bit color components is always performed using filtering (the scaler unit filter). Chrominance resampling and rescaling can be combined to support native rescaling of YUV formats.

The usage of the scaler unit for resampling the chrominance of YUV420 and YUV422 is shown in [Figure 12-573](#) and [Figure 12-574](#), respectively. The settings of the scaler unit to perform chrominance resampling are described in [Section 12.6.4.8.5, DISPC VID Scaler Unit](#).



**Figure 12-573. DISPC VID YUV420 to ARGB48 Using Scaler Unit for Resampling Chrominance**



**Figure 12-574. DISPC VID YUV422 to ARGB48 Using Scaler Unit for Resampling Chrominance**

The vertical scaler is not a mandatory block for resampling the chrominance of YUV422 data, as shown in [Figure 12-574](#).

Using the filter it is possible to re-sample the chrominance from either MPEG1 or MPEG2 sub-sampled chroma source.

##### 12.6.4.8.4.2 Chrominance Resampling for VIDL Pipeline

The VIDL pipeline does not include a scaler unit. Chroma upsampling is done using dedicated YUV420 to YUV422, and YUV422 to YUV444 chroma upsamplers.

The VIDL pipeline converts YUV420 (Chroma) data to YUV422 (Chroma) using a simple vertical average filter (average of two adjacent chroma lines to generate a missing line except for the very bottom line which is generated by repeating the previous chroma line). If the video image is a sub-image of a larger video frame data, then the vertical Luma line offset from the top should be an even number.

The VIDL pipeline converts YUV422 data to YUV444 format using a 4-tap filter (implementing the Catmull-Rom algorithm) to generate missing chroma data as follows:

$$\text{Cout}[2*i] = \text{Cin}[i]$$

$$\text{Cout}[2*i+1] = \text{CLIP} \left( -\frac{1}{16} * \text{Cin}[i-1] + \frac{9}{16} * \text{Cin}[i] + \frac{9}{16} * \text{Cin}[i+1] - \frac{1}{16} * \text{Cin}[i+2] \right)$$

The missing chroma data is generated as a simple weighted average of the surrounding 4 chroma values, which is equivalent to a 4x1 filter kernel with values:  $[-1/16, 9/16, 9/16, -1/16]$ . In this algorithm, edge effects are treated by repeating the first and last pixel.

### 12.6.4.8.5 DISPC VID Scaler Unit

#### Note

Only VID pipeline includes a resizer unit (scaler). The VIDL pipeline does not include a scaler.

The programmable scaler filter works with all supported video formats, including formats with alpha channel. The alpha channel is scaled with the same parameters as the RGB color components. For the YUV formats, Y and Cb/Cr are processed independently. A RGB 64-bit source (16 bits per component) is truncated to 8-bit/component, if the scaler is enabled. Otherwise, it will be truncated to ARGB48 format in the scaler bypass mode. A 10-bit/12-bit YUV source is also truncated to 8-bit, since the scaler is enabled to either upsample the chroma component and/or resize the YUV frame data directly (for memory-to-memory operation with YUV format as the memory destination type).

The filter is based on a finite impulse response (FIR) filter with 16 phases. The filter is a 5-tap for horizontal filtering, and can be configured for 3 or 5 taps for vertical filtering. The filtering can be used for various processing:

- Up-sampling of the picture
- Down-sampling of the picture
- Anti-aliasing reduction
- Chrominance resampling in case of YUV data formats

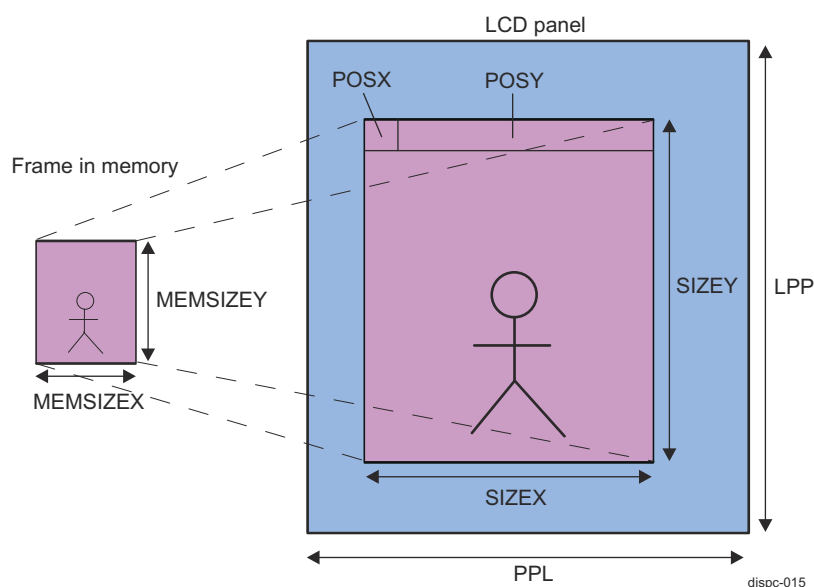
The following limitations must be considered:

- The up-sampling ratio is up to x16.
- The down-sampling ratio using 3-tap configuration is down to x0.5 for RGB format.
- The down-sampling ratio using 5-tap configuration is down to x0.25 for RGB format.

#### Note

The user must set the correct size and position of the original video before resize in order for the up-sampled/down-sampled video to be displayed inside the screen boundaries.

Figure 12-575 shows an example of video up-sampling, with corresponding video window attributes.



**Figure 12-575. DISPC Video Window Attributes**

The video window attributes can be configured in the following register bit-fields:



- Source data format (input to the scaler unit) in DSS0\_VID\_ATTRIBUTES[6-1] FORMAT bit-field
- Video picture width in system memory (frame buffer) in DSS0\_VID\_PICTURE\_SIZE[13-0] MEMSIZE\_X bit-field. The window width is from 1 up to 4096 pixels. All the integer values in the range [1:4096] are allowed except for YUV422 formats. In case of YUV422 and UYVY422 formats the width has to be a multiple of 2 pixels.
- Video picture height in system memory (frame buffer) in DSS0\_VID\_PICTURE\_SIZE[29-16] MEMSIZE\_Y bit-field. The window width is from 1 up to 4096 pixels. All the integer values in the range [1:4096] are allowed.
- Video window width at pipeline output (after resizing) in DSS0\_VID\_SIZE[13-0] SIZE\_X bit-field. The window width is from 1 up to 4096 pixels. All the integer values in the range [1:4096] are allowed up to the screen width minus the horizontal start location of the window on the screen.
- Video window height at pipeline output (after resizing) in DSS0\_VID\_SIZE[29-16] SIZE\_Y bit-field. The window height is from 1 up to 4096 pixels. All the integer values in the range [1:4096] are allowed up to the screen height minus the vertical start location of the window on the screen.
- Video window overlay X-position in DSS0\_OVR\_ATTRIBUTES2\_0 through DSS0\_OVR\_ATTRIBUTES2\_4[13-0] POS\_X bit-field. See [Section 12.6.4.10, DISPC Overlay Managers](#).
- Video window overlay Y-position in DSS0\_OVR\_ATTRIBUTES2\_0 through DSS0\_OVR\_ATTRIBUTES2\_4[29-16] POS\_Y bit-field. See [Section 12.6.4.10, DISPC Overlay Managers](#).
- For configuration of LPP and PPL video port output display parameters, see [Section 12.6.4.11.8, DISPC VP Timing Generator and Display Panel Settings](#).

For vertical up-sampling and down-sampling in a 3-tap configuration, the equations are:

For RGB formats

$$Aout(n) = \left( \sum_{i=-1}^{i=1} Ci(\Phi) \times Ain(n+i) \right) \gg (Cwidth - (OutWidth - InWidth) - 1)$$

$$Rout(n) = \left( \sum_{i=-1}^{i=1} Ci(\Phi) \times Rin(n+i) \right) \gg (Cwidth - (OutWidth - InWidth) - 1)$$

$$Gout(n) = \left( \sum_{i=-1}^{i=1} Ci(\Phi) \times Gin(n+i) \right) \gg (Cwidth - (OutWidth - InWidth) - 1)$$

$$Bout(n) = \left( \sum_{i=-1}^{i=1} Ci(\Phi) \times Bin(n+i) \right) \gg (Cwidth - (OutWidth - InWidth) - 1)$$

For YUV formats

$$Yout(n) = \left( \sum_{i=-1}^{i=1} Cyi(\Phi_y) \times Yin(n+i) \right) \gg (Cwidth - (OutWidth - InWidth) - 1)$$

$$Cbout(n) = \left( \sum_{i=-1}^{i=1} Cci(\Phi_c) \times Cbin(n+i) \right) \gg (Cwidth - (OutWidth - InWidth) - 1)$$

$$Crout(n) = \left( \sum_{i=-1}^{i=1} Cci(\Phi_c) \times Crin(n+i) \right) \gg (Cwidth - (OutWidth - InWidth) - 1)$$

Component	Vertical	Horizontal
Cwidth (Coefficient Width)	10 bits	10 bits
InWidth (Input Width)	10 bits	12 bits
OutWidth (Output Width)	12 bits	12 bits

dispc-013

(23)

For vertical and horizontal up-sampling and down-sampling in a 5-tap configuration, the equations are:

For RGB formats

$$Aout(n) = \left( \sum_{i=-2}^{i=2} Ci(\Phi) \times Ain(n+i) \right) \gg (Cwidth - (OutWidth - InWidth) - 1)$$

$$Rout(n) = \left( \sum_{i=-2}^{i=2} Ci(\Phi) \times Rin(n+i) \right) \gg (Cwidth - (OutWidth - InWidth) - 1)$$

$$Gout(n) = \left( \sum_{i=-2}^{i=2} Ci(\Phi) \times Gin(n+i) \right) \gg (Cwidth - (OutWidth - InWidth) - 1)$$

$$Bout(n) = \left( \sum_{i=-2}^{i=2} Ci(\Phi) \times Bin(n+i) \right) \gg (Cwidth - (OutWidth - InWidth) - 1)$$

For YUV formats

$$Yout(n) = \left( \sum_{i=-2}^{i=2} Cyi(\Phi_y) \times Yin(n+i) \right) \gg (Cwidth - (OutWidth - InWidth) - 1)$$

$$Cbout(n) = \left( \sum_{i=-2}^{i=2} Cci(\Phi_c) \times Cbin(n+i) \right) \gg (Cwidth - (OutWidth - InWidth) - 1)$$

$$Crout(n) = \left( \sum_{i=-2}^{i=2} Cci(\Phi_c) \times Crin(n+i) \right) \gg (Cwidth - (OutWidth - InWidth) - 1)$$

Component	Vertical	Horizontal
Cwidth (Coefficient Width)	10 bits	10 bits
InWidth (Input Width)	10 bits	12 bits
OutWidth (Output Width)	12 bits	12 bits

dispc-012

(24)

### Note

The pixel (n + 1) is the previous pixel with respect to pixel (n). The line (n + 1) is the previous line with respect to line (n).

The coefficients  $C_i()$  depend on the phase between input and output pixels.

The coefficients are different for Y and Cr/Cb filtering because the calculations are independent due to the chrominance resampling for YUV422 and YUV420.

First, the vertical filter is applied to the encoded input pixel data, and then the horizontal filter is applied on the resulting pixel values to generate the output pixel values. The vertical input of the filter consists of:

- Six lines of 2048 × 32 bits for 5-tap configuration
- Three lines of 4096 × 32 bits for 3-tap configuration

Table 12-551 lists some of the scaler supported configurations.

**Table 12-551. DISPC VID Line Buffer Width for Scaler Unit**

Pixel Format	Maximum Input Width (Pixels) for 5-tap	Maximum Input Width (Pixels) for 3-tap
32 bits per pixel (ARGB32-8888, ARGB-2101010, etc.)	2048 pixels wide <sup>(1)</sup>	4096 pixels wide
YUV420, YUV422	4096 pixels wide	4096 pixels wide

- (1) For the 5-tap configuration, the 6th video line is used on the output of the horizontal scaler in order to start the next output line generation while the video pipeline output is being stalled by the overlay manager (either due to display in blanking period and/or in background pixel period). The 6th line buffer is actually 48-bit wide to allow storage of ARGB48 format pixel data.

At the beginning of frame scaling processing, the first line may be duplicated multiple times depending on the initial vertical phase programmed for the poly-phase filter.

At the end of frame scaling processing the last line is duplicated, if the scaling logic requires loading more lines and the last line has been reached.

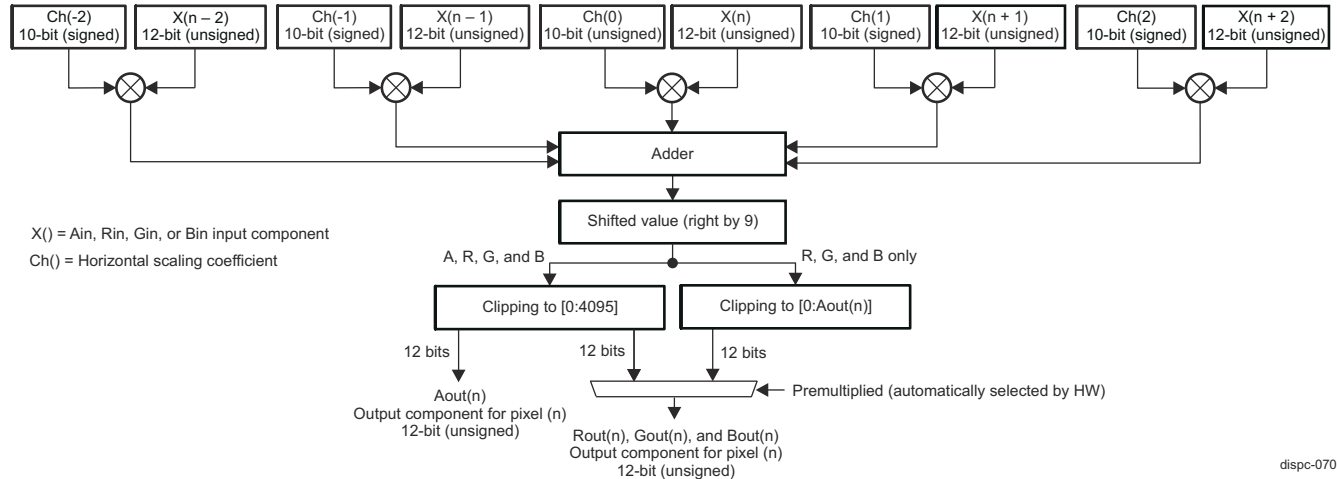
Similarly, the first pixel may be duplicated multiple times depending on the initial horizontal phase programmed for the poly-phase filter. The last pixel is duplicated, if the scaling logic requires loading more pixels and the last pixel has been reached.

The programmable coefficients of the polyphase filters are signed 10-bit values (except for the central coefficient, which is unsigned). The vertical video scaler has an 10-bit input and a 12-bit output. The horizontal scaling stage takes the resulting 12-bit input and produces 12-bit output.

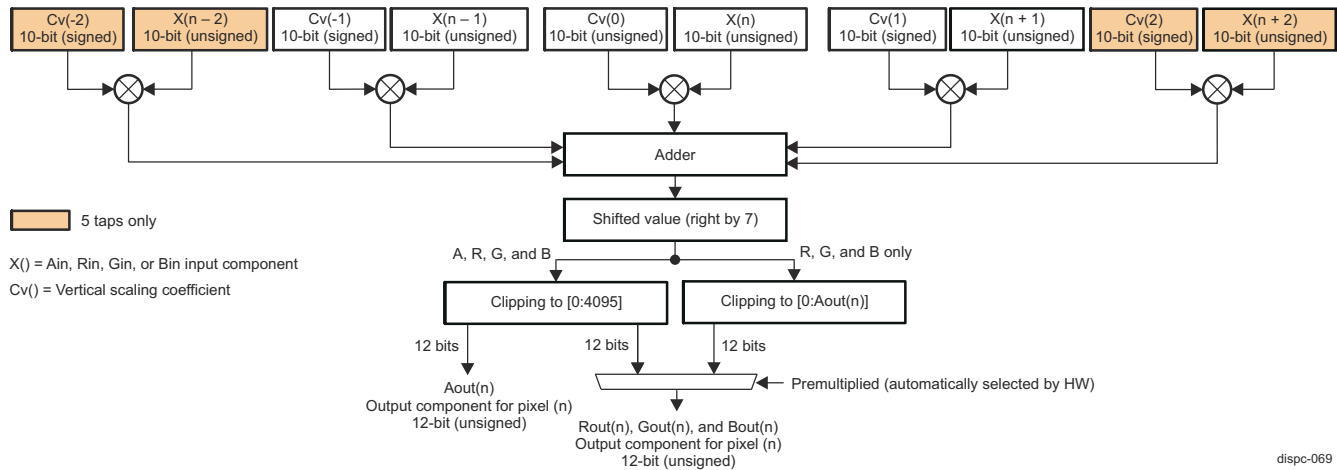
Figure 12-576 and Figure 12-577 show the scaler macro-architecture for components A, R, G, and B. Figure 12-578 and Figure 12-579 show the scaler macro-architecture for components Y, Cr, and Cb.

### Note

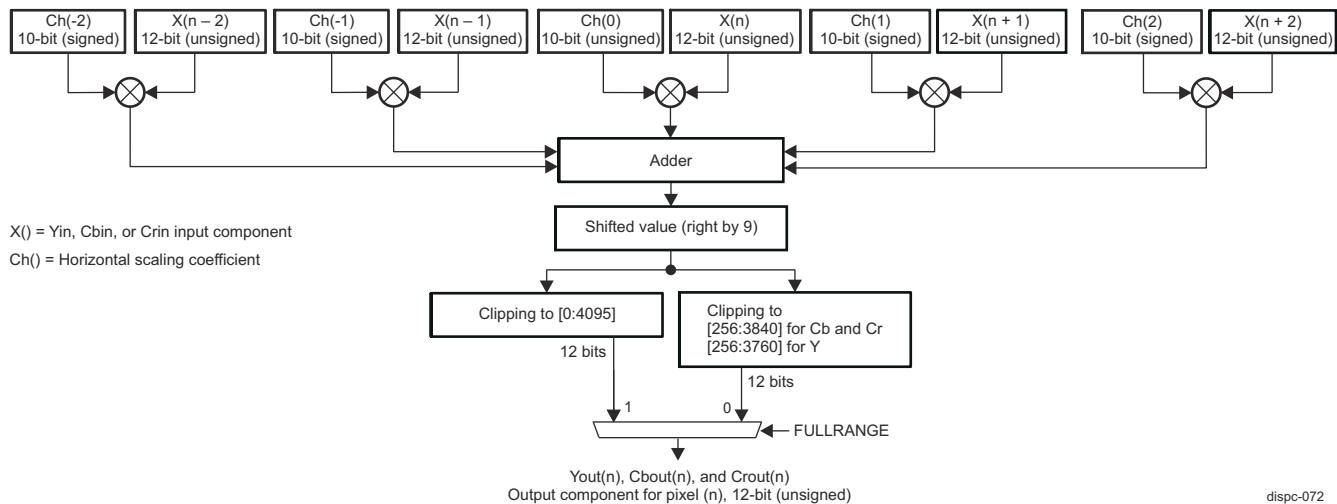
The scaling and CSC clipping is set by the same bit, DSS0\_VID\_ATTRIBUTES[11] FULLRANGE.



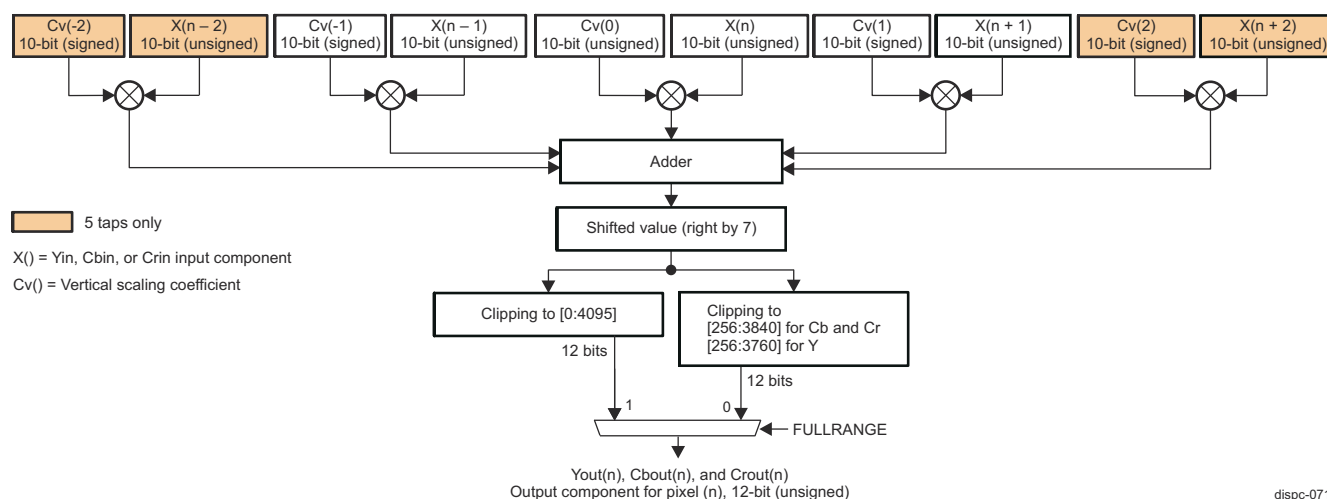
**Figure 12-576. DISPC VID Macro-Architecture of the Horizontal Scaling for A, R, G, and B Components (5-tap Restriction)**



**Figure 12-577. DISPC VID Macro-Architecture of the Vertical Scaling for A, R, G, and B Components (5 and 3 taps)**



**Figure 12-578. DISPC VID Macro-Architecture of the Horizontal Scaling for Y, Cr, and Cb Components (5-tap Restriction)**



**Figure 12-579. DISPC VID Macro-Architecture of the Vertical Scaling for Y, Cr, and Cb Components (5 and 3 taps)**

Table 12-552 list the register fields in the function of the coefficients for the VID horizontal scaler in the DSS0\_VID\_FIR\_COEF\_H0\_0 to DSS0\_VID\_FIR\_COEF\_H0\_8, and DSS0\_VID\_FIR\_COEF\_H12\_0 to DSS0\_VID\_FIR\_COEF\_H12\_15 registers.

**Table 12-552. DISPC Register Fields Associated to Coefficients for ARGB and Y Configuration in VID Horizontal Scaler**

Phases	Ch(2)	Ch(1)	Ch(0)	Ch(-1)	Ch(-2)
	Signed coefficient [29-20] FIRHC2 bitfield	Signed coefficient [19-10] FIRHC1 bitfield	Unsigned central coefficient [9-0] FIRHC0 bitfield	Signed coefficient [19-10] FIRHC1 bitfield	Signed coefficient [29-20] FIRHC2 bitfield
0	DSS0_VID_FIR_COEF_F_H12_0	DSS0_VID_FIR_COEF_F_H12_0	DSS0_VID_FIR_COEF_F_H0_0	DSS0_VID_FIR_COEF_F_H12_0	DSS0_VID_FIR_COEF_F_H12_0
1	DSS0_VID_FIR_COEF_F_H12_1	DSS0_VID_FIR_COEF_F_H12_1	DSS0_VID_FIR_COEF_F_H0_1	DSS0_VID_FIR_COEF_F_H12_15	DSS0_VID_FIR_COEF_F_H12_15
2	DSS0_VID_FIR_COEF_F_H12_2	DSS0_VID_FIR_COEF_F_H12_2	DSS0_VID_FIR_COEF_F_H0_2	DSS0_VID_FIR_COEF_F_H12_14	DSS0_VID_FIR_COEF_F_H12_14
3	DSS0_VID_FIR_COEF_F_H12_3	DSS0_VID_FIR_COEF_F_H12_3	DSS0_VID_FIR_COEF_F_H0_3	DSS0_VID_FIR_COEF_F_H12_13	DSS0_VID_FIR_COEF_F_H12_13
4	DSS0_VID_FIR_COEF_F_H12_4	DSS0_VID_FIR_COEF_F_H12_4	DSS0_VID_FIR_COEF_F_H0_4	DSS0_VID_FIR_COEF_F_H12_12	DSS0_VID_FIR_COEF_F_H12_12
5	DSS0_VID_FIR_COEF_F_H12_5	DSS0_VID_FIR_COEF_F_H12_5	DSS0_VID_FIR_COEF_F_H0_5	DSS0_VID_FIR_COEF_F_H12_11	DSS0_VID_FIR_COEF_F_H12_11
6	DSS0_VID_FIR_COEF_F_H12_6	DSS0_VID_FIR_COEF_F_H12_6	DSS0_VID_FIR_COEF_F_H0_6	DSS0_VID_FIR_COEF_F_H12_10	DSS0_VID_FIR_COEF_F_H12_10
7	DSS0_VID_FIR_COEF_F_H12_7	DSS0_VID_FIR_COEF_F_H12_7	DSS0_VID_FIR_COEF_F_H0_7	DSS0_VID_FIR_COEF_F_H12_9	DSS0_VID_FIR_COEF_F_H12_9
8	DSS0_VID_FIR_COEF_F_H12_8	DSS0_VID_FIR_COEF_F_H12_8	DSS0_VID_FIR_COEF_F_H0_8	DSS0_VID_FIR_COEF_F_H12_8	DSS0_VID_FIR_COEF_F_H12_8
9	DSS0_VID_FIR_COEF_F_H12_9	DSS0_VID_FIR_COEF_F_H12_9	DSS0_VID_FIR_COEF_F_H0_7	DSS0_VID_FIR_COEF_F_H12_7	DSS0_VID_FIR_COEF_F_H12_7
10	DSS0_VID_FIR_COEF_F_H12_10	DSS0_VID_FIR_COEF_F_H12_10	DSS0_VID_FIR_COEF_F_H0_6	DSS0_VID_FIR_COEF_F_H12_6	DSS0_VID_FIR_COEF_F_H12_6
11	DSS0_VID_FIR_COEF_F_H12_11	DSS0_VID_FIR_COEF_F_H12_11	DSS0_VID_FIR_COEF_F_H0_5	DSS0_VID_FIR_COEF_F_H12_5	DSS0_VID_FIR_COEF_F_H12_5
12	DSS0_VID_FIR_COEF_F_H12_12	DSS0_VID_FIR_COEF_F_H12_12	DSS0_VID_FIR_COEF_F_H0_4	DSS0_VID_FIR_COEF_F_H12_4	DSS0_VID_FIR_COEF_F_H12_4

**Table 12-552. DISPC Register Fields Associated to Coefficients for ARGB and Y Configuration in VID Horizontal Scaler (continued)**

Phases	Ch(2)	Ch(1)	Ch(0)	Ch(-1)	Ch(-2)
13	DSS0_VID_FIR_COEF_F_H12_13	DSS0_VID_FIR_COEF_F_H12_13	DSS0_VID_FIR_COEF_F_H0_3	DSS0_VID_FIR_COEF_F_H12_3	DSS0_VID_FIR_COEF_F_H12_3
14	DSS0_VID_FIR_COEF_F_H12_14	DSS0_VID_FIR_COEF_F_H12_14	DSS0_VID_FIR_COEF_F_H0_2	DSS0_VID_FIR_COEF_F_H12_2	DSS0_VID_FIR_COEF_F_H12_2
15	DSS0_VID_FIR_COEF_F_H12_15	DSS0_VID_FIR_COEF_F_H12_15	DSS0_VID_FIR_COEF_F_H0_1	DSS0_VID_FIR_COEF_F_H12_1	DSS0_VID_FIR_COEF_F_H12_1

#### Note

In [Table 12-552](#), the cells without color are duplicated from the grey cells.

Similar table approach applies to the vertical scaler (registers DSS0\_VID\_FIR\_COEF\_V0\_0 to DSS0\_VID\_FIR\_COEF\_V0\_8, and DSS0\_VID\_FIR\_COEF\_V12\_0 to DSS0\_VID\_FIR\_COEF\_V12\_15 are used).

Similar table approach applies to the coefficients for Cb/Cr filtering in case of YUV format (registers DSS0\_VID\_FIR\_COEF\_H0\_C\_0 to DSS0\_VID\_FIR\_COEF\_H0\_C\_8, and DSS0\_VID\_FIR\_COEF\_H12\_C\_0 to DSS0\_VID\_FIR\_COEF\_H12\_C\_15, and DSS0\_VID\_FIR\_COEF\_V0\_C\_0 to DSS0\_VID\_FIR\_COEF\_V0\_C\_8 and DSS0\_VID\_FIR\_COEF\_V12\_C\_0 to DSS0\_VID\_FIR\_COEF\_V12\_C\_15 are used) .

The VID scaler unit vertical and/or horizontal sampling is selected by configuring the DSS0\_VID\_ATTRIBUTES[8-7] RESIZEENABLE bit field.

Prior to enabling the video up/down-sampling block a valid configuration has to be set by the user. After configuring the required VID registers change the DSS0\_VP\_CONTROL[5] GOBIT register bit of the video port the video pipeline is associated with. The software has to wait before setting the GO bit that the hardware has reset the bit. The software reset is not recommended since the application cannot guarantee to be able to reset it before the HW.

The following fields define the configuration of the video up-sampling/down-sampling block in the VID pipeline for ARGB and YUV formats:

- Vertical upsampling and downsampling increment value in the [23-0] FIRVINC bit field of DSS0\_VID\_FIRV (for ARGB and Y) and DSS0\_VID\_FIRV2 (for CbCr) registers. The unsigned integer value range is  $2^{23}$ . Software calculates the value using the following equation:

$$FIRVINC = 2^{21} * \left( \frac{MEMSIZEY+1}{SIZEY+1} \right)$$

dispc-066

- Horizontal upsampling and downsampling increment value in the [23-0] FIRHINC bit field of the DSS0\_VID\_FIRH (for ARGB and Y) and DSS0\_VID\_FIRH2 (for CbCr) registers. The unsigned integer value range is [1:16384]. Software calculates the value using the following equation:

$$FIRHINC = 2^{21} * \left( \frac{MEMSIZEH+1}{SIZEH+1} \right)$$

dispc-067

- Vertical up/downsampling accumulator value in the [23-0] VERTICALACCU bit field of the DSS0\_VID\_ACCUV\_0 and DSS0\_VID\_ACCUV\_1 (for ARGB and Y), and DSS0\_VID\_ACCUV2\_0 and DSS0\_VID\_ACCUV2\_1 (for CbCr) registers. The accumulator value indicates on which phase the vertical filtering starts. The DSS0\_VID\_ACCUV\_0 register is used for progressive output, while for interlace output both DSS0\_VID\_ACCUV\_0 and DSS0\_VID\_ACCUV\_1 registers are used. The DSS0\_VID\_ACCUV2\_0 and DSS0\_VID\_ACCUV2\_1 registers are used in the same manner for progressive or interlace output.

- Vertical upsampling and downsampling line buffer configuration via the DSS0\_VID\_ATTRIBUTES[21] VERTICALTAPS bit: The default value at reset time is 0x0 (3-tap configuration is used). If the bit field is reset, the 3-tap configuration is used.
- Horizontal upsampling and downsampling accumulator value in the [23-0] HORIZONTALACCU bit field of the DSS0\_VID\_ACCUH\_0 and DSS0\_VID\_ACCUH\_1 (for ARGB and Y), and DSS0\_VID\_ACCUH2\_0 and DSS0\_VID\_ACCUH2\_1 (for CbCr) registers. The accumulator value indicates on which phase the horizontal filtering starts. The register DSS0\_VID\_ACCUH\_0 is used for progressive output, while for interlace output both DSS0\_VID\_ACCUH\_0 and DSS0\_VID\_ACCUH\_1 registers are used. The DSS0\_VID\_ACCUH2\_0 and DSS0\_VID\_ACCUH2\_1 are used in the same manner for progressive or interlace output.

Table 12-553 lists the scaler vertical and horizontal accumulator values and phases.

**Table 12-553. DISPC VID Scaler Vertical and Horizontal Accumulator Phases**

Accumulator Value (MSB bits)	Phases f
0	0
256 or -3840	1
512 or -3584	2
768 or -3328	3
1024 or -3072	4
1280 or -2816	5
1536 or -2560	6
1792 or -2304	7
2048 or -2048	8
2304 or -1792	9
2560 or -1536	10
2816 or -1280	11
3072 or -1024	12
3328 or -768	13
3584 or -512	14
3840 or -256	15

- Vertical upsampling and downsampling central coefficients:
  - For ARGB and Y, the vertical upsampling and downsampling central coefficients are defined in the DSS0\_VID\_FIR\_COEF\_V0\_0 to DSS0\_VID\_FIR\_COEF\_V0\_8 registers. There are 9 registers for the 16 phases with 1 coefficient for each of them. Symetrical implementation is used, so only 9 coefficients are used. Each register contains one 10-bit unsigned coefficient (the central one).
  - Four CbCr, the vertical upsampling and downsampling central coefficients are set in DSS0\_VID\_FIR\_COEF\_V0\_C\_0 to DSS0\_VID\_FIR\_COEF\_V0\_C\_8 registers.
- Vertical upsampling and downsampling coefficients:
  - For ARGB and Y, the vertical upsampling and downsampling coefficients are defined in the DSS0\_VID\_FIR\_COEF\_V12\_0 to DSS0\_VID\_FIR\_COEF\_V12\_15 registers. There are 16 registers for the 16 phases with 2 coefficient for each of them, so a total of 32 programmable coefficients for the vertical up/down-sampling block. Each register contains two 10-bit signed coefficients.
  - Four CbCr, the vertical upsampling and downsampling coefficients are set in DSS0\_VID\_FIR\_COEF\_V12\_C\_0 to DSS0\_VID\_FIR\_COEF\_V12\_C\_15 registers.
- Horizontal upsampling and downsampling central coefficients:
  - For ARGB and Y, the horizontal upsampling and downsampling central coefficients are defined in the DSS0\_VID\_FIR\_COEF\_H0\_0 to DSS0\_VID\_FIR\_COEF\_H0\_8 registers. There are 9 registers for the 16 phases with 1 coefficient for each of them. Symetrical implementation is used, so only 9 coefficients are used. Each register contains one 10-bit unsigned coefficient (the central one).



- Four CbCr, the horizontal upsampling and downsampling central coefficients are set in DSS0\_VID\_FIR\_COEF\_H0\_C\_0 to DSS0\_VID\_FIR\_COEF\_H0\_C\_8 registers.
- Horizontal upsampling and downsampling coefficients:
  - For ARGB and Y, the horizontal upsampling and downsampling coefficients are defined in the DSS0\_VID\_FIR\_COEF\_H12\_0 to DSS0\_VID\_FIR\_COEF\_H12\_15 registers. There are 16 registers for the 16 phases with 2 coefficient for each of them, so a total of 32 programmable coefficients for the horizontal up/down-sampling block. Each register contains two 10-bit signed coefficients.
  - Four CbCr, the horizontal upsampling and downsampling coefficients are set in DSS0\_VID\_FIR\_COEF\_H12\_C\_0 to DSS0\_VID\_FIR\_COEF\_H12\_C\_15 registers.

The YUV filtering is based on the equations of the ARGB filtering. In addition to the registers used for ARGB filtering configuration, a second set of registers for filtering is used. The first set of registers is used for Y configuration (instead of ARGB configuration) and the second set of registers is used for CbCr filtering configuration. The two sets of registers can be the same when the YUV format is not converted to RGB after filtering. When the RGB conversion is required after filtering, then the chrominance needs to be re-sampled with a different filtering configuration because:

- Chrominance samples are offset to the luminance samples. That is, DSS0\_VID\_FIRH2 and DSS0\_VID\_FIRV2, and DSS0\_VID\_ACCUV2\_0/DSS0\_VID\_ACCUV2\_1 and DSS0\_VID\_ACCUH2\_0/DSS0\_VID\_ACCUH2\_1 registers need to be configured differently than DSS0\_VID\_FIRH and DSS0\_VID\_FIRV, and DSS0\_VID\_ACCUH\_0/DSS0\_VID\_ACCUH\_1 and DSS0\_VID\_ACCUV\_0/DSS0\_VID\_ACCUV\_1 registers used for the luminance filtering.
- Chrominance is sub-sampled by two horizontally only in case of YUV422, and horizontally and vertically in case of YUV420. The coefficients for the vertical chrominance re-sampling are identical to the coefficients for vertical luminance filtering in case of YUV422. The coefficients for the horizontal and vertical chrominance re-sampling are different than the ones for luminance filtering in case of YUV420.

#### 12.6.4.8.6 DISPC VID Color Space Conversion YUV to RGB

The Color Space Conversion (CSC) unit converts the video-encoded pixel values from YUV444 format into ARGB48 format (12-bit value per component A, R, G, and B, with A fixed at 0xFFFF).

In case of YUV420 or YUV422 formats, a chrominance resampling to YUV444 is performed before converting the YUV into RGB values (see [Section 12.6.4.8.4, DISPC VID Chrominance Resampling](#)). The YUV422/YUV420 to YUV444 chrominance resampling is a pre-processing to the color space conversion.

[Figure 12-580](#) and [Figure 12-581](#) show the 3 × 3 11-bit coefficients used to convert from YUV444 into ARGB48. The value of A component is fixed at 0xFFFF in the output. The coefficients are set according to the standard used to encode the pixel data in YUV color space. [Table 12-554](#) summarizes the coefficients with their respective register bit fields.

**Table 12-554. DISPC VID CSC - YUV to RGB Register Settings**

Coefficients	Register Fields
R <sub>Y</sub>	DSS0_VID_CSC_COEF0[10-0] C00
R <sub>Cr</sub>	DSS0_VID_CSC_COEF0[26-16] C01
R <sub>Cb</sub>	DSS0_VID_CSC_COEF1[10-0] C02
G <sub>Y</sub>	DSS0_VID_CSC_COEF1[26-16] C10
G <sub>Cr</sub>	DSS0_VID_CSC_COEF2[10-0] C11
G <sub>Cb</sub>	DSS0_VID_CSC_COEF2[26-16] C12
B <sub>Y</sub>	DSS0_VID_CSC_COEF3[10-0] C20
B <sub>Cr</sub>	DSS0_VID_CSC_COEF3[26-16] C21
B <sub>Cb</sub>	DSS0_VID_CSC_COEF4[10-0] C22
Y offset	DSS0_VID_CSC_COEF5[15-3] PREOFFSET1
Cr offset	DSS0_VID_CSC_COEF5[31-19] PREOFFSET2
Cb offset	DSS0_VID_CSC_COEF6[15-3] PREOFFSET3

- Limited data range conversion (video data range)

If the active range for the luminance samples (Y) is [256:3760] and for the chrominance samples (Cb and Cr) is [256:3840], the following equation in [Figure 12-580](#) (based on 11-bit coefficients) must be used to convert the YUV to RGB. To clip the values of R, G, and B output components to the full output data range [0:4095], set the DSS0\_VID\_ATTRIBUTES[11] FULLRANGE bit to 0x1.

$$\begin{bmatrix} R_{OUT} \\ G_{OUT} \\ B_{OUT} \end{bmatrix} = \frac{1}{256} * \begin{bmatrix} R_Y & R_{Cr} & R_{Cb} \\ G_Y & G_{Cr} & G_{Cb} \\ B_Y & B_{Cr} & B_{Cb} \end{bmatrix} * \begin{bmatrix} Y_{IN} - 256 \\ Cr_{IN} - 2048 \\ Cb_{IN} - 2048 \end{bmatrix}$$

dispc-005

**Figure 12-580. DISPC VID CSC YCbCr to RGB Equation (Limited Data Range), 12-Bit Outputs**

In [Figure 12-580](#), the offset values (-256, -2048, -2048) are also programmed via corresponding register bit-fields shown in [Table 12-554, DISPC VID CSC - YUV to RGB Register Settings](#).

- Full data range conversion (graphics data range)

If the active range for the luminance samples (Y) and chrominance samples (Cb and Cr) is [0:4095], the following equation in [Figure 12-581](#) (based on 11-bit coefficients) must be used to convert the YUV to RGB. To clip the values of R, G, and B output components to the full output data range [0:4095], set the DSS0\_VID\_ATTRIBUTES[11] FULLRANGE bit to 0x1.

$$\begin{bmatrix} R_{OUT} \\ G_{OUT} \\ B_{OUT} \end{bmatrix} = \frac{1}{256} * \begin{bmatrix} R_Y & R_{Cr} & R_{Cb} \\ G_Y & G_{Cr} & G_{Cb} \\ B_Y & B_{Cr} & B_{Cb} \end{bmatrix} * \begin{bmatrix} Y_{IN} \\ Cr_{IN} - 2048 \\ Cb_{IN} - 2048 \end{bmatrix}$$

dispc-006

**Figure 12-581. DISPC VID CSC YCbCr to RGB Equation (Full Data Range), 12-Bit Outputs**

In [Figure 12-580](#) and [Figure 12-581](#), the offset values (0, -256, -2048) can be programmed via the corresponding register bit-fields shown in [Table 12-554, DISPC VID CSC - YUV to RGB Register Settings](#).

### Note

In case of YUV420/YUV422-NV21 data, the coefficients for Cr and Cb must be swapped in order to correctly support YUV420/YUV422-NV21 format.

The scaling and CSC clipping is set by the same bit, DSS0\_VID\_ATTRIBUTES[11] FULLRANGE.

## Output Clipping

It is possible for the output results to be larger (overflow) than the maximum or smaller (underflow) than the minimum value the output representation supports. The Color Space Conversion module clips the overflow/underflow result to the maximum/minimum value as follows:

- FULLRANGE = 1 (Graphics range)
  - Maximum: 4095
  - Minimum : 0
- FULLRANGE = 0 (Video range)
  - Maximum: 3760 for Luma, 3840 for Chroma
  - Minimum: 256 for both Luma and Chroma

The FULLRANGE = 0 setting should only be used when the CSC logic is used to convert RGB to YCrCb and the output needs to be in the video data range.

For YCrCb to RGB conversion and other RGB to RGB processing, the FULLRANGE bit should be set to 1 to enable full data range in the output.



#### 12.6.4.8.7 DISPC VID Brightness/Contrast/Saturation/Hue Control

The brightness/contrast/saturation controls are implemented in the same Color Space Conversion (CSC) block by making adjustments to the conversion coefficients and input/output offset values as shown in [Figure 12-582](#):

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} m * A0 & m * k * B0 & m * k * C0 \\ m * A1 & m * k * B1 & m * k * C1 \\ m * A2 & m * k * B2 & m * k * C2 \end{bmatrix} \begin{bmatrix} Y + Y\_offset\_adj \\ Cr + Cr\_offset \\ Cb + Cb\_offset \end{bmatrix} + \begin{bmatrix} R\_offset \\ G\_offset \\ B\_offset \end{bmatrix}$$

dispc-101

**Figure 12-582. DISPC VID Brightness/Contrast/Saturation Equation for YUV to RGB**

In [Figure 12-582](#) the following terminology is used:

- A/B/C coefficients and the offset parameters are for YUV to RGB conversion. See [Table 12-555](#) below for coefficients and offsets mapping to register fields.
- The gain term (m) is used for contrast control. The allowed range is:  $0 < m \leq 2.0$  (where  $m=1$  means contrast bypass).
- The gain term (k) is used for saturation control. The allowed range is:  $0 < k \leq 2.0$  (where  $k=1$  means saturation bypass).
- The brightness offset (b) is added to  $Y\_offset\_adj$ . The allowed range (for 8-bit example) is:  $-128 \leq b \leq +127$  (where  $b=0$  means brightness bypass).

The hue adjustment can also be built into the above equation ([Figure 12-582](#)) to comprehend the following Cb/Cr hue angle adjustments:

$$Cb\_hue\_adj = (Cb * \cos \emptyset + Cr * \sin \emptyset)$$

$$Cr\_hue\_adj = (Cr * \cos \emptyset - Cb * \sin \emptyset), \text{ where } \emptyset \text{ is the desired hue angle (with } \emptyset=0 \text{ meaning hue adjustment bypass)}$$

The final combined matrix equation for controlling brightness/contrast/saturation/hue (BCSH) during YUV to RGB conversion is shown in [Figure 12-583](#)

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} m * A0 & m * k * [B0 * \cos(Hue) + C0 * \sin(Hue)] & m * k * [C0 * \cos(Hue) - B0 * \sin(Hue)] \\ m * A1 & m * k * [B1 * \cos(Hue) + C1 * \sin(Hue)] & m * k * [C1 * \cos(Hue) - B1 * \sin(Hue)] \\ m * A2 & m * k * [B2 * \cos(Hue) + C2 * \sin(Hue)] & m * k * [C2 * \cos(Hue) - B2 * \sin(Hue)] \end{bmatrix} \begin{bmatrix} Y + Y\_offset\_adj \\ Cr + Cr\_offset \\ Cb + Cb\_offset \end{bmatrix} + \begin{bmatrix} R\_offset \\ G\_offset \\ B\_offset \end{bmatrix}$$

dispc-102

**Figure 12-583. DISPC VID Brightness/Contrast/Saturation/Hue Equation for YUV to RGB**

Gain and offset terms in the above equation ([Figure 12-583](#)) are programmed in the hardware registers as adjusted CSC coefficients (11-bit signed) and offset values (13-bit signed) to control the brightness, contrast, saturation and hue. Mapping of the coefficients and offsets to register fields is provided in [Table 12-555](#). Outputs are clipped to keep the output values in the proper range.

**Table 12-555. DISPC VID BCSH Register Settings for YUV to RGB**

Coefficients	Register Fields
A0	DSS0_VID_CSC_COEF0[10-0] C00
B0	DSS0_VID_CSC_COEF0[26-16] C01
C0	DSS0_VID_CSC_COEF1[10-0] C02
A1	DSS0_VID_CSC_COEF1[26-16] C10
B1	DSS0_VID_CSC_COEF2[10-0] C11
C1	DSS0_VID_CSC_COEF2[26-16] C12
A2	DSS0_VID_CSC_COEF3[10-0] C20
B2	DSS0_VID_CSC_COEF3[26-16] C21
C2	DSS0_VID_CSC_COEF4[10-0] C22

**Table 12-555. DISPC VID BCSH Register Settings for YUV to RGB (continued)**

Coefficients	Register Fields
Y offset	DSS0_VID_CSC_COEF5[15-3] PREOFFSET1
Cr offset	DSS0_VID_CSC_COEF5[31-19] PREOFFSET2
Cb offset	DSS0_VID_CSC_COEF6[15-3] PREOFFSET3
R offset	DSS0_VID_CSC_COEF6[31-19] POSTOFFSET1
G offset	DSS0_VID_CSC_COEF7[15-3] POSTOFFSET2
B offset	DSS0_VID_CSC_COEF7[31-19] POSTOFFSET3

For RGB input formats, the same CSC block can be used to adjust contrast, brightness, saturation, and hue using the matrix equation from [Figure 12-584](#).

$$\begin{bmatrix} R_{out} \\ G_{out} \\ B_{out} \end{bmatrix} = \begin{bmatrix} m * A0 & m * k * [B0 * \cos(Hue) + C0 * \sin(Hue)] & m * k * [C0 * \cos(Hue) - B0 * \sin(Hue)] \\ m * A1 & m * k * [B1 * \cos(Hue) + C1 * \sin(Hue)] & m * k * [C1 * \cos(Hue) - B1 * \sin(Hue)] \\ m * A2 & m * k * [B2 * \cos(Hue) + C2 * \sin(Hue)] & m * k * [C2 * \cos(Hue) - B2 * \sin(Hue)] \end{bmatrix} \begin{bmatrix} a0 & b0 & c0 \\ a1 & b1 & c1 \\ a2 & b2 & c2 \end{bmatrix} \begin{bmatrix} R_{in} \\ G_{in} \\ B_{in} \end{bmatrix} + \begin{bmatrix} R\_offset \\ G\_offset \\ B\_offset \end{bmatrix}$$

Coefficients a/b/c are related to A/B/C (for RGB to YUV and YUV to RGB conversions) as follows:

$$\begin{bmatrix} a0 & b0 & c0 \\ a1 & b1 & c1 \\ a2 & b2 & c2 \end{bmatrix} = \begin{bmatrix} A0 & B0 & C0 \\ A1 & B1 & C1 \\ A2 & B2 & C2 \end{bmatrix}^{-1}$$

dispc-103

**Figure 12-584. DISPC VID Brightness/Contrast/Saturation/Hue Equation for RGB Input**

In [Figure 12-584](#) the following specifics apply:

- A/B/C coefficients and the RGB offset parameters mapping to register fields is provided in [Table 12-555](#).
- The gain term (m) is used for contrast control. The allowed range is:  $0 < m \leq 2.0$  (where  $m=1$  means contrast bypass).
- The gain term (k) is used for saturation control. The allowed range is:  $0 < k \leq 2.0$  (where  $k=1$  means saturation bypass).
- The brightness is controlled by the RGB offset values.
- The Hue angle is used to adjust hue. Hue  $\emptyset=0$  means hue adjustment bypass.

Outputs are clipped to keep the output values in the proper range.

#### 12.6.4.8.8 DISPC VID Luma Key Support

The video pipeline supports Luma key transparency for Blue-ray video layer composition. When enabled, Y value of each pixel will be checked against luma key range values. The range values are defined in DSS0\_VID\_LUMAKEY register fields, as follows: [11-0] LUMAKEYMIN < Y < [27-16] LUMAKEYMAX. If this condition is true, then the pixel alpha value will be forced to zero (transparent) to make the pixel transparent during the blending process in the overlay manager.

#### 12.6.4.8.9 DISPC VID Cropping Support

At the output of the video pipeline a final crop stage is added which allows further cropping of the video window to the required dimensions. The crop feature is enabled by DSS0\_VID\_ATTRIBUTES[13] CROP register field and the cropping dimensions are further set by DSS0\_VID\_CROP[...] CROPLEFT/CROPTOP/CROPBOTTOM/CROPRIGHT register fields. Cropping of up to 31 pixels/lines is supported in every direction.

The crop feature is especially useful while dealing with the compressed frame buffer using the FBDC module, as explained in [Section 12.6.4.6.11, DISPC Compressed Data Format Support](#).

#### 12.6.4.9 DISPC Write-Back Pipeline

The write-back (WB) pipeline is used to store in the system memory the capture of the overlay output or the output of one of the pipelines. The WB pipeline consists of a CSC unit, a scaler unit, and an RGB truncation logic. The format from the overlay managers is always ARGB48. The format from the output of the VID pipeline

can be YUV422, YUV420 or ARGB48. Because the overlay works on ARGB48 format and the video accelerator works on YUV format, the color space conversion from RGB to YUV is used to directly output to memory the format that can be encoded with no extra processing.

The write-back pipeline can be connected either to the VID pipeline output or to the output of one of the overlay managers, either in M2M (memory-to-memory) or Capture mode (see [Section 12.6.4.6.3, DISPC Write DMA Buffer](#)), by configuring the DSS0\_COMMON\_DISPC\_CONNECTIONS[20-16] WB\_CONN register field.

- In M2M mode, the pipeline or overlay manager connected to WB must not be connected to any VP.
- In Capture mode, the pipeline or overlay manager connected to WB must be connected to at least one VP.

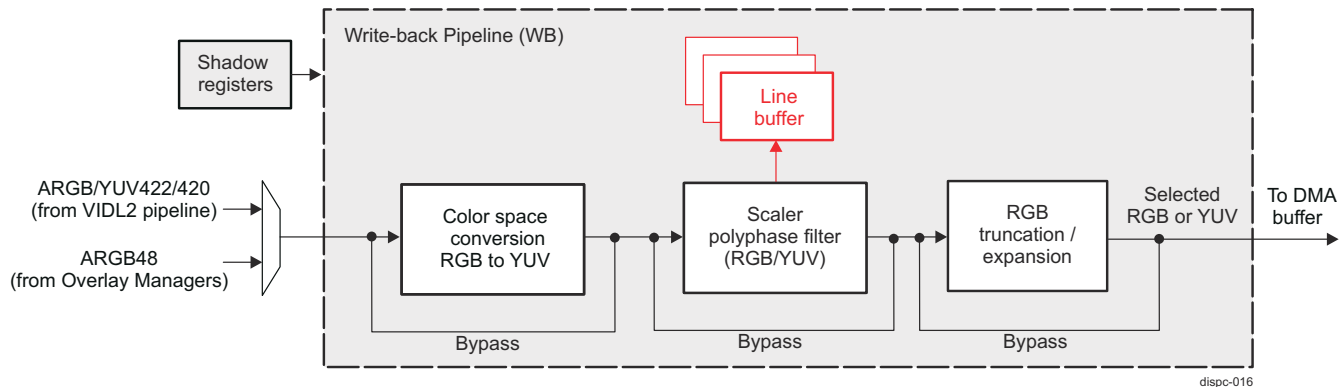
The capture frame rate can be set in the DSS0\_WB\_ATTRIBUTES[26:24] CAPTUREMODE bit field.

The write-back pipeline also supports input source cropping to allow a sub-frame capture.

The ARGB48 format is truncated (LSB drop) or expanded to match the output color depth formats. No dithering on the output is supported. When there is no alpha field mentioned by x in the pixel format description, 0's shall be used. For example, for RGB12 pixel format the upper 4 bits are 0's since RGB value is only 12 bits inside a 16-bit container.

The WB pipeline is enabled by setting the DSS0\_WB\_ATTRIBUTES[0] ENABLE bit to 0x1.

[Figure 12-585](#) shows the WB pipeline.



**Figure 12-585. DISPC Write-Back Pipeline**

#### 12.6.4.9.1 DISPC WB Color Space Conversion RGB to YUV

The WB CSC unit converts the encoded pixel values from RGB48 into YUV444 format. For YUV420 or YUV422 as output formats, a chrominance subsampling (YUV444 to YUV422, and YUV422 to YUV420) is performed in the WB scaler block after the CSC operation.

[Figure 12-586](#) and [Figure 12-587](#) show the  $3 \times 3$  11-bit coefficients used for the CSC operation. The user sets the coefficients according to the standard used to encode the pixel data in YUV color space. [Table 12-556](#) lists the coefficients with their respective bit fields, together with the post offsets (that are 13-bit signed integer numbers).

**Table 12-556. DISPC WB CSC - RGB to YUV Register Settings**

Coefficients	Register Bit Fields
$Y_R$	DSS0_WB_CSC_COEF0[10:0] C00
$Y_G$	DSS0_WB_CSC_COEF0[26:16] C01
$Y_B$	DSS0_WB_CSC_COEF1[10:0] C02
$Cr_R$	DSS0_WB_CSC_COEF1[26:16] C10
$Cr_G$	DSS0_WB_CSC_COEF2[10:0] C11
$Cr_B$	DSS0_WB_CSC_COEF2[26:16] C12
$Cb_R$	DSS0_WB_CSC_COEF3[10:0] C20

**Table 12-556. DISPC WB CSC - RGB to YUV Register Settings (continued)**

Coefficients	Register Bit Fields
Cb <sub>G</sub>	DSS0_WB_CSC_COEF3[26:16] C21
Cb <sub>B</sub>	DSS0_WB_CSC_COEF4[10:0] C22
R <sub>POSTOFFSET</sub>	DSS0_WB_CSC_COEF6[31-19] POSTOFFSET1
G <sub>POSTOFFSET</sub>	DSS0_WB_CSC_COEF7[15-3] POSTOFFSET2
B <sub>POSTOFFSET</sub>	DSS0_WB_CSC_COEF7[31-19] POSTOFFSET3

**Limited data range conversion (video data range)**

If the active range for the 12-bit output luminance samples (Y) is to be [256:3760] and for the output chrominance samples (Cb and Cr) is to be [256:3840], then [Figure 12-586](#) gives the equation (based on 11-bit coefficients) to convert the RGB to YUV. The range selection is done by setting the DSS0\_WB\_ATTRIBUTES[11] FULLRANGE bit to 0x0. The [256, 2048, 2048] offset values are programmable through the registers from [Table 12-556](#).

$$\begin{bmatrix} Y_{OUT} \\ Cb_{OUT} \\ Cr_{OUT} \end{bmatrix} = \frac{1}{256} * \begin{bmatrix} Y_R & Y_G & Y_B \\ Cb_R & Cb_G & Cb_B \\ Cr_R & Cr_G & Cr_B \end{bmatrix} * \begin{bmatrix} R_{IN} \\ G_{IN} \\ B_{IN} \end{bmatrix} + \begin{bmatrix} 256 \\ 2048 \\ 2048 \end{bmatrix}$$

dispc-017

**Figure 12-586. DISPC WB CSC RGB to YCbCr Equation (Limited Data Range), 12-bit Outputs****Full data range conversion (graphics data range)**

If the active range for the output luminance samples (Y) and or the chrominance samples (Cb and Cr) is [0:4095], then [Figure 12-587](#) gives the equation (based on 11-bit coefficients) to convert the RGB to YUV. The [0, 2048, 2048] offset values are programmable through the registers from [Table 12-556](#).

$$\begin{bmatrix} Y_{OUT} \\ Cb_{OUT} \\ Cr_{OUT} \end{bmatrix} = \frac{1}{256} * \begin{bmatrix} Y_R & Y_G & Y_B \\ Cb_R & Cb_G & Cb_B \\ Cr_R & Cr_G & Cr_B \end{bmatrix} * \begin{bmatrix} R_{IN} \\ G_{IN} \\ B_{IN} \end{bmatrix} + \begin{bmatrix} 0 \\ 2048 \\ 2048 \end{bmatrix}$$

dispc-018

**Figure 12-587. DISPC WB CSC RGB to YCbCr Equation (Full Data Range), 12-bit Outputs****12.6.4.9.2 DISPC WB Scaler Unit**

The functional aspect of the WB pipeline scaler unit is identical to the VID pipeline scaler unit (see [Section 12.6.4.8.5, DISPC VID Scaler Unit](#)), except that the WB scaler is 8-bit only and takes only the 8 most significant bits of the data fed through. The WB scaler, similarly to the one in the video pipeline, can perform scaling in the native ARGB, YUV422, or YUV420 formats. The support for scaling in YUV formats allows a YUV source to be scaled through video and write-back scalers without chroma up and downsampling, if the output is to remain in the same format. In addition, the scaling capability of VID and WB pipelines can be combined to double the maximum rescaling factor (applicable for memory-to-memory operations only).

The downscaling capability is further reduced, if the pipeline is doing a format conversion where downsampling is inherent. When doing RGB/YUV422-to-YUV420 conversion in WB pipeline, the downsampling capability of WB scaler is reduced to:

- If 5 taps is selected, then the downsampling is limited to 0.5 in RGB
- If 3 taps is selected, then no down-scaling is available

The write-back window attributes of the WB pipeline can be configured in the following register bit-fields:

- Source data format for write-back (input to the scaler unit) in DSS0\_WB\_ATTRIBUTES[6-1] FORMAT bit-field

- Start position (offset) of the window on the overlay which WB will capture in DSS0\_WB\_POSITION[29-16] POSY and [13-0] POSX register field. Only applicable when the WB pipeline is operating in capture\_mode.
- Write-back picture width in system memory (frame buffer) in DSS0\_WB\_PICTURE\_SIZE[13-0] MEMSIZEX bit-field. The window width is from 1 up to 4096 pixels. All the integer values in the range [1:4096] are allowed in the case of RGB16 and RGB24 video data. In the case of YUV422 and UYVY422 formats the width has to be a multiple of 2 pixels.
- Write-back picture height in system memory (frame buffer) in DSS0\_WB\_PICTURE\_SIZE[29-16] MEMSIZEY bit-field. The window width is from 1 up to 4096 pixels. All the integer values in the range [1:4096] are allowed. In the case of YUV420 the height has to be a multiple of 2 lines.
- Write-back window width at pipeline input (before resizing) in DSS0\_WB\_SIZE[13-0] SIZEX bit-field. The window width is from 1 up to 4096 pixels. The width of the image sent to the write back pipe has to be a multiple of 2 pixels, if its format (input) is YUV420/YUV422.
- Write-back window height at pipeline input (before resizing) in DSS0\_WB\_SIZE[29-16] SIZEY bit-field. The window height is from 1 up to 4096 pixels. The width of the image sent to the write back pipe has to be a multiple of 2 lines, if its format (input) is YUV420.

### Note

The WB pipeline supports input source cropping to allow a sub-frame capture by specifying the offset position and the size of the sub-frame. With the offset position parameters (POSY, POSX) set to (0,0) and the size parameters (SIZEY, SIZEX) set to the full size, no clipping should be applied. The software must make sure that the clipping parameters are set up correctly to avoid going over the WB input frame boundaries. Otherwise, the pipeline may lock up.

**Table 12-557** list the register fields in the function of the coefficients for the VID horizontal scaler in the DSS0\_WB\_FIR\_COEF\_H0\_0 to DSS0\_WB\_FIR\_COEF\_H0\_8, and DSS0\_WB\_FIR\_COEF\_H12\_0 to DSS0\_WB\_FIR\_COEF\_H12\_15 registers.

**Table 12-557. DISPC Register Fields Associated to Coefficients for ARGB and Y Configuration in WB Horizontal Scaler**

Phases	Ch(2)	Ch(1)	Ch(0)	Ch(-1)	Ch(-2)
	Signed coefficient [29-20] FIRHC2 bitfield	Signed coefficient [19-10] FIRHC1 bitfield	Unsigned central coefficient [9-0] FIRHC0 bitfield	Signed coefficient [19-10] FIRHC1 bitfield	Signed coefficient [29-20] FIRHC2 bitfield
0	DSS0_WB_FIR_COEF_F_H12_0	DSS0_WB_FIR_COEF_F_H12_0	DSS0_WB_FIR_COEF_F_H0_0	DSS0_WB_FIR_COEF_F_H12_0	DSS0_WB_FIR_COEF_F_H12_0
1	DSS0_WB_FIR_COEF_F_H12_1	DSS0_WB_FIR_COEF_F_H12_1	DSS0_WB_FIR_COEF_F_H0_1	DSS0_WB_FIR_COEF_F_H12_15	DSS0_WB_FIR_COEF_F_H12_15
2	DSS0_WB_FIR_COEF_F_H12_2	DSS0_WB_FIR_COEF_F_H12_2	DSS0_WB_FIR_COEF_F_H0_2	DSS0_WB_FIR_COEF_F_H12_14	DSS0_WB_FIR_COEF_F_H12_14
3	DSS0_WB_FIR_COEF_F_H12_3	DSS0_WB_FIR_COEF_F_H12_3	DSS0_WB_FIR_COEF_F_H0_3	DSS0_WB_FIR_COEF_F_H12_13	DSS0_WB_FIR_COEF_F_H12_13
4	DSS0_WB_FIR_COEF_F_H12_4	DSS0_WB_FIR_COEF_F_H12_4	DSS0_WB_FIR_COEF_F_H0_4	DSS0_WB_FIR_COEF_F_H12_12	DSS0_WB_FIR_COEF_F_H12_12
5	DSS0_WB_FIR_COEF_F_H12_5	DSS0_WB_FIR_COEF_F_H12_5	DSS0_WB_FIR_COEF_F_H0_5	DSS0_WB_FIR_COEF_F_H12_11	DSS0_WB_FIR_COEF_F_H12_11
6	DSS0_WB_FIR_COEF_F_H12_6	DSS0_WB_FIR_COEF_F_H12_6	DSS0_WB_FIR_COEF_F_H0_6	DSS0_WB_FIR_COEF_F_H12_10	DSS0_WB_FIR_COEF_F_H12_10
7	DSS0_WB_FIR_COEF_F_H12_7	DSS0_WB_FIR_COEF_F_H12_7	DSS0_WB_FIR_COEF_F_H0_7	DSS0_WB_FIR_COEF_F_H12_9	DSS0_WB_FIR_COEF_F_H12_9
8	DSS0_WB_FIR_COEF_F_H12_8	DSS0_WB_FIR_COEF_F_H12_8	DSS0_WB_FIR_COEF_F_H0_8	DSS0_WB_FIR_COEF_F_H12_8	DSS0_WB_FIR_COEF_F_H12_8
9	DSS0_WB_FIR_COEF_F_H12_9	DSS0_WB_FIR_COEF_F_H12_9	DSS0_WB_FIR_COEF_F_H0_7	DSS0_WB_FIR_COEF_F_H12_7	DSS0_WB_FIR_COEF_F_H12_7

**Table 12-557. DISPC Register Fields Associated to Coefficients for ARGB and Y Configuration in WB Horizontal Scaler (continued)**

Phases	Ch(2)	Ch(1)	Ch(0)	Ch(-1)	Ch(-2)
10	DSS0_WB_FIR_COEF_F_H12_10	DSS0_WB_FIR_COEF_F_H12_10	DSS0_WB_FIR_COEF_F_H0_6	DSS0_WB_FIR_COEF_F_H12_6	DSS0_WB_FIR_COEF_F_H12_6
11	DSS0_WB_FIR_COEF_F_H12_11	DSS0_WB_FIR_COEF_F_H12_11	DSS0_WB_FIR_COEF_F_H0_5	DSS0_WB_FIR_COEF_F_H12_5	DSS0_WB_FIR_COEF_F_H12_5
12	DSS0_WB_FIR_COEF_F_H12_12	DSS0_WB_FIR_COEF_F_H12_12	DSS0_WB_FIR_COEF_F_H0_4	DSS0_WB_FIR_COEF_F_H12_4	DSS0_WB_FIR_COEF_F_H12_4
13	DSS0_WB_FIR_COEF_F_H12_13	DSS0_WB_FIR_COEF_F_H12_13	DSS0_WB_FIR_COEF_F_H0_3	DSS0_WB_FIR_COEF_F_H12_3	DSS0_WB_FIR_COEF_F_H12_3
14	DSS0_WB_FIR_COEF_F_H12_14	DSS0_WB_FIR_COEF_F_H12_14	DSS0_WB_FIR_COEF_F_H0_2	DSS0_WB_FIR_COEF_F_H12_2	DSS0_WB_FIR_COEF_F_H12_2
15	DSS0_WB_FIR_COEF_F_H12_15	DSS0_WB_FIR_COEF_F_H12_15	DSS0_WB_FIR_COEF_F_H0_1	DSS0_WB_FIR_COEF_F_H12_1	DSS0_WB_FIR_COEF_F_H12_1

### Note

In [Table 12-557](#), the cells without color are duplicated from the grey cells.

Similar table approach applies to the vertical scaler (registers DSS0\_WB\_FIR\_COEF\_V0\_0 to DSS0\_WB\_FIR\_COEF\_V0\_8, and DSS0\_WB\_FIR\_COEF\_V12\_0 to DSS0\_WB\_FIR\_COEF\_V12\_15 are used).

Similar table approach applies to the coefficients for Cb/Cr filtering in case of YUV format (registers DSS0\_WB\_FIR\_COEF\_H0\_C\_0 to DSS0\_WB\_FIR\_COEF\_H0\_C\_8, and DSS0\_WB\_FIR\_COEF\_H12\_C\_0 to DSS0\_WB\_FIR\_COEF\_H12\_C\_15, and DSS0\_WB\_FIR\_COEF\_V0\_C\_0 to DSS0\_WB\_FIR\_COEF\_V0\_C\_8 and DSS0\_WB\_FIR\_COEF\_V12\_C\_0 to DSS0\_WB\_FIR\_COEF\_V12\_C\_15 are used).

The WB scaler unit vertical and/or horizontal sampling is selected by configuring the DSS0\_WB\_ATTRIBUTES[8-7] RESIZEENABLE register bit field.

Prior to enabling the WB scaler a valid configuration has to be set by the user.

In case of capturing data of one of the output channels, the corresponding DSS0\_VP\_CONTROL[5] GOBIT bit has to be set to update the configuration depending to which VP output the write-back pipeline is associated with. Refers also to [Section 12.6.4.11.12, DISPC Shadow Mechanism for Registers](#) when write back channel is active. The SW has to wait before setting the GOBIT bit that the HW has reset the same bit. The DSS0\_WB\_ATTRIBUTES[0] ENABLE bit can be set to update those registers, if it has been previously disabled.

The following register fields define the configuration of the video up/downsampling block in the WB pipeline:

- Vertical up/downsampling increment value in the[23-0] FIRVINC bit field of DSS0\_WB\_FIRV (for ARGB and Y) and DSS0\_WB\_FIRV2 (for CbCr) registers. Software calculates the value using the following equation:

$$FIRVINC = 2^{21} * \left( \frac{SIZEY+1}{MEMSIZEY+1} \right) \quad \text{dispc-066} \quad (25)$$

- Horizontal up/downsampling increment value in the [23-0] FIRHINC bit field of the DSS0\_WB\_FIRH (for ARGB and Y) and DSS0\_WB\_FIRH2 (for CbCr) registers. Software calculates the value using the following equation:

$$FIRHINC = 2^{21} * \left( \frac{SIZEX+1}{MEMSIZEX+1} \right) \quad \text{dispc-067} \quad (26)$$



- Vertical up/downsampling accumulator value in the [23-0] VERTICALACCU bit field of the DSS0\_WB\_ACCUV\_0 and DSS0\_WB\_ACCUV\_1 (for ARGB and Y), and DSS0\_WB\_ACCUV2\_0 and DSS0\_WB\_ACCUV2\_1 (for CbCr) registers. The accumulator value indicates on which phase the vertical filtering starts.
- Vertical upsampling and downsampling line buffer configuration via the DSS0\_WB\_ATTRIBUTES[21] VERTICALTAPS bit. The default value at reset time is 0x0 (3-tap configuration is used). If the bit field is reset, the 3-tap configuration is used.
- Horizontal upsampling and downsampling accumulator value in the [23-0] HORIZONTALACCU bit field of the DSS0\_WB\_ACCUH\_0 and DSS0\_WB\_ACCUH\_1 (for ARGB and Y), and DSS0\_WB\_ACCUH2\_0 and DSS0\_WB\_ACCUH2\_1 (for CbCr) registers. The accumulator value indicates on which phase the horizontal filtering starts.

Table 12-558 lists the DISPC vertical and horizontal accumulator values and phases. Other accumulator values are also supported. The HW determines the nearest phase value in order to load the corresponding one.

**Table 12-558. DISPC WB Scaler Vertical and Horizontal Accumulator Phases**

Accumulator Value (MSB bits)	Phases f
0	0
256 or -3840	1
512 or -3584	2
768 or -3328	3
1024 or -3072	4
1280 or -2816	5
1536 or -2560	6
1792 or -2304	7
2048 or -2048	8
2304 or -1792	9
2560 or -1536	10
2816 or -1280	11
3072 or -1024	12
3328 or -768	13
3584 or -512	14
3840 or -256	15

- Vertical upsampling and downsampling central coefficients:
  - For ARGB and Y, the vertical upsampling and downsampling central coefficients are defined in the DSS0\_WB\_FIR\_COEF\_V0\_0 to DSS0\_WB\_FIR\_COEF\_V0\_8 registers. There are 9 registers for the 16 phases with 1 coefficient for each of them. Symetrical implementation is used, so only 9 coefficients are used. Each register contains one 10-bit unsigned coefficient (the central one).
  - Four CbCr, the vertical upsampling and downsampling central coefficients are set in DSS0\_WB\_FIR\_COEF\_V0\_C\_0 to DSS0\_WB\_FIR\_COEF\_V0\_C\_8 registers.
- Vertical upsampling and downsampling coefficients:
  - For ARGB and Y, the vertical upsampling and downsampling coefficients are defined in the DSS0\_WB\_FIR\_COEF\_V12\_0 to DSS0\_WB\_FIR\_COEF\_V12\_15 registers. There are 16 registers for the 16 phases with 2 coefficient for each of them, so a total of 32 programmable coefficients for the vertical up/down-sampling block. Each register contains two 10-bit signed coefficients.
  - Four CbCr, the vertical upsampling and downsampling coefficients are set in DSS0\_WB\_FIR\_COEF\_V12\_C\_0 to DSS0\_WB\_FIR\_COEF\_V12\_C\_15 registers.
- Horizontal upsampling and downsampling central coefficients:
  - For ARGB and Y, the horizontal upsampling and downsampling central coefficients are defined in the DSS0\_WB\_FIR\_COEF\_H0\_0 to DSS0\_WB\_FIR\_COEF\_H0\_8 registers. There are 9 registers for the 16

phases with 1 coefficient for each of them. Symmetrical implementation is used, so only 9 coefficients are used. Each register contains one 10-bit unsigned coefficient (the central one).

- Four CbCr, the horizontal upsampling and downsampling central coefficients are set in DSS0\_WB\_FIR\_COEF\_H0\_C\_0 to DSS0\_WB\_FIR\_COEF\_H0\_C\_8 registers.
- Horizontal upsampling and downsampling coefficients:
  - For ARGB and Y, the horizontal upsampling and downsampling coefficients are defined in the DSS0\_WB\_FIR\_COEF\_H12\_0 to DSS0\_WB\_FIR\_COEF\_H12\_15 registers. There are 16 registers for the 16 phases with 2 coefficient for each of them, so a total of 32 programmable coefficients for the horizontal up/down-sampling block. Each register contains two 10-bit signed coefficients.
  - Four CbCr, the horizontal upsampling and downsampling coefficients are set in DSS0\_WB\_FIR\_COEF\_H12\_C\_0 to DSS0\_WB\_FIR\_COEF\_H12\_C\_15 registers.

The YUV filtering is based on the equations of the ARGB filtering. In addition to the registers used for ARGB filtering configuration, a second set of registers for filtering is used. The first set of registers is used for Y configuration (instead of ARGB configuration) and the second set of registers is used for CbCr filtering configuration. The two sets of registers can be the same when the YUV format is not converted to RGB after filtering. When the RGB conversion is required after filtering, then the chrominance needs to be re-sampled with a different filtering configuration because:

- Chrominance samples are offset to the luminance samples. That is, DSS0\_WB\_FIRH2 and DSS0\_WB\_FIRV2, and DSS0\_WB\_ACCUV2\_0/DSS0\_WB\_ACCUV2\_1 and DSS0\_WB\_ACCUH2\_0/DSS0\_WB\_ACCUH2\_1 registers need to be configured differently than DSS0\_WB\_FIRH and DSS0\_WB\_FIRV, and DSS0\_WB\_ACCUH\_0/DSS0\_WB\_ACCUH\_1 and DSS0\_WB\_ACCUV\_0/DSS0\_WB\_ACCUV\_1 registers used for the luminance filtering.
- Chrominance is sub-sampled by two horizontally only in case of YUV422, and horizontally and vertically in case of YUV420. The coefficients for the vertical chrominance re-sampling are identical to the coefficients for vertical luminance filtering in case of YUV422. The coefficients for the horizontal and vertical chrominance re-sampling are different than the ones for luminance filtering in case of YUV420.

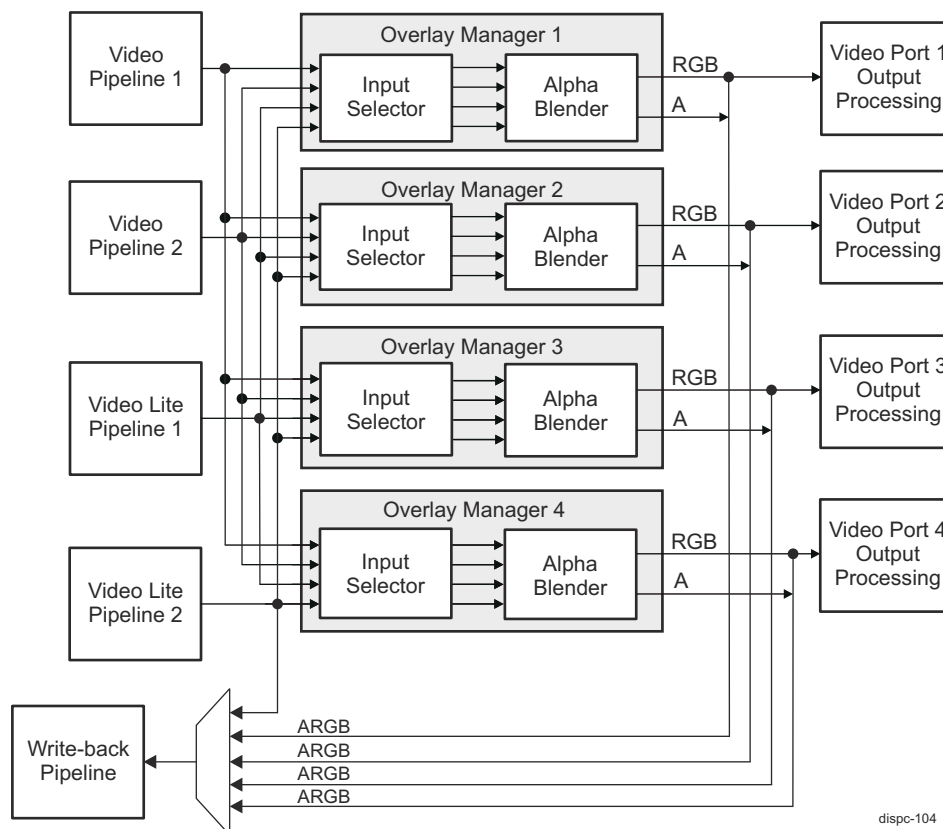
#### 12.6.4.10 DISPC Overlay Manager

The overlay manager (OVR) is responsible for compositing selected input layers together to generate the final display output frame. The DISPC implements four identical overlay managers, one for each display output (see [Figure 12-547, DISPC Architecture Overview](#)). Any of these overlay managers can be used to source also the Write-back pipeline.

- OVR1 is connected to Video Port 1 (VP1).
- OVR2 is connected to Video Port 2 (VP2).
- OVR3 is connected to Video Port 3 (VP3).
- OVR4 is connected to Video Port 4 (VP4).

Each OVR is preceded by a programmable input pipeline selector which re-maps the output of the selected input pipelines according to the overlay manager Z-order configuration.





**Figure 12-588. DISPC Overlay Manager and Input Selector**

#### Note

The DISPC overlay managers, OVR1 through OVR4, will be commonly referred to as *overlay manager* (OVR) in the following sections.

#### 12.6.4.10.1 DISPC Overlay Input Selector

The overlay input selector before the overlay manager is a n-input to n-output crossbar switch. As such, any input can be connected to any output in the selector. The selection is based on the Z-order layer selection in the overlay configuration through the following register fields:

- DSS0\_OVR\_ATTRIBUTES\_0[4-1] CHANNELIN field for layer 0, Z-order 0
- DSS0\_OVR\_ATTRIBUTES\_1[4-1] CHANNELIN field for layer 1, Z-order 1
- DSS0\_OVR\_ATTRIBUTES\_2[4-1] CHANNELIN field for layer 2, Z-order 2
- DSS0\_OVR\_ATTRIBUTES\_3[4-1] CHANNELIN field for layer 3, Z-order 3
- DSS0\_OVR\_ATTRIBUTES\_4[4-1] CHANNELIN field for layer 4, Z-order 4

The z-order determines the order in which the selected layers are blended together to generate the final output by the overlay manager:

- The lowest enabled order input is the bottom-most layer, just above the background color
- The highest enabled order input is the top-most layer

The OVR input selector also passes the following attributes from the selected input pipeline to the corresponding selector output port (*zsel*):

- Source pipeline size attributes:
  - Number of pixels per line (from DSS0\_VID\_SIZE[13-0] SIZEX register field)
  - Number of lines (from DSS0\_VID\_SIZE[29-16] SIZEY register field)

- Source pipeline global blending level (from DSS0\_VID\_GLOBAL\_ALPHA register)

Then, the overlay manager uses the above listed attributes as input layer attributes along with POSX and POSY overlay manager configuration parameters. The POSX and POSY can be configured in the [13-0] POSX and [29-16] POSY fields of the following registers:

- For layer 0, Z-order 0, in DSS0\_OVR\_ATTRIBUTES2\_0 register
- For layer 1, Z-order 1, in DSS0\_OVR\_ATTRIBUTES2\_1 register
- For layer 2, Z-order 2, in DSS0\_OVR\_ATTRIBUTES2\_2 register
- For layer 3, Z-order 3, in DSS0\_OVR\_ATTRIBUTES2\_3 register
- For layer 4, Z-order 4, in DSS0\_OVR\_ATTRIBUTES2\_4 register

#### Note

The output of an input pipeline can be mapped to more than one overlay manager simultaneously, if and only if the following conditions are true:

- All video port timing generators controlling the overlay managers are identically programmed (same frame width/height with same blanking parameters running at same frame rate) and clocked by the same pixel clock source, AND ...
- The pipeline output is positioned on each display output at the same frame position, OR ...
- One of the overlay managers is connected to the write-back path only.

The software is responsible for managing the pipeline assignment and setting up the video port timing generators properly in order to avoid any resource conflicts that may cause the DISPC to lock up (sync loss).

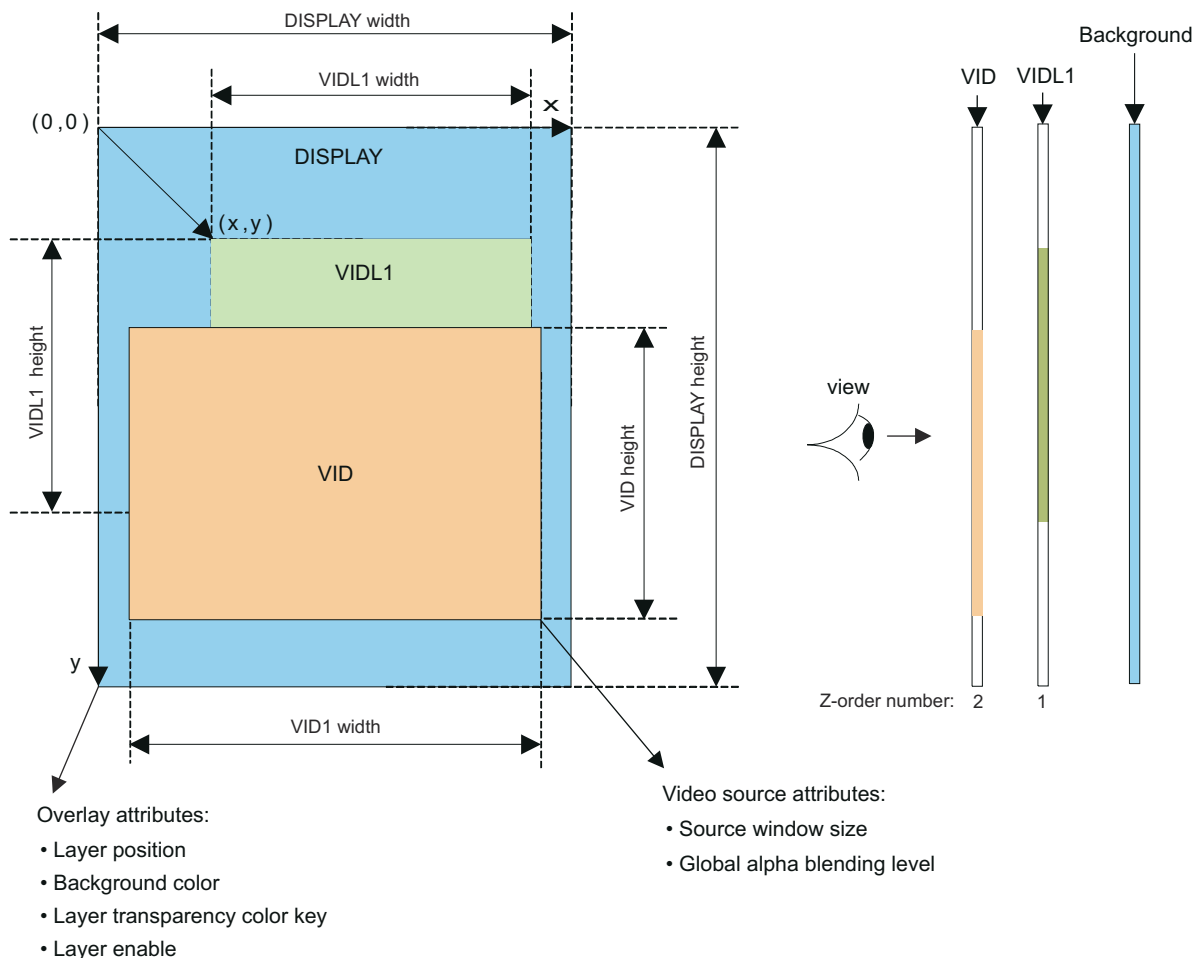
#### 12.6.4.10.2 DISPC Overlay Mechanism

The overlay mechanism consists of displaying more than one input layer (graphics and/or video layers) using:

- A priority rule based on the display Z-order (selected by configuring the overlay input selector, as described in [Section 12.6.4.10.1](#))
- Transparency color keys configuration (destination and source transparency)
- Alpha blending values (using the alpha component of each pixel or a global alpha value set by the user)
- Size and position attributes of the source window data, as described in [Section 12.6.4.10.1](#)

The overlay manager is configured by using the Z-order parameter. The Z-order value defined for each pipeline indicates the visibility order of the window on the screen. If the Z-order value of window A is lower than the Z-order of layer B, then layer A is displayed below layer B. The transparency color keys and the alpha blending factors are then used to blend the layers together (see [Section 12.6.4.10.2.1](#), *Overlay Alpha Blender*, and [Section 12.6.4.10.2.2](#), *Overlay Transparency Color Keys*).

[Figure 12-589](#) gives an simplified example of how two input pipeline frame outputs are merged together to generate the final display output.



**Figure 12-589. DISPC Overlay Example and Display Attributes**

As shown in [Figure 12-589](#), each input pipeline generates one rectangular sub-frame output in ARGB48 format. The overlay is responsible for positioning the sub-frame in the display output frame (specified by DSS0\_VP\_SIZE\_SCREEN[13-0] PPL and [29-16] LPP video port register fields), using the overlay window position parameters (specified by the [13-0] POSX and [29-16] POSY overlay fields in the DSS0\_OVR\_ATTRIBUTES2\_0 through DSS0\_OVR\_ATTRIBUTES2\_4 registers), and the window size parameters (specified by the [13-0] SIZEX and [29-16] SIZEY input pipeline fields in the DSS0\_VID\_SIZE registers). When the overlay manager does not require the input pipeline data (either because it is currently outputting the background color pixel or it is being stalled by the VP output module), the overlay manager stalls the input pipeline.

When there are no video-encoded pixels at a specific position, the programmable solid background color appears. The solid background color can be set in the DSS0\_OVR\_DEFAULT\_COLOR and DSS0\_OVR\_DEFAULT\_COLOR2 registers.

The entire pixels of the video window have to be inside the display screen. Depending on the width of the buffer to be displayed in the video layer and the position, the width must be adjusted by software to limit the right edge of the window inside the display screen. The same is also true for the video layer height in the input pipe line configuration.

#### 12.6.4.10.2.1 Overlay Alpha Blender

The overlay manager can blend as many as four input pipelines using three cascaded stages of basically the same alpha-blending logic. Alpha output generation is supported only for the write-back path.

The alpha blending value is defined by:

- The component value A when using an ARGB or RGBA pixel format (alpha in the source pixel data is converted to 8-bit alpha value in the input pipeline logic):
  - For ARGB-1555, the alpha blending is defined using a 1-bit value. It is converted into an 8-bit value by duplicating the 1-bit value (see [Table 12-559](#)).
  - For ARGB-4444, the alpha blending is defined using a 4-bit value. It is converted into an 8-bit value by duplicating the 4-bit value (see [Table 12-559](#)).
  - If the pixel format contains no alpha blending value, the pixel alpha value is considered to be 0xFF, and if alpha is equal to 0xFF, there is no multiplication.
  - For BITMAP or YUV formats, there is no alpha blending factor associated with each pixel value. Only the global alpha blending factor associated with the window displaying the BITMAP or YUV format is used.
- The global alpha blending value can be set in the DSS0\_VID\_GLOBAL\_ALPHA register of the input pipeline, and is updated in synch with the selected output channel.
- The modulated alpha blending value (that is, a total alpha blending value, when a combination of the pixel alpha blending value A and a global alpha blending ia present) is determined as: Alpha = (Pixel\_Alpha × Global\_Alpha) / 256.

[Table 12-559](#) shows the remapping and the percentage of alpha blending achieved.

**Table 12-559. DISPC Overlay Alpha Blending – ARGB**

Alpha Blending 1-Bit Value (ARGB16-1555)	Alpha Blending 4-Bit Value (ARGB16-4444)	Alpha Blending 8-Bit Value (Converted Value)	% Blending
0x0	0x0	0x00	100% (transparent)
N/A	0x1	0x11	93.33%
N/A	0x2	0x22	86.6%
N/A	...	...	...
N/A	0xE	0xEE	6.6%
0x1	0xF	0xFF	0% (opaque)

## Premultiplied Alpha and Alpha Output Generation

The source image in ARGB format may have its RGB component already premultiplied with the alpha (AR'G'B') where:

- $R' = A * R$
- $G' = A * G$
- $B' = A * B$

In that case, the processing is as follows:

- Color components of premultiplied layers are multiplied with the global alpha, if Global\_Alpha is not equal to 0.
- Color components of the composed underlying layer are multiplied with  $(1 - A * \text{Global\_Alpha})$ .

The premultiplied alpha option is accessible through the DSS0\_VID\_ATTRIBUTES[28] PREMULIPLYALPHA register bit of the input pipeline. The following settings are available:

- PREMULIPLYALPHA bit = 0: Source is not premultiplied with alpha. Full blending is done in the overlay manager.
- PREMULIPLYALPHA bit = 1: Source is premultiplied with alpha. Partial blending is done.

When the write-back channel copies back to memory the premultiplied color component, the the alpha value is computed as follows:

$$A(\text{dst}) = A(\text{src}) + (1 - A(\text{src}))A(\text{dst})$$

When the DSS0\_WB\_ATTRIBUTES[9] ALPHAENABLE register bit is cleared or when the overlay channel is not selected for write-back, the computation of the A(dst) is disabled.

### Note

There is no alpha blending between the background and the next layer (layer-0) above it.

Layer-0 cannot be pre-multiplied, if the overlay output is driving the display, since that layer (or the background color) defines the bottom most layer. In this case, layer-0 alpha is 1 always.

Layer-0 can have an alpha value other than 1 when the overlay output is only used to generate an intermediate layer composition result in a memory-to-memory operation (through the WB pipeline). But, if alpha is not 1, the resulting data will be "pre-multiplied". If a non-premultiplied data is required, the resulting overlay output must be fed through a Porter-Duff compositing engine to have the result divided by the overlay alpha output.

#### 12.6.4.10.2.2 Overlay Transparency Color Keys

The overlay manager supports two color transparency modes - source and destination. These modes cannot be active at the same time.

The source transparency is tied to the top most layer, so the layer to have this transparency applied to has to be mapped to the highest Z-ordered input (layer-4) of the overlay manager. The destination transparency is tied to the bottom most layer, so the layer to have this transparency applied to has to be mapped to the lowest Z-ordered input (layer-0) of the overlay manager.

The source and destination transparencies are defined as following:

- Source color transparency: Top layer pixel that meets the source transparency color check is made transparent.
- Destination color transparency: Top layer pixel is made transparent, only if the bottom layer pixel does not meet the destination transparency color check.

The transparency color key is enabled via the DSS0\_OVR\_CONFIG[10] TCKLCDENABLE register bit.

The minimum transparency color values are specified in the TRANSCOLORKEY field of the DSS0\_OVR\_TRANS\_COLOR\_MIN and DSS0\_OVR\_TRANS\_COLOR\_MIN2 registers.

The maximum transparency color values are specified in the TRANSCOLORKEY field of the DSS0\_OVR\_TRANS\_COLOR\_MAX and DSS0\_OVR\_TRANS\_COLOR\_MAX2 registers.

The transparency color key values defined in the registers are compared with the pixel values. If each color component (R, G, and B) is in the range defined by MIN and MAX, then there is a match. If at least one of the color component is not in the range, then there is no match.

#### • Source transparency color key

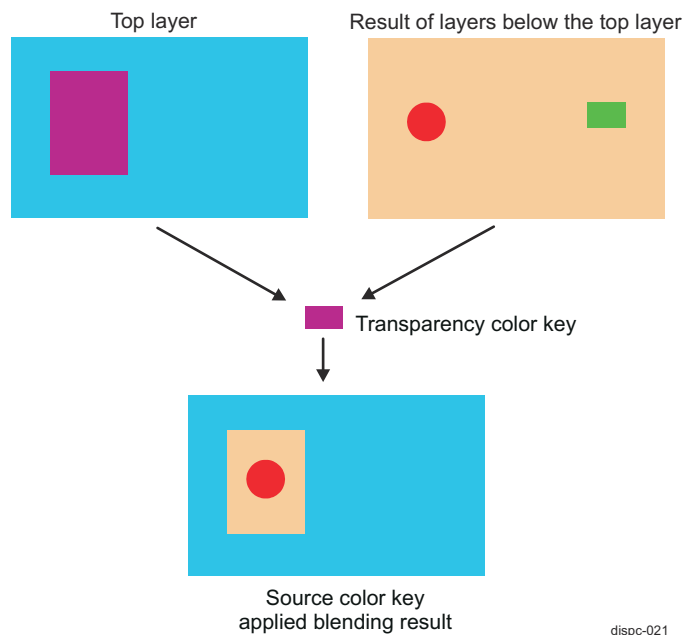
The values of the source transparency color key define the range of encoded pixel data considered as a transparent pixel. The encoded pixel values with the source color key value inside the valid range are not visible, and the encoded pixel values of the under layers or solid background color are visible.

The source transparency color key can be used with YUV and RGB formats (ARGB, RGB, RGBA, xRGB, and RGBx). In that case the A information is ignored for the comparison between the pixel value and the color key value. It is possible to use YUV formats with some care, since the comparison is between the input pixel value of the overlay manager from pipeline and the color key value. The YUV data is converted to RGB format. The user must consider the color space conversion processing in order to define the RGB color key value used for the comparison, in case the original format is YUV.

The scaler can be enabled as a preprocessor in the video pipelines. The pixel scaling processing can be considered in order to define the color key value to be used after the rescaling for the comparison between the input pixel value to the overlay manager and the color key value.

The source transparency color key mode is selected by setting the DSS0\_OVR\_CONFIG[11] TCKLCDSELECTION register bit to 0x1.

Figure 12-590 shows an example of source transparency color key usage. The pixels with the transparency color key are not displayed. Instead, pixels of the resulting layer underneath are shown.



**Figure 12-590. DISPC Overlay Source Transparency Color Key Example**

- **Destination transparency color key**

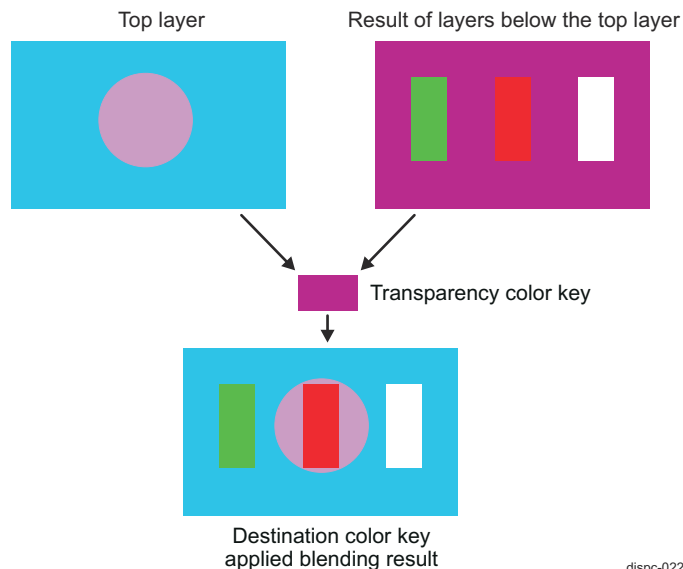
The destination transparency color key values define the range of the encoded pixels in layer-0 (bottom layer), which are not going to be displayed. That is, the encoded pixel values matching the destination color key value range are pixels not visible on the screen. Pixels at the same position in the layers above layer-0 are visible. The destination transparency color key is applicable in the layer in the background only when there is an overlap between the layer in background position and the layer just above it, otherwise, the destination transparency color key is ignored. See [Section 12.6.4.10.2, DISPC Overlay Mechanism](#), for details on layer position depending on the Z-order parameter.

The destination transparency color key can be used only with RGB formats (ARGB, RGB, xRGB, RGBA, and RGBx). In that case the A information is ignored for the comparison between the pixel value and the color key value. It is possible to use YUV formats with some care since the comparison is between the input pixel value of the overlay manager from pipeline and the color key value. The YUV data is converted to RGB format. The user must consider the color space conversion processing in order to define the RGB color key value used for the comparison in case the original format is YUV.

The scaler can be enabled as a preprocessor in the pipelines. The pixel scaling processing must be considered in order to define the color key value to be used after rescaling for the comparison between the input pixel value to the overlay manager and the color key value.

The source transparency color key mode is selected by setting the DSS0\_OVR\_CONFIG[11] TCKLCDSELECTION register bit to 0x0.

Figure 12-591 shows an example of the destination color key usage. The pixels from layer-0 (bottom layer) equal to the transparency color key are not displayed and are replaced by the pixels from layer-1 (top layer). All other layer-0 pixels, different from the transparency color key, are displayed over layer-1.



**Figure 12-591. DISPC Overlay Destination Transparency Color Key Example**

#### 12.6.4.10.3 Overlay 3D Support

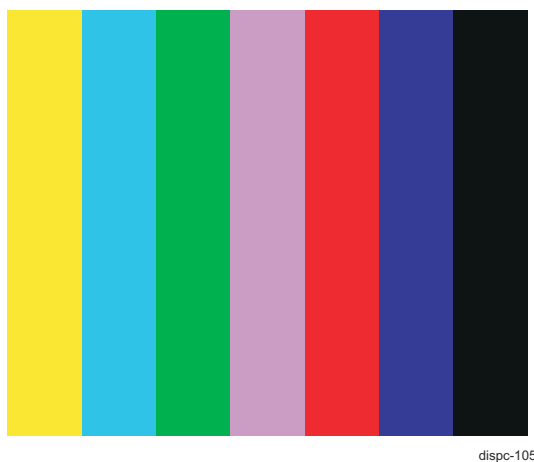
In order to support S3D (Stereoscopic 3D) by combining in hardware left and right frames into the same output frame, the following configurations (defined by HDMI 1.4b standard specification) can be used:

- Separate pipeline mode: Left/Right or Top/Bottom. One layer is used for left/top and another layer is used for right/bottom.

#### 12.6.4.10.4 Overlay Color Bar Insertion

Each overlay manager supports a simple internal color bar insertion in each display output path to enable testing of display output interface without using the frame buffer data from the memory.

The colors are: White, Yellow, Cyan, Green, Magenta, Red, Blue, Black, as shown in below. These make three primary colors, three secondary colors, white, and black.



**Figure 12-592. DISPC Overlay Internal Color Bar**

When the DSS0\_OVR\_CONFIG[1] COLORBAREN register bit is set to 1, the overlay output data is replaced by the predefined ARGB48 color bar data.

When color bar mode is used, the color space conversion matrix should be appropriately programmed, so that it can convert the full-range RGB to the desired color format.

**Table 12-560. DISPC Overlay Color Bar Table**

Color	G	B	R
White	0xFFFF	0xFFFF	0xFFFF
Yellow	0xFFFF	0	0xFFFF
Cyan	0xFFFF	0xFFFF	0
Green	0xFFFF	0	0
Magenta	0	0xFFFF	0xFFFF
Red	0	0	0xFFFF
Blue	0	0xFFFF	0
Black	0	0	0

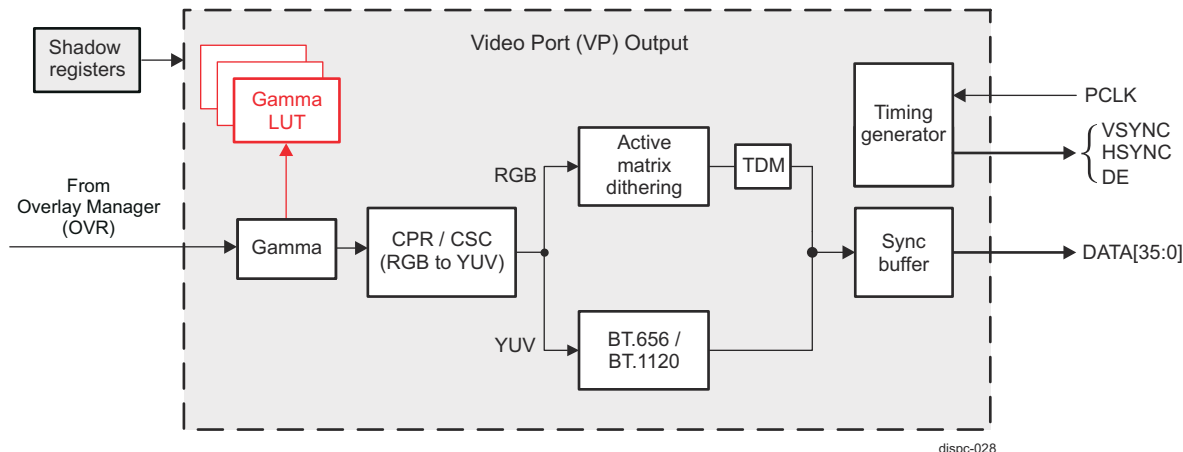
#### 12.6.4.11 DISPC Video Port Output

The DISPC implements four identical Video Port (VP) outputs:

- VP1 processes data received from Overlay Manager 1 (OVR1)
- VP2 processes data received from Overlay Manager 2 (OVR2)
- VP3 processes data received from Overlay Manager 3 (OVR3)
- VP4 processes data received from Overlay Manager 4 (OVR4)

The VP output path consists of several processing blocks (see [Figure 12-593](#)):

- Gamma correction unit
- Color phase rotation (CPR) unit, also used for RGB to YUV color space conversion (CSC)
- Active matrix dithering with TDM (multiple cycle output format)
- BT.656/BT.1120 block
- Timing generator
- Async buffer


**Figure 12-593. DISPC VP Output Architecture**

The CSC processing can be configured to occur either before or after the gamma correction in the VP output module via the DSS0\_VP\_CONFIG[26] COLORCONVPOS register bit. For RGB to YUV color space conversion, the conversion must be done after the gamma correction. For other RGB output applications, the conversion must be done before the final gamma correction.

#### Note

The DISPC video port (VP) outputs, VP1 through VP4, will be commonly referred to as *video port (VP)* in the following sections.



#### 12.6.4.11.1 DISPC VP Gamma Correction Unit

The VP supports a look up table to perform the gamma correction on the display output. The look up table consists of 3 separate 1024x10-bit memories, which are indexed by the R/G/B component data.

When performing gamma correction, the selected encoded pixel values from the video pipeline path are sent by the overlay manager to the gamma curve table. Each R/G/B component of the encoded pixel value is used as a pointer to index 1 out of 1024x30-bit gamma curve entries in the table. Each 10-bit component is replaced with the 10-bit table value corresponding to R, G, or B component. The table is loaded by software via the DSS0\_VP\_GAMMA\_TABLE\_0 through DSS0\_VP\_GAMMA\_TABLE\_15 registers. It is possible to load only part of the table.

The sequence to load the gamma table is:

1. SW writes (only writes are supported) 32-bit gamma correction values using single access, or burst access in linear increment burst mode, into the 64-byte region strating at DSS0\_VP\_GAMMA\_TABLE\_0 register address. The LSB 30 bits [29:0] are used for the value, and the MSB 1 bit [31] is used for resetting the index into the table at the end.
2. Loop to step #1, if there is a new access to the gamma table register. The software can access other registers between two accesses to the gamma table register.

Software needs to ensure that there is no visible effect when modifying the table, since it is not under hardware control.

The usage of the gamma table is activated by setting DSS0\_VP\_CONFIG[2] GAMMAENABLE register bit.

Figure 12-594 describes the format of one of the gamma curve values in the memory.

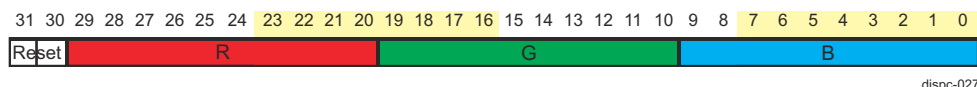


Figure 12-594. DISPC VP Output Gamma Table Write Data Format

#### 12.6.4.11.2 DISPC VP Color Phase Rotation Unit

If required, a color phase rotation (CPR) processing can be applied on the "gamma" data before the spatial/temporal dithering.

The CPR can be selected to correct the non-pure white backlight of the display panel by using a programmable matrix to convert the 36-bit RGB pixel value into a new 36-bit RGB pixel value. The matrix is programmed through a set of nine 11-bit signed coefficients. The output of the calculation is clipped to [0:4095]. The CPR is enabled by setting the DSS0\_VP\_CONFIG[15] CPR bit to 0x1.

The CPR processing is expressed by the equation shown in Figure 12-595. Table 12-561 lists all coefficients with their respective register bitfields for configuration.

$$\begin{bmatrix} R_{out} \\ G_{out} \\ B_{out} \end{bmatrix} = \frac{1}{256} * \begin{bmatrix} R_r & R_g & R_b \\ G_r & G_g & G_b \\ B_r & B_g & B_b \end{bmatrix} * \begin{bmatrix} R_{in} \\ G_{in} \\ B_{in} \end{bmatrix}$$

dispc-023

Figure 12-595. DISPC VP CPR Matrix

Table 12-561. DISPC VP CPR and CSC Coefficients with Associated Register Fields

Register Field	Color Phase Rotation	Color Space Conversion RGB to YUV
DSS0_VP_CSC_COEF0[10-0] C00	RR	YR
DSS0_VP_CSC_COEF0[26-16] C01	RG	YG

**Table 12-561. DISPC VP CPR and CSC Coefficients with Associated Register Fields (continued)**

Register Field	Color Phase Rotation	Color Space Conversion RGB to YUV
DSS0_VP_CSC_COEF1[10-0] C02	RB	YB
DSS0_VP_CSC_COEF1[26-16] C10	GR	CrR
DSS0_VP_CSC_COEF2[10-0] C11	GG	CrG
DSS0_VP_CSC_COEF2[26-16] C12	GB	CrB
DSS0_VP_CSC_COEF3[10-0] C20	BR	CbR
DSS0_VP_CSC_COEF3[26-16] C21	BG	CbG
DSS0_VP_CSC_COEF4[10-0] C22	BB	CbB
DSS0_VP_CSC_COEF6[31-19] POSTOFFSET1	-	R offset
DSS0_VP_CSC_COEF7[15-3] POSTOFFSET2	-	G offset
DSS0_VP_CSC_COEF7[31-19] POSTOFFSET3	-	B offset

### 12.6.4.11.3 DISPC VP Color Space Conversion - RGB to YUV

The RGB to YUV color space conversion (CSC) in the video port is done by using the same programmable logic used for CPR. If CPR is also required, then the process can be combined with the CSC processing by adjusting matrix coefficients. [Table 12-561](#) lists the associated coefficients with their respective register bit-fields. The CSC processing is enabled by setting the DSS0\_VP\_CONFIG[24] COLORCONVENABLE register bit.

The color space conversion can be selected in order to convert from 36-bit RGB to YUV444. The conversion matrix is programmed through a set of nine 11-bit signed coefficients. The result is clipped to [256:3760] for the luminance and [256:3840] for the chrominance, when a full-range YUV output is not desired (equivalent to clipping to [16..235] and [16..240] in 8-bit video). In this case the DSS0\_VP\_CONFIG[25] FULLRANGE register bit must be set to 0x0.

$$\begin{bmatrix} Y_{OUT} \\ Cr_{OUT} \\ Cb_{OUT} \end{bmatrix} = \frac{1}{256} * \begin{bmatrix} Y_R & Y_G & Y_B \\ Cr_R & Cr_G & Cr_B \\ Cb_R & Cb_G & Cb_B \end{bmatrix} * \begin{bmatrix} R_{IN} \\ G_{IN} \\ B_{IN} \end{bmatrix} + \begin{bmatrix} 256 \\ 2048 \\ 2048 \end{bmatrix}$$

dispc-098

**Figure 12-596. DISPC VP CSC RGB to YUV Equation (FULLRANGE=0)**

If the programmed active range for the luminance samples (Y) and chrominance samples (Cb and Cr) is [0:4095], then the values Y, Cb, and Cr are clipped to the range [0:4095]. The following equation gives the 11-bit coefficients of the RGB to YUV color space conversion, for full range (when DSS0\_VP\_CONFIG[25] FULLRANGE = 0x1).

$$\begin{bmatrix} Y_{OUT} \\ Cr_{OUT} \\ Cb_{OUT} \end{bmatrix} = \frac{1}{256} * \begin{bmatrix} Y_R & Y_G & Y_B \\ Cr_R & Cr_G & Cr_B \\ Cb_R & Cb_G & Cb_B \end{bmatrix} * \begin{bmatrix} R_{IN} \\ G_{IN} \\ B_{IN} \end{bmatrix} + \begin{bmatrix} 0 \\ 2048 \\ 2048 \end{bmatrix}$$

dispc-099

**Figure 12-597. DISPC VP CSC RGB to YUV Equation (FULLRANGE=1)**

In order to provide YUV422 data to the BT.656 / BT.1120 block, the YUV444 format is further converted to YUV422 by sub-sampling the chrominance values. The hardware does this by averaging two-by-two the

chrominance samples. The conversion from YUV444 to YUV422 is performed when the color space conversion in the VP is enabled.

#### 12.6.4.11.4 DISPC VP BT.656 and BT.1120 Modes

Each VP output can be configured in BT.656 or BT.1120 mode. The following standards are not supported in BT.656 mode:

- BT.470 (Support for conventional Analog TV Systems)
- BT.803 (The avoidance of interference generated by digital television studio equipment)
- BT.1364 (Format of ancillary data signals carried in digital component studio interfaces)
- BT.601 (Studio encoding parameters for digital TV standards for standard 4:3 and wide 16:9 aspect ratios)

Unsupported formats when BT.1120 mode is used:

- BT.1364 (Support for Ancillary Data during blanking period)
- BT.709 (Square pixel support)

BT.656/BT.1120 modes use embedded EAV/SAV syncs.

Enabling BT.656 or BT.1120 format for a VP output is done by setting DSS0\_VP\_CONFIG[20] BT656ENABLE or [21] BT1120ENABLE register bit, respectively

---

#### Note

It is not possible to enable BT.656 and BT.1120 modes simultaneously on the same VP output.

---

##### 12.6.4.11.4.1 DISPC BT Mode Blanking

During the transmission of the video signal, the portion of the stream in-between active video data segments is known as the horizontal blanking interval.

Strictly speaking this entire region is the blanking interval, but this interval also includes the EAV and SAV codes. The remaining bytes of information in a digital blanking interval are filled with values corresponding to the blanking levels of the Cb, Y and Cr signals respectively, and in accordance with the standard multiplex sequence for the stream (CbYCrY..). The blanking levels are as follows:

- Cb = 80h
- Y = 10h
- Cr = 80h.

The sequence in the BLANKING region of the data stream is therefore: 80h, 10h, 80h, 10h.....80h, 10h

For more details on setting the blanking timing values for BT.1120 and/or BT.656 mode, see [Section 12.6.4.11.8, DISPC VP Timing Generator and Display Panel Settings](#).

##### 12.6.4.11.4.2 DISPC BT Mode EAV and SAV

The End of Active Video (EAV) and Start of Active Video (SAV) parts of the stream are timing codes. Their function can be summarized as follows:

- EAV – marks the end of the active video data within the current line and therefore also the start of the subsequent line.
- SAV – heralds the start of the active video data within the current line.

These codes are embedded within the BT.656 video data stream, thereby eliminating the need for additional timing signals (HSYNC, VSYNC) to be included as part of the interface.

Both EAV and SAV codes are comprised of a sequence of four bytes ( FFh – 00h – 00h - XY). The first three bytes in the sequence constitute a fixed preamble. The fourth byte, contains information about the field being transmitted (Field 1 or Field 2 in an interlaced video signal), the state of field blanking (Vertical) and the state of line blanking (Horizontal). The bit assignment for this byte of the code is shown in [Figure 12-598](#), with the function of each bit described in [Table 12-562](#).

MSB							LSB
1	F	V	H	P3	P2	P1	P0

**Figure 12-598. DISPC BT Mode Bit-Assignment for the Fourth Byte of EAV/SAV Codes**
**Table 12-562. DISPC BT Mode Bit Function**

Bit	Symbol	Function
7	1	Always set to '1'.
6	F	Field bit. 0 – Field 1 1 – Field 2
5	V	Vertical Blanking Status bit. This bit goes High during a vertical field blanking interval, otherwise it remains Low.
4	H	Horizontal Blanking Status bit. 0 – byte is part of SAV code (i.e. stream is entering an active video data region for the current line) 1 – byte is part of EAV code (i.e. stream has entered a horizontal blanking interval - start of a new line)
3	P3	Protection bit 3
2	P2	Protection bit 2
1	P1	Protection bit 1
0	P0	Protection bit 0

The protection bits allow for detection and correction of 1-bit errors and the detection of 2-bit errors. The status of P3, P2, P1 and P0 depend on the states of bits F, V, and H. This dependency is shown in [Table 12-563](#).

**Table 12-563. DISPC BT Mode Status of Protection Bits in Function of F, V, and H**

F	V	H	P3	P2	P1	P0
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	1	1
1	0	1	1	0	1	0
1	1	0	1	1	0	0
1	1	1	0	0	0	1

#### 12.6.4.11.5 DISPC VP Spatial/Temporal Dithering

The active spatial/temporal dithering logic can be enabled to minimize the color banding when displaying the data on a LCD panel with color depth lower than 24-bit. The dithering logic is integrated after the color/gamma conversion blocks and before the TDM (Time Division Multiplexing) block. The spatial/temporal dithering algorithm is based on the (x,y) pixel position and frame rate control (the value of removed bits and the frame number). The dithering logic can process the pixels over one frame, two frames, or four frames. The number of frames is selected by setting the DSS0\_VP\_CONTROL[31-30] SPATIALTEMPORALDITHERINGFRAMES register field. In the case of a single frame, only spatial processing is applied. In case of multiple frames, both spatial and temporal processing are applied to the pixels. The number of frame is initialized before enabling the spatial/temporal dithering logic. It must be never changed by the software while the spatial/temporal logic is enabled. The spatial/temporal dithering logic is enabled by setting the DSS0\_VP\_CONTROL[7] STDITHERENABLE bit to 0x1.

### Note

- If the interface data bus is smaller than the pixel format size and the spatial/temporal dithering is not enabled, the MSBs of the pixel color components are output on the interface data bus.
- If the interface data bus is larger than the pixel format size, then by programming the pixel components replication active/inactive the MSB is replicated to the LSB of the interface data bus.

#### 12.6.4.11.6 DISPC VP Multiple Cycle Output Format (TDM)

The pixels (only RGB components) after the active matrix dithering unit are formatted on one or multiple cycles (from 1 to 3 cycles). On three cycles, two pixels can concatenate and send to the panel. The cycle format is selected through the DSS0\_VP\_CONTROL[24-23] TDMCYCLEFORMAT bit field. The number of bits for each cycle is set in the DSS0\_VP\_DATA\_CYCLE\_0 register for the first cycle, the DSS0\_VP\_DATA\_CYCLE1 register for the second cycle, and the DSS0\_VP\_DATA\_CYCLE2 register for the third cycle. The output interface data bus width, when TDM mode is enabled (DSS0\_VP\_CONTROL[20] TDMENABLE register bit = 1), can be 8, 9, 12, or 16 bits, configurable through the DSS0\_VP\_CONTROL[22-21] TDMPARALLELMODE register field.

When the TDM is disabled (DSS0\_VP\_CONTROL[20] TDMENABLE = 0), the video port output interface data bus width is configured through the DSS0\_VP\_CONTROL[10-8] DATALINES register field.

When using TDM mode, only up to 24 bits per pixel can be output on the interface. For higher color depth, only the upper bits are kept before converting each pixel into TDM output.

[Figure 12-599](#) through [Figure 12-602](#) show various examples of TDM settings in the function of pixel data formats and the interface data bus width.

24-bpp			
	1st cycle	2nd cycle	3rd cycle
Data[7]	R0[7]	G0[7]	B0[7]
Data[6]	R0[6]	G0[6]	B0[6]
Data[5]	R0[5]	G0[5]	B0[5]
Data[4]	R0[4]	G0[4]	B0[4]
Data[3]	R0[3]	G0[3]	B0[3]
Data[2]	R0[2]	G0[2]	B0[2]
Data[1]	R0[1]	G0[1]	B0[1]
Data[0]	R0[0]	G0[0]	B0[0]

CONTROL[24-23] TDMCYCLEFORMAT = 0x2  
DATA\_CYCLE0 = 0x00000008  
DATA\_CYCLE1 = 0x00000008  
DATA\_CYCLE2 = 0x00000008

18-bpp			
	1st cycle	2nd cycle	3rd cycle
Data[7]	R0[5]	G0[3]	x
Data[6]	R0[4]	G0[2]	x
Data[5]	R0[3]	G0[1]	x
Data[4]	R0[2]	G0[0]	x
Data[3]	R0[1]	B0[5]	x
Data[2]	R0[0]	B0[4]	x
Data[1]	G0[5]	B0[3]	B0[1]
Data[0]	G0[4]	B0[2]	B0[0]

CONTROL[24-23] TDMCYCLEFORMAT = 0x2  
DATA\_CYCLE0 = 0x00000008  
DATA\_CYCLE1 = 0x00000008  
DATA\_CYCLE2 = 0x00000002

16-bpp		
	1st cycle	2nd cycle
Data[7]	R0[4]	G0[2]
Data[6]	R0[3]	G0[1]
Data[5]	R0[2]	G0[0]
Data[4]	R0[1]	B0[4]
Data[3]	R0[0]	B0[3]
Data[2]	G0[5]	B0[2]
Data[1]	G0[4]	B0[1]
Data[0]	G0[3]	B0[0]

CONTROL[24-23] TDMCYCLEFORMAT = 0x1  
DATA\_CYCLE0 = 0x00000008  
DATA\_CYCLE1 = 0x00000008

12-bpp		
	1st cycle	2nd cycle
Data[7]	R0[3]	x
Data[6]	R0[2]	x
Data[5]	R0[1]	x
Data[4]	R0[0]	x
Data[3]	G0[3]	B0[3]
Data[2]	G0[2]	B0[2]
Data[1]	G0[1]	B0[1]
Data[0]	G0[0]	B0[0]

CONTROL[24-23] TDMCYCLEFORMAT = 0x1  
DATA\_CYCLE0 = 0x00000008  
DATA\_CYCLE1 = 0x00000004

dispc-057

**Figure 12-599. DISPC VP TDM 8-Bit Interface Settings**

24-bpp			
	1st cycle	2nd cycle	3rd cycle
Data[8]	R0[7]	G0[6]	x
Data[7]	R0[6]	G0[5]	x
Data[6]	R0[5]	G0[4]	x
Data[5]	R0[4]	G0[3]	B0[5]
Data[4]	R0[3]	G0[2]	B0[4]
Data[3]	R0[2]	G0[1]	B0[3]
Data[2]	R0[1]	G0[0]	B0[2]
Data[1]	R0[0]	B0[7]	B0[1]
Data[0]	G0[7]	B0[6]	B0[0]

CONTROL[24-23] TDMCYCLEFORMAT = 0x2  
DATA\_CYCLE0 = 0x00000009  
DATA\_CYCLE1 = 0x00000009  
DATA\_CYCLE2 = 0x00000006

18-bpp		
	1st cycle	2nd cycle
Data[8]	R0[5]	G0[2]
Data[7]	R0[4]	G0[1]
Data[6]	R0[3]	G0[0]
Data[5]	R0[2]	B0[5]
Data[4]	R0[1]	B0[4]
Data[3]	R0[0]	B0[3]
Data[2]	G0[5]	B0[2]
Data[1]	G0[4]	B0[1]
Data[0]	G0[3]	B0[0]

CONTROL[24-23] TDMCYCLEFORMAT = 0x1  
DATA\_CYCLE0 = 0x00000009  
DATA\_CYCLE1 = 0x00000009

16-bpp		
	1st cycle	2nd cycle
Data[8]	R0[4]	x
Data[7]	R0[3]	x
Data[6]	R0[2]	G0[1]
Data[5]	R0[1]	G0[0]
Data[4]	R0[0]	B0[4]
Data[3]	G0[5]	B0[3]
Data[2]	G0[4]	B0[2]
Data[1]	G0[3]	B0[1]
Data[0]	G0[2]	B0[0]

CONTROL[24-23] TDMCYCLEFORMAT = 0x1  
DATA\_CYCLE0 = 0x00000009  
DATA\_CYCLE1 = 0x00000007

12-bpp		
	1st cycle	2nd cycle
Data[8]	R0[3]	x
Data[7]	R0[2]	x
Data[6]	R0[1]	x
Data[5]	R0[0]	x
Data[4]	G0[3]	x
Data[3]	G0[2]	x
Data[2]	G0[1]	B0[2]
Data[1]	G0[0]	B0[1]
Data[0]	B0[3]	B0[0]

CONTROL[24-23] TDMCYCLEFORMAT = 0x1  
DATA\_CYCLE0 = 0x00000009  
DATA\_CYCLE1 = 0x00000003

dispc-058

**Figure 12-600. DISPC VP TDM 9-Bit Interface Settings**

24-bpp		
	1st cycle	2nd cycle
Data[11]	R0[7]	G0[3]
Data[10]	R0[6]	G0[2]
Data[9]	R0[5]	G0[1]
Data[8]	R0[4]	G0[0]
Data[7]	R0[3]	B0[7]
Data[6]	R0[2]	B0[6]
Data[5]	R0[1]	B0[5]
Data[4]	R0[0]	B0[4]
Data[3]	G0[7]	B0[3]
Data[2]	G0[6]	B0[2]
Data[1]	G0[5]	B0[1]
Data[0]	G0[4]	B0[0]

CONTROL[24-23] TDMCYCLEFORMAT = 0x1  
DATA\_CYCLE0 = 0x0000000C  
DATA\_CYCLE1 = 0x0000000C

18-bpp			
	1st cycle	2nd cycle	3rd cycle
Data[11]	R0[5]	B0[5]	G1[5]
Data[10]	R0[4]	B0[4]	G1[4]
Data[9]	R0[3]	B0[3]	G1[3]
Data[8]	R0[2]	B0[2]	G1[2]
Data[7]	R0[1]	B0[1]	G1[1]
Data[6]	R0[0]	B0[0]	G1[0]
Data[5]	G0[5]	R1[5]	B1[5]
Data[4]	G0[4]	R1[4]	B1[4]
Data[3]	G0[3]	R1[3]	B1[3]
Data[2]	G0[2]	R1[2]	B1[2]
Data[1]	G0[1]	R1[1]	B1[1]
Data[0]	G0[0]	R1[0]	B1[0]

CONTROL[24-23] TDMCYCLEFORMAT = 0x3  
DATA\_CYCLE0 = 0x0000000C  
DATA\_CYCLE1 = 0x00060606  
DATA\_CYCLE2 = 0x000C0000

16-bpp		
	1st cycle	2nd cycle
Data[11]	R0[4]	x
Data[10]	R0[3]	x
Data[9]	R0[2]	x
Data[8]	R0[1]	x
Data[7]	R0[0]	x
Data[6]	G0[5]	x
Data[5]	G0[4]	x
Data[4]	G0[3]	x
Data[3]	G0[2]	B0[3]
Data[2]	G0[1]	B0[2]
Data[1]	G0[0]	B0[1]
Data[0]	B0[4]	B0[0]

CONTROL[24-23] TDMCYCLEFORMAT = 0x1  
DATA\_CYCLE0 = 0x0000000C  
DATA\_CYCLE1 = 0x00000004

12-bpp	
	1st cycle
Data[11]	R0[3]
Data[10]	R0[2]
Data[9]	R0[1]
Data[8]	R0[0]
Data[7]	G0[3]
Data[6]	G0[2]
Data[5]	G0[1]
Data[4]	G0[0]
Data[3]	B0[3]
Data[2]	B0[2]
Data[1]	B0[1]
Data[0]	B0[0]

CONTROL[24-23] TDMCYCLEFORMAT = 0x0  
DATA\_CYCLE0 = 0x0000000C

dispc-059

**Figure 12-601. DISPC VP TDM 12-Bit Interface Settings**



24-bpp			
	1st cycle	2nd cycle	3rd cycle
Data[15]	R0[7]	B0[7]	G1[7]
Data[14]	R0[6]	B0[6]	G1[6]
Data[13]	R0[5]	B0[5]	G1[5]
Data[12]	R0[4]	B0[4]	G1[4]
Data[11]	R0[3]	B0[3]	G1[3]
Data[10]	R0[2]	B0[2]	G1[2]
Data[9]	R0[1]	B0[1]	G1[1]
Data[8]	R0[0]	B0[0]	G1[0]
Data[7]	G0[7]	R1[7]	B1[7]
Data[6]	G0[6]	R1[6]	B1[6]
Data[5]	G0[5]	R1[5]	B1[5]
Data[4]	G0[4]	R1[4]	B1[4]
Data[3]	G0[3]	R1[3]	B1[3]
Data[2]	G0[2]	R1[2]	B1[2]
Data[1]	G0[1]	R1[1]	B1[1]
Data[0]	G0[0]	R1[0]	B1[0]

CONTROL[24-23] TDMCYCLEFORMAT = 0x3

DATA\_CYCLE0 = 0x00000010

DATA\_CYCLE1 = 0x00080808

DATA\_CYCLE2 = 0x00100000

18-bpp		
	1st cycle	2nd cycle
Data[15]	R0[5]	x
Data[14]	R0[4]	x
Data[13]	R0[3]	x
Data[12]	R0[2]	x
Data[11]	R0[1]	x
Data[10]	R0[0]	x
Data[9]	G0[5]	x
Data[8]	G0[4]	x
Data[7]	G0[3]	X
Data[6]	G0[2]	x
Data[5]	G0[1]	x
Data[4]	G0[0]	x
Data[3]	B0[5]	x
Data[2]	B0[4]	x
Data[1]	B0[3]	B0[1]
Data[0]	B0[2]	B0[0]

CONTROL[24-23] TDMCYCLEFORMAT = 0x1

DATA\_CYCLE0 = 0x00000010

DATA\_CYCLE1 = 0x00000002

16-bpp	
	1st cycle
Data[15]	R0[4]
Data[14]	R0[3]
Data[13]	R0[2]
Data[12]	R0[1]
Data[11]	R0[0]
Data[10]	G0[5]
Data[9]	G0[4]
Data[8]	G0[3]
Data[7]	G0[2]
Data[6]	G0[1]
Data[5]	G0[0]
Data[4]	B0[4]
Data[3]	B0[3]
Data[2]	B0[2]
Data[1]	B0[1]
Data[0]	B0[0]

CONTROL[24-23] TDMCYCLEFORMAT = 0x0

DATA\_CYCLE0 = 0x00000010

12-bpp	
	1st cycle
Data[15]	x
Data[14]	x
Data[13]	x
Data[12]	x
Data[11]	R0[3]
Data[10]	R0[2]
Data[9]	R0[1]
Data[8]	R0[0]
Data[7]	G0[3]
Data[6]	G0[2]
Data[5]	G0[1]
Data[4]	G0[0]
Data[3]	B0[3]
Data[2]	B0[2]
Data[1]	B0[1]
Data[0]	B0[0]

CONTROL[24-23] TDMCYCLEFORMAT = 0x0

DATA\_CYCLE0 = 0x0000000C

dispc-060

**Figure 12-602. DISPC VP TDM 16-Bit Interface Settings**

#### 12.6.4.11.7 DISPC VP Stall Mode

The VP output can be programmed to operate in Stall Mode, where there is a stall handshake mechanism (back-pressure) with the display peripheral to which the VP is connected. No sync signals (HS, VS) are generated in Stall Mode of operation. Data and Enable (DE) signals are provided only when the stall signal is de-asserted (1'b0) by the peripheral. The Stall Mode can be enabled via the DSS0\_VP\_CONTROL[11] STALLMODE register bit.

### Note

In this SoC, the Stall Mode of operation is supported only on the VP outputs connected to the DSI peripheral module.

There are two types of data transfer, if Stall Mode is enabled, that can be selected by the DSS0\_VP\_CONTROL[12] STALLMODETYPE register bit:

- In DSI Video Mode, multiple frames are sent out on VP until the VP is disabled. A FRAMEDONE\_IRQ interrupt is generated at the end of each frame. No VSYNC\_IRQ interrupt is generated.
- In DSI Command Mode, a single frame is sent out on VP and the VP is auto disabled. A FRAMEDONE\_IRQ interrupt is generated at the end of frame. To transfer another frame in Command Mode, the user needs to re-enable the VP. No VSYNC\_IRQ interrupt is generated.

#### 12.6.4.11.8 DISPC VP Timing Generator and Display Panel Settings

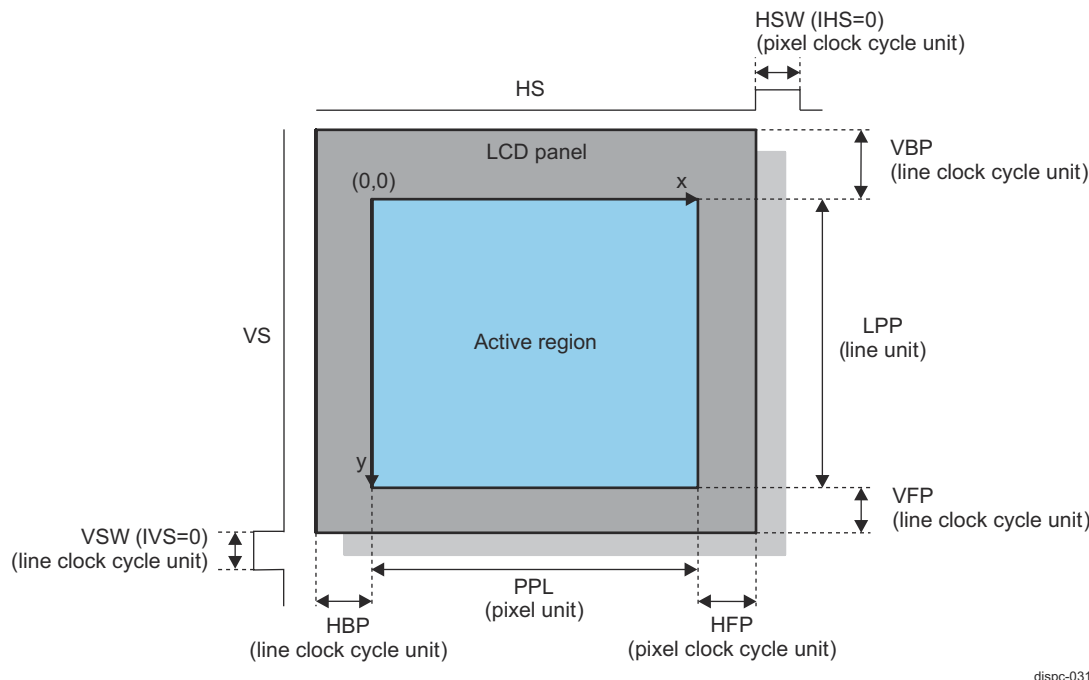
Each video port output has a dedicated timing generator supporting progressive and interlaced modes. It is clocked using the pixel clock and is configured to generate the video data and sync signals to match the desired video display standard timings.

Two-level configuration for enabling the video ports exists:

- Via the individual DSS0\_VP\_CONTROL[0] ENABLE register bit for each VP. It allows each VP to be independently controlled by two different hosts.
- Via the common DSS0\_COMMON\_DISPC\_GLOBAL\_OUTPUT\_ENABLE[2-0] VP\_ENABLE register field. It allows two or more VP timing generators to be in sync by enabling them simultaneously with a single register write.

Both ENABLE and VP\_ENABLE register fields must be set in order for a VP (timing generator) to start.

Figure 12-603 shows the timing generator display parameters.



**Figure 12-603. DISPC VP Display Timing Parameters**

The width of VP output data bus is configured in the DSS0\_VP\_CONTROL[10-8] DATALINES register field, when TDM feature is disabled. When TDM is enabled, the width of the output data bus is defined according to [Section 12.6.4.11.6, DISPC VP Multiple Cycle Output Format \(TDM\)](#).

The size of the display panel is defined by:

- Number of lines per frame, DSS0\_VP\_SIZE\_SCREEN[29-16] LPP bit field
- Number of pixels per line, DSS0\_VP\_SIZE\_SCREEN[13-0] PPL bit field

Standard HSYNC/VSYNC timing generation is programmable as follows:

- Horizontal front porch is set in the DSS0\_VP\_TIMING\_H[19-8] HFP bit field.
- Horizontal back porch is set in the DSS0\_VP\_TIMING\_H[31-20] HBP bit field.
- Horizontal synchronization pulse width is set in the DSS0\_VP\_TIMING\_H[7-0] HSW bit field.
- Vertical front porch is set in the DSS0\_VP\_TIMING\_V[19-8] VFP bit field.
- Vertical back porch is set in the DSS0\_VP\_TIMING\_V[31-20] VBP bit field.
- Vertical synchronization pulse width is set in the DSS0\_VP\_TIMING\_V[7-0] VSW bit field.

When the output is in BT.1120 or BT.656 mode, the following timing constants are mapped onto the DSS0\_VP\_TIMING\_H and DSS0\_VP\_TIMING\_V registers:

- Progressive mode:
  - Horizontal blanking (12 bits) is set in the {DSS0\_VP\_TIMING\_V[3-0] VSW, DSS0\_VP\_TIMING\_H[7-0] HSW} register fields (up to 2048 bytes of horizontal blanking supported).
  - Vertical frame blanking No 1 is set in the DSS0\_VP\_TIMING\_V[19-8] VFP bit field.
  - Vertical frame blanking No 2 is set in the DSS0\_VP\_TIMING\_V[31-20] VBP bit field.
  - Number of lines per frame is set in the DSS0\_VP\_SIZE\_SCREEN[29-16] LPP bit field.
  - Number of pixels per line is set in the DSS0\_VP\_SIZE\_SCREEN[13-0] PPL bit field.
- Interlaced mode:
  - Horizontal blanking (12 bits) is set in the {DSS0\_VP\_TIMING\_V[3-0] VSW, DSS0\_VP\_TIMING\_H[7-0] HSW} register fields (up to 2048 bytes of horizontal blanking supported).
  - Vertical field blanking No.1 for Even Field is set in the DSS0\_VP\_TIMING\_H[19-8] HFP bit field.
  - Vertical field blanking No.2 for Even Field is set in the DSS0\_VP\_TIMING\_H[31-20] HBP bit field.
  - Vertical field blanking No.1 for Odd Field is set in the DSS0\_VP\_TIMING\_V[19-8] VFP bit field.
  - Vertical field blanking No.2 for Odd Field is set in the DSS0\_VP\_TIMING\_V[31-20] VBP bit field.
  - Number of lines per field (even) is set in the DSS0\_VP\_SIZE\_SCREEN[29-16] LPP bit field.
  - Delta number of odd field compared to even field (in a single line) is set in the DSS0\_VP\_SIZE\_SCREEN[15-14] DELTA\_LPP bit field. The DELTA\_LPP field only controls the output channel and not the size of the field fetched from the frame buffer in memory. This field must be set to zero for YUV420 format.
  - Number of pixels per line is set in the DSS0\_VP\_SIZE\_SCREEN[13-0] PPL bit field.

Horizontal/vertical synchronization and output enable signals polarity are programmable by setting the DSS0\_VP\_POL\_FREQ[12] IVS, [13] IHS, and [15] IEO register bits. These signals can be gated by setting the DSS0\_VP\_CONFIG[7] VSYNCGATED and [6] HSYNCGATED register bits. In addition, the alignment between VSYNC and HSYNC signals can be programmed via the DSS0\_VP\_POL\_FREQ[18] ALIGN register bit.

The latch of data can be driven on the rising or falling edge of the pixel clock by setting the DSS0\_VP\_POL\_FREQ[14] IPC register bit. The drive of the SYNC and VSYNC signals in the function of the pixel clock is done by setting the DSS0\_VP\_POL\_FREQ[16] RF bit.

Each VP output can be configured in progressive output mode or interlaced output mode. The selection is done by writing into the bit-field DSS0\_VP\_CONFIG[22] OUTPUTMODEENABLE register bit. The default setting is for progressive mode. The selection can be changed only when the corresponding VP output is disabled. The configuration is independent for each VP output. It is possible to change the configuration of one of the VP outputs while the other VP is enabled.

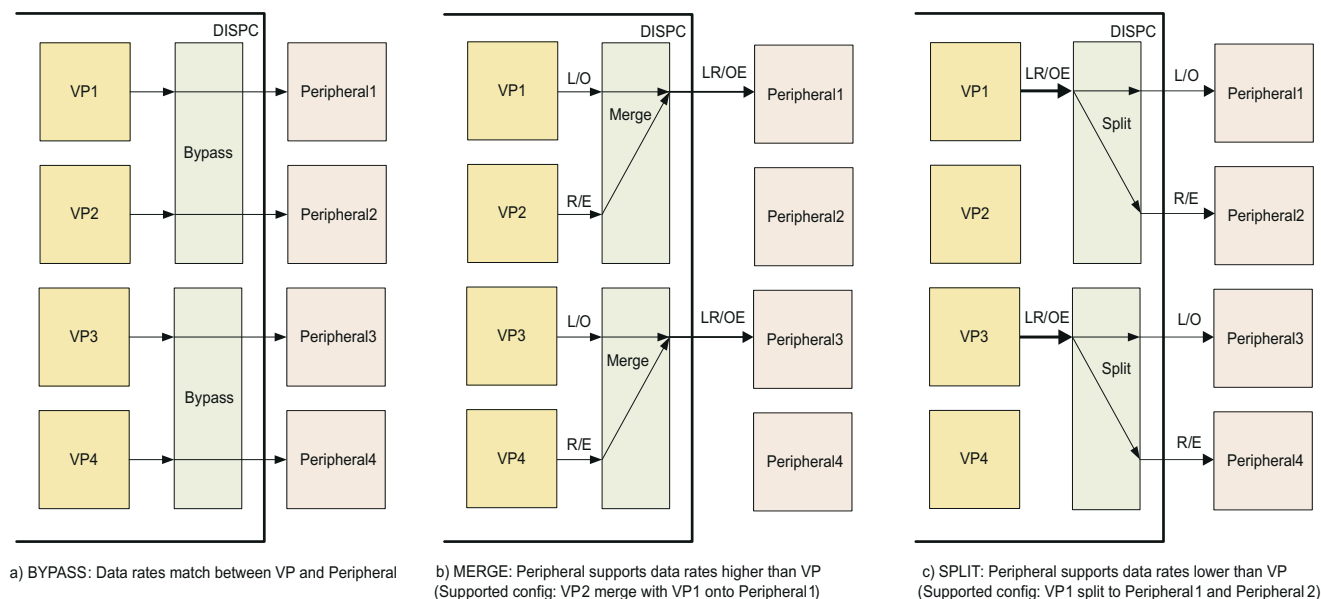
The pixel clock for each VP output can be gated by setting the DSS0\_VP\_CONFIG[5] PIXELCLOCKGATED register bit.

The hold time of the pixels on the data bus is determined in clock cycles by the DSS0\_VP\_CONTROL[16-14] HT register field.

### 12.6.4.11.9 DISPC VP Merge-Split-Sync (MSS) Module

In order to interface with peripherals which can support a different data rate than the VP output, a Merge-Split-Sync (MSS) module is introduced between the VP outputs and DISPC final output. When interfacing with a peripheral which can support a higher data rate than the VP, two VP outputs can be merged to generate one high data rate stream on the DISPC output (MERGE mode). When interfacing with a peripheral which requires dual physical links to support a high resolution video, one VP output can be split into two half data rate pixel streams on two separate DISPC outputs (SPLIT mode).

Both left/right (L/R) or odd/even (O/E) pixel selection types are supported.



**Figure 12-604. DISPC VP Merge-Split Scheme**

Additionally, the MSS module can also operate in a SYNC mode where two VP outputs get the same source pixel clock. The SYNC mode only affects the clock muxing and does not manipulate the data going out of each VP output.

Only the following configurations, as shown in [Figure 12-604](#), are supported:

- MERGE Mode
  - VP2 (Right/Even stream) merged with VP1 (Left/Odd stream) output onto Peripheral #1 (P1)
  - VP4 (Right/Even stream) merged with VP3 (Left/Odd stream) output onto Peripheral #3 (P3)
- SPLIT Mode
  - VP1 split into Peripheral #1 (P1)(Left/Odd stream) and Peripheral #2 (P2)(Right/Even stream)
  - VP3 split into Peripheral #3 (P3)(Left/Odd stream) and Peripheral #4 (P4)(Right/Even stream)
- SYNC Mode
  - VP1 and VP2 in sync and get the same pixel clock
  - VP3 and VP4 in sync and get the same pixel clock

The DSS0\_COMMON\_DISPC\_MSS\_VP1 and DSS0\_COMMON\_DISPC\_MSS\_VP3 registers control the MSS operation.

The MSS line buffers are sized to support up to 6K RGB30 video data. Therefore, in MERGE use-case, two 3K VP outputs can be merged to create a 6K output at DISPC. In SPLIT use-case, one 6K output can be split into two 3K outputs.

In both, MERGE and SPLIT use-cases, all the control signals (VS, HS, DE) are re-generated so as to match with the timing of the new merged frame or split frames.

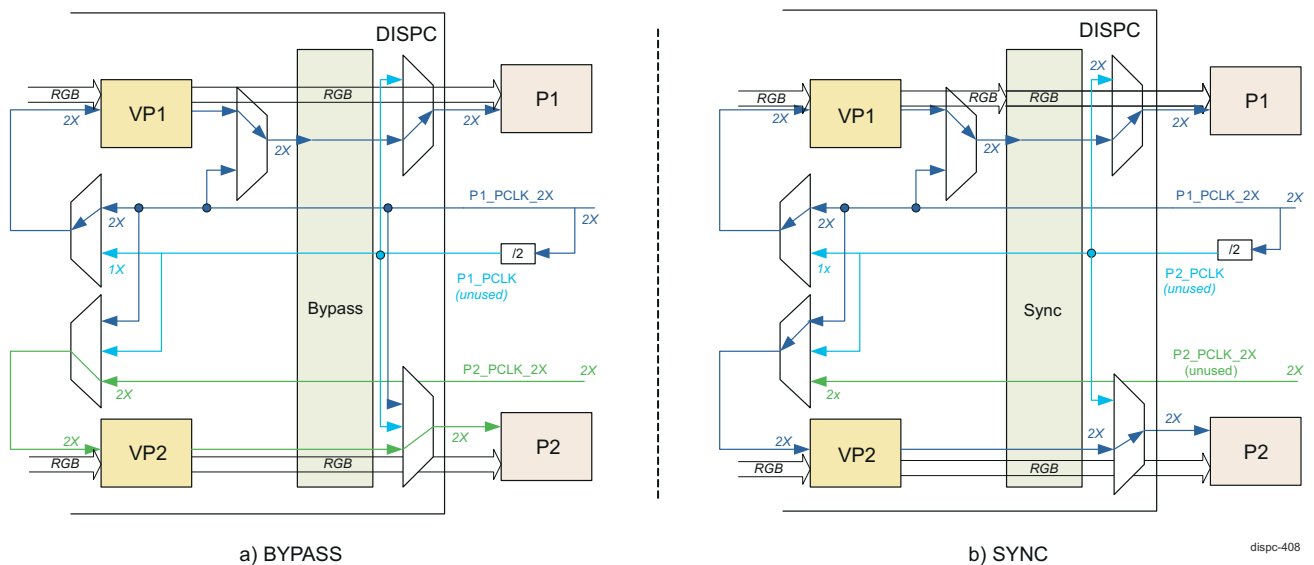
### Note

For Merge-Split-Sync mode of operation, the VP outputs need to be programmed to be active-high for the different signals (DE, VS, HS). For SPLIT mode of operation, it is required that the total number of pixels in a line is a multiple of 2. For both SPLIT mode and SYNC mode of operation, both the VP blocks need to be enabled at the same cycle. This is achieved by making a single write to the global enable register (DSS0\_COMMON\_DISPC\_GLOBAL\_OUTPUT\_ENABLE[3-0] VP\_ENABLE field).

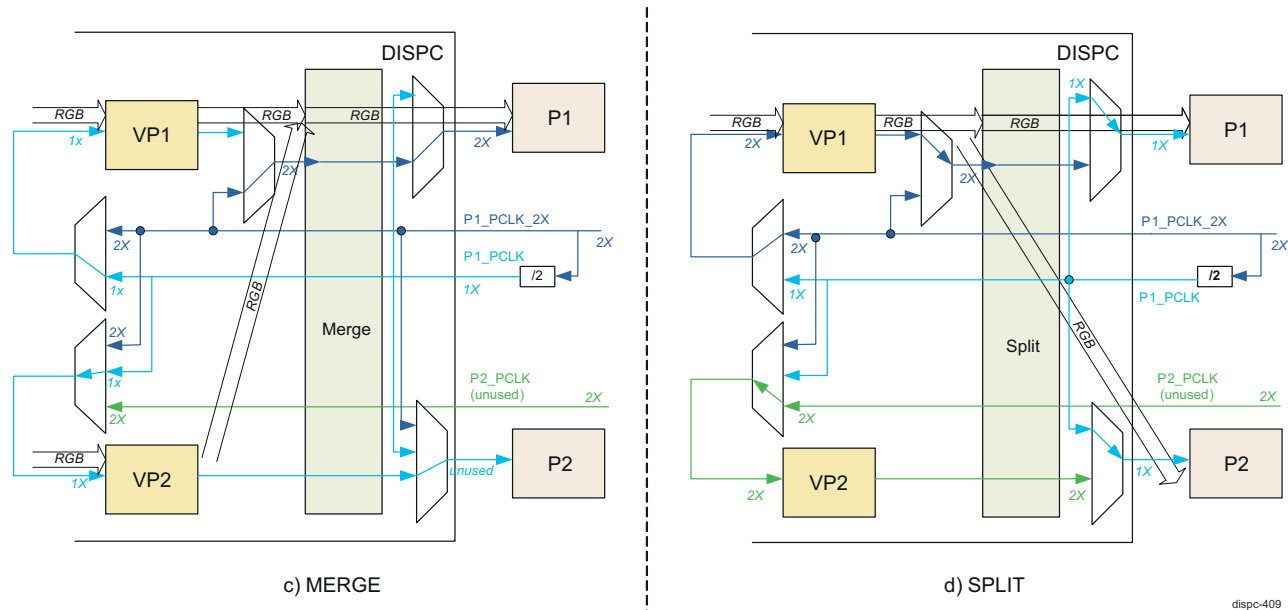
When the DISPC is interfaced with the DP/eDP peripheral, the SPLIT mode corresponds to the Split (single stream transport) mode of operation of the DSC (Digital Stream Compression) sub-module within the EDP module. In this case, the two streams at half the pixel clock rate are first provided to DSC\_ENC0 and DSC\_ENC1 within the EDP. Then, the combined encoded stream is provided to the EDP MHDPTX Controller VIF0 port. The SYNC mode corresponds to the Dual (multi stream transport) mode of operation of the EDP module. In this case, the two streams at full pixel clock rate are provided to the EDP VIF0 and VIF1 ports (the EDP DSC has to be bypassed in this mode). For more details, see the EDP [Section 12.6.6.2.1, Video Stream Clock/Data Muxing](#).

#### 12.6.4.11.9.1 MSS Clocking Scheme

The pixel clocks going to the VPn as well as the pixel clocks going to the peripherals (Pn) will be different under different cases, depending on whether the output is in BYPASS, SYNC, MERGE or SPLIT mode. The different combinations of the clocks are as shown in [Figure 12-605](#) and [Figure 12-606](#).



**Figure 12-605. DISPC VP Clocking Scheme for MSS BYPASS and SYNC Modes**



**Figure 12-606. DISPC VP Clocking Scheme for MSS MERGE and SPLIT Modes**

Figure 12-606 show VP1 and VP2. Same clock scheme is applicable to VP3 and VP4.

For all cases in Figure 12-605 and Figure 12-606 there is no async interaction between P1\_PCLKs and P2\_PCLKs.

#### 12.6.4.11.9.2 MSS Merge with Scaling

Merge operation with scaling enabled in the VID pipelines can result in visual artifacts at the merged boundary. The artifacts can be avoided by scaling each half to a larger window size and then using the crop feature of the VID pipelines to get to the required size for each half.

#### 12.6.4.11.10 DISPC Internal Diagnostic Features

The DSS supports the following internal diagnostic features in DISPC hardware:

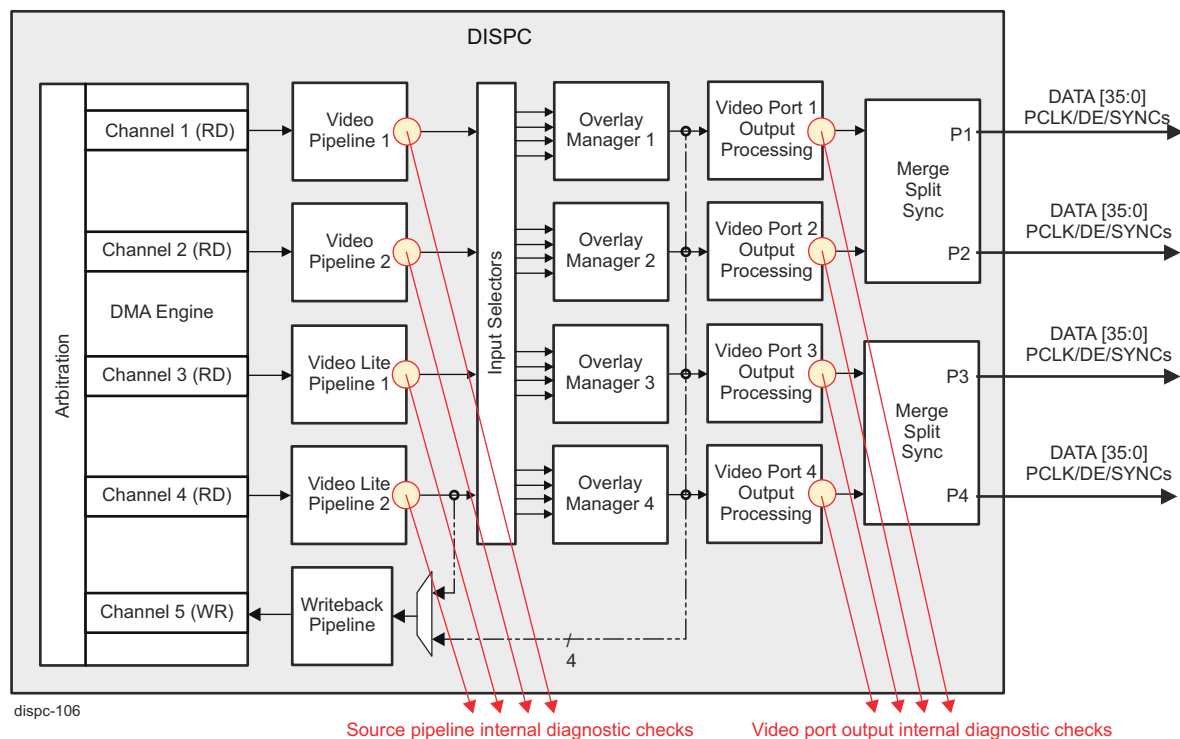
- Data correctness check: To verify intended data is shown correctly on the display.
- Freeze frame detection: To notify a possible frame freeze, when there is no change in the display frame over a multiple frame periods.

These features are implemented by collecting signatures from user-defined regions within each pipeline output frame and the final display output frame, and comparing them to reference signatures provided by the user (software) and/or to previously saved signatures. The signature from each region is generated by using a MISR (Multiple Input Signature Register) module.

#### 12.6.4.11.10.1 Internal Diagnostic Check Regions

The DSS supports the following internal diagnostic check regions:

- Video pipelines: One internal diagnostic check region at the output of each video pipeline.
- Video port outputs: Up to eight internal diagnostic sub-regions within the active video output area of the final display output of each video port.



**Figure 12-607. DISPC Internal Diagnostic Check Regions**

Table 12-564 lists the parameters supported by each of the internal diagnostic check regions, together with the associated register control fields.

**Table 12-564. DISPC Internal Diagnostic Check Regions Parameters**

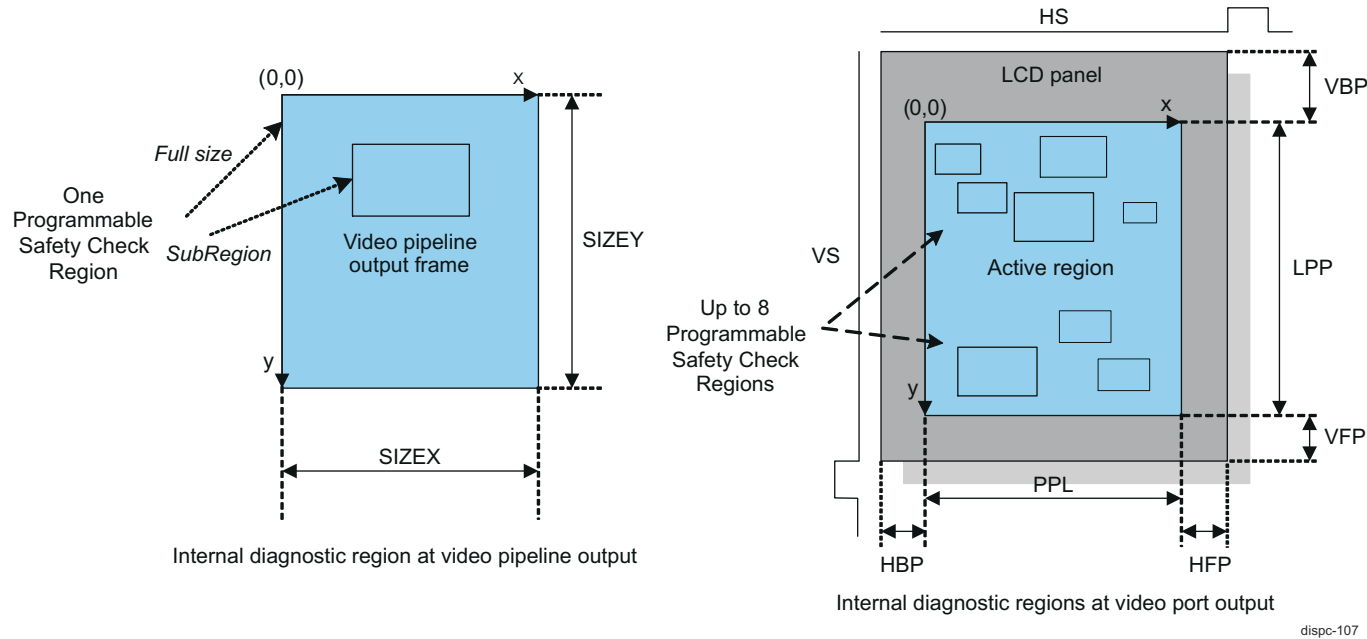
Internal Diagnostic Region Parameter	Video Pipeline Control Register	Video Port Sub-region m (m = 0 to 7) Control Register
X position	DSS0_VID_SAFETY_POSITION[11-0] POSX	DSS0_VP_SAFETY_POSITION_m[11-0] POSX
Y position	DSS0_VID_SAFETY_POSITION[27-16] POSY	DSS0_VP_SAFETY_POSITION_m[27-16] POSY
Width	DSS0_VID_SAFETY_SIZE[11-0] SIZEX	DSS0_VP_SAFETY_SIZE_m[11-0] SIZEX

**Table 12-564. DISPC Internal Diagnostic Check Regions Parameters (continued)**

Internal Diagnostic Region Parameter	Video Pipeline Control Register	Video Port Sub-region m (m = 0 to 7) Control Register
Height	DSS0_VID_SAFETY_SIZE[27-16] SIZEY	DSS0_VP_SAFETY_SIZE_m[27-16] SIZEY
Data Correctness Check Mode Enable	DSS0_VID_SAFETY_ATTRIBUTES[1] CAPTUREMODE	DSS0_VP_SAFETY_ATTRIBUTES_m[1] CAPTUREMODE
Freeze Frame Detection Mode Enable	DSS0_VID_SAFETY_ATTRIBUTES[1] CAPTUREMODE	DSS0_VP_SAFETY_ATTRIBUTES_m[1] CAPTUREMODE
Region Internal Diagnostic Check Enable	DSS0_VID_SAFETY_ATTRIBUTES[0] ENABLE	DSS0_VP_SAFETY_ATTRIBUTES_m[0] ENABLE
Freeze Frame Detection Threshold	DSS0_VID_SAFETY_ATTRIBUTES[10-3] THRESHOLD	DSS0_VP_SAFETY_ATTRIBUTES_m[10-3] THRESHOLD
Frames to Skip	DSS0_VID_SAFETY_ATTRIBUTES[12-11] FRAMESKIP	DSS0_VP_SAFETY_ATTRIBUTES_m[12-11] FRAMESKIP
Reference Signature	DSS0_VID_SAFETY_REF_SIGNATURE[31-0] SIGNATURE	DSS0_VP_SAFETY_REF_SIGNATURE_m[31-0] SIGNATURE
MISR Seed	DSS0_VID_SAFETY_LFSR_SEED[31-0] SEED	DSS0_VP_SAFETY_LFSR_SEED[31-0] SEED
MISR Seed Selection	DSS0_VID_SAFETY_ATTRIBUTES[2] SEEDSELECT	DSS0_VP_SAFETY_ATTRIBUTES_m[2] SEEDSELECT
MISR Captured Signature	DSS0_VID_SAFETY_CAPT_SIGNATURE[31-0] SIGNATURE	DSS0_VP_SAFETY_CAPT_SIGNATURE_m[31-0] SIGNATURE



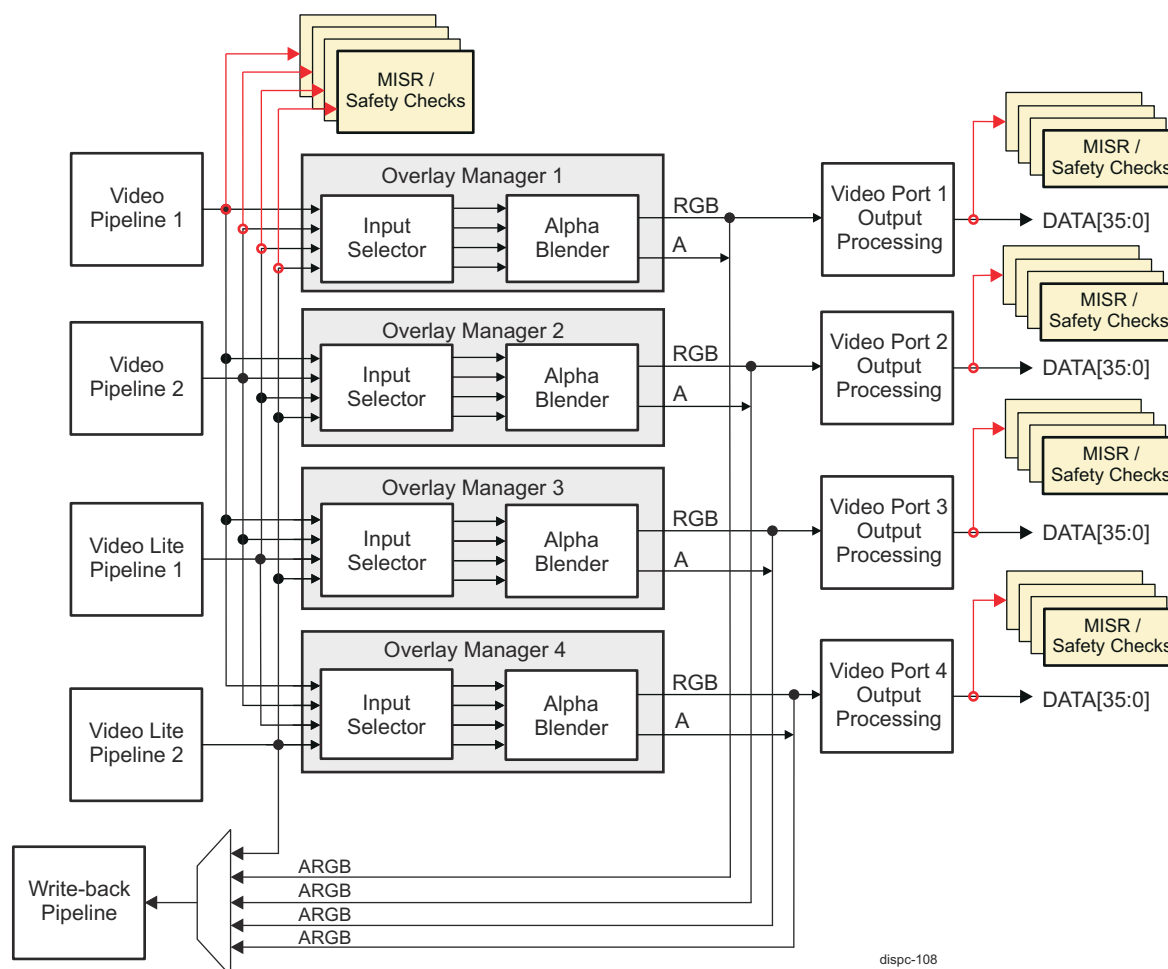
Figure 12-608 shows examples of one internal diagnostic region in the video pipeline output stage and up to 8 possible regions in the final video port output stage.



**Figure 12-608. DISPC Internal Diagnostic Check Region Examples**

The internal diagnostic region in the video pipeline captures data, only if the embedded alpha data is not equal to 0 (that is, non-transparent pixels). The internal diagnostic regions in the display video port output captures all active video pixels within the region boundary. The regions (up to eight) in the display output should be typically non-overlapping areas of the screen, but the hardware does not restrict them to be non-overlapping.

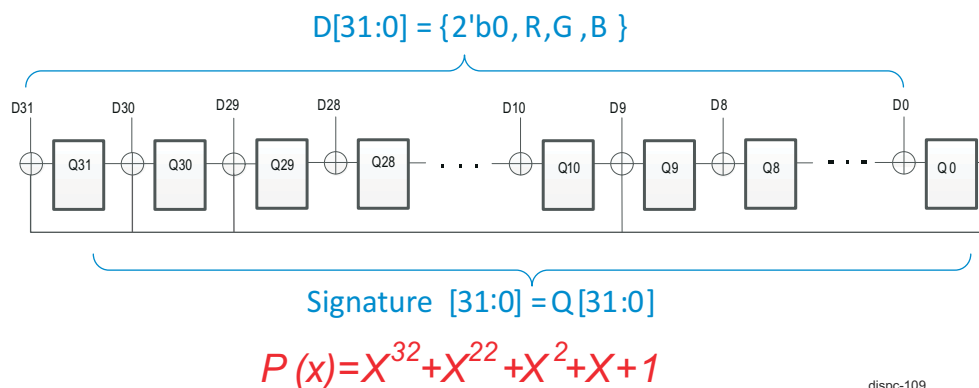
Figure 12-609 shows the locations within DISPC data path where the data is analyzed.



**Figure 12-609. DISPC Internal Diagnostic Check Locations**

### 12.6.4.11.10.2 Internal Diagnostic Signature Generator Using MISR

A 32-bit multiple input signature register (MISR) with 32-bit Galois LFSR polynomial,  $P(x)=X^{32}+X^{22}+X^2+X+1$ , is used to capture a signature of active video pixel data (the 10 MSB bits of each color component) for each internal diagnostic region, as shown in Figure 12-610 and the algorithm below it.



**Figure 12-610. DISPC Internal Diagnostic 32-bit MISR Implementation**

The MISR (32-bit Galois LFSR based) signature generation algorithm is as follows:

```
Tap_polynomial = 32'hE000_0200
```

```
lfsr_q = seed // initial LFSR value - any non zero value should work
```

```
For each (pixel data in the internal diagnostic region) {
```

```
data_in[31:0] = { 2'b0, r,g,b } // 10 MSB bits of R,G,B components
```

```
lfsr_d = (data_in(lfsr_q >> 1))
```

```
If (lfsr_q[0]) lfsr_d = lfsr_d ^ Tap_polynomial
```

```
lfsr_q = lfsr_d
```

```
}
```

```
signature = lfsr_q
```

All MISRs are initialized at the beginning of each frame with a non-zero "seed" value either with the default constant value (0xFFFF\_FFFF) or with the programmable seed configuration value in SAFETY\_LFSR\_SEED register for each group of regions (refer to Table 12-564, *DISPC Internal Diagnostic Check Regions Parameters*). Note that a same seed must be used to compare signatures between two frames. If different than the default seed value is desired, then a SEEDSELECT parameter must be set before the corresponding SAFETY\_LFSR\_SEED register is configured with the customer seed value.

All captured signatures from the internal diagnostic regions are memory mapped to read-only SAFETY\_CAPT\_SIGNATURE registers for each video port (refer to Table 12-564, *DISPC Internal Diagnostic Check Regions Parameters*). A SAFETY\_CAPT\_SIGNATURE register is updated with the signature from each sub-region at the end of every frame. This register returns the signature of the last frame data when the MISR is enabled. When MISR is disabled, this register is cleared.

The MISR aliasing (faulty signature matches fault-free signature) probability is:

$$(2^{(L-n)} - 1) / (2^L - 1) \approx 2^{(L-n)} / 2^L = 2^{-n} \text{ for large } L, \text{ where}$$

$n$  = length of signature register

$L$  = length of input sequence

Using the above approximation, the result is:

n=4, aliasing probability = 6.25%

n=16, aliasing probability = 0.0015%

...

n=32 (as in DISPC MISR), aliasing probability is  $\sim 2^{-32}$  (negligible)

#### 12.6.4.11.10.3 Internal Diagnostic Checks

If the data correctness check is enabled (see [Table 12-564](#), *DISPC Internal Diagnostic Check Regions Parameters*):

- When a new signature is generated, it is compared against the reference signature provided by the software. A SAFETY\_REF\_SIGNATURE register must be configured with a reference signature data, see [Table 12-564](#), *DISPC Internal Diagnostic Check Regions Parameters*.
- If signatures do not match, an interrupt event is generated to indicate data mismatch.

If the freeze frame detection is enabled (see [Table 12-564](#), *DISPC Internal Diagnostic Check Regions Parameters*):

- When a new signature is generated, it is first compared against the saved signature (from previous frame).
  - There exists a capability within the frame freeze detection logic to skip alternate frames during comparison. This feature is useful while handling ping-pong buffers or interlaced input video. This feature is controlled via the [12-11] FRAMESKIP field within a SAFETY\_ATTRIBUTES register.
- If signatures match, an internal counter used to keep track of the number of frames with no data change (for the region) is incremented.
- If signatures do not match, the counter is cleared.
- If the counter value is greater than the user programmed freeze frame detection threshold value (see [Table 12-564](#), *DISPC Internal Diagnostic Check Regions Parameters*), an interrupt event (VIDSAFETYREGION\_IRQ or VPSAFETYREGION\_IRQ, see [Section 12.6.4.5](#), *DISPC Interrupt Requests*) is generated to indicate a possible freeze frame detection. The threshold value must be configured with the maximum number of identical successive frames allowed before an interrupt is generated.
- After the comparison, the signature is saved as the previous signature.
- The counter is cleared when the interrupt event is generated or when freeze frame detection check is disabled.

This frame freeze is different from the display getting frozen due to pipeline lock up. In that case, the DISPC will generate an SYNCLOST\_IRQ and/or DMA VIDBUFFERUNDERFLOW\_IRQ interrupt. The freeze frame detection is for source data getting frozen while the DSS is working normally.

The two internal diagnostic checks are continuously performed over multiple frames as long as their mode enable register bits are set.

#### 12.6.4.11.10.4 Internal Diagnostic Check Limitations

The internal diagnostic check is only be available when the DISPC is outputting RGB/YUV component data with separate sync signals. The internal diagnostic functions are not available for the following output modes:

- YUV422 embedded sync modes (BT.656 and BT.1120)
- RGB TDM (Time Division Multiplex) mode

#### 12.6.4.11.11 DISPC Security Management

##### 12.6.4.11.11.1 Security Implementation

DSS supports the following security features:

- Secure mode configuration
- Illegal connection prevention

#### Secure Mode

The DISPC supports secure mode configuration registers (DSS0\_VID\_SECURE, DSS0\_WB\_SECURE, DSS0\_OVR\_SECURE and DSS0\_VP\_SECURE) which defines the "secure mode" attribute of each pipeline,

overlay manager, and video port instance in DISPC. These registers can only be modified by a host with an appropriate secure privilege (MReqSecure=1). When the SECURE bit corresponding to an instance is set by a secure host, the instance is deemed to be in "secure mode" and the DISPC hardware prevents the output of the instance getting connected to a non-secure downstream module. Also, any DMA transfer initiated by a secure pipeline will have its OCP in-band signal MReqSecure set to HIGH to indicate that is a secure mode transaction request.

By default, all pipelines, overlay managers, and video ports are in a "non-secure mode". The corresponding SECURE bits in the DSS0\_VID\_SECURE, DSS0\_WB\_SECURE, DSS0\_OVR\_SECURE and DSS0\_VP\_SECURE registers are active, only when the DSS0\_COMMON\_DISPC\_SECURE\_DISABLE[0] SECURE\_DISABLE register bit is configured properly to 0x0. When the SECURE\_DISABLE bit is set to 0x1, the SECURE register bits are non-active and DISPC will behave as non-secure module.

### Illegal Connection Prevention

The DISPC hardware enforces the following rules to prevent an illegal connection:

- Secure input pipeline can only connect to a secure overlay manager: If any host (secure or non-secure) configures an overlay manager input selection to connect a secure input pipeline to a non-secure overlay manager, then the DISPC hardware will block the connection and issue a "security violation" interrupt (SECURITYVIOLATION\_IRQ) to alert the host.
- Non-Secure WB pipeline can take its input only from a non-secure input pipeline or a non-secure overlay manager output. Otherwise, the DISPC hardware will block the data transfer through WB and issue a "security violation" interrupt to alert the host.
- Non-Secure video port can only display data from a non-secure overlay manager output. Otherwise, the DISPC hardware will block the data display on the video port and issue a "security violation" interrupt to alert the host.

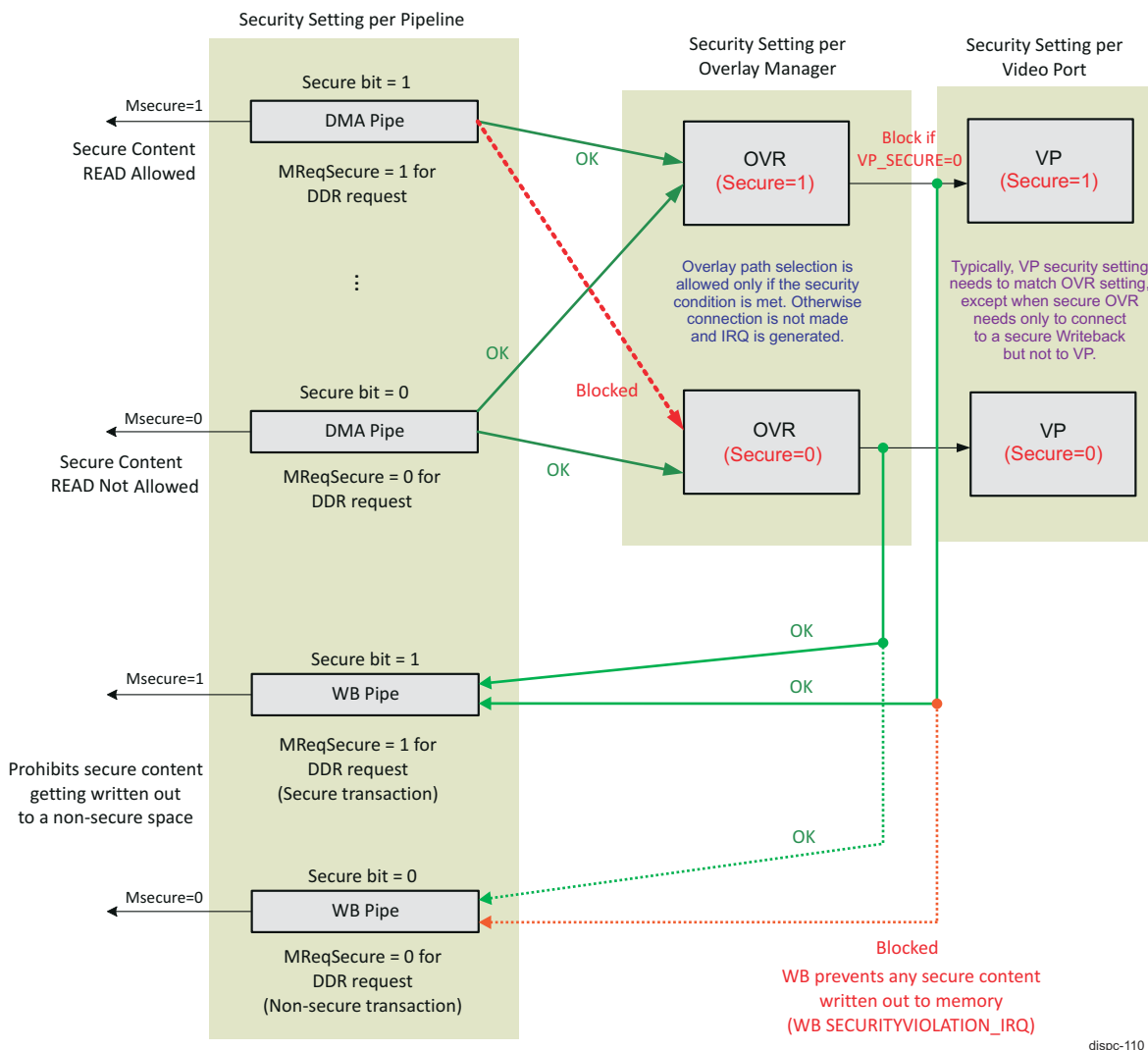


Figure 12-611. DISPC Secure Bit Setting and Illegal Connection Block

#### 12.6.4.11.11.2 Secure Mode Configuration

The SECURE bit in DSS0\_VID\_SECURE, DSS0\_WB\_SECURE, DSS0\_OVR\_SECURE and DSS0\_VP\_SECURE registers is set/reset by a secure transaction. When the SECURE bit has been set, the software in "secure mode" is responsible for checking the DISPC configuration. The SECURE bit is propagated by DISPC to the system Interconnect in order to qualify all DISPC requests as secure or non-secure requests, based on the secure bits defined in the control register.

When DISPC accesses the frame buffer, in the case the SECURE bit has been reset and the frame buffer has been set secure, the DISPC will receive an "error" in response of non-secure requests.

#### 12.6.4.11.12 DISPC Shadow Mechanism for Registers

Some DISPC registers are termed *shadow registers*. The shadow registers allow the software to modify them at any time, without direct effect on the DISPC hardware configuration. When all the values for a given configuration are written into the shadow registers, software must set only one register bit to validate the configuration. When the hardware reaches the end of the current frame and sees that the bit has been set by software, the new configuration is now the configuration used by the hardware.

The DSS0\_VP\_CONTROL[5] GOBIT bit enables the hardware to use the new configuration, for all shadow registers associated with each VP output.

The registers are statically associated to a particular output (for example, timing registers) or dynamically associated to one output at a time (for example, video registers).

#### 12.6.4.11.13 DISPC Registers

This section describes the DISPC registers.

##### 12.6.4.11.13.1 DSS\_COMMON Registers

Table 12-566 lists the memory-mapped registers for the DSS\_COMMON. All register offset addresses not listed in Table 12-566 should be considered as reserved locations and the register contents should not be modified.

DSS COMMON Registers

**Table 12-565. DSS\_COMMON Instances**

Instance	Base Address
DSS0_DISPC_0_COMMON_M	04A0 0000h
DSS0_DISPC_0_COMMON_S0	04A1 0000h
DSS0_DISPC_0_COMMON_S1	04B0 0000h
DSS0_DISPC_0_COMMON_S2	04B1 0000h

**Table 12-566. DSS\_COMMON Registers**

Offset	Acronym	Register Name	DSS0_DISPC_0_COMMON_M Physical Address	DSS0_DISPC_0_COMMON_S0 Physical Address	DSS0_DISPC_0_COMMON_S1 Physical Address	DSS0_DISPC_0_COMMON_S2 Physical Address
4h	<a href="#">DSS0_COMMON_DSS_REVISION</a>		04A0 0004h	N/A <sup>(1)</sup>	N/A	N/A
8h	<a href="#">DSS0_COMMON_DSS_SYSCONFIG</a>		04A0 0008h	N/A	N/A	N/A
20h	<a href="#">DSS0_COMMON_DSS_SYSSTATUS</a>		04A0 0020h	N/A	N/A	N/A
28h	<a href="#">DSS0_COMMON_DISPC_IRQSTATUS_RAW</a>		04A0 0028h	04A1 0028h	04B0 0028h	04B1 0028h
2Ch	<a href="#">DSS0_COMMON_DISPC_IRQSTATUS</a>		04A0 002Ch	04A1 002Ch	04B0 002Ch	04B1 002Ch
30h	<a href="#">DSS0_COMMON_DISPC_IRQENABLE_SET</a>		04A0 0030h	04A1 0030h	04B0 0030h	04B1 0030h
34h	<a href="#">DSS0_COMMON_DISPC_IRQENABLE_CLR</a>		04A0 0034h	04A1 0034h	04B0 0034h	04B1 0034h
38h	<a href="#">DSS0_COMMON_VID_IRQENABLE_0</a>		04A0 0038h	04A1 0038h	04B0 0038h	04B1 0038h
3Ch	<a href="#">DSS0_COMMON_VID_IRQENABLE_1</a>		04A0 003Ch	04A1 003Ch	04B0 003Ch	04B1 003Ch
40h	<a href="#">DSS0_COMMON_VID_IRQENABLE_2</a>		04A0 0040h	04A1 0040h	04B0 0040h	04B1 0040h
44h	<a href="#">DSS0_COMMON_VID_IRQENABLE_3</a>		04A0 0044h	04A1 0044h	04B0 0044h	04B1 0044h
48h	<a href="#">DSS0_COMMON_VID_IRQSTATUS_0</a>		04A0 0048h	04A1 0048h	04B0 0048h	04B1 0048h
4Ch	<a href="#">DSS0_COMMON_VID_IRQSTATUS_1</a>		04A0 004Ch	04A1 004Ch	04B0 004Ch	04B1 004Ch
50h	<a href="#">DSS0_COMMON_VID_IRQSTATUS_2</a>		04A0 0050h	04A1 0050h	04B0 0050h	04B1 0050h
54h	<a href="#">DSS0_COMMON_VID_IRQSTATUS_3</a>		04A0 0054h	04A1 0054h	04B0 0054h	04B1 0054h
58h	<a href="#">DSS0_COMMON_VP_IRQENABLE_0</a>		04A0 0058h	04A1 0058h	04B0 0058h	04B1 0058h
5Ch	<a href="#">DSS0_COMMON_VP_IRQENABLE_1</a>		04A0 005Ch	04A1 005Ch	04B0 005Ch	04B1 005Ch
60h	<a href="#">DSS0_COMMON_VP_IRQENABLE_2</a>		04A0 0060h	04A1 0060h	04B0 0060h	04B1 0060h
64h	<a href="#">DSS0_COMMON_VP_IRQENABLE_3</a>		04A0 0064h	04A1 0064h	04B0 0064h	04B1 0064h
68h	<a href="#">DSS0_COMMON_VP_IRQSTATUS_0</a>		04A0 0068h	04A1 0068h	04B0 0068h	04B1 0068h
6Ch	<a href="#">DSS0_COMMON_VP_IRQSTATUS_1</a>		04A0 006Ch	04A1 006Ch	04B0 006Ch	04B1 006Ch
70h	<a href="#">DSS0_COMMON_VP_IRQSTATUS_2</a>		04A0 0070h	04A1 0070h	04B0 0070h	04B1 0070h
74h	<a href="#">DSS0_COMMON_VP_IRQSTATUS_3</a>		04A0 0074h	04A1 0074h	04B0 0074h	04B1 0074h
78h	<a href="#">DSS0_COMMON_WB_IRQENABLE</a>		04A0 0078h	04A1 0078h	04B0 0078h	04B1 0078h
7Ch	<a href="#">DSS0_COMMON_WB_IRQSTATUS</a>		04A0 007Ch	04A1 007Ch	04B0 007Ch	04B1 007Ch

**Table 12-566. DSS\_COMMON Registers (continued)**

Offset	Acronym	Register Name	DSS0_DISPC_0_COMMON_M Physical Address	DSS0_DISPC_0_COMMON_S0 Physical Address	DSS0_DISPC_0_COMMON_S1 Physical Address	DSS0_DISPC_0_COMMON_S2 Physical Address
80h	<a href="#">DSS0_COMMON_DISPC_IRQ_EOI_FUNC</a>		04A0 0080h	04A1 0080h	04B0 0080h	04B1 0080h
84h	<a href="#">DSS0_COMMON_DISPC_IRQ_EOI_SAFETY</a>		04A0 0084h	04A1 0084h	04B0 0084h	04B1 0084h
88h	<a href="#">DSS0_COMMON_DISPC_IRQ_EOI_SECURITY</a>		04A0 0088h	04A1 0088h	04B0 0088h	04B1 0088h
90h	<a href="#">DSS0_COMMON_DISPC_SECURE_DISABLE</a>		04A0 0090h	N/A	N/A	N/A
98h	<a href="#">DSS0_COMMON_DISPC_GLOBAL_MFLAG_ATTRIBUTE</a>		04A0 0098h	N/A	N/A	N/A
9Ch	<a href="#">DSS0_COMMON_DISPC_GLOBAL_OUTPUT_ENABLE</a>		04A0 009Ch	N/A	N/A	N/A
A0h	<a href="#">DSS0_COMMON_DISPC_GLOBAL_BUFFER</a>		04A0 00A0h	N/A	N/A	N/A
A4h	<a href="#">DSS0_COMMON_DSS_CBA_CFG</a>		04A0 00A4h	N/A	N/A	N/A
A8h	<a href="#">DSS0_COMMON_DISPC_DBG_CONTROL</a>		04A0 00A8h	N/A	N/A	N/A
ACh	<a href="#">DSS0_COMMON_DISPC_DBG_STATUS</a>		04A0 00ACh	N/A	N/A	N/A
B0h	<a href="#">DSS0_COMMON_DISPC_CLKGATING_DISABLE</a>		04A0 00B0h	N/A	N/A	N/A
B8h	<a href="#">DSS0_COMMON_FBDC_REVISION_1</a>		04A0 00B8h	N/A	N/A	N/A
BCh	<a href="#">DSS0_COMMON_FBDC_REVISION_2</a>		04A0 00BCh	N/A	N/A	N/A
C0h	<a href="#">DSS0_COMMON_FBDC_REVISION_3</a>		04A0 00C0h	N/A	N/A	N/A
C4h	<a href="#">DSS0_COMMON_FBDC_REVISION_4</a>		04A0 00C4h	N/A	N/A	N/A
C8h	<a href="#">DSS0_COMMON_FBDC_REVISION_5</a>		04A0 00C8h	N/A	N/A	N/A
CCh	<a href="#">DSS0_COMMON_FBDC_REVISION_6</a>		04A0 00CCh	N/A	N/A	N/A
D0h	<a href="#">DSS0_COMMON_FBDC_COMMON_CONTROL</a>		04A0 00D0h	N/A	N/A	N/A
D4h	<a href="#">DSS0_COMMON_FBDC_CONSTANT_COLOR_0</a>		04A0 00D4h	N/A	N/A	N/A
D8h	<a href="#">DSS0_COMMON_FBDC_CONSTANT_COLOR_1</a>		04A0 00D8h	N/A	N/A	N/A
E4h	<a href="#">DSS0_COMMON_DISPC_CONNECTIONS</a>		04A0 00E4h	N/A	N/A	N/A
E8h	<a href="#">DSS0_COMMON_DISPC_MSS_VP1</a>		04A0 00E8h	N/A	N/A	N/A
ECh	<a href="#">DSS0_COMMON_DISPC_MSS_VP3</a>		04A0 00ECh	N/A	N/A	N/A
F0h	<a href="#">DSS0_COMMON_GLOBAL_DMA_THREAD_SIZE</a>		04A0 00F0h	N/A	N/A	N/A
F4h	<a href="#">DSS0_COMMON_GLOBAL_DMA_THREAD_SIZESTATUS</a>		04A0 00F4h	N/A	N/A	N/A
F8h	<a href="#">DSS0_COMMON_GLOBAL_GOBITMODE</a>		04A0 00F8h	N/A	N/A	N/A

(1) N/A = Not Applicable



### 12.6.4.11.13.1.1 DSS0\_COMMON\_DSS\_REVISION Register (Offset = 4h) [reset = 64003201h]

DSS0\_COMMON\_DSS\_REVISION is shown in [Figure 12-612](#) and described in [Table 12-568](#).

Return to [Summary Table](#).

This register contains the K3\_DSS revision number

**Table 12-567. DSS0\_COMMON\_DSS\_REVISION  
Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 0004h
DSS0_DISPC_0_COMMON_S0	N/A
DSS0_DISPC_0_COMMON_S1	N/A
DSS0_DISPC_0_COMMON_S2	N/A

**Figure 12-612. DSS0\_COMMON\_DSS\_REVISION Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODID															
R-6400h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REVRTL				REVMAJOR				CUSTOM				REVMIN			
R-6h				R-2h				R-0h				R-1h			

LEGEND: R = Read Only; -n = value after reset

**Table 12-568. DSS0\_COMMON\_DSS\_REVISION Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	MODID	R	6400h	Module ID Field
15-11	REVRTL	R	6h	RTL Revision
10-8	REVMAJOR	R	2h	Major Revision
7-6	CUSTOM	R	0h	Custom
5-0	REVMIN	R	1h	Minor Revision

### 12.6.4.11.13.1.2 DSS0\_COMMON\_DSS\_SYSCONFIG Register (Offset = 8h) [reset = 11h]

DSS0\_COMMON\_DSS\_SYSCONFIG is shown in [Figure 12-613](#) and described in [Table 12-570](#).

Return to [Summary Table](#).

This register controls various parameters related to software reset and IP idle

**Table 12-569. DSS0\_COMMON\_DSS\_SYSCONFIG Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 0008h
DSS0_DISPC_0_COMMON_S0	N/A
DSS0_DISPC_0_COMMON_S1	N/A
DSS0_DISPC_0_COMMON_S2	N/A

**Figure 12-613. DSS0\_COMMON\_DSS\_SYSCONFIG Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED		RESERVED					
R-0h		R-0h					
7	6	5	4	3	2	1	0
RESERVED		WARMRESET	IDLEMODE		RESERVED	SOFTRESET	AUTOCLKGATING
R-0h		R/W-0h	R/W-2h		R-0h	R/W-0h	R/W-1h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-570. DSS0\_COMMON\_DSS\_SYSCONFIG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-14	RESERVED	R	0h	Write 0's for future compatibility. Read returns 0
13-8	RESERVED	R	0h	Write 0's for future compatibility. Read returns 0
7-6	RESERVED	R	0h	Write 0's for future compatibility. Read returns 0
5	WARMRESET	R/W	0h	Warm reset. Setting this bit to 1 triggers a module warm reset. The bit is automatically reset by the hardware. During read, it always returns 0. The warm reset keeps the configuration registers unchanged
4-3	IDLEMODE	R/W	2h	Deprecated
2	RESERVED	R	0h	Write 0's for future compatibility. Read returns 0

**Table 12-570. DSS0\_COMMON\_DSS\_SYSCONFIG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	SOFTRESET	R/W	0h	Software reset. Setting this bit to 1 triggers a module reset. The bit is automatically reset by the hardware. During read, it always returns 0
0	AUTOCLKGATING	R/W	1h	Internal clock gating strategy  0h = Clocks are free-running  1h = Automatic clock gating strategy is applied, clocks are gated based on module activity

### 12.6.4.11.13.1.3 DSS0\_COMMON\_DSS\_SYSSTATUS Register (Offset = 20h) [reset = 21Fh]

DSS0\_COMMON\_DSS\_SYSSTATUS is shown in [Figure 12-614](#) and described in [Table 12-572](#).

Return to [Summary Table](#).

This register provides status information about the module, excluding the interrupt status information

**Table 12-571. DSS0\_COMMON\_DSS\_SYSSTATUS Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 0020h
DSS0_DISPC_0_COMMON_S0	N/A
DSS0_DISPC_0_COMMON_S1	N/A
DSS0_DISPC_0_COMMON_S2	N/A

**Figure 12-614. DSS0\_COMMON\_DSS\_SYSSTATUS Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						DISPC_IDLE_S TATUS	RESERVED
R-0h						R-1h	R-0h
7	6	5	4	3	2	1	0
RESERVED			DISPC_VP_RESETDONE				DISPC_FUNC_ RESETDONE
R-0h			R-Fh				R-1h

LEGEND: R = Read Only; -n = value after reset

**Table 12-572. DSS0\_COMMON\_DSS\_SYSSTATUS Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0h	Reserved
9	DISPC_IDLE_STATUS	R	1h	Idle status of DISPC 0h = DISPC is not in Idle 1h = DISPC is in Idle
8-5	RESERVED	R	0h	Reserved 0h = Internal module reset is on-going 1h = Reset completed
4-1	DISPC_VP_RESETDONE	R	Fh	Reset status of VP [3:0] pixel clock domain 0h = Internal module reset is on-going 1h = Reset completed

**Table 12-572. DSS0\_COMMON\_DSS\_SYSSTATUS Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	DISPC_FUNC_RESETDONE	R	1h	Reset status of DISPC Functional clock domain  0h = Internal module reset is on-going  1h = Reset completed

#### 12.6.4.11.13.1.4 DSS0\_COMMON\_DISPC\_IRQSTATUS\_RAW Register (Offset = 28h) [reset = 0h]

DSS0\_COMMON\_DISPC\_IRQSTATUS\_RAW is shown in [Figure 12-615](#) and described in [Table 12-574](#).

Return to [Summary Table](#).

RAW Interrupt status. Raw status is set even if interrupt is not enabled. Write 1 to set the RAW status

**Table 12-573.**  
**DSS0\_COMMON\_DISPC\_IRQSTATUS\_RAW**  
**Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 0028h
DSS0_DISPC_0_COMMON_S0	04A1 0028h
DSS0_DISPC_0_COMMON_S1	04B0 0028h
DSS0_DISPC_0_COMMON_S2	04B1 0028h

**Figure 12-615. DSS0\_COMMON\_DISPC\_IRQSTATUS\_RAW Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							DUMMY_IRQ
R-0h							R/W1S-0h
15	14	13	12	11	10	9	8
DUMMY1_IRQ	WB_IRQ	RESERVED	RESERVED				
R/W1S-0h	R/W1S-0h	R-0h	R-0h				
7	6	5	4	3	2	1	0
VID_IRQ				VP_IRQ			
R/W1S-0h				R/W1S-0h			

LEGEND: R = Read Only; R/W1S = Read/Write 1 to Set Bit; -n = value after reset

**Table 12-574. DSS0\_COMMON\_DISPC\_IRQSTATUS\_RAW Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reserved
16	DUMMY_IRQ	R/W1S	0h	Dummy IRQ STATUS- Reserved for future use 0h = Write-0 : No action, Read-0 : No event pending 1h = Write-1 : Set event, Read-1 : IRQ event pending
15	DUMMY1_IRQ	R/W1S	0h	Dummy IRQ STATUS- Reserved for future use 0h = Write-0 : No action, Read-0 : No event pending 1h = Write-1 : Set event, Read-1 : IRQ event pending
14	WB_IRQ	R/W1S	0h	WB IRQ STATUS. Register indicates the WB pipeline interrupt events if WB pipeline is present 0h = Write-0 : No action, Read-0 : No event pending 1h = Write-1 : Set event, Read-1 : IRQ event pending
13	RESERVED	R	0h	Reserved

**Table 12-574. DSS0\_COMMON\_DISPC\_IRQSTATUS\_RAW Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
12-8	RESERVED	R	0h	Reserved
7-4	VID_IRQ	R/W1S	0h	VID IRQ STATUS. Register indicates the Video pipeline[s] interrupt events. [0] -> VID1, [1] -> VIDL1, [2] -> VID2, [3] -> VIDL2  0h = Write-0 : No action, Read-0 : No event pending 1h = Write-1 : Set event, Read-1 : IRQ event pending
3-0	VP_IRQ	R/W1S	0h	VP [3:0] IRQ STATUS. Register indicates the Video Port[s] interrupt events  0h = Write-0 : No action, Read-0 : No event pending 1h = Write-1 : Set event, Read-1 : IRQ event pending

### 12.6.4.11.13.1.5 DSS0\_COMMON\_DISPC\_IRQSTATUS Register (Offset = 2Ch) [reset = 0h]

DSS0\_COMMON\_DISPC\_IRQSTATUS is shown in [Figure 12-616](#) and described in [Table 12-576](#).

Return to [Summary Table](#).

Interrupt status. Enabled status, isn't set unless event is enabled. Write 1 to clear the status after interrupt has been serviced. RAW status also gets cleared, i.e. even if not enabled

**Table 12-575. DSS0\_COMMON\_DISPC\_IRQSTATUS Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 002Ch
DSS0_DISPC_0_COMMON_S0	04A1 002Ch
DSS0_DISPC_0_COMMON_S1	04B0 002Ch
DSS0_DISPC_0_COMMON_S2	04B1 002Ch

**Figure 12-616. DSS0\_COMMON\_DISPC\_IRQSTATUS Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							DUMMY_IRQ
R-0h							R/W1C-0h
15	14	13	12	11	10	9	8
DUMMY1_IRQ	WB_IRQ	RESERVED	RESERVED				
R/W1C-0h	R/W1C-0h	R-0h	R-0h				
7	6	5	4	3	2	1	0
VID_IRQ				VP_IRQ			
R/W1C-0h				R/W1C-0h			

LEGEND: R = Read Only; R/W1C = Read/Write 1 to Clear Bit; -n = value after reset

**Table 12-576. DSS0\_COMMON\_DISPC\_IRQSTATUS Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reserved
16	DUMMY_IRQ	R/W1C	0h	Dummy IRQ STATUS-Reserved for future use 0h = Write-0 : No action, Read-0 : No event pending 1h = Write-1 : Clear pending event, if any, Read-1 : IRQ event pending
15	DUMMY1_IRQ	R/W1C	0h	Dummy IRQ STATUS-Reserved for future use 0h = Write-0 : No action, Read-0 : No event pending 1h = Write-1 : Clear pending event, if any, Read-1 : IRQ event pending



**Table 12-576. DSS0\_COMMON\_DISPC\_IRQSTATUS Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
14	WB_IRQ	R/W1C	0h	WB IRQ STATUS. Register indicates the WB pipeline interrupt events if WB pipeline is present  0h = Write-0 : No action, Read-0 : No event pending  1h = Write-1 : Clear pending event, if any, Read-1 : IRQ event pending
13	RESERVED	R	0h	Reserved
12-8	RESERVED	R	0h	Reserved
7-4	VID_IRQ	R/W1C	0h	VID IRQ STATUS. Register indicates the Video pipeline[s] interrupt events.[0] -> VID1, [1] -> VIDL1, [2] -> VID2, [3] -> VIDL2  0h = Write-0 : No action, Read-0 : No event pending  1h = Write-1 : Clear pending event, if any, Read-1 : IRQ event pending
3-0	VP_IRQ	R/W1C	0h	VP [3:0] IRQ STATUS. Register indicates the Video Port[s] interrupt events  0h = Write-0 : No action, Read-0 : No event pending  1h = Write-1 : Clear pending event, if any, Read-1 : IRQ event pending

### 12.6.4.11.13.1.6 DSS0\_COMMON\_DISPC\_IRQENABLE\_SET Register (Offset = 30h) [reset = 0h]

DSS0\_COMMON\_DISPC\_IRQENABLE\_SET is shown in [Figure 12-617](#) and described in [Table 12-578](#).

Return to [Summary Table](#).

SET Interrupt enable. Write 1 to set interrupt enable. Readout equal to corresponding \_CLR register

**Table 12-577.**  
**DSS0\_COMMON\_DISPC\_IRQENABLE\_SET**  
**Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 0030h
DSS0_DISPC_0_COMMON_S0	04A1 0030h
DSS0_DISPC_0_COMMON_S1	04B0 0030h
DSS0_DISPC_0_COMMON_S2	04B1 0030h

**Figure 12-617. DSS0\_COMMON\_DISPC\_IRQENABLE\_SET Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							SET_DUMMY_I RQ
R-0h							R/W1S-0h
15	14	13	12	11	10	9	8
SET_DUMMY1 _IRQ	SET_WB_IRQ	RESERVED	RESERVED				
R/W1S-0h	R/W1S-0h	R-0h	R-0h				
7	6	5	4	3	2	1	0
SET_VID_IRQ				SET_VP_IRQ			
R/W1S-0h				R/W1S-0h			

LEGEND: R = Read Only; R/W1S = Read/Write 1 to Set Bit; -n = value after reset

**Table 12-578. DSS0\_COMMON\_DISPC\_IRQENABLE\_SET Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reserved
16	SET_DUMMY_IRQ	R/W1S	0h	Dummy IRQ 0h = Write-0 : No action, Read-0 : Interrupt Disabled 1h = Write-1 : Enable interrupt, Read-1 : Interrupt Enabled
15	SET_DUMMY1_IRQ	R/W1S	0h	Dummy IRQ 0h = Write-0 : No action, Read-0 : Interrupt Disabled 1h = Write-1 : Enable interrupt, Read-1 : Interrupt Enabled
14	SET_WB_IRQ	R/W1S	0h	WB IRQ, if WB pipeline is present 0h = Write-0 : No action, Read-0 : Interrupt Disabled 1h = Write-1 : Enable interrupt, Read-1 : Interrupt Enabled
13	RESERVED	R	0h	Reserved

**Table 12-578. DSS0\_COMMON\_DISPC\_IRQENABLE\_SET Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
12-8	RESERVED	R	0h	Reserved
7-4	SET_VID_IRQ	R/W1S	0h	VID IRQ. [0] -> VID1, [1] -> VIDL1, [2] -> VID2, [3] -> VIDL2  0h = Write-0 : No action, Read-0 : Interrupt Disabled 1h = Write-1 : Enable interrupt, Read-1 : Interrupt Enabled
3-0	SET_VP_IRQ	R/W1S	0h	VP [3:0] IRQ  0h = Write-0 : No action, Read-0 : Interrupt Disabled 1h = Write-1 : Enable interrupt, Read-1 : Interrupt Enabled

### 12.6.4.11.13.1.7 DSS0\_COMMON\_DISPC\_IRQENABLE\_CLR Register (Offset = 34h) [reset = 0h]

DSS0\_COMMON\_DISPC\_IRQENABLE\_CLR is shown in [Figure 12-618](#) and described in [Table 12-580](#).

Return to [Summary Table](#).

CLR Interrupt enable. Write 1 to clear interrupt enable

**Table 12-579.**  
**DSS0\_COMMON\_DISPC\_IRQENABLE\_CLR**  
**Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 0034h
DSS0_DISPC_0_COMMON_S0	04A1 0034h
DSS0_DISPC_0_COMMON_S1	04B0 0034h
DSS0_DISPC_0_COMMON_S2	04B1 0034h

**Figure 12-618. DSS0\_COMMON\_DISPC\_IRQENABLE\_CLR Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							CLR_DUMMY_I RQ
R-0h							R/W1C-0h
15	14	13	12	11	10	9	8
CLR_DUMMY1 _IRQ	CLR_WB_IRQ	RESERVED	RESERVED				
R/W1C-0h	R/W1C-0h	R-0h	R-0h				
7	6	5	4	3	2	1	0
CLR_VID_IRQ				CLR_VP_IRQ			
R/W1C-0h				R/W1C-0h			

LEGEND: R = Read Only; R/W1C = Read/Write 1 to Clear Bit; -n = value after reset

**Table 12-580. DSS0\_COMMON\_DISPC\_IRQENABLE\_CLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reserved
16	CLR_DUMMY_IRQ	R/W1C	0h	Dummy IRQ 0h = Write-0 : No action, Read-0 : Interrupt Disabled 1h = Write-1 : Clear interrupt, Read-1 : Interrupt Enabled
15	CLR_DUMMY1_IRQ	R/W1C	0h	Dummy IRQ 0h = Write-0 : No action, Read-0 : Interrupt Disabled 1h = Write-1 : Clear interrupt, Read-1 : Interrupt Enabled
14	CLR_WB_IRQ	R/W1C	0h	WB IRQ, if WB pipeline is present 0h = Write-0 : No action, Read-0 : Interrupt Disabled 1h = Write-1 : Clear interrupt, Read-1 : Interrupt Enabled
13	RESERVED	R	0h	Reserved

**Table 12-580. DSS0\_COMMON\_DISPC\_IRQENABLE\_CLR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
12-8	RESERVED	R	0h	Reserved
7-4	CLR_VID_IRQ	R/W1C	0h	VID IRQ.[0] -> VID1, [1] -> VIDL1, [2] -> VID2, [3] -> VIDL2  0h = Write-0 : No action, Read-0 : Interrupt Disabled  1h = Write-1 : Clear interrupt, Read-1 : Interrupt Enabled
3-0	CLR_VP_IRQ	R/W1C	0h	VP [3:0] IRQ  0h = Write-0 : No action, Read-0 : Interrupt Disabled  1h = Write-1 : Clear interrupt, Read-1 : Interrupt Enabled

### 12.6.4.11.13.1.8 DSS0\_COMMON\_VID\_IRQENABLE\_0 Register (Offset = 38h) [reset = 0h]

DSS0\_COMMON\_VID\_IRQENABLE\_0 is shown in [Figure 12-619](#) and described in [Table 12-582](#).

Return to [Summary Table](#).

This register allows to mask/unmask the VID internal sources of interrupt, on an event-by-event basis.[0] -> VID1, [1] -> VIDL1, [2] -> VID2, [3] -> VIDL2

**Table 12-581. DSS0\_COMMON\_VID\_IRQENABLE\_0 Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 0038h
DSS0_DISPC_0_COMMON_S0	04A1 0038h
DSS0_DISPC_0_COMMON_S1	04B0 0038h
DSS0_DISPC_0_COMMON_S2	04B1 0038h

**Figure 12-619. DSS0\_COMMON\_VID\_IRQENABLE\_0 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED			FBDC_ILLEGALTILEREQ_EN	FBDC_CORRUPTTILE_EN	SAFETYREGION_EN	VIDENDWINDOW_EN	VIDBUFFERUNDERFLOW_EN
R-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-582. DSS0\_COMMON\_VID\_IRQENABLE\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reserved
4	FBDC_ILLEGALTILEREQ_EN	R/W	0h	FBDC IRQ, Illegal tile req detected 0h = Event is masked 1h = Event generates an interrupt when it occurs
3	FBDC_CORRUPTTILE_EN	R/W	0h	FBDC IRQ. Corrupt tile detected 0h = Event is masked 1h = Event generates an interrupt when it occurs
2	SAFETYREGION_EN	R/W	0h	Safety Feature IRQ. This is raised when FrameFreeze is detected or data mismatch occurs within the safety region 0h = Event is masked 1h = Event generates an interrupt when it occurs

**Table 12-582. DSS0\_COMMON\_VID\_IRQENABLE\_0 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	VIDENDWINDOW_EN	R/W	0h	Video End Window. This is raised by the DMA engine when the full video data has been sent to the pipeline  0h = Event is masked 1h = Event generates an interrupt when it occurs
0	VIDBUFFERUNDERFLOW_EN	R/W	0h	Video DMA Buffer Underflow. This is raised when the DMA buffer does not have the data requested by the Video pipeline  0h = Event is masked 1h = Event generates an interrupt when it occurs

### 12.6.4.11.13.1.9 DSS0\_COMMON\_VID\_IRQENABLE\_1 Register (Offset = 3Ch) [reset = 0h]

DSS0\_COMMON\_VID\_IRQENABLE\_1 is shown in [Figure 12-620](#) and described in [Table 12-584](#).

Return to [Summary Table](#).

This register allows to mask/unmask the VID internal sources of interrupt, on an event-by-event basis.[0] -> VID1, [1] -> VIDL1, [2] -> VID2, [3] -> VIDL2

**Table 12-583. DSS0\_COMMON\_VID\_IRQENABLE\_1 Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 003Ch
DSS0_DISPC_0_COMMON_S0	04A1 003Ch
DSS0_DISPC_0_COMMON_S1	04B0 003Ch
DSS0_DISPC_0_COMMON_S2	04B1 003Ch

**Figure 12-620. DSS0\_COMMON\_VID\_IRQENABLE\_1 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED			FBDC_ILLEGAL_TILE_REQ_EN	FBDC_CORRUPT_TILE_EN	SAFETY_REGION_EN	VID_END_WINDOW_EN	VID_BUFFER_UNDERFLOW_EN
R-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-584. DSS0\_COMMON\_VID\_IRQENABLE\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reserved
4	FBDC_ILLEGAL_TILE_REQ_EN	R/W	0h	FBDC IRQ, Illegal tile req detected 0h = Event is masked 1h = Event generates an interrupt when it occurs
3	FBDC_CORRUPT_TILE_EN	R/W	0h	FBDC IRQ. Corrupt tile detected 0h = Event is masked 1h = Event generates an interrupt when it occurs
2	SAFETY_REGION_EN	R/W	0h	Safety Feature IRQ. This is raised when FrameFreeze is detected or data mismatch occurs within the safety region 0h = Event is masked 1h = Event generates an interrupt when it occurs



**Table 12-584. DSS0\_COMMON\_VID\_IRQENABLE\_1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	VIDENDWINDOW_EN	R/W	0h	Video End Window. This is raised by the DMA engine when the full video data has been sent to the pipeline  0h = Event is masked 1h = Event generates an interrupt when it occurs
0	VIDBUFFERUNDERFLOW_EN	R/W	0h	Video DMA Buffer Underflow. This is raised when the DMA buffer does not have the data requested by the Video pipeline  0h = Event is masked 1h = Event generates an interrupt when it occurs

### 12.6.4.11.13.1.10 DSS0\_COMMON\_VID\_IRQENABLE\_2 Register (Offset = 40h) [reset = 0h]

DSS0\_COMMON\_VID\_IRQENABLE\_2 is shown in [Figure 12-621](#) and described in [Table 12-586](#).

Return to [Summary Table](#).

This register allows to mask/unmask the VID internal sources of interrupt, on an event-by-event basis.[0] -> VID1, [1] -> VIDL1, [2] -> VID2, [3] -> VIDL2

**Table 12-585. DSS0\_COMMON\_VID\_IRQENABLE\_2 Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 0040h
DSS0_DISPC_0_COMMON_S0	04A1 0040h
DSS0_DISPC_0_COMMON_S1	04B0 0040h
DSS0_DISPC_0_COMMON_S2	04B1 0040h

**Figure 12-621. DSS0\_COMMON\_VID\_IRQENABLE\_2 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED			FBDC_ILLEGAL_TILE_REQ_EN	FBDC_CORRUPT_TILE_EN	SAFETY_REGION_EN	VID_END_WINDOW_EN	VID_BUFFER_UNDERFLOW_EN
R-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-586. DSS0\_COMMON\_VID\_IRQENABLE\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reserved
4	FBDC_ILLEGAL_TILE_REQ_EN	R/W	0h	FBDC IRQ, Illegal tile req detected 0h = Event is masked 1h = Event generates an interrupt when it occurs
3	FBDC_CORRUPT_TILE_EN	R/W	0h	FBDC IRQ. Corrupt tile detected 0h = Event is masked 1h = Event generates an interrupt when it occurs
2	SAFETY_REGION_EN	R/W	0h	Safety Feature IRQ. This is raised when FrameFreeze is detected or data mismatch occurs within the safety region 0h = Event is masked 1h = Event generates an interrupt when it occurs

**Table 12-586. DSS0\_COMMON\_VID\_IRQENABLE\_2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	VIDENDWINDOW_EN	R/W	0h	Video End Window. This is raised by the DMA engine when the full video data has been sent to the pipeline  0h = Event is masked 1h = Event generates an interrupt when it occurs
0	VIDBUFFERUNDERFLOW_EN	R/W	0h	Video DMA Buffer Underflow. This is raised when the DMA buffer does not have the data requested by the Video pipeline  0h = Event is masked 1h = Event generates an interrupt when it occurs

### 12.6.4.11.13.1.11 DSS0\_COMMON\_VID\_IRQENABLE\_3 Register (Offset = 44h) [reset = 0h]

DSS0\_COMMON\_VID\_IRQENABLE\_3 is shown in [Figure 12-622](#) and described in [Table 12-588](#).

Return to [Summary Table](#).

This register allows to mask/unmask the VID internal sources of interrupt, on an event-by-event basis.[0] -> VID1, [1] -> VIDL1, [2] -> VID2, [3] -> VIDL2

**Table 12-587. DSS0\_COMMON\_VID\_IRQENABLE\_3 Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 0044h
DSS0_DISPC_0_COMMON_S0	04A1 0044h
DSS0_DISPC_0_COMMON_S1	04B0 0044h
DSS0_DISPC_0_COMMON_S2	04B1 0044h

**Figure 12-622. DSS0\_COMMON\_VID\_IRQENABLE\_3 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED			FBDC_ILLEGALTILEREQ_EN	FBDC_CORRUPTTILE_EN	SAFETYREGION_EN	VIDENDWINDOW_EN	VIDBUFFERUNDERFLOW_EN
R-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-588. DSS0\_COMMON\_VID\_IRQENABLE\_3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reserved
4	FBDC_ILLEGALTILEREQ_EN	R/W	0h	FBDC IRQ, Illegal tile req detected 0h = Event is masked 1h = Event generates an interrupt when it occurs
3	FBDC_CORRUPTTILE_EN	R/W	0h	FBDC IRQ. Corrupt tile detected 0h = Event is masked 1h = Event generates an interrupt when it occurs
2	SAFETYREGION_EN	R/W	0h	Safety Feature IRQ. This is raised when FrameFreeze is detected or data mismatch occurs within the safety region 0h = Event is masked 1h = Event generates an interrupt when it occurs

**Table 12-588. DSS0\_COMMON\_VID\_IRQENABLE\_3 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	VIDENDWINDOW_EN	R/W	0h	Video End Window. This is raised by the DMA engine when the full video data has been sent to the pipeline  0h = Event is masked 1h = Event generates an interrupt when it occurs
0	VIDBUFFERUNDERFLOW_EN	R/W	0h	Video DMA Buffer Underflow. This is raised when the DMA buffer does not have the data requested by the Video pipeline  0h = Event is masked 1h = Event generates an interrupt when it occurs

### 12.6.4.11.13.1.12 DSS0\_COMMON\_VID\_IRQSTATUS\_0 Register (Offset = 48h) [reset = 0h]

DSS0\_COMMON\_VID\_IRQSTATUS\_0 is shown in [Figure 12-623](#) and described in [Table 12-590](#).

Return to [Summary Table](#).

This register groups all the status of the VID internal events that generate an interrupt. Write 1 to a clear a bit field.[0] -> VID1, [1] -> VIDL1, [2] -> VID2, [3] -> VIDL2

**Table 12-589. DSS0\_COMMON\_VID\_IRQSTATUS\_0 Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 0048h
DSS0_DISPC_0_COMMON_S0	04A1 0048h
DSS0_DISPC_0_COMMON_S1	04B0 0048h
DSS0_DISPC_0_COMMON_S2	04B1 0048h

**Figure 12-623. DSS0\_COMMON\_VID\_IRQSTATUS\_0 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED			FBDC_ILLEGAL_TILE_REQ_IRQ	FBDC_CORRUPT_TILE_IRQ	SAFETY_REGION_IRQ	VID_END_WINDOW_IRQ	VID_BUFFER_UNDERFLOW_IRQ
R-0h			R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h

LEGEND: R = Read Only; R/W1C = Read/Write 1 to Clear Bit; -n = value after reset

**Table 12-590. DSS0\_COMMON\_VID\_IRQSTATUS\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reserved
4	FBDC_ILLEGAL_TILE_REQ_IRQ	R/W1C	0h	FBDC IRQ, Illegal tile req detected  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
3	FBDC_CORRUPT_TILE_IRQ	R/W1C	0h	FBDC IRQ. Corrupt tile detected  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending

**Table 12-590. DSS0\_COMMON\_VID\_IRQSTATUS\_0 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	SAFETYREGION_IRQ	R/W1C	0h	<p>Safety Feature IRQ.</p> <p>This is raised when FrameFreeze is detected or data mismatch occurs within the safety region</p> <p>0h = Write: Status is unchanged, Read: Event is not pending</p> <p>1h = Write: Status is reset, Read: Event is Pending</p>
1	VIDENDWINDOW_IRQ	R/W1C	0h	<p>Video End Window.</p> <p>This is raised by the DMA engine when the full video data has been sent to the pipeline</p> <p>0h = Write: Status is unchanged, Read: Event is not pending</p> <p>1h = Write: Status is reset, Read: Event is Pending</p>
0	VIDBUFFERUNDERFLOW_IRQ	R/W1C	0h	<p>Video DMA Buffer Underflow.</p> <p>This is raised when the DMA buffer does not have the data requested by the Video pipeline</p> <p>0h = Write: Status is unchanged, Read: Event is not pending</p> <p>1h = Write: Status is reset, Read: Event is Pending</p>

### 12.6.4.11.13.1.13 DSS0\_COMMON\_VID\_IRQSTATUS\_1 Register (Offset = 4Ch) [reset = 0h]

DSS0\_COMMON\_VID\_IRQSTATUS\_1 is shown in [Figure 12-624](#) and described in [Table 12-592](#).

Return to [Summary Table](#).

This register groups all the status of the VID internal events that generate an interrupt. Write 1 to a clear a bit field.[0] -> VID1, [1] -> VIDL1, [2] -> VID2, [3] -> VIDL2

**Table 12-591. DSS0\_COMMON\_VID\_IRQSTATUS\_1 Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 004Ch
DSS0_DISPC_0_COMMON_S0	04A1 004Ch
DSS0_DISPC_0_COMMON_S1	04B0 004Ch
DSS0_DISPC_0_COMMON_S2	04B1 004Ch

**Figure 12-624. DSS0\_COMMON\_VID\_IRQSTATUS\_1 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED			FBDC_ILLEGAL_TILE_REQ_IRQ	FBDC_CORRUPT_TILE_IRQ	SAFETYREGION_IRQ	VIDENDWINDOW_IRQ	VIDBUFFERUNDERFLOW_IRQ
R-0h			R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h

LEGEND: R = Read Only; R/W1C = Read/Write 1 to Clear Bit; -n = value after reset

**Table 12-592. DSS0\_COMMON\_VID\_IRQSTATUS\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reserved
4	FBDC_ILLEGAL_TILE_REQ_IRQ	R/W1C	0h	FBDC IRQ, Illegal tile req detected  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
3	FBDC_CORRUPT_TILE_IRQ	R/W1C	0h	FBDC IRQ. Corrupt tile detected  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending



**Table 12-592. DSS0\_COMMON\_VID\_IRQSTATUS\_1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	SAFETYREGION_IRQ	R/W1C	0h	<p>Safety Feature IRQ.</p> <p>This is raised when FrameFreeze is detected or data mismatch occurs within the safety region</p> <p>0h = Write: Status is unchanged, Read: Event is not pending</p> <p>1h = Write: Status is reset, Read: Event is Pending</p>
1	VIDENDWINDOW_IRQ	R/W1C	0h	<p>Video End Window.</p> <p>This is raised by the DMA engine when the full video data has been sent to the pipeline</p> <p>0h = Write: Status is unchanged, Read: Event is not pending</p> <p>1h = Write: Status is reset, Read: Event is Pending</p>
0	VIDBUFFERUNDERFLOW_IRQ	R/W1C	0h	<p>Video DMA Buffer Underflow.</p> <p>This is raised when the DMA buffer does not have the data requested by the Video pipeline</p> <p>0h = Write: Status is unchanged, Read: Event is not pending</p> <p>1h = Write: Status is reset, Read: Event is Pending</p>

### 12.6.4.11.13.1.14 DSS0\_COMMON\_VID\_IRQSTATUS\_2 Register (Offset = 50h) [reset = 0h]

DSS0\_COMMON\_VID\_IRQSTATUS\_2 is shown in [Figure 12-625](#) and described in [Table 12-594](#).

Return to [Summary Table](#).

This register groups all the status of the VID internal events that generate an interrupt. Write 1 to a clear a bit field.[0] -> VID1, [1] -> VIDL1, [2] -> VID2, [3] -> VIDL2

**Table 12-593. DSS0\_COMMON\_VID\_IRQSTATUS\_2 Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 0050h
DSS0_DISPC_0_COMMON_S0	04A1 0050h
DSS0_DISPC_0_COMMON_S1	04B0 0050h
DSS0_DISPC_0_COMMON_S2	04B1 0050h

**Figure 12-625. DSS0\_COMMON\_VID\_IRQSTATUS\_2 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED			FBDC_ILLEGAL_TILE_REQ_IRQ	FBDC_CORRUPT_TILE_IRQ	SAFETY_REGION_IRQ	VID_END_WINDOW_IRQ	VID_BUFFER_UNDERFLOW_IRQ
R-0h			R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h

LEGEND: R = Read Only; R/W1C = Read/Write 1 to Clear Bit; -n = value after reset

**Table 12-594. DSS0\_COMMON\_VID\_IRQSTATUS\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reserved
4	FBDC_ILLEGAL_TILE_REQ_IRQ	R/W1C	0h	FBDC IRQ, Illegal tile req detected  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
3	FBDC_CORRUPT_TILE_IRQ	R/W1C	0h	FBDC IRQ. Corrupt tile detected  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending

**Table 12-594. DSS0\_COMMON\_VID\_IRQSTATUS\_2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	SAFETYREGION_IRQ	R/W1C	0h	<p>Safety Feature IRQ.</p> <p>This is raised when FrameFreeze is detected or data mismatch occurs within the safety region</p> <p>0h = Write: Status is unchanged, Read: Event is not pending</p> <p>1h = Write: Status is reset, Read: Event is Pending</p>
1	VIDENDWINDOW_IRQ	R/W1C	0h	<p>Video End Window.</p> <p>This is raised by the DMA engine when the full video data has been sent to the pipeline</p> <p>0h = Write: Status is unchanged, Read: Event is not pending</p> <p>1h = Write: Status is reset, Read: Event is Pending</p>
0	VIDBUFFERUNDERFLOW_IRQ	R/W1C	0h	<p>Video DMA Buffer Underflow.</p> <p>This is raised when the DMA buffer does not have the data requested by the Video pipeline</p> <p>0h = Write: Status is unchanged, Read: Event is not pending</p> <p>1h = Write: Status is reset, Read: Event is Pending</p>

### 12.6.4.11.13.1.15 DSS0\_COMMON\_VID\_IRQSTATUS\_3 Register (Offset = 54h) [reset = 0h]

DSS0\_COMMON\_VID\_IRQSTATUS\_3 is shown in [Figure 12-626](#) and described in [Table 12-596](#).

Return to [Summary Table](#).

This register groups all the status of the VID internal events that generate an interrupt. Write 1 to a clear a bit field.[0] -> VID1, [1] -> VIDL1, [2] -> VID2, [3] -> VIDL2

**Table 12-595. DSS0\_COMMON\_VID\_IRQSTATUS\_3 Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 0054h
DSS0_DISPC_0_COMMON_S0	04A1 0054h
DSS0_DISPC_0_COMMON_S1	04B0 0054h
DSS0_DISPC_0_COMMON_S2	04B1 0054h

**Figure 12-626. DSS0\_COMMON\_VID\_IRQSTATUS\_3 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED			FBDC_ILLEGAL_TILE_REQ_IRQ	FBDC_CORRUPT_TILE_IRQ	SAFETY_REGION_IRQ	VID_END_WINDOW_IRQ	VID_BUFFER_UNDERFLOW_IRQ
R-0h			R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h

LEGEND: R = Read Only; R/W1C = Read/Write 1 to Clear Bit; -n = value after reset

**Table 12-596. DSS0\_COMMON\_VID\_IRQSTATUS\_3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reserved
4	FBDC_ILLEGAL_TILE_REQ_IRQ	R/W1C	0h	FBDC IRQ, Illegal tile req detected  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
3	FBDC_CORRUPT_TILE_IRQ	R/W1C	0h	FBDC IRQ. Corrupt tile detected  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending

**Table 12-596. DSS0\_COMMON\_VID\_IRQSTATUS\_3 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
2	SAFETYREGION_IRQ	R/W1C	0h	<p>Safety Feature IRQ.</p> <p>This is raised when FrameFreeze is detected or data mismatch occurs within the safety region</p> <p>0h = Write: Status is unchanged, Read: Event is not pending</p> <p>1h = Write: Status is reset, Read: Event is Pending</p>
1	VIDENDWINDOW_IRQ	R/W1C	0h	<p>Video End Window.</p> <p>This is raised by the DMA engine when the full video data has been sent to the pipeline</p> <p>0h = Write: Status is unchanged, Read: Event is not pending</p> <p>1h = Write: Status is reset, Read: Event is Pending</p>
0	VIDBUFFERUNDERFLOW_IRQ	R/W1C	0h	<p>Video DMA Buffer Underflow.</p> <p>This is raised when the DMA buffer does not have the data requested by the Video pipeline</p> <p>0h = Write: Status is unchanged, Read: Event is not pending</p> <p>1h = Write: Status is reset, Read: Event is Pending</p>

### 12.6.4.11.13.1.16 DSS0\_COMMON\_VP\_IRQENABLE\_0 Register (Offset = 58h) [reset = 0h]

DSS0\_COMMON\_VP\_IRQENABLE\_0 is shown in [Figure 12-627](#) and described in [Table 12-598](#).

Return to [Summary Table](#).

This register allows to mask/unmask the VP\_0 internal sources of interrupt, on an event-by-event basis

**Table 12-597. DSS0\_COMMON\_VP\_IRQENABLE\_0  
Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 0058h
DSS0_DISPC_0_COMMON_S0	04A1 0058h
DSS0_DISPC_0_COMMON_S1	04B0 0058h
DSS0_DISPC_0_COMMON_S2	04B1 0058h

**Figure 12-627. DSS0\_COMMON\_VP\_IRQENABLE\_0 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							SAFETYREGION1_EN
R-0h							R/W-0h
15	14	13	12	11	10	9	8
SAFETYREGION1_EN			DUMMY_EN	VPSYNC_EN	SECURITYVIO LATION_EN	SAFETYREGION_EN	
R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	
7	6	5	4	3	2	1	0
SAFETYREGION_EN		ACBIASCOUNT STATUS_EN	VPSYNCLST _EN	VPVPROGRAM MEDLINENUM BER_EN	VPVSYNC_OD D_EN	VPVSYNC_EN	VPFRAMEDON E_EN
R/W-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-598. DSS0\_COMMON\_VP\_IRQENABLE\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reserved
16-13	SAFETYREGION1_EN	R/W	0h	Safety Feature IRQ. This is raised when FrameFreeze is detected or data mismatch occurs within the safety region 4-7  0h = Event is masked 1h = Event generates an interrupt when it occurs
12	DUMMY_EN	R/W	0h	Dummy IRQ for future use  0h = Event is masked 1h = Event generates an interrupt when it occurs

**Table 12-598. DSS0\_COMMON\_VP\_IRQENABLE\_0 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	VPSYNC_EN	R/W	0h	Go bit clear event 0h = Event is masked 1h = Event generates an interrupt when it occurs
10	SECURITYVIOLATION_EN	R/W	0h	Security Violation interrupt for OVR/VP. Non-secure OVR/VP connected to secure VID 0h = Event is masked 1h = Event generates an interrupt when it occurs
9-6	SAFETYREGION_EN	R/W	0h	Safety Feature IRQ. This is raised when FrameFreeze is detected or data mismatch occurs within the safety region 0-3 0h = Event is masked 1h = Event generates an interrupt when it occurs
5	ACBIASCOUNTSTATUS_EN	R/W	0h	AC BIAS transition counter has decremented to zero 0h = Event is masked 1h = Event generates an interrupt when it occurs
4	VPSYNCLOST_EN	R/W	0h	Synchronization Lost for Video Port 0h = Event is masked 1h = Event generates an interrupt when it occurs
3	VPPROGRAMMEDLINENUMBER_EN	R/W	0h	Programmed Line Number. It indicates that the scan of the display has reached the programmed user line number 0h = Event is masked 1h = Event generates an interrupt when it occurs
2	VPVSYNC_ODD_EN	R/W	0h	VSYSN for odd field from interlace mode only 0h = Event is masked 1h = Event generates an interrupt when it occurs
1	VPVSYNC_EN	R/W	0h	Vertical Synchronization for VP 0h = Event is masked 1h = Event generates an interrupt when it occurs
0	VPFRAMEDONE_EN	R/W	0h	Frame Done for Video Port. VP output has been disabled by user. All the data have been sent 0h = Event is masked 1h = Event generates an interrupt when it occurs

### 12.6.4.11.13.1.17 DSS0\_COMMON\_VP\_IRQENABLE\_1 Register (Offset = 5Ch) [reset = 0h]

DSS0\_COMMON\_VP\_IRQENABLE\_1 is shown in [Figure 12-628](#) and described in [Table 12-600](#).

Return to [Summary Table](#).

This register allows to mask/unmask the VP\_1 internal sources of interrupt, on an event-by-event basis

**Table 12-599. DSS0\_COMMON\_VP\_IRQENABLE\_1  
Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 005Ch
DSS0_DISPC_0_COMMON_S0	04A1 005Ch
DSS0_DISPC_0_COMMON_S1	04B0 005Ch
DSS0_DISPC_0_COMMON_S2	04B1 005Ch

**Figure 12-628. DSS0\_COMMON\_VP\_IRQENABLE\_1 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							SAFETYREGION1_EN
R-0h							R/W-0h
15	14	13	12	11	10	9	8
SAFETYREGION1_EN			DUMMY_EN	VPSYNC_EN	SECURITYVIO LATION_EN	SAFETYREGION_EN	
R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	
7	6	5	4	3	2	1	0
SAFETYREGION_EN		ACBIASCOUNT STATUS_EN	VPSYNCLOST _EN	VPVPROGRAM MEDLINENUM BER_EN	VPVSYNC_OD D_EN	VPVSYNC_EN	VPFRAMEDON E_EN
R/W-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-600. DSS0\_COMMON\_VP\_IRQENABLE\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reserved
16-13	SAFETYREGION1_EN	R/W	0h	Safety Feature IRQ. This is raised when FrameFreeze is detected or data mismatch occurs within the safety region 4-7  0h = Event is masked 1h = Event generates an interrupt when it occurs
12	DUMMY_EN	R/W	0h	Dummy IRQ for future use  0h = Event is masked 1h = Event generates an interrupt when it occurs



**Table 12-600. DSS0\_COMMON\_VP\_IRQENABLE\_1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	VPSYNC_EN	R/W	0h	Go bit clear event 0h = Event is masked 1h = Event generates an interrupt when it occurs
10	SECURITYVIOLATION_EN	R/W	0h	Security Violation interrupt for OVR/VP. Non-secure OVR/VP connected to secure VID 0h = Event is masked 1h = Event generates an interrupt when it occurs
9-6	SAFETYREGION_EN	R/W	0h	Safety Feature IRQ. This is raised when FrameFreeze is detected or data mismatch occurs within the safety region 0-3 0h = Event is masked 1h = Event generates an interrupt when it occurs
5	ACBIASCOUNTSTATUS_EN	R/W	0h	AC BIAS transition counter has decremented to zero 0h = Event is masked 1h = Event generates an interrupt when it occurs
4	VPSYNCLOST_EN	R/W	0h	Synchronization Lost for Video Port 0h = Event is masked 1h = Event generates an interrupt when it occurs
3	VPPROGRAMMEDLINENUMBER_EN	R/W	0h	Programmed Line Number. It indicates that the scan of the display has reached the programmed user line number 0h = Event is masked 1h = Event generates an interrupt when it occurs
2	VPVSYNC_ODD_EN	R/W	0h	VSYNC for odd field from interlace mode only 0h = Event is masked 1h = Event generates an interrupt when it occurs
1	VPVSYNC_EN	R/W	0h	Vertical Synchronization for VP 0h = Event is masked 1h = Event generates an interrupt when it occurs
0	VPFRAMEDONE_EN	R/W	0h	Frame Done for Video Port. VP output has been disabled by user. All the data have been sent 0h = Event is masked 1h = Event generates an interrupt when it occurs

### 12.6.4.11.13.1.18 DSS0\_COMMON\_VP\_IRQENABLE\_2 Register (Offset = 60h) [reset = 0h]

DSS0\_COMMON\_VP\_IRQENABLE\_2 is shown in [Figure 12-629](#) and described in [Table 12-602](#).

Return to [Summary Table](#).

This register allows to mask/unmask the VP\_2 internal sources of interrupt, on an event-by-event basis

**Table 12-601. DSS0\_COMMON\_VP\_IRQENABLE\_2  
Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 0060h
DSS0_DISPC_0_COMMON_S0	04A1 0060h
DSS0_DISPC_0_COMMON_S1	04B0 0060h
DSS0_DISPC_0_COMMON_S2	04B1 0060h

**Figure 12-629. DSS0\_COMMON\_VP\_IRQENABLE\_2 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							SAFETYREGION1_EN
R-0h							R/W-0h
15	14	13	12	11	10	9	8
SAFETYREGION1_EN			DUMMY_EN	VPSYNC_EN	SECURITYVIO LATION_EN	SAFETYREGION_EN	
R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	
7	6	5	4	3	2	1	0
SAFETYREGION_EN		ACBIASCOUNT STATUS_EN	VPSYNCLST _EN	VPPROGRAM MEDLINENUM BER_EN	VPVSYNC_OD D_EN	VPVSYNC_EN	VPFRAMEDON E_EN
R/W-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-602. DSS0\_COMMON\_VP\_IRQENABLE\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reserved
16-13	SAFETYREGION1_EN	R/W	0h	Safety Feature IRQ. This is raised when FrameFreeze is detected or data mismatch occurs within the safety region 4-7 0h = Event is masked 1h = Event generates an interrupt when it occurs
12	DUMMY_EN	R/W	0h	Dummy IRQ for future use 0h = Event is masked 1h = Event generates an interrupt when it occurs

**Table 12-602. DSS0\_COMMON\_VP\_IRQENABLE\_2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	VPSYNC_EN	R/W	0h	Go bit clear event 0h = Event is masked 1h = Event generates an interrupt when it occurs
10	SECURITYVIOLATION_EN	R/W	0h	Security Violation interrupt for OVR/VP. Non-secure OVR/VP connected to secure VID 0h = Event is masked 1h = Event generates an interrupt when it occurs
9-6	SAFETYREGION_EN	R/W	0h	Safety Feature IRQ. This is raised when FrameFreeze is detected or data mismatch occurs within the safety region 0-3 0h = Event is masked 1h = Event generates an interrupt when it occurs
5	ACBIASCOUNTSTATUS_EN	R/W	0h	AC BIAS transition counter has decremented to zero 0h = Event is masked 1h = Event generates an interrupt when it occurs
4	VPSYNCLOST_EN	R/W	0h	Synchronization Lost for Video Port 0h = Event is masked 1h = Event generates an interrupt when it occurs
3	VPPROGRAMMEDLINENUMBER_EN	R/W	0h	Programmed Line Number. It indicates that the scan of the display has reached the programmed user line number 0h = Event is masked 1h = Event generates an interrupt when it occurs
2	VPVSYNC_ODD_EN	R/W	0h	VSYNC for odd field from interlace mode only 0h = Event is masked 1h = Event generates an interrupt when it occurs
1	VPVSYNC_EN	R/W	0h	Vertical Synchronization for VP 0h = Event is masked 1h = Event generates an interrupt when it occurs
0	VPFRAMEDONE_EN	R/W	0h	Frame Done for Video Port. VP output has been disabled by user. All the data have been sent 0h = Event is masked 1h = Event generates an interrupt when it occurs

### 12.6.4.11.13.1.19 DSS0\_COMMON\_VP\_IRQENABLE\_3 Register (Offset = 64h) [reset = 0h]

DSS0\_COMMON\_VP\_IRQENABLE\_3 is shown in [Figure 12-630](#) and described in [Table 12-604](#).

Return to [Summary Table](#).

This register allows to mask/unmask the VP\_3 internal sources of interrupt, on an event-by-event basis

**Table 12-603. DSS0\_COMMON\_VP\_IRQENABLE\_3  
Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 0064h
DSS0_DISPC_0_COMMON_S0	04A1 0064h
DSS0_DISPC_0_COMMON_S1	04B0 0064h
DSS0_DISPC_0_COMMON_S2	04B1 0064h

**Figure 12-630. DSS0\_COMMON\_VP\_IRQENABLE\_3 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							SAFETYREGION1_EN
R-0h							R/W-0h
15	14	13	12	11	10	9	8
SAFETYREGION1_EN			DUMMY_EN	VPSYNC_EN	SECURITYVIO LATION_EN	SAFETYREGION_EN	
R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	
7	6	5	4	3	2	1	0
SAFETYREGION_EN		ACBIASCOUNT STATUS_EN	VPSYNCLST _EN	VPVPROGRAM MEDLINENUM BER_EN	VPVSYNC_OD D_EN	VPVSYNC_EN	VPFRAMEDON E_EN
R/W-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-604. DSS0\_COMMON\_VP\_IRQENABLE\_3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reserved
16-13	SAFETYREGION1_EN	R/W	0h	Safety Feature IRQ. This is raised when FrameFreeze is detected or data mismatch occurs within the safety region 4-7 0h = Event is masked 1h = Event generates an interrupt when it occurs
12	DUMMY_EN	R/W	0h	Dummy IRQ for future use 0h = Event is masked 1h = Event generates an interrupt when it occurs

**Table 12-604. DSS0\_COMMON\_VP\_IRQENABLE\_3 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	VPSYNC_EN	R/W	0h	Go bit clear event 0h = Event is masked 1h = Event generates an interrupt when it occurs
10	SECURITYVIOLATION_EN	R/W	0h	Security Violation interrupt for OVR/VP. Non-secure OVR/VP connected to secure VID 0h = Event is masked 1h = Event generates an interrupt when it occurs
9-6	SAFETYREGION_EN	R/W	0h	Safety Feature IRQ. This is raised when FrameFreeze is detected or data mismatch occurs within the safety region 0-3 0h = Event is masked 1h = Event generates an interrupt when it occurs
5	ACBIASCOUNTSTATUS_EN	R/W	0h	AC BIAS transition counter has decremented to zero 0h = Event is masked 1h = Event generates an interrupt when it occurs
4	VPSYNCLOST_EN	R/W	0h	Synchronization Lost for Video Port 0h = Event is masked 1h = Event generates an interrupt when it occurs
3	VPPROGRAMMEDLINENUMBER_EN	R/W	0h	Programmed Line Number. It indicates that the scan of the display has reached the programmed user line number 0h = Event is masked 1h = Event generates an interrupt when it occurs
2	VPVSYNC_ODD_EN	R/W	0h	VSYNC for odd field from interlace mode only 0h = Event is masked 1h = Event generates an interrupt when it occurs
1	VPVSYNC_EN	R/W	0h	Vertical Synchronization for VP 0h = Event is masked 1h = Event generates an interrupt when it occurs
0	VPFRAMEDONE_EN	R/W	0h	Frame Done for Video Port. VP output has been disabled by user. All the data have been sent 0h = Event is masked 1h = Event generates an interrupt when it occurs

### 12.6.4.11.13.1.20 DSS0\_COMMON\_VP\_IRQSTATUS\_0 Register (Offset = 68h) [reset = 0h]

DSS0\_COMMON\_VP\_IRQSTATUS\_0 is shown in [Figure 12-631](#) and described in [Table 12-606](#).

Return to [Summary Table](#).

This register groups all the status of the VP\_0 internal events that generate an interrupt. Write 1 to a given bit resets this bit

**Table 12-605. DSS0\_COMMON\_VP\_IRQSTATUS\_0 Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 0068h
DSS0_DISPC_0_COMMON_S0	04A1 0068h
DSS0_DISPC_0_COMMON_S1	04B0 0068h
DSS0_DISPC_0_COMMON_S2	04B1 0068h

**Figure 12-631. DSS0\_COMMON\_VP\_IRQSTATUS\_0 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							SAFETYREGION1_IRQ
R-0h							R/W1C-0h
15	14	13	12	11	10	9	8
SAFETYREGION1_IRQ			DUMMY_IRQ	VPSYNC_IRQ	SECURITYVIO LATION_IRQ	SAFETYREGION_IRQ	
R/W1C-0h			R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	
7	6	5	4	3	2	1	0
SAFETYREGION_IRQ		ACBIASCOUNT STATUS_IRQ	VPSYNCLIST _IRQ	VPVPROGRAM MEDLINENUM BER_IRQ	VPVSYNC_OD D_IRQ	VPVSYNC_IRQ	VPFRAMEDON E_IRQ
R/W1C-0h		R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h

LEGEND: R = Read Only; R/W1C = Read/Write 1 to Clear Bit; -n = value after reset

**Table 12-606. DSS0\_COMMON\_VP\_IRQSTATUS\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reserved
16-13	SAFETYREGION1_IRQ	R/W1C	0h	Safety Feature IRQ. This is raised when FrameFreeze is detected or data mismatch occurs within the safety region 4-7 0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
12	DUMMY_IRQ	R/W1C	0h	Dummy IRQ for future use 0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending

**Table 12-606. DSS0\_COMMON\_VP\_IRQSTATUS\_0 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	VPSYNC_IRQ	R/W1C	0h	Go bit clear event  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
10	SECURITYVIOLATION_IRQ	R/W1C	0h	Security Violation IRQ. Non-secure OVR/VP connected to secure VID  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
9-6	SAFETYREGION_IRQ	R/W1C	0h	Safety Feature IRQ. This is raised when FrameFreeze is detected or data mismatch occurs within the safety region 0-3  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
5	ACBIASCOUNTSTATUS_IRQ	R/W1C	0h	AC BIAS transition counter has decremented to zero  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
4	VPSYNCLOST_IRQ	R/W1C	0h	Synchronization Lost on VP output. The required data are not output at the correct time due to too short blanking periods or stall of at least one pipelines associated with VP output  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
3	VPPROGRAMMEDLINENUMBER_IRQ	R/W1C	0h	Programmed Line Number. It indicates that the scan of the display has reached the programmed user line number  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
2	VPVSYNC_ODD_IRQ	R/W1C	0h	VSYNC for odd field. For interlace mode only  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
1	VPVSYNC_IRQ	R/W1C	0h	Vertical Synchronization for VP output. It is used as VSYNC_EVEN in case of interlace mode  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
0	VPFRAMEDONE_IRQ	R/W1C	0h	Frame Done for VP. VP output has been disabled by user All the data have been sent  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending

### 12.6.4.11.13.1.21 DSS0\_COMMON\_VP\_IRQSTATUS\_1 Register (Offset = 6Ch) [reset = 0h]

DSS0\_COMMON\_VP\_IRQSTATUS\_1 is shown in [Figure 12-632](#) and described in [Table 12-608](#).

Return to [Summary Table](#).

This register groups all the status of the VP\_1 internal events that generate an interrupt. Write 1 to a given bit resets this bit

**Table 12-607. DSS0\_COMMON\_VP\_IRQSTATUS\_1 Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 006Ch
DSS0_DISPC_0_COMMON_S0	04A1 006Ch
DSS0_DISPC_0_COMMON_S1	04B0 006Ch
DSS0_DISPC_0_COMMON_S2	04B1 006Ch

**Figure 12-632. DSS0\_COMMON\_VP\_IRQSTATUS\_1 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							SAFETYREGION1_IRQ
R-0h							R/W1C-0h
15	14	13	12	11	10	9	8
SAFETYREGION1_IRQ			DUMMY_IRQ	VPSYNC_IRQ	SECURITYVIO LATION_IRQ	SAFETYREGION_IRQ	
R/W1C-0h			R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	
7	6	5	4	3	2	1	0
SAFETYREGION_IRQ		ACBIASCOUNT STATUS_IRQ	VPSYNCLIST _IRQ	VPVPROGRAM MEDLINENUM BER_IRQ	VPVSYNC_OD D_IRQ	VPVSYNC_IRQ	VPFRAMEDON E_IRQ
R/W1C-0h		R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h

LEGEND: R = Read Only; R/W1C = Read/Write 1 to Clear Bit; -n = value after reset

**Table 12-608. DSS0\_COMMON\_VP\_IRQSTATUS\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reserved
16-13	SAFETYREGION1_IRQ	R/W1C	0h	Safety Feature IRQ. This is raised when FrameFreeze is detected or data mismatch occurs within the safety region 4-7 0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
12	DUMMY_IRQ	R/W1C	0h	Dummy IRQ for future use 0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending



**Table 12-608. DSS0\_COMMON\_VP\_IRQSTATUS\_1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	VPSYNC_IRQ	R/W1C	0h	Go bit clear event  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
10	SECURITYVIOLATION_IRQ	R/W1C	0h	Security Violation IRQ. Non-secure OVR/VP connected to secure VID  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
9-6	SAFETYREGION_IRQ	R/W1C	0h	Safety Feature IRQ. This is raised when FrameFreeze is detected or data mismatch occurs within the safety region 0-3  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
5	ACBIASCOUNTSTATUS_IRQ	R/W1C	0h	AC BIAS transition counter has decremented to zero  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
4	VPSYNCLOST_IRQ	R/W1C	0h	Synchronization Lost on VP output. The required data are not output at the correct time due to too short blanking periods or stall of at least one pipelines associated with VP output  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
3	VPPROGRAMMEDLINENUMBER_IRQ	R/W1C	0h	Programmed Line Number. It indicates that the scan of the display has reached the programmed user line number  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
2	VPVSYNC_ODD_IRQ	R/W1C	0h	VSYNC for odd field. For interlace mode only  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
1	VPVSYNC_IRQ	R/W1C	0h	Vertical Synchronization for VP output. It is used as VSYNC_EVEN in case of interlace mode  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
0	VPFRAMEDONE_IRQ	R/W1C	0h	Frame Done for VP. VP output has been disabled by user All the data have been sent  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending

### 12.6.4.11.13.1.22 DSS0\_COMMON\_VP\_IRQSTATUS\_2 Register (Offset = 70h) [reset = 0h]

DSS0\_COMMON\_VP\_IRQSTATUS\_2 is shown in [Figure 12-633](#) and described in [Table 12-610](#).

Return to [Summary Table](#).

This register groups all the status of the VP\_2 internal events that generate an interrupt. Write 1 to a given bit resets this bit

**Table 12-609. DSS0\_COMMON\_VP\_IRQSTATUS\_2 Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 0070h
DSS0_DISPC_0_COMMON_S0	04A1 0070h
DSS0_DISPC_0_COMMON_S1	04B0 0070h
DSS0_DISPC_0_COMMON_S2	04B1 0070h

**Figure 12-633. DSS0\_COMMON\_VP\_IRQSTATUS\_2 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							SAFETYREGION1_IRQ
R-0h							R/W1C-0h
15	14	13	12	11	10	9	8
SAFETYREGION1_IRQ			DUMMY_IRQ	VPSYNC_IRQ	SECURITYVIO LATION_IRQ	SAFETYREGION_IRQ	
R/W1C-0h			R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	
7	6	5	4	3	2	1	0
SAFETYREGION_IRQ		ACBIASCOUNT STATUS_IRQ	VPSYNCLIST _IRQ	VPVPROGRAM MEDLINENUM BER_IRQ	VPVSYNC_OD D_IRQ	VPVSYNC_IRQ	VPFRAMEDON E_IRQ
R/W1C-0h		R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h

LEGEND: R = Read Only; R/W1C = Read/Write 1 to Clear Bit; -n = value after reset

**Table 12-610. DSS0\_COMMON\_VP\_IRQSTATUS\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reserved
16-13	SAFETYREGION1_IRQ	R/W1C	0h	Safety Feature IRQ. This is raised when FrameFreeze is detected or data mismatch occurs within the safety region 4-7 0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
12	DUMMY_IRQ	R/W1C	0h	Dummy IRQ for future use 0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending

**Table 12-610. DSS0\_COMMON\_VP\_IRQSTATUS\_2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	VPSYNC_IRQ	R/W1C	0h	Go bit clear event  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
10	SECURITYVIOLATION_IRQ	R/W1C	0h	Security Violation IRQ. Non-secure OVR/VP connected to secure VID  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
9-6	SAFETYREGION_IRQ	R/W1C	0h	Safety Feature IRQ. This is raised when FrameFreeze is detected or data mismatch occurs within the safety region 0-3  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
5	ACBIASCOUNTSTATUS_IRQ	R/W1C	0h	AC BIAS transition counter has decremented to zero  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
4	VPSYNCLOST_IRQ	R/W1C	0h	Synchronization Lost on VP output. The required data are not output at the correct time due to too short blanking periods or stall of at least one pipelines associated with VP output  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
3	VPPROGRAMMEDLINENUMBER_IRQ	R/W1C	0h	Programmed Line Number. It indicates that the scan of the display has reached the programmed user line number  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
2	VPVSYNC_ODD_IRQ	R/W1C	0h	VSYNC for odd field. For interlace mode only  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
1	VPVSYNC_IRQ	R/W1C	0h	Vertical Synchronization for VP output. It is used as VSYNC_EVEN in case of interlace mode  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
0	VPFRAMEDONE_IRQ	R/W1C	0h	Frame Done for VP. VP output has been disabled by user All the data have been sent  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending

### 12.6.4.11.13.1.23 DSS0\_COMMON\_VP\_IRQSTATUS\_3 Register (Offset = 74h) [reset = 0h]

DSS0\_COMMON\_VP\_IRQSTATUS\_3 is shown in [Figure 12-634](#) and described in [Table 12-612](#).

Return to [Summary Table](#).

This register groups all the status of the VP\_3 internal events that generate an interrupt. Write 1 to a given bit resets this bit

**Table 12-611. DSS0\_COMMON\_VP\_IRQSTATUS\_3 Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 0074h
DSS0_DISPC_0_COMMON_S0	04A1 0074h
DSS0_DISPC_0_COMMON_S1	04B0 0074h
DSS0_DISPC_0_COMMON_S2	04B1 0074h

**Figure 12-634. DSS0\_COMMON\_VP\_IRQSTATUS\_3 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							SAFETYREGION1_IRQ
R-0h							R/W1C-0h
15	14	13	12	11	10	9	8
SAFETYREGION1_IRQ			DUMMY_IRQ	VPSYNC_IRQ	SECURITYVIO LATION_IRQ	SAFETYREGION_IRQ	
R/W1C-0h			R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	
7	6	5	4	3	2	1	0
SAFETYREGION_IRQ		ACBIASCOUNT STATUS_IRQ	VPSYNCLIST _IRQ	VPVPROGRAM MEDLINENUM BER_IRQ	VPVSYNC_OD D_IRQ	VPVSYNC_IRQ	VPFRAMEDON E_IRQ
R/W1C-0h		R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h

LEGEND: R = Read Only; R/W1C = Read/Write 1 to Clear Bit; -n = value after reset

**Table 12-612. DSS0\_COMMON\_VP\_IRQSTATUS\_3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reserved
16-13	SAFETYREGION1_IRQ	R/W1C	0h	Safety Feature IRQ. This is raised when FrameFreeze is detected or data mismatch occurs within the safety region 4-7 0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
12	DUMMY_IRQ	R/W1C	0h	Dummy IRQ for future use 0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending

**Table 12-612. DSS0\_COMMON\_VP\_IRQSTATUS\_3 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11	VPSYNC_IRQ	R/W1C	0h	Go bit clear event  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
10	SECURITYVIOLATION_IRQ	R/W1C	0h	Security Violation IRQ. Non-secure OVR/VP connected to secure VID  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
9-6	SAFETYREGION_IRQ	R/W1C	0h	Safety Feature IRQ. This is raised when FrameFreeze is detected or data mismatch occurs within the safety region 0-3  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
5	ACBIASCOUNTSTATUS_IRQ	R/W1C	0h	AC BIAS transition counter has decremented to zero  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
4	VPSYNCLOST_IRQ	R/W1C	0h	Synchronization Lost on VP output. The required data are not output at the correct time due to too short blanking periods or stall of at least one pipelines associated with VP output  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
3	VPPROGRAMMEDLINENUMBER_IRQ	R/W1C	0h	Programmed Line Number. It indicates that the scan of the display has reached the programmed user line number  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
2	VPVSYNC_ODD_IRQ	R/W1C	0h	VSYNC for odd field. For interlace mode only  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
1	VPVSYNC_IRQ	R/W1C	0h	Vertical Synchronization for VP output. It is used as VSYNC_EVEN in case of interlace mode  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
0	VPFRAMEDONE_IRQ	R/W1C	0h	Frame Done for VP. VP output has been disabled by user All the data have been sent  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending

### 12.6.4.11.13.1.24 DSS0\_COMMON\_WB\_IRQENABLE Register (Offset = 78h) [reset = 0h]

DSS0\_COMMON\_WB\_IRQENABLE is shown in [Figure 12-635](#) and described in [Table 12-614](#).

Return to [Summary Table](#).

This register allows to mask/unmask the WB internal sources of interrupt, if WB pipeline is present, on an event-by-event basis

**Table 12-613. DSS0\_COMMON\_WB\_IRQENABLE Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 0078h
DSS0_DISPC_0_COMMON_S0	04A1 0078h
DSS0_DISPC_0_COMMON_S1	04B0 0078h
DSS0_DISPC_0_COMMON_S2	04B1 0078h

**Figure 12-635. DSS0\_COMMON\_WB\_IRQENABLE Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED			WBSYNC_EN	SECURITYVIO LATION_EN	WBFRAMEDO NE_EN	WBUNCOMPL ETEERROR_E N	WBBUFFEROV ERFLOW_EN
R-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-614. DSS0\_COMMON\_WB\_IRQENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reserved
4	WBSYNC_EN	R/W	0h	Write-back sync IRQ. Configuration copied from shadow to work for WB for next frame 0h = Event is masked 1h = Event generates an interrupt when it occurs
3	SECURITYVIOLATION_EN	R/W	0h	Security Violation IRQ. Non-secure WB connected to a secure VID/OVR 0h = Event is masked 1h = Event generates an interrupt when it occurs
2	WBFRAMEDONE_EN	R/W	0h	Write-back Frame Done 0h = Event is masked 1h = Event generates an interrupt when it occurs

**Table 12-614. DSS0\_COMMON\_WB\_IRQENABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	WBUNCOMPLETEERRO R_EN	R/W	0h	The write back buffer has been flushed before been fully drained. Can only occur in WB Capture Mode use-case  0h = Event is masked  1h = Event generates an interrupt when it occurs
0	WBBUFFEROVERFLOW_ EN	R/W	0h	Write-back DMA Buffer Overflow. Can only occur in WB Capture Mode use-case  0h = Event is masked  1h = Event generates an interrupt when it occurs

### 12.6.4.11.13.1.25 DSS0\_COMMON\_WB\_IRQSTATUS Register (Offset = 7Ch) [reset = 0h]

DSS0\_COMMON\_WB\_IRQSTATUS is shown in [Figure 12-636](#) and described in [Table 12-616](#).

Return to [Summary Table](#).

This register groups all the status of the WB internal events that generate an interrupt, if WB pipeline is present. Write 1 to a given bit resets this bit

**Table 12-615. DSS0\_COMMON\_WB\_IRQSTATUS Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 007Ch
DSS0_DISPC_0_COMMON_S0	04A1 007Ch
DSS0_DISPC_0_COMMON_S1	04B0 007Ch
DSS0_DISPC_0_COMMON_S2	04B1 007Ch

**Figure 12-636. DSS0\_COMMON\_WB\_IRQSTATUS Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED			WBSYNC_IRQ	SECURITYVIO LATION_IRQ	WBFRAMEDO NE_IRQ	WBUNCOMPL ETEERROR_IR Q	WBBUFFEROV ERFLOW_IRQ
R-0h			R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h

LEGEND: R = Read Only; R/W1C = Read/Write 1 to Clear Bit; -n = value after reset

**Table 12-616. DSS0\_COMMON\_WB\_IRQSTATUS Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reserved
4	WBSYNC_IRQ	R/W1C	0h	Write-back sync IRQ. Configuration copied from shadow to work for WB for next frame 0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
3	SECURITYVIOLATION_IRQ	R/W1C	0h	Security Violation IRQ. Non-secure WB connected to a secure VID/OVR 0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
2	WBFRAMEDONE_IRQ	R/W1C	0h	Write-back Frame Done 0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending



**Table 12-616. DSS0\_COMMON\_WB\_IRQSTATUS Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	WBUNCOMPLETEERRO R_IRQ	R/W1C	0h	Write back DMA buffer is flushed before been completely drained. Can only occur in WB Capture Mode use-case  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending
0	WBBUFFEROVERFLOW_ IRQ	R/W1C	0h	Write-back DMA Buffer Overflow The DMA buffer is full  0h = Write: Status is unchanged, Read: Event is not pending 1h = Write: Status is reset, Read: Event is Pending

### 12.6.4.11.13.1.26 DSS0\_COMMON\_DISPC\_IRQ\_EOI\_FUNC Register (Offset = 80h) [reset = 0h]

DSS0\_COMMON\_DISPC\_IRQ\_EOI\_FUNC is shown in [Figure 12-637](#) and described in [Table 12-618](#).

Return to [Summary Table](#).

End-Of-Interrupt register for FUNC interrupts, to be used if pulse interrupts are used

The EOI register is used to re-trigger the pulse interrupt signal to ensure that any nested interrupt events are serviced. The software interrupt handler must write to the EOI register at the end of the current interrupt processing routine, so that new events can re-trigger the pulse interrupt signal again. For level interrupt signals the EOI register is not functional and must not be used.

**Table 12-617.**  
**DSS0\_COMMON\_DISPC\_IRQ\_EOI\_FUNC Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 0080h
DSS0_DISPC_0_COMMON_S0	04A1 0080h
DSS0_DISPC_0_COMMON_S1	04B0 0080h
DSS0_DISPC_0_COMMON_S2	04B1 0080h

**Figure 12-637. DSS0\_COMMON\_DISPC\_IRQ\_EOI\_FUNC Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															EOI
R-0h															W-0h

LEGEND: R = Read Only; W = Write Only; -n = value after reset

**Table 12-618. DSS0\_COMMON\_DISPC\_IRQ\_EOI\_FUNC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	EOI	W	0h	Write 1 to acknowledge a pulse IRQ  0h = Write-0 : No action 1h = Write-1 : End-of-Interrupt

#### 12.6.4.11.13.1.27 DSS0\_COMMON\_DISPC\_IRQ\_EOI\_SAFETY Register (Offset = 84h) [reset = 0h]

DSS0\_COMMON\_DISPC\_IRQ\_EOI\_SAFETY is shown in [Figure 12-638](#) and described in [Table 12-620](#).

Return to [Summary Table](#).

End-Of-Interrupt register for SAFETY interrupts, to be used if pulse interrupts are used

The EOI register is used to re-trigger the pulse interrupt signal to ensure that any nested interrupt events are serviced. The software interrupt handler must write to the EOI register at the end of the current interrupt processing routine, so that new events can re-trigger the pulse interrupt signal again. For level interrupt signals the EOI register is not functional and must not be used.

**Table 12-619.**  
**DSS0\_COMMON\_DISPC\_IRQ\_EOI\_SAFETY**  
**Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 0084h
DSS0_DISPC_0_COMMON_S0	04A1 0084h
DSS0_DISPC_0_COMMON_S1	04B0 0084h
DSS0_DISPC_0_COMMON_S2	04B1 0084h

**Figure 12-638. DSS0\_COMMON\_DISPC\_IRQ\_EOI\_SAFETY Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															EOI
R-0h															W-0h

LEGEND: R = Read Only; W = Write Only; -n = value after reset

**Table 12-620. DSS0\_COMMON\_DISPC\_IRQ\_EOI\_SAFETY Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	EOI	W	0h	Write 1 to acknowledge a pulse IRQ  0h = Write-0 : No action 1h = Write-1 : End-of-Interrupt

### 12.6.4.11.13.1.28 DSS0\_COMMON\_DISPC\_IRQ\_EOI\_SECURITY Register (Offset = 88h) [reset = 0h]

DSS0\_COMMON\_DISPC\_IRQ\_EOI\_SECURITY is shown in [Figure 12-639](#) and described in [Table 12-622](#).

Return to [Summary Table](#).

End-Of-Interrupt register for SECURITY interrupts, to be used if pulse interrupts are used

The EOI register is used to re-trigger the pulse interrupt signal to ensure that any nested interrupt events are serviced. The software interrupt handler must write to the EOI register at the end of the current interrupt processing routine, so that new events can re-trigger the pulse interrupt signal again. For level interrupt signals the EOI register is not functional and must not be used.

**Table 12-621.**  
**DSS0\_COMMON\_DISPC\_IRQ\_EOI\_SECURITY**  
**Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 0088h
DSS0_DISPC_0_COMMON_S0	04A1 0088h
DSS0_DISPC_0_COMMON_S1	04B0 0088h
DSS0_DISPC_0_COMMON_S2	04B1 0088h

**Figure 12-639. DSS0\_COMMON\_DISPC\_IRQ\_EOI\_SECURITY Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															EOI
R-0h															W-0h

LEGEND: R = Read Only; W = Write Only; -n = value after reset

**Table 12-622. DSS0\_COMMON\_DISPC\_IRQ\_EOI\_SECURITY Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	EOI	W	0h	Write 1 to acknowledge a pulse IRQ  0h = Write-0 : No action 1h = Write-1 : End-of-Interrupt

### 12.6.4.11.13.1.29 DSS0\_COMMON\_DISPC\_SECURE\_DISABLE Register (Offset = 90h) [reset = 0h]

DSS0\_COMMON\_DISPC\_SECURE\_DISABLE is shown in [Figure 12-640](#) and described in [Table 12-624](#).

Return to [Summary Table](#).

Disable security settings throughout DSS IP. COMMON\_1.DISPC\_SECURE bits are honoured only if COMMON.DSS0\_COMMON\_DISPC\_SECURE\_DISABLE = 0

**Table 12-623.**  
**DSS0\_COMMON\_DISPC\_SECURE\_DISABLE**  
**Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 0090h
DSS0_DISPC_0_COMMON_S0	N/A
DSS0_DISPC_0_COMMON_S1	N/A
DSS0_DISPC_0_COMMON_S2	N/A

**Figure 12-640. DSS0\_COMMON\_DISPC\_SECURE\_DISABLE Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							SECURE_DISABLE
R-0h							R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-624. DSS0\_COMMON\_DISPC\_SECURE\_DISABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	SECURE_DISABLE	R/W	0h	Secure disable bit  0h = Secure bits, in COMMON_1.DISPC_SECURE, are active  1h = Secure bits, in COMMON_1.DISPC_SECURE, are not active and IP will behave as non-secure

### 12.6.4.11.13.1.30 DSS0\_COMMON\_DISPC\_GLOBAL\_MFLAG\_ATTRIBUTE Register (Offset = 98h) [reset = 0h]

DSS0\_COMMON\_DISPC\_GLOBAL\_MFLAG\_ATTRIBUTE is shown in [Figure 12-641](#) and described in [Table 12-626](#).

Return to [Summary Table](#).

MFLAG control register

**Table 12-625.**  
**DSS0\_COMMON\_DISPC\_GLOBAL\_MFLAG\_ATTRIB**  
**UTE Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 0098h
DSS0_DISPC_0_COMMON_S0	N/A
DSS0_DISPC_0_COMMON_S1	N/A
DSS0_DISPC_0_COMMON_S2	N/A

**Figure 12-641. DSS0\_COMMON\_DISPC\_GLOBAL\_MFLAG\_ATTRIBUTE Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						RESERVED	
R-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED	MFLAG_START	RESERVED				MFLAG_CTRL	
R-0h	R/W-0h	R-0h				R/W-0h	

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-626. DSS0\_COMMON\_DISPC\_GLOBAL\_MFLAG\_ATTRIBUTE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0h	Reserved
8-7	RESERVED	R	0h	Reserved
6	MFLAG_START	R/W	0h	MFLAG_START for DMA master port  0h = When the DMA buffer is empty at the beginning of the frame, the MFLAG of each pipe is kept at 0 until PRELOAD is reached, then based on MFLAG_CTRL, MFLAG is generated and internal logic is arbitrating between pipeline requests  1h = Even at the beginning of the frame when the DMA buffer is empty, MFLAG_CTRL is used to determine how MFLAG dedicated to each pipe signal shall be driven
5-2	RESERVED	R	0h	Reserved

**Table 12-626. DSS0\_COMMON\_DISPC\_GLOBAL\_MFLAG\_ATTRIBUTE Register Field Descriptions  
(continued)**

Bit	Field	Type	Reset	Description
1-0	MFLAG_CTRL	R/W	0h	<p>MFLAG_CTRL for DMA master port</p> <p>0h = MFLAG mechanism is disabled: MFLAG out band signal is set to 0</p> <p>1h = MFLAG mechanism is enabled: MFLAG out band signal is always set to 1</p> <p>2h = MFLAG mechanism is enabled and MFLAG out band signal is dynamically set to 0 or 1 depending on the MFLAG rules</p>

### 12.6.4.11.13.1.31 DSS0\_COMMON\_DISPC\_GLOBAL\_OUTPUT\_ENABLE Register (Offset = 9Ch) [reset = 0h]

DSS0\_COMMON\_DISPC\_GLOBAL\_OUTPUT\_ENABLE is shown in [Figure 12-642](#) and described in [Table 12-628](#).

Return to [Summary Table](#).

DISPC global output enable register. The ENABLE or GO bit for a particular output port is set when either the corresponding bit in this register is set or the corresponding bit within the sub-module is set. This register allows enabling multiple outputs synchronously [simultaneously], which is not possible if the ENABLE/GO bits are present only within the sub-module

**Table 12-627.**  
**DSS0\_COMMON\_DISPC\_GLOBAL\_OUTPUT\_ENABLE Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 009Ch
DSS0_DISPC_0_COMMON_S0	N/A
DSS0_DISPC_0_COMMON_S1	N/A
DSS0_DISPC_0_COMMON_S2	N/A

**Figure 12-642. DSS0\_COMMON\_DISPC\_GLOBAL\_OUTPUT\_ENABLE Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED												VP_GO			
R-0h												R/W-0h			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												VP_ENABLE			
R-0h												R/W-0h			

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-628. DSS0\_COMMON\_DISPC\_GLOBAL\_OUTPUT\_ENABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	0h	Reserved
19-16	VP_GO	R/W	0h	Global GO Command for the VP [3:0] output. It is used to synchronize the pipelines associated with the VP output. wr: immediate  0h = The hardware has finished updating the internal shadow registers of the pipeline(s) connected to the VP output using the user values. The hardware resets the bit when the update is completed  1h = The user has finished to program the shadow registers of the pipeline(s) associated with the VP output and the hardware can update the internal registers at the VFP start period
15-4	RESERVED	R	0h	Reserved
3-0	VP_ENABLE	R/W	0h	Global VP [3:0] Enable  0h = VP port is disabled  1h = VP port is enabled



### 12.6.4.11.13.1.32 DSS0\_COMMON\_DISPC\_GLOBAL\_BUFFER Register (Offset = A0h) [reset = 4688h]

DSS0\_COMMON\_DISPC\_GLOBAL\_BUFFER is shown in [Figure 12-643](#) and described in [Table 12-630](#).

Return to [Summary Table](#).

The register configures the DMA buffers allocations to the pipelines for DMA

**Table 12-629.**  
**DSS0\_COMMON\_DISPC\_GLOBAL\_BUFFER**  
**Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 00A0h
DSS0_DISPC_0_COMMON_S0	N/A
DSS0_DISPC_0_COMMON_S1	N/A
DSS0_DISPC_0_COMMON_S2	N/A

**Figure 12-643. DSS0\_COMMON\_DISPC\_GLOBAL\_BUFFER Register**

31	30	29	28	27	26	25	24
BUFFERFILLIN G	SHAREDBUGE NABLE	RESERVED	RESERVED				
R/W-0h	R/W-0h	R/W-0h	R-0h				
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED	WB_BUFFER			VIDL2_BUFFER			VID2_BUFFER
R-0h	R/W-4h			R/W-3h			R/W-2h
7	6	5	4	3	2	1	0
VID2_BUFFER		VIDL1_BUFFER			VID1_BUFFER		
R/W-2h		R/W-1h			R/W-0h		

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-630. DSS0\_COMMON\_DISPC\_GLOBAL\_BUFFER Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	BUFFERFILLING	R/W	0h	Controls if the DMA buffers are re-filled only when the LOW threshold is reached or if all DMA buffers are re-filled when at least one of them reaches the LOW threshold  0h = Each DMA buffer is re-filled when it reaches LOW threshold  1h = All DMA buffers are re-filled up to high threshold when at least one of them reaches the LOW threshold. Only active DMA buffers shall be considered and when reaching the end of the frame the DMA buffer goes to empty condition so no need to fill it again
30	SHAREDBUGENABLE	R/W	0h	Enable Shared DMA Buffer feature  0h = Shared Buffer scheme is disabled. Design follows the buffer settings as defined in this register  1h = Shared Buffer scheme is enabled. Design follows the buffer settings as defined in VID#.DMA_BUFSIZE and VP#/WB.DMA_THREADSIZE registers

**Table 12-630. DSS0\_COMMON\_DISPC\_GLOBAL\_BUFFER Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
29	RESERVED	R/W	0h	Reserved1
28-15	RESERVED	R	0h	Reserved
14-12	WB_BUFFER	R/W	4h	WB DMA buffer allocation to one of the pipelines, if WB pipeline is present  0h = DMA buffer allocated to the VID-1 pipeline 1h = DMA buffer allocated to the VIDL-1 pipeline 2h = DMA buffer allocated to the VID-2 pipeline 3h = DMA buffer allocated to the VIDL-2 pipeline 4h = DMA buffer allocated to the WB pipeline
11-9	VIDL2_BUFFER	R/W	3h	VIDL2 DMA buffer allocation to one of the pipelines, if VIDL2 is present  0h = DMA buffer allocated to the VID-1 pipeline 1h = DMA buffer allocated to the VIDL-1 pipeline 2h = DMA buffer allocated to the VID-2 pipeline 3h = DMA buffer allocated to the VIDL-2 pipeline 4h = DMA buffer allocated to the WB pipeline
8-6	VID2_BUFFER	R/W	2h	VID2 DMA buffer allocation to one of the pipelines, if VID2 is present  0h = DMA buffer allocated to the VID-1 pipeline 1h = DMA buffer allocated to the VIDL-1 pipeline 2h = DMA buffer allocated to the VID-2 pipeline 3h = DMA buffer allocated to the VIDL-2 pipeline 4h = DMA buffer allocated to the WB pipeline
5-3	VIDL1_BUFFER	R/W	1h	VIDL1 DMA buffer allocation to one of the pipelines  0h = DMA buffer allocated to the VID-1 pipeline 1h = DMA buffer allocated to the VIDL-1 pipeline 2h = DMA buffer allocated to the VID-2 pipeline 3h = DMA buffer allocated to the VIDL-2 pipeline 4h = DMA buffer allocated to the WB pipeline
2-0	VID1_BUFFER	R/W	0h	VID1 DMA buffer allocation to one of the pipelines  0h = DMA buffer allocated to the VID-1 pipeline 1h = DMA buffer allocated to the VIDL-1 pipeline 2h = DMA buffer allocated to the VID-2 pipeline 3h = DMA buffer allocated to the VIDL-2 pipeline 4h = DMA buffer allocated to the WB pipeline

### 12.6.4.11.13.1.33 DSS0\_COMMON\_DSS\_CBA\_CFG Register (Offset = A4h) [reset = Ch]

DSS0\_COMMON\_DSS\_CBA\_CFG is shown in [Figure 12-644](#) and described in [Table 12-632](#).

Return to [Summary Table](#).

This register contains CBA specific config bits in DSS

**Table 12-631. DSS0\_COMMON\_DSS\_CBA\_CFG Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 00A4h
DSS0_DISPC_0_COMMON_S0	N/A
DSS0_DISPC_0_COMMON_S1	N/A
DSS0_DISPC_0_COMMON_S2	N/A

**Figure 12-644. DSS0\_COMMON\_DSS\_CBA\_CFG Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							DMA_BACKLOGSTATUS_DISABLE_VAL
R-0h							R/W-0h
7	6	5	4	3	2	1	0
DMA_BACKLOGSTATUS_DISABLE_VAL	DMA_BACKLOGSTATUS_DISABLE_VAL	PRI_HI			PRI_LO		
R/W-0h	R/W-0h	R/W-1h			R/W-4h		

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-632. DSS0\_COMMON\_DSS\_CBA\_CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0h	Reserved
8-7	DMA_BACKLOGSTATUS_DISABLE_VAL	R/W	0h	IP Internal - Tie-off value on DMA_BACKLOGSTATUS pins when DMA Backlog Status reporting is disabled
6	DMA_BACKLOGSTATUS_DISABLE	R/W	0h	IP Internal - Disable generation of DMA Backlog Status reporting to interconnect
5-3	PRI_HI	R/W	1h	The value sent out on the PRI_HI bus from DSS to CBA Indicates the priority level for high-priority [MFLAG] transactions. Value of 0x0 indicates highest priority Value of 0x7 indicates lowest priority
2-0	PRI_LO	R/W	4h	The value sent out on the PRI_LO bus from DSS to CBA Indicates the priority level for normal [non-MFLAG] transactions. Value of 0x0 indicates highest priority Value of 0x7 indicates lowest priority

### 12.6.4.11.13.1.34 DSS0\_COMMON\_DISPC\_DBG\_CONTROL Register (Offset = A8h) [reset = 0h]

DSS0\_COMMON\_DISPC\_DBG\_CONTROL is shown in [Figure 12-645](#) and described in [Table 12-634](#).

Return to [Summary Table](#).

DISPC debug status control register

**Table 12-633.**

**DSS0\_COMMON\_DISPC\_DBG\_CONTROL Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 00A8h
DSS0_DISPC_0_COMMON_S0	N/A
DSS0_DISPC_0_COMMON_S1	N/A
DSS0_DISPC_0_COMMON_S2	N/A

**Figure 12-645. DSS0\_COMMON\_DISPC\_DBG\_CONTROL Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							DBGMUXSEL
R-0h							R/W-0h
7	6	5	4	3	2	1	0
DBGMUXSEL							DBGEN
R/W-0h							R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-634. DSS0\_COMMON\_DISPC\_DBG\_CONTROL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0h	Reserved

**Table 12-634. DSS0\_COMMON\_DISPC\_DBG\_CONTROL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
8-1	DBGMUXSEL	R/W	0h	<p>Mux select for the debug status</p> <p>00h = 8 values to select any 4bytes, order from LSB 4bytes to MSB 4bytes, at a time from the 32byte VID1 debug bus</p> <p>08h = 8 values to select any 4bytes, order from LSB 4bytes to MSB 4bytes, at a time from the 32byte VIDL1 debug bus</p> <p>10h = 8 values to select any 4bytes, order from LSB 4bytes to MSB 4bytes, at a time from the 32byte VID2 debug bus</p> <p>18h = 8 values to select any 4bytes, order from LSB 4bytes to MSB 4bytes, at a time from the 32byte VIDL2 debug bus</p> <p>20h = Select WB debug bus , 4bytes</p> <p>21h = Select OVR1 debug bus, 4bytes</p> <p>22h = Select OVR2 debug bus, 4bytes</p> <p>23h = Select OVR3 debug bus, 4bytes</p> <p>24h = Select OVR3 debug bus, 4bytes</p> <p>25h = Select VP1 debug bus , 8bytes</p> <p>27h = Select VP2 debug bus , 8bytes</p> <p>29h = Select VP3 debug bus , 8bytes</p> <p>2Bh = Select VP3 debug bus , 8bytes</p> <p>2Dh = Select MISC debug bus, 4bytes</p>
0	DBGGEN	R/W	0h	<p>Enable debug ports</p> <p>0h = 0</p> <p>1h = 1</p>

### 12.6.4.11.13.1.35 DSS0\_COMMON\_DISPC\_DBG\_STATUS Register (Offset = ACh) [reset = 0h]

DSS0\_COMMON\_DISPC\_DBG\_STATUS is shown in [Figure 12-646](#) and described in [Table 12-636](#).

Return to [Summary Table](#).

DISPC debug status register

**Table 12-635.**  
**DSS0\_COMMON\_DISPC\_DBG\_STATUS Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 00ACh
DSS0_DISPC_0_COMMON_S0	N/A
DSS0_DISPC_0_COMMON_S1	N/A
DSS0_DISPC_0_COMMON_S2	N/A

**Figure 12-646. DSS0\_COMMON\_DISPC\_DBG\_STATUS Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBGOUT																															
R-0h																															

LEGEND: R = Read Only; -n = value after reset

**Table 12-636. DSS0\_COMMON\_DISPC\_DBG\_STATUS Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	DBGOUT	R	0h	Debug status

### 12.6.4.11.13.1.36 DSS0\_COMMON\_DISPC\_CLKGATING\_DISABLE Register (Offset = B0h) [reset = 0h]

DSS0\_COMMON\_DISPC\_CLKGATING\_DISABLE is shown in [Figure 12-647](#) and described in [Table 12-638](#).

Return to [Summary Table](#).

Register to control clock gating at DISPC sub-module level

**Table 12-637.**  
**DSS0\_COMMON\_DISPC\_CLKGATING\_DISABLE**  
**Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 00B0h
DSS0_DISPC_0_COMMON_S0	N/A
DSS0_DISPC_0_COMMON_S1	N/A
DSS0_DISPC_0_COMMON_S2	N/A

**Figure 12-647. DSS0\_COMMON\_DISPC\_CLKGATING\_DISABLE Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED		VP				OVR	
R-0h		R/W-0h				R/W-0h	
15	14	13	12	11	10	9	8
OVR		WB	RESERVED	RESERVED			
R/W-0h		R/W-0h	R-0h	R-0h			
7	6	5	4	3	2	1	0
RESERVED	VID				RESERVED		DMA
R-0h	R/W-0h				R-0h		R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-638. DSS0\_COMMON\_DISPC\_CLKGATING\_DISABLE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-22	RESERVED	R	0h	Reserved
21-18	VP	R/W	0h	Clock gating control for VP [3:0] 0h = Clock-Gating is enabled 1h = Clock-gating is disabled. Clocks are free running
17-14	OVR	R/W	0h	Clock gating control for OVR [3:0] 0h = Clock-Gating is enabled 1h = Clock-gating is disabled. Clocks are free running
13	WB	R/W	0h	Clock gating control for WB, if WB pipeline is present 0h = Clock-Gating is enabled 1h = Clock-gating is disabled. Clocks are free running
12	RESERVED	R	0h	Reserved

**Table 12-638. DSS0\_COMMON\_DISPC\_CLKGATING\_DISABLE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-7	RESERVED	R	0h	Reserved
6-3	VID	R/W	0h	Clock gating control for VID. [0] -> VID1, [1] -> VIDL1, [2] -> VID2, [3] -> VIDL2  0h = Clock-Gating is enabled  1h = Clock-gating is disabled. Clocks are free running
2-1	RESERVED	R	0h	Reserved
0	DMA	R/W	0h	Clock gating control for DMA  0h = Clock-Gating is enabled  1h = Clock-gating is disabled. Clocks are free running



### 12.6.4.11.13.1.37 DSS0\_COMMON\_FBDC\_REVISION\_1 Register (Offset = B8h) [reset = 0h]

DSS0\_COMMON\_FBDC\_REVISION\_1 is shown in [Figure 12-648](#) and described in [Table 12-640](#).

Return to [Summary Table](#).

This register contains the FBDC Product Code

**Table 12-639. DSS0\_COMMON\_FBDC\_REVISION\_1 Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 00B8h
DSS0_DISPC_0_COMMON_S0	N/A
DSS0_DISPC_0_COMMON_S1	N/A
DSS0_DISPC_0_COMMON_S2	N/A

**Figure 12-648. DSS0\_COMMON\_FBDC\_REVISION\_1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																PRODUCTCODE															
R-0h																R-0h															

LEGEND: R = Read Only; -n = value after reset

**Table 12-640. DSS0\_COMMON\_FBDC\_REVISION\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	PRODUCTCODE	R	0h	FBDC Product Code

### 12.6.4.11.13.1.38 DSS0\_COMMON\_FBDC\_REVISION\_2 Register (Offset = BCh) [reset = 0h]

DSS0\_COMMON\_FBDC\_REVISION\_2 is shown in [Figure 12-649](#) and described in [Table 12-642](#).

Return to [Summary Table](#).

This register contains the FBDC Branch Code

**Table 12-641. DSS0\_COMMON\_FBDC\_REVISION\_2 Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 00BCh
DSS0_DISPC_0_COMMON_S0	N/A
DSS0_DISPC_0_COMMON_S1	N/A
DSS0_DISPC_0_COMMON_S2	N/A

**Figure 12-649. DSS0\_COMMON\_FBDC\_REVISION\_2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																BRANCHCODE															
R-0h																R-0h															

LEGEND: R = Read Only; -n = value after reset

**Table 12-642. DSS0\_COMMON\_FBDC\_REVISION\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	BRANCHCODE	R	0h	FBDC Branch Code

### 12.6.4.11.13.1.39 DSS0\_COMMON\_FBDC\_REVISION\_3 Register (Offset = C0h) [reset = 0h]

DSS0\_COMMON\_FBDC\_REVISION\_3 is shown in [Figure 12-650](#) and described in [Table 12-644](#).

Return to [Summary Table](#).

This register contains the FBDC Version Code

**Table 12-643. DSS0\_COMMON\_FBDC\_REVISION\_3  
Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 00C0h
DSS0_DISPC_0_COMMON_S0	N/A
DSS0_DISPC_0_COMMON_S1	N/A
DSS0_DISPC_0_COMMON_S2	N/A

**Figure 12-650. DSS0\_COMMON\_FBDC\_REVISION\_3 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																VERSIONCODE															
R-0h																R-0h															

LEGEND: R = Read Only; -n = value after reset

**Table 12-644. DSS0\_COMMON\_FBDC\_REVISION\_3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	VERSIONCODE	R	0h	FBDC Version Code

#### 12.6.4.11.13.1.40 DSS0\_COMMON\_FBDC\_REVISION\_4 Register (Offset = C4h) [reset = 0h]

DSS0\_COMMON\_FBDC\_REVISION\_4 is shown in [Figure 12-651](#) and described in [Table 12-646](#).

Return to [Summary Table](#).

This register contains the FBDC Scalable Core Code

**Table 12-645. DSS0\_COMMON\_FBDC\_REVISION\_4 Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 00C4h
DSS0_DISPC_0_COMMON_S0	N/A
DSS0_DISPC_0_COMMON_S1	N/A
DSS0_DISPC_0_COMMON_S2	N/A

**Figure 12-651. DSS0\_COMMON\_FBDC\_REVISION\_4 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																CORECODE															
R-0h																R-0h															

LEGEND: R = Read Only; -n = value after reset

**Table 12-646. DSS0\_COMMON\_FBDC\_REVISION\_4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	CORECODE	R	0h	FBDC Scalable Core Code

#### 12.6.4.11.13.1.41 DSS0\_COMMON\_FBDC\_REVISION\_5 Register (Offset = C8h) [reset = 0h]

DSS0\_COMMON\_FBDC\_REVISION\_5 is shown in [Figure 12-652](#) and described in [Table 12-648](#).

Return to [Summary Table](#).

This register contains the FBDC Configuration Code

**Table 12-647. DSS0\_COMMON\_FBDC\_REVISION\_5  
Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 00C8h
DSS0_DISPC_0_COMMON_S0	N/A
DSS0_DISPC_0_COMMON_S1	N/A
DSS0_DISPC_0_COMMON_S2	N/A

**Figure 12-652. DSS0\_COMMON\_FBDC\_REVISION\_5 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																CONFIGCODE															
R-0h																R-0h															

LEGEND: R = Read Only; -n = value after reset

**Table 12-648. DSS0\_COMMON\_FBDC\_REVISION\_5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	CONFIGCODE	R	0h	FBDC Configuration Code

#### 12.6.4.11.13.1.42 DSS0\_COMMON\_FBDC\_REVISION\_6 Register (Offset = CCh) [reset = 0h]

DSS0\_COMMON\_FBDC\_REVISION\_6 is shown in [Figure 12-653](#) and described in [Table 12-650](#).

Return to [Summary Table](#).

This register contains the FBDC Changelist Code

**Table 12-649. DSS0\_COMMON\_FBDC\_REVISION\_6 Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 00CCh
DSS0_DISPC_0_COMMON_S0	N/A
DSS0_DISPC_0_COMMON_S1	N/A
DSS0_DISPC_0_COMMON_S2	N/A

**Figure 12-653. DSS0\_COMMON\_FBDC\_REVISION\_6 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHANGELISTCODE																															
R-0h																															

LEGEND: R = Read Only; -n = value after reset

**Table 12-650. DSS0\_COMMON\_FBDC\_REVISION\_6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	CHANGELISTCODE	R	0h	FBDC Changelist Code

#### 12.6.4.11.13.1.43 DSS0\_COMMON\_FBDC\_COMMON\_CONTROL Register (Offset = D0h) [reset = X]

DSS0\_COMMON\_FBDC\_COMMON\_CONTROL is shown in [Figure 12-654](#) and described in [Table 12-652](#).

Return to [Summary Table](#).

This register contains the common control signals for FBDC

**Table 12-651.**  
**DSS0\_COMMON\_FBDC\_COMMON\_CONTROL**  
**Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 00D0h
DSS0_DISPC_0_COMMON_S0	N/A
DSS0_DISPC_0_COMMON_S1	N/A
DSS0_DISPC_0_COMMON_S2	N/A

**Figure 12-654. DSS0\_COMMON\_FBDC\_COMMON\_CONTROL Register**

31	30	29	28	27	26	25	24
RESERVED							
R/W-X							
23	22	21	20	19	18	17	16
RESERVED							
R/W-X							
15	14	13	12	11	10	9	8
RESERVED							
R/W-X							
7	6	5	4	3	2	1	0
RESERVED					GPATYPE	CLKGATE	IDLEGATE
R/W-X					R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-652. DSS0\_COMMON\_FBDC\_COMMON\_CONTROL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-3	RESERVED	R/W	X	
2	GPATYPE	R/W	0h	GPU Selection 0h = 8XE GPU is used for compression 1h = 8XT GPU is used for compression
1	CLKGATE	R/W	0h	Reserved
0	IDLEGATE	R/W	0h	Reserved

#### 12.6.4.11.13.1.44 DSS0\_COMMON\_FBDC\_CONSTANT\_COLOR\_0 Register (Offset = D4h) [reset = 0h]

DSS0\_COMMON\_FBDC\_CONSTANT\_COLOR\_0 is shown in [Figure 12-655](#) and described in [Table 12-654](#).

Return to [Summary Table](#).

Defines the Constant Color-0 value to be used for the FBDC

**Table 12-653.**  
**DSS0\_COMMON\_FBDC\_CONSTANT\_COLOR\_0**  
**Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 00D4h
DSS0_DISPC_0_COMMON_S0	N/A
DSS0_DISPC_0_COMMON_S1	N/A
DSS0_DISPC_0_COMMON_S2	N/A

**Figure 12-655. DSS0\_COMMON\_FBDC\_CONSTANT\_COLOR\_0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CONSTCOLOR																															
R/W-0h																															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-654. DSS0\_COMMON\_FBDC\_CONSTANT\_COLOR\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	CONSTCOLOR	R/W	0h	Defines the Constant Color-0 value to be used for the FBDC



#### 12.6.4.11.13.1.45 DSS0\_COMMON\_FBDC\_CONSTANT\_COLOR\_1 Register (Offset = D8h) [reset = 0h]

DSS0\_COMMON\_FBDC\_CONSTANT\_COLOR\_1 is shown in [Figure 12-656](#) and described in [Table 12-656](#).

Return to [Summary Table](#).

Defines the Constant Color-1 value to be used for the FBDC

**Table 12-655.**  
**DSS0\_COMMON\_FBDC\_CONSTANT\_COLOR\_1**  
**Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 00D8h
DSS0_DISPC_0_COMMON_S0	N/A
DSS0_DISPC_0_COMMON_S1	N/A
DSS0_DISPC_0_COMMON_S2	N/A

**Figure 12-656. DSS0\_COMMON\_FBDC\_CONSTANT\_COLOR\_1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CONSTCOLOR																															
R/W-0h																															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-656. DSS0\_COMMON\_FBDC\_CONSTANT\_COLOR\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	CONSTCOLOR	R/W	0h	Defines the Constant Color-1 value to be used for the FBDC

### 12.6.4.11.13.1.46 DSS0\_COMMON\_DISPC\_CONNECTIONS Register (Offset = E4h) [reset = X]

DSS0\_COMMON\_DISPC\_CONNECTIONS is shown in [Figure 12-657](#) and described in [Table 12-658](#).

Return to [Summary Table](#).

Connections of various sub-modules within DISPC, as well as some peripherals outside. One hot encoding

**Table 12-657.**

**DSS0\_COMMON\_DISPC\_CONNECTIONS Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 00E4h
DSS0_DISPC_0_COMMON_S0	N/A
DSS0_DISPC_0_COMMON_S1	N/A
DSS0_DISPC_0_COMMON_S2	N/A

**Figure 12-657. DSS0\_COMMON\_DISPC\_CONNECTIONS Register**

31	30	29	28	27	26	25	24
RESERVED				VIRTUALVP_CONN			
R/W-X				R/W-0h			
23	22	21	20	19	18	17	16
RESERVED				WB_CONN			
R/W-X				R/W-0h			
15	14	13	12	11	10	9	8
RESERVED							
R/W-X							
7	6	5	4	3	2	1	0
DPI_1_CONN				DPI_0_CONN			
R/W-0h				R/W-0h			

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-658. DSS0\_COMMON\_DISPC\_CONNECTIONS Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	RESERVED	R/W	X	
27-24	VIRTUALVP_CONN	R/W	0h	Defines the connection to VIRTUAL_VP output 0h = None connected to VIRTUAL_VP output 1h = OVR1 connected to VIRTUAL_VP output 2h = OVR2 connected to VIRTUAL_VP output 4h = OVR3 connected to VIRTUAL_VP output 8h = OVR4 connected to VIRTUAL_VP output
23-21	RESERVED	R/W	X	

**Table 12-658. DSS0\_COMMON\_DISPC\_CONNECTIONS Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
20-16	WB_CONN	R/W	0h	Defines the connection to WB pipe 0h = None connected to WB 1h = VIDL2 connected to WB in m2m 2h = OVR1 connected to WB in m2m or capture 4h = OVR2 connected to WB in m2m or capture 8h = OVR3 connected to WB in m2m or capture 10h = OVR4 connected to WB in m2m or capture
15-8	RESERVED	R/W	X	
7-4	DPI_1_CONN	R/W	0h	Defines the connection to DPI-1 output. For J7, valid values are 0x0, 0x2 and 0x8 0h = None connected to DPI-#n output 1h = VP1 connected to DPI-#n output 2h = VP2 connected to DPI-#n output 4h = VP3 connected to DPI-#n output 8h = VP4 connected to DPI-#n output
3-0	DPI_0_CONN	R/W	0h	Defines the connection to DPI-0 output. For J7, valid values are 0x0, 0x2 and 0x8 0h = None connected to DPI-#n output 1h = VP1 connected to DPI-#n output 2h = VP2 connected to DPI-#n output 4h = VP3 connected to DPI-#n output 8h = VP4 connected to DPI-#n output

### 12.6.4.11.13.1.47 DSS0\_COMMON\_DISPC\_MSS\_VP1 Register (Offset = E8h) [reset = 0h]

DSS0\_COMMON\_DISPC\_MSS\_VP1 is shown in [Figure 12-658](#) and described in [Table 12-660](#).

Return to [Summary Table](#).

This register controls the Merge\_Split\_Sync operation for VP1

**Table 12-659. DSS0\_COMMON\_DISPC\_MSS\_VP1 Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 00E8h
DSS0_DISPC_0_COMMON_S0	N/A
DSS0_DISPC_0_COMMON_S1	N/A
DSS0_DISPC_0_COMMON_S2	N/A

**Figure 12-658. DSS0\_COMMON\_DISPC\_MSS\_VP1 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				MSSFORMAT	MSSTYPE		MSENABLE
R-0h				R/W-0h	R/W-0h		R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-660. DSS0\_COMMON\_DISPC\_MSS\_VP1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3	MSSFORMAT	R/W	0h	Merge Split format 0h = The format is LEFT-RIGHT 1h = The format is ODD-EVEN
2-1	MSSTYPE	R/W	0h	Merge-Split-Sync operation type 0h = VP-1 is going to be merged with VP-2, and the merged output goes on VP-1 1h = VP-1 is going to be split into two, with the split outputs going on VP-1 and VP-2 2h = VP-1 is going to be in SYNC with VP-2 in terms of PCLK
0	MSENABLE	R/W	0h	Merge-Split-Sync operation Enable 0h = Merge-split operation is bypassed 1h = Merge-split operation is enabled

#### 12.6.4.11.13.1.48 DSS0\_COMMON\_DISPC\_MSS\_VP3 Register (Offset = ECh) [reset = 0h]

DSS0\_COMMON\_DISPC\_MSS\_VP3 is shown in [Figure 12-659](#) and described in [Table 12-662](#).

Return to [Summary Table](#).

This register controls the Merge\_Split\_Sync operation for VP3

**Table 12-661. DSS0\_COMMON\_DISPC\_MSS\_VP3 Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 00ECh
DSS0_DISPC_0_COMMON_S0	N/A
DSS0_DISPC_0_COMMON_S1	N/A
DSS0_DISPC_0_COMMON_S2	N/A

**Figure 12-659. DSS0\_COMMON\_DISPC\_MSS\_VP3 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				MSSFORMAT	MSSTYPE		MSENABLE
R-0h				R/W-0h	R/W-0h		R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-662. DSS0\_COMMON\_DISPC\_MSS\_VP3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reserved
3	MSSFORMAT	R/W	0h	Merge Split format 0h = The format is ODD-EVEN 1h = The format is LEFT-RIGHT
2-1	MSSTYPE	R/W	0h	Merge-Split-Sync operation type 0h = VP-3 is going to be merged with VP-4, and the merged output goes on VP-3 1h = VP-3 is going to be split into two, with the split outputs going on VP-3 and VP-4 2h = VP-3 is going to be in SYNC with VP-4 in terms of PCLK
0	MSENABLE	R/W	0h	Merge-Split-Sync operation Enable 0h = Merge-split operation is bypassed 1h = Merge-split operation is enabled

### 12.6.4.11.13.1.49 DSS0\_COMMON\_GLOBAL\_DMA\_THREADSIZE Register (Offset = F0h) [reset = 10h]

DSS0\_COMMON\_GLOBAL\_DMA\_THREADSIZE is shown in [Figure 12-660](#) and described in [Table 12-664](#).

Return to [Summary Table](#).

This register configures the DMA buffer size allocated to the different threads - Shared memory feature

**Table 12-663.**  
**DSS0\_COMMON\_GLOBAL\_DMA\_THREADSIZE**  
**Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 00F0h
DSS0_DISPC_0_COMMON_S0	N/A
DSS0_DISPC_0_COMMON_S1	N/A
DSS0_DISPC_0_COMMON_S2	N/A

**Figure 12-660. DSS0\_COMMON\_GLOBAL\_DMA\_THREADSIZE Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED								WBTHREADSIZE				VP3THREADSIZE			
R-0h								R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VP3TH READ SIZE	VP2THREADSIZE					VP1THREADSIZE					VP0THREADSIZE				
	R/W-0h					R/W-0h					R/W-10h				

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-664. DSS0\_COMMON\_GLOBAL\_DMA\_THREADSIZE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24-20	WBTHREADSIZE	R/W	0h	Total DMA buffer size for all the pipelines connected to WB THREAD4.If the value programmed is n, then the allocated buffer size is 16KB*n. Default:0KB
19-15	VP3THREADSIZE	R/W	0h	Total DMA buffer size for all the pipelines connected to VP3 THREAD3.If the value programmed is n, then the allocated buffer size is 16KB*n. Default:0KB
14-10	VP2THREADSIZE	R/W	0h	Total DMA buffer size for all the pipelines connected to VP2 THREAD2.If the value programmed is n, then the allocated buffer size is 16KB*n. Default:0KB
9-5	VP1THREADSIZE	R/W	0h	Total DMA buffer size for all the pipelines connected to VP1 THREAD1.If the value programmed is n, then the allocated buffer size is 16KB*n. Default:0KB
4-0	VP0THREADSIZE	R/W	10h	Total DMA buffer size for all the pipelines connected to VP0 THREAD0.If the value programmed is n, then the allocated buffer size is 16KB*n. Default:256KB

### 12.6.4.11.13.1.50 DSS0\_COMMON\_GLOBAL\_DMA\_THREADSIZESTATUS Register (Offset = F4h) [reset = 10h]

DSS0\_COMMON\_GLOBAL\_DMA\_THREADSIZESTATUS is shown in [Figure 12-661](#) and described in [Table 12-666](#).

Return to [Summary Table](#).

This register read the synchronized value of DMA buffer size allocated to the different threads - Shared memory feature

**Table 12-665.**  
**DSS0\_COMMON\_GLOBAL\_DMA\_THREADSIZESTATUS**  
**US Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 00F4h
DSS0_DISPC_0_COMMON_S0	N/A
DSS0_DISPC_0_COMMON_S1	N/A
DSS0_DISPC_0_COMMON_S2	N/A

**Figure 12-661. DSS0\_COMMON\_GLOBAL\_DMA\_THREADSIZESTATUS Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED							WBTHREADSIZE					VP3THREADSIZE			
R-0h							R-0h					R-0h			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VP3TH READ SIZE	VP2THREADSIZE					VP1THREADSIZE					VP0THREADSIZE				
R-0h	R-0h					R-0h					R-10h				

LEGEND: R = Read Only; -n = value after reset

**Table 12-666. DSS0\_COMMON\_GLOBAL\_DMA\_THREADSIZESTATUS Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24-20	WBTHREADSIZE	R	0h	Synchronized version of WB THREADSIZE. Value used by HW
19-15	VP3THREADSIZE	R	0h	Synchronized version of VP3 THREADSIZE. Value used by HW
14-10	VP2THREADSIZE	R	0h	Synchronized version of VP2 THREADSIZE. Value used by HW
9-5	VP1THREADSIZE	R	0h	Synchronized version of VP1 THREADSIZE. Value used by HW
4-0	VP0THREADSIZE	R	10h	Synchronized version of VP0 THREADSIZE. Value used by HW

### 12.6.4.11.13.1.51 DSS0\_COMMON\_GLOBAL\_GOBITMODE Register (Offset = F8h) [reset = 0h]

DSS0\_COMMON\_GLOBAL\_GOBITMODE is shown in [Figure 12-662](#) and described in [Table 12-668](#).

Return to [Summary Table](#).

**Table 12-667.**  
**DSS0\_COMMON\_GLOBAL\_GOBITMODE Instances**

Instance	Physical Address
DSS0_DISPC_0_COMMON_M	04A0 00F8h
DSS0_DISPC_0_COMMON_S0	N/A
DSS0_DISPC_0_COMMON_S1	N/A
DSS0_DISPC_0_COMMON_S2	N/A

**Figure 12-662. DSS0\_COMMON\_GLOBAL\_GOBITMODE Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							MODE
R-0h							R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-668. DSS0\_COMMON\_GLOBAL\_GOBITMODE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	MODE	R/W	0h	Go bit 0h = Pipe GO bit settings are disabled 1h = Pipe GO bit settings are enabled



### 12.6.4.11.13.2 DSS\_VID Registers

Table 12-670 lists the memory-mapped registers for the DSS\_VID. All register offset addresses not listed in Table 12-670 should be considered as reserved locations and the register contents should not be modified.

VID and VIDL Registers

**Table 12-669. DSS\_VID Instances**

Instance	Base Address
DSS0_VIDL1	04A2 0000h
DSS0_VIDL2	04A3 0000h
DSS0_VID1	04A5 0000h
DSS0_VID2	04A6 0000h

**Table 12-670. DSS\_VID Registers**

Offset	Acronym	Register Name	DSS0_VIDL1 Physical Address	DSS0_VIDL2 Physical Address	DSS0_VID1 Physical Address	DSS0_VID2 Physical Address
0h	<a href="#">DSS0_VID_ACCUH_0</a>		N/A <sup>(1)</sup>	N/A	04A5 0000h	04A6 0000h
4h	<a href="#">DSS0_VID_ACCUH_1</a>		N/A	N/A	04A5 0004h	04A6 0004h
8h	<a href="#">DSS0_VID_ACCUH2_0</a>		N/A	N/A	04A5 0008h	04A6 0008h
Ch	<a href="#">DSS0_VID_ACCUH2_1</a>		N/A	N/A	04A5 000Ch	04A6 000Ch
10h	<a href="#">DSS0_VID_ACCUV_0</a>		N/A	N/A	04A5 0010h	04A6 0010h
14h	<a href="#">DSS0_VID_ACCUV_1</a>		N/A	N/A	04A5 0014h	04A6 0014h
18h	<a href="#">DSS0_VID_ACCUV2_0</a>		N/A	N/A	04A5 0018h	04A6 0018h
1Ch	<a href="#">DSS0_VID_ACCUV2_1</a>		N/A	N/A	04A5 001Ch	04A6 001Ch
20h	<a href="#">DSS0_VID_ATTRIBUTES</a>		04A2 0020h	04A3 0020h	04A5 0020h	04A6 0020h
24h	<a href="#">DSS0_VID_ATTRIBUTES2</a>		04A2 0024h	04A3 0024h	04A5 0024h	04A6 0024h
28h	<a href="#">DSS0_VID_BA_0</a>		04A2 0028h	04A3 0028h	04A5 0028h	04A6 0028h
2Ch	<a href="#">DSS0_VID_BA_1</a>		04A2 002Ch	04A3 002Ch	04A5 002Ch	04A6 002Ch
30h	<a href="#">DSS0_VID_BA_UV_0</a>		04A2 0030h	04A3 0030h	04A5 0030h	04A6 0030h
34h	<a href="#">DSS0_VID_BA_UV_1</a>		04A2 0034h	04A3 0034h	04A5 0034h	04A6 0034h
38h	<a href="#">DSS0_VID_BUF_SIZE_STATUS</a>		04A2 0038h	04A3 0038h	04A5 0038h	04A6 0038h
3Ch	<a href="#">DSS0_VID_BUF_THRESHOLD</a>		04A2 003Ch	04A3 003Ch	04A5 003Ch	04A6 003Ch
40h	<a href="#">DSS0_VID_CSC_COEF0</a>		04A2 0040h	04A3 0040h	04A5 0040h	04A6 0040h
44h	<a href="#">DSS0_VID_CSC_COEF1</a>		04A2 0044h	04A3 0044h	04A5 0044h	04A6 0044h
48h	<a href="#">DSS0_VID_CSC_COEF2</a>		04A2 0048h	04A3 0048h	04A5 0048h	04A6 0048h
4Ch	<a href="#">DSS0_VID_CSC_COEF3</a>		04A2 004Ch	04A3 004Ch	04A5 004Ch	04A6 004Ch
50h	<a href="#">DSS0_VID_CSC_COEF4</a>		04A2 0050h	04A3 0050h	04A5 0050h	04A6 0050h
54h	<a href="#">DSS0_VID_CSC_COEF5</a>		04A2 0054h	04A3 0054h	04A5 0054h	04A6 0054h
58h	<a href="#">DSS0_VID_CSC_COEF6</a>		04A2 0058h	04A3 0058h	04A5 0058h	04A6 0058h
5Ch	<a href="#">DSS0_VID_FIRH</a>		N/A	N/A	04A5 005Ch	04A6 005Ch
60h	<a href="#">DSS0_VID_FIRH2</a>		N/A	N/A	04A5 0060h	04A6 0060h
64h	<a href="#">DSS0_VID_FIRV</a>		N/A	N/A	04A5 0064h	04A6 0064h
68h	<a href="#">DSS0_VID_FIRV2</a>		N/A	N/A	04A5 0068h	04A6 0068h
6Ch	<a href="#">DSS0_VID_FIR_COEF_H0_0</a>		N/A	N/A	04A5 006Ch	04A6 006Ch
70h	<a href="#">DSS0_VID_FIR_COEF_H0_1</a>		N/A	N/A	04A5 0070h	04A6 0070h
74h	<a href="#">DSS0_VID_FIR_COEF_H0_2</a>		N/A	N/A	04A5 0074h	04A6 0074h
78h	<a href="#">DSS0_VID_FIR_COEF_H0_3</a>		N/A	N/A	04A5 0078h	04A6 0078h
7Ch	<a href="#">DSS0_VID_FIR_COEF_H0_4</a>		N/A	N/A	04A5 007Ch	04A6 007Ch
80h	<a href="#">DSS0_VID_FIR_COEF_H0_5</a>		N/A	N/A	04A5 0080h	04A6 0080h

**Table 12-670. DSS\_VID Registers (continued)**

Offset	Acronym	Register Name	DSS0_VIDL1 Physical Address	DSS0_VIDL2 Physical Address	DSS0_VID1 Physical Address	DSS0_VID2 Physical Address
84h	<a href="#">DSS0_VID_FIR_COEF_H0_6</a>		N/A	N/A	04A5 0084h	04A6 0084h
88h	<a href="#">DSS0_VID_FIR_COEF_H0_7</a>		N/A	N/A	04A5 0088h	04A6 0088h
8Ch	<a href="#">DSS0_VID_FIR_COEF_H0_8</a>		N/A	N/A	04A5 008Ch	04A6 008Ch
90h	<a href="#">DSS0_VID_FIR_COEF_H0_C_0</a>		N/A	N/A	04A5 0090h	04A6 0090h
94h	<a href="#">DSS0_VID_FIR_COEF_H0_C_1</a>		N/A	N/A	04A5 0094h	04A6 0094h
98h	<a href="#">DSS0_VID_FIR_COEF_H0_C_2</a>		N/A	N/A	04A5 0098h	04A6 0098h
9Ch	<a href="#">DSS0_VID_FIR_COEF_H0_C_3</a>		N/A	N/A	04A5 009Ch	04A6 009Ch
A0h	<a href="#">DSS0_VID_FIR_COEF_H0_C_4</a>		N/A	N/A	04A5 00A0h	04A6 00A0h
A4h	<a href="#">DSS0_VID_FIR_COEF_H0_C_5</a>		N/A	N/A	04A5 00A4h	04A6 00A4h
A8h	<a href="#">DSS0_VID_FIR_COEF_H0_C_6</a>		N/A	N/A	04A5 00A8h	04A6 00A8h
ACh	<a href="#">DSS0_VID_FIR_COEF_H0_C_7</a>		N/A	N/A	04A5 00ACh	04A6 00ACh
B0h	<a href="#">DSS0_VID_FIR_COEF_H0_C_8</a>		N/A	N/A	04A5 00B0h	04A6 00B0h
B4h	<a href="#">DSS0_VID_FIR_COEF_H12_0</a>		N/A	N/A	04A5 00B4h	04A6 00B4h
B8h	<a href="#">DSS0_VID_FIR_COEF_H12_1</a>		N/A	N/A	04A5 00B8h	04A6 00B8h
BCh	<a href="#">DSS0_VID_FIR_COEF_H12_2</a>		N/A	N/A	04A5 00BCh	04A6 00BCh
C0h	<a href="#">DSS0_VID_FIR_COEF_H12_3</a>		N/A	N/A	04A5 00C0h	04A6 00C0h
C4h	<a href="#">DSS0_VID_FIR_COEF_H12_4</a>		N/A	N/A	04A5 00C4h	04A6 00C4h
C8h	<a href="#">DSS0_VID_FIR_COEF_H12_5</a>		N/A	N/A	04A5 00C8h	04A6 00C8h
CCh	<a href="#">DSS0_VID_FIR_COEF_H12_6</a>		N/A	N/A	04A5 00CCh	04A6 00CCh
D0h	<a href="#">DSS0_VID_FIR_COEF_H12_7</a>		N/A	N/A	04A5 00D0h	04A6 00D0h
D4h	<a href="#">DSS0_VID_FIR_COEF_H12_8</a>		N/A	N/A	04A5 00D4h	04A6 00D4h
D8h	<a href="#">DSS0_VID_FIR_COEF_H12_9</a>		N/A	N/A	04A5 00D8h	04A6 00D8h
DCh	<a href="#">DSS0_VID_FIR_COEF_H12_10</a>		N/A	N/A	04A5 00DCh	04A6 00DCh
E0h	<a href="#">DSS0_VID_FIR_COEF_H12_11</a>		N/A	N/A	04A5 00E0h	04A6 00E0h
E4h	<a href="#">DSS0_VID_FIR_COEF_H12_12</a>		N/A	N/A	04A5 00E4h	04A6 00E4h
E8h	<a href="#">DSS0_VID_FIR_COEF_H12_13</a>		N/A	N/A	04A5 00E8h	04A6 00E8h
ECh	<a href="#">DSS0_VID_FIR_COEF_H12_14</a>		N/A	N/A	04A5 00ECh	04A6 00ECh
F0h	<a href="#">DSS0_VID_FIR_COEF_H12_15</a>		N/A	N/A	04A5 00F0h	04A6 00F0h
F4h	<a href="#">DSS0_VID_FIR_COEF_H12_C_0</a>		N/A	N/A	04A5 00F4h	04A6 00F4h
F8h	<a href="#">DSS0_VID_FIR_COEF_H12_C_1</a>		N/A	N/A	04A5 00F8h	04A6 00F8h
FCh	<a href="#">DSS0_VID_FIR_COEF_H12_C_2</a>		N/A	N/A	04A5 00FCh	04A6 00FCh
100h	<a href="#">DSS0_VID_FIR_COEF_H12_C_3</a>		N/A	N/A	04A5 0100h	04A6 0100h
104h	<a href="#">DSS0_VID_FIR_COEF_H12_C_4</a>		N/A	N/A	04A5 0104h	04A6 0104h
108h	<a href="#">DSS0_VID_FIR_COEF_H12_C_5</a>		N/A	N/A	04A5 0108h	04A6 0108h
10Ch	<a href="#">DSS0_VID_FIR_COEF_H12_C_6</a>		N/A	N/A	04A5 010Ch	04A6 010Ch
110h	<a href="#">DSS0_VID_FIR_COEF_H12_C_7</a>		N/A	N/A	04A5 0110h	04A6 0110h
114h	<a href="#">DSS0_VID_FIR_COEF_H12_C_8</a>		N/A	N/A	04A5 0114h	04A6 0114h
118h	<a href="#">DSS0_VID_FIR_COEF_H12_C_9</a>		N/A	N/A	04A5 0118h	04A6 0118h
11Ch	<a href="#">DSS0_VID_FIR_COEF_H12_C_10</a>		N/A	N/A	04A5 011Ch	04A6 011Ch
120h	<a href="#">DSS0_VID_FIR_COEF_H12_C_11</a>		N/A	N/A	04A5 0120h	04A6 0120h
124h	<a href="#">DSS0_VID_FIR_COEF_H12_C_12</a>		N/A	N/A	04A5 0124h	04A6 0124h
128h	<a href="#">DSS0_VID_FIR_COEF_H12_C_13</a>		N/A	N/A	04A5 0128h	04A6 0128h
12Ch	<a href="#">DSS0_VID_FIR_COEF_H12_C_14</a>		N/A	N/A	04A5 012Ch	04A6 012Ch
130h	<a href="#">DSS0_VID_FIR_COEF_H12_C_15</a>		N/A	N/A	04A5 0130h	04A6 0130h

**Table 12-670. DSS0\_VID Registers (continued)**

Offset	Acronym	Register Name	DSS0_VIDL1 Physical Address	DSS0_VIDL2 Physical Address	DSS0_VID1 Physical Address	DSS0_VID2 Physical Address
134h	<a href="#">DSS0_VID_FIR_COEF_V0_0</a>		N/A	N/A	04A5 0134h	04A6 0134h
138h	<a href="#">DSS0_VID_FIR_COEF_V0_1</a>		N/A	N/A	04A5 0138h	04A6 0138h
13Ch	<a href="#">DSS0_VID_FIR_COEF_V0_2</a>		N/A	N/A	04A5 013Ch	04A6 013Ch
140h	<a href="#">DSS0_VID_FIR_COEF_V0_3</a>		N/A	N/A	04A5 0140h	04A6 0140h
144h	<a href="#">DSS0_VID_FIR_COEF_V0_4</a>		N/A	N/A	04A5 0144h	04A6 0144h
148h	<a href="#">DSS0_VID_FIR_COEF_V0_5</a>		N/A	N/A	04A5 0148h	04A6 0148h
14Ch	<a href="#">DSS0_VID_FIR_COEF_V0_6</a>		N/A	N/A	04A5 014Ch	04A6 014Ch
150h	<a href="#">DSS0_VID_FIR_COEF_V0_7</a>		N/A	N/A	04A5 0150h	04A6 0150h
154h	<a href="#">DSS0_VID_FIR_COEF_V0_8</a>		N/A	N/A	04A5 0154h	04A6 0154h
158h	<a href="#">DSS0_VID_FIR_COEF_V0_C_0</a>		N/A	N/A	04A5 0158h	04A6 0158h
15Ch	<a href="#">DSS0_VID_FIR_COEF_V0_C_1</a>		N/A	N/A	04A5 015Ch	04A6 015Ch
160h	<a href="#">DSS0_VID_FIR_COEF_V0_C_2</a>		N/A	N/A	04A5 0160h	04A6 0160h
164h	<a href="#">DSS0_VID_FIR_COEF_V0_C_3</a>		N/A	N/A	04A5 0164h	04A6 0164h
168h	<a href="#">DSS0_VID_FIR_COEF_V0_C_4</a>		N/A	N/A	04A5 0168h	04A6 0168h
16Ch	<a href="#">DSS0_VID_FIR_COEF_V0_C_5</a>		N/A	N/A	04A5 016Ch	04A6 016Ch
170h	<a href="#">DSS0_VID_FIR_COEF_V0_C_6</a>		N/A	N/A	04A5 0170h	04A6 0170h
174h	<a href="#">DSS0_VID_FIR_COEF_V0_C_7</a>		N/A	N/A	04A5 0174h	04A6 0174h
178h	<a href="#">DSS0_VID_FIR_COEF_V0_C_8</a>		N/A	N/A	04A5 0178h	04A6 0178h
17Ch	<a href="#">DSS0_VID_FIR_COEF_V12_0</a>		N/A	N/A	04A5 017Ch	04A6 017Ch
180h	<a href="#">DSS0_VID_FIR_COEF_V12_1</a>		N/A	N/A	04A5 0180h	04A6 0180h
184h	<a href="#">DSS0_VID_FIR_COEF_V12_2</a>		N/A	N/A	04A5 0184h	04A6 0184h
188h	<a href="#">DSS0_VID_FIR_COEF_V12_3</a>		N/A	N/A	04A5 0188h	04A6 0188h
18Ch	<a href="#">DSS0_VID_FIR_COEF_V12_4</a>		N/A	N/A	04A5 018Ch	04A6 018Ch
190h	<a href="#">DSS0_VID_FIR_COEF_V12_5</a>		N/A	N/A	04A5 0190h	04A6 0190h
194h	<a href="#">DSS0_VID_FIR_COEF_V12_6</a>		N/A	N/A	04A5 0194h	04A6 0194h
198h	<a href="#">DSS0_VID_FIR_COEF_V12_7</a>		N/A	N/A	04A5 0198h	04A6 0198h
19Ch	<a href="#">DSS0_VID_FIR_COEF_V12_8</a>		N/A	N/A	04A5 019Ch	04A6 019Ch
1A0h	<a href="#">DSS0_VID_FIR_COEF_V12_9</a>		N/A	N/A	04A5 01A0h	04A6 01A0h
1A4h	<a href="#">DSS0_VID_FIR_COEF_V12_10</a>		N/A	N/A	04A5 01A4h	04A6 01A4h
1A8h	<a href="#">DSS0_VID_FIR_COEF_V12_11</a>		N/A	N/A	04A5 01A8h	04A6 01A8h
1ACh	<a href="#">DSS0_VID_FIR_COEF_V12_12</a>		N/A	N/A	04A5 01ACh	04A6 01ACh
1B0h	<a href="#">DSS0_VID_FIR_COEF_V12_13</a>		N/A	N/A	04A5 01B0h	04A6 01B0h
1B4h	<a href="#">DSS0_VID_FIR_COEF_V12_14</a>		N/A	N/A	04A5 01B4h	04A6 01B4h
1B8h	<a href="#">DSS0_VID_FIR_COEF_V12_15</a>		N/A	N/A	04A5 01B8h	04A6 01B8h
1BCh	<a href="#">DSS0_VID_FIR_COEF_V12_C_0</a>		N/A	N/A	04A5 01BCh	04A6 01BCh
1C0h	<a href="#">DSS0_VID_FIR_COEF_V12_C_1</a>		N/A	N/A	04A5 01C0h	04A6 01C0h
1C4h	<a href="#">DSS0_VID_FIR_COEF_V12_C_2</a>		N/A	N/A	04A5 01C4h	04A6 01C4h
1C8h	<a href="#">DSS0_VID_FIR_COEF_V12_C_3</a>		N/A	N/A	04A5 01C8h	04A6 01C8h
1CCh	<a href="#">DSS0_VID_FIR_COEF_V12_C_4</a>		N/A	N/A	04A5 01CCh	04A6 01CCh
1D0h	<a href="#">DSS0_VID_FIR_COEF_V12_C_5</a>		N/A	N/A	04A5 01D0h	04A6 01D0h
1D4h	<a href="#">DSS0_VID_FIR_COEF_V12_C_6</a>		N/A	N/A	04A5 01D4h	04A6 01D4h
1D8h	<a href="#">DSS0_VID_FIR_COEF_V12_C_7</a>		N/A	N/A	04A5 01D8h	04A6 01D8h
1DCh	<a href="#">DSS0_VID_FIR_COEF_V12_C_8</a>		N/A	N/A	04A5 01DCh	04A6 01DCh
1E0h	<a href="#">DSS0_VID_FIR_COEF_V12_C_9</a>		N/A	N/A	04A5 01E0h	04A6 01E0h

**Table 12-670. DSS0\_VID Registers (continued)**

Offset	Acronym	Register Name	DSS0_VIDL1 Physical Address	DSS0_VIDL2 Physical Address	DSS0_VID1 Physical Address	DSS0_VID2 Physical Address
1E4h	<a href="#">DSS0_VID_FIR_COEF_V12_C_10</a>		N/A	N/A	04A5 01E4h	04A6 01E4h
1E8h	<a href="#">DSS0_VID_FIR_COEF_V12_C_11</a>		N/A	N/A	04A5 01E8h	04A6 01E8h
1ECh	<a href="#">DSS0_VID_FIR_COEF_V12_C_12</a>		N/A	N/A	04A5 01ECh	04A6 01ECh
1F0h	<a href="#">DSS0_VID_FIR_COEF_V12_C_13</a>		N/A	N/A	04A5 01F0h	04A6 01F0h
1F4h	<a href="#">DSS0_VID_FIR_COEF_V12_C_14</a>		N/A	N/A	04A5 01F4h	04A6 01F4h
1F8h	<a href="#">DSS0_VID_FIR_COEF_V12_C_15</a>		N/A	N/A	04A5 01F8h	04A6 01F8h
1FCh	<a href="#">DSS0_VID_GLOBAL_ALPHA</a>		04A2 01FCh	04A3 01FCh	04A5 01FCh	04A6 01FCh
208h	<a href="#">DSS0_VID_MFLAG_THRESHOLD</a>		04A2 0208h	04A3 0208h	04A5 0208h	04A6 0208h
20Ch	<a href="#">DSS0_VID_PICTURE_SIZE</a>		04A2 020Ch	04A3 020Ch	04A5 020Ch	04A6 020Ch
210h	<a href="#">DSS0_VID_PIXEL_INC</a>		04A2 0210h	04A3 0210h	04A5 0210h	04A6 0210h
218h	<a href="#">DSS0_VID_PRELOAD</a>		04A2 0218h	04A3 0218h	04A5 0218h	04A6 0218h
21Ch	<a href="#">DSS0_VID_ROW_INC</a>		04A2 021Ch	04A3 021Ch	04A5 021Ch	04A6 021Ch
220h	<a href="#">DSS0_VID_SIZE</a>		04A2 0220h	04A3 0220h	04A5 0220h	04A6 0220h
22Ch	<a href="#">DSS0_VID_BA_EXT_0</a>		04A2 022Ch	04A3 022Ch	04A5 022Ch	04A6 022Ch
230h	<a href="#">DSS0_VID_BA_EXT_1</a>		04A2 0230h	04A3 0230h	04A5 0230h	04A6 0230h
234h	<a href="#">DSS0_VID_BA_UV_EXT_0</a>		04A2 0234h	04A3 0234h	04A5 0234h	04A6 0234h
238h	<a href="#">DSS0_VID_BA_UV_EXT_1</a>		04A2 0238h	04A3 0238h	04A5 0238h	04A6 0238h
23Ch	<a href="#">DSS0_VID_CSC_COEF7</a>		04A2 023Ch	04A3 023Ch	04A5 023Ch	04A6 023Ch
248h	<a href="#">DSS0_VID_ROW_INC_UV</a>		04A2 0248h	04A3 0248h	04A5 0248h	04A6 0248h
24Ch	<a href="#">DSS0_VID_TILE</a>		04A2 024Ch	04A3 024Ch	04A5 024Ch	04A6 024Ch
250h	<a href="#">DSS0_VID_TILE2</a>		04A2 0250h	04A3 0250h	04A5 0250h	04A6 0250h
254h	<a href="#">DSS0_VID_FBDC_ATTRIBUTES</a>		04A2 0254h	04A3 0254h	04A5 0254h	04A6 0254h
258h	<a href="#">DSS0_VID_FBDC_CLEAR_COLOR</a>		04A2 0258h	04A3 0258h	04A5 0258h	04A6 0258h
260h	<a href="#">DSS0_VID_CLUT_0</a>		04A2 0260h	04A3 0260h	04A5 0260h	04A6 0260h
264h	<a href="#">DSS0_VID_CLUT_1</a>		04A2 0264h	04A3 0264h	04A5 0264h	04A6 0264h
268h	<a href="#">DSS0_VID_CLUT_2</a>		04A2 0268h	04A3 0268h	04A5 0268h	04A6 0268h
26Ch	<a href="#">DSS0_VID_CLUT_3</a>		04A2 026Ch	04A3 026Ch	04A5 026Ch	04A6 026Ch
270h	<a href="#">DSS0_VID_CLUT_4</a>		04A2 0270h	04A3 0270h	04A5 0270h	04A6 0270h
274h	<a href="#">DSS0_VID_CLUT_5</a>		04A2 0274h	04A3 0274h	04A5 0274h	04A6 0274h
278h	<a href="#">DSS0_VID_CLUT_6</a>		04A2 0278h	04A3 0278h	04A5 0278h	04A6 0278h
27Ch	<a href="#">DSS0_VID_CLUT_7</a>		04A2 027Ch	04A3 027Ch	04A5 027Ch	04A6 027Ch
280h	<a href="#">DSS0_VID_CLUT_8</a>		04A2 0280h	04A3 0280h	04A5 0280h	04A6 0280h
284h	<a href="#">DSS0_VID_CLUT_9</a>		04A2 0284h	04A3 0284h	04A5 0284h	04A6 0284h
288h	<a href="#">DSS0_VID_CLUT_10</a>		04A2 0288h	04A3 0288h	04A5 0288h	04A6 0288h
28Ch	<a href="#">DSS0_VID_CLUT_11</a>		04A2 028Ch	04A3 028Ch	04A5 028Ch	04A6 028Ch
290h	<a href="#">DSS0_VID_CLUT_12</a>		04A2 0290h	04A3 0290h	04A5 0290h	04A6 0290h
294h	<a href="#">DSS0_VID_CLUT_13</a>		04A2 0294h	04A3 0294h	04A5 0294h	04A6 0294h
298h	<a href="#">DSS0_VID_CLUT_14</a>		04A2 0298h	04A3 0298h	04A5 0298h	04A6 0298h
29Ch	<a href="#">DSS0_VID_CLUT_15</a>		04A2 029Ch	04A3 029Ch	04A5 029Ch	04A6 029Ch
2A0h	<a href="#">DSS0_VID_SAFETY_ATTRIBUTES</a>		04A2 02A0h	04A3 02A0h	04A5 02A0h	04A6 02A0h
2A4h	<a href="#">DSS0_VID_SAFETY_CAPT_SIGNATURE</a>		04A2 02A4h	04A3 02A4h	04A5 02A4h	04A6 02A4h
2A8h	<a href="#">DSS0_VID_SAFETY_POSITION</a>		04A2 02A8h	04A3 02A8h	04A5 02A8h	04A6 02A8h
2ACh	<a href="#">DSS0_VID_SAFETY_REF_SIGNATURE</a>		04A2 02ACh	04A3 02ACh	04A5 02ACh	04A6 02ACh
2B0h	<a href="#">DSS0_VID_SAFETY_SIZE</a>		04A2 02B0h	04A3 02B0h	04A5 02B0h	04A6 02B0h

**Table 12-670. DSS\_VID Registers (continued)**

Offset	Acronym	Register Name	DSS0_VIDL1 Physical Address	DSS0_VIDL2 Physical Address	DSS0_VID1 Physical Address	DSS0_VID2 Physical Address
2B4h	<a href="#">DSS0_VID_SAFETY_LFSR_SEED</a>		04A2 02B4h	04A3 02B4h	04A5 02B4h	04A6 02B4h
2B8h	<a href="#">DSS0_VID_LUMAKEY</a>		04A2 02B8h	04A3 02B8h	04A5 02B8h	04A6 02B8h
2BCh	<a href="#">DSS0_VID_DMA_BUFSIZE</a>		04A2 02BCh	04A3 02BCh	04A5 02BCh	04A6 02BCh
2C0h	<a href="#">DSS0_VID_CROP</a>		04A2 02C0h	04A3 02C0h	04A5 02C0h	04A6 02C0h
2C4h	<a href="#">DSS0_VID_SECURE</a>		04A2 02C4h	04A3 02C4h	04A5 02C4h	04A6 02C4h
2C8h	<a href="#">DSS0_VID_PIPE_GO</a>		04A2 02C8h	04A3 02C8h	04A5 02C8h	04A6 02C8h

(1) N/A = Not Applicable

### 12.6.4.11.13.2.1 DSS0\_VID\_ACCUH\_0 Register (Offset = 0h) [reset = 0h]

DSS0\_VID\_ACCUH\_0 is shown in [Figure 12-663](#) and described in [Table 12-672](#).

Return to [Summary Table](#).

The register configures the resize accumulator init values for horizontal up/down-sampling of the video window. DISPC\_VIDx\_ACCU\_\_0 & DISPC\_VIDx\_ACCU\_\_1 for ping-pong mechanism with external trigger, based on the field polarity. This register is used for ARGB and Y in YUV420/YUV422. Shadow register

**Table 12-671. DSS0\_VID\_ACCUH\_0 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0000h
DSS0_VID2	04A6 0000h

**Figure 12-663. DSS0\_VID\_ACCUH\_0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								HORIZONTALACCU																							
R-0h								R/W-0h																							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-672. DSS0\_VID\_ACCUH\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-0	HORIZONTALACCU	R/W	0h	Horizontal initialization accu signed value

#### 12.6.4.11.13.2.2 DSS0\_VID\_ACCUH\_1 Register (Offset = 4h) [reset = 0h]

DSS0\_VID\_ACCUH\_1 is shown in [Figure 12-664](#) and described in [Table 12-674](#).

Return to [Summary Table](#).

The register configures the resize accumulator init values for horizontal up/down-sampling of the video window. DISPC\_VIDx\_ACCU\_\_0 & DISPC\_VIDx\_ACCU\_\_1 for ping-pong mechanism with external trigger, based on the field polarity. This register is used for ARGB and Y in YUV420/YUV422. Shadow register

**Table 12-673. DSS0\_VID\_ACCUH\_1 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0004h
DSS0_VID2	04A6 0004h

**Figure 12-664. DSS0\_VID\_ACCUH\_1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								HORIZONTALACCU																							
R-0h								R/W-0h																							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-674. DSS0\_VID\_ACCUH\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-0	HORIZONTALACCU	R/W	0h	Horizontal initialization accu signed value

### 12.6.4.11.13.2.3 DSS0\_VID\_ACCUH2\_0 Register (Offset = 8h) [reset = 0h]

DSS0\_VID\_ACCUH2\_0 is shown in [Figure 12-665](#) and described in [Table 12-676](#).

Return to [Summary Table](#).

The register configures the resize accumulator init value for horizontal up/down-sampling of the video window. DISPC\_VID n\_ACCU2\_\_0 & DISPC\_VID n\_ACCU2\_\_1 for ping-pong mechanism with external trigger, based on the field polarity. This register is used for U/V components in YUV420/YUV422. It is not used when the input format is any RGB format. Shadow register

**Table 12-675. DSS0\_VID\_ACCUH2\_0 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0008h
DSS0_VID2	04A6 0008h

**Figure 12-665. DSS0\_VID\_ACCUH2\_0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								HORIZONTALACCU																							
R-0h								R/W-0h																							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-676. DSS0\_VID\_ACCUH2\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-0	HORIZONTALACCU	R/W	0h	Horizontal initialization accu signed value



#### 12.6.4.11.13.2.4 DSS0\_VID\_ACCUH2\_1 Register (Offset = Ch) [reset = 0h]

DSS0\_VID\_ACCUH2\_1 is shown in [Figure 12-666](#) and described in [Table 12-678](#).

Return to [Summary Table](#).

The register configures the resize accumulator init value for horizontal up/down-sampling of the video window. DISPC\_VID n\_ACCU2\_\_0 & DISPC\_VID n\_ACCU2\_\_1 for ping-pong mechanism with external trigger, based on the field polarity. This register is used for U/V components in YUV420/YUV422. It is not used when the input format is any RGB format. Shadow register

**Table 12-677. DSS0\_VID\_ACCUH2\_1 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 000Ch
DSS0_VID2	04A6 000Ch

**Figure 12-666. DSS0\_VID\_ACCUH2\_1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								HORIZONTALACCU																							
R-0h								R/W-0h																							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-678. DSS0\_VID\_ACCUH2\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-0	HORIZONTALACCU	R/W	0h	Horizontal initialization accu signed value

### 12.6.4.11.13.2.5 DSS0\_VID\_ACCUV\_0 Register (Offset = 10h) [reset = 0h]

DSS0\_VID\_ACCUV\_0 is shown in [Figure 12-667](#) and described in [Table 12-680](#).

Return to [Summary Table](#).

The register configures the resize accumulator init values for horizontal and vertical up/down-sampling of the video window. DISPC\_VIDx\_ACCU\_\_0 & DISPC\_VIDx\_ACCU\_\_1 for ping-pong mechanism with external trigger, based on the field polarity. It is used for ARGB and Y in YUV420/YUV422. Shadow register

**Table 12-679. DSS0\_VID\_ACCUV\_0 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0010h
DSS0_VID2	04A6 0010h

**Figure 12-667. DSS0\_VID\_ACCUV\_0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								VERTICALACCU																							
R-0h								R/W-0h																							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-680. DSS0\_VID\_ACCUV\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-0	VERTICALACCU	R/W	0h	Vertical initialization accu signed value

#### 12.6.4.11.13.2.6 DSS0\_VID\_ACCUV\_1 Register (Offset = 14h) [reset = 0h]

DSS0\_VID\_ACCUV\_1 is shown in [Figure 12-668](#) and described in [Table 12-682](#).

Return to [Summary Table](#).

The register configures the resize accumulator init values for horizontal and vertical up/down-sampling of the video window. DISPC\_VIDx\_ACCU\_\_0 & DISPC\_VIDx\_ACCU\_\_1 for ping-pong mechanism with external trigger, based on the field polarity. It is used for ARGB and Y in YUV420/YUV422. Shadow register

**Table 12-681. DSS0\_VID\_ACCUV\_1 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0014h
DSS0_VID2	04A6 0014h

**Figure 12-668. DSS0\_VID\_ACCUV\_1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								VERTICALACCU																							
R-0h								R/W-0h																							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-682. DSS0\_VID\_ACCUV\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-0	VERTICALACCU	R/W	0h	Vertical initialization accu signed value

### 12.6.4.11.13.2.7 DSS0\_VID\_ACCUV2\_0 Register (Offset = 18h) [reset = 0h]

DSS0\_VID\_ACCUV2\_0 is shown in [Figure 12-669](#) and described in [Table 12-684](#).

Return to [Summary Table](#).

The register configures the resize accumulator init value for vertical up/down-sampling of the video window. ACCU2\_\_0 & ACCU2\_\_1 for ping-pong mechanism with external trigger, based on the field polarity. It is used for U/V components for YUV420. It is not used when the input format is any RGB format or YUV422. Shadow register

**Table 12-683. DSS0\_VID\_ACCUV2\_0 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0018h
DSS0_VID2	04A6 0018h

**Figure 12-669. DSS0\_VID\_ACCUV2\_0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								VERTICALACCU																							
R-0h								R/W-0h																							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-684. DSS0\_VID\_ACCUV2\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-0	VERTICALACCU	R/W	0h	Vertical initialization accu signed value

#### 12.6.4.11.13.2.8 DSS0\_VID\_ACCUV2\_1 Register (Offset = 1Ch) [reset = 0h]

DSS0\_VID\_ACCUV2\_1 is shown in [Figure 12-670](#) and described in [Table 12-686](#).

Return to [Summary Table](#).

The register configures the resize accumulator init value for vertical up/down-sampling of the video window. ACCU2\_\_0 & ACCU2\_\_1 for ping-pong mechanism with external trigger, based on the field polarity. It is used for U/V components for YUV420. It is not used when the input format is any RGB format or YUV422. Shadow register

**Table 12-685. DSS0\_VID\_ACCUV2\_1 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 001Ch
DSS0_VID2	04A6 001Ch

**Figure 12-670. DSS0\_VID\_ACCUV2\_1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								VERTICALACCU																							
R-0h								R/W-0h																							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-686. DSS0\_VID\_ACCUV2\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-0	VERTICALACCU	R/W	0h	Vertical initialization accu signed value

### 12.6.4.11.13.2.9 DSS0\_VID\_ATTRIBUTES Register (Offset = 20h) [reset = 0h]

DSS0\_VID\_ATTRIBUTES is shown in [Figure 12-671](#) and described in [Table 12-688](#).

Return to [Summary Table](#).

The register configures the DSS0\_VID\_ATTRIBUTES of the video window. Shadow register

**Table 12-687. DSS0\_VID\_ATTRIBUTES Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0020h
DSS0_VIDL2	04A3 0020h
DSS0_VID1	04A5 0020h
DSS0_VID2	04A6 0020h

**Figure 12-671. DSS0\_VID\_ATTRIBUTES Register**

31	30	29	28	27	26	25	24
LUMAKEYENABLE	GAMMAINVERSION	GAMMAINVERSIONPOS	PREMULTIPLYALPHA	RESERVED		SELFREFRESH	
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R-0h		R/W-0h	
23	22	21	20	19	18	17	16
ARBITRATION	RESERVED	VERTICALTAPS	RESERVED	BUFPRELOAD	RESERVED	SELFREFRESHAUTO	RESERVED
R/W-0h	R-0h	R/W-0h	R-0h	R/W-0h	R-0h	R/W-0h	R-0h
15	14	13	12	11	10	9	8
RESERVED		CROP	FLIP	FULLRANGE	NIBBLEMODE	COLORCONVERTABLE	RESIZEENABLE
R-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RESIZEENABLE	FORMAT						ENABLE
R/W-0h	R/W-0h						R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-688. DSS0\_VID\_ATTRIBUTES Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	LUMAKEYENABLE	R/W	0h	Enable Luma Key transparency matching 0h = Luma Key operation is disabled 1h = Luma Key operation is enabled
30	GAMMAINVERSION	R/W	0h	Inverse Gamma support [using the CLUT table] 0h = Gamma inversion is disabled 1h = Gamma inversion is enabled
29	GAMMAINVERSIONPOS	R/W	0h	Position of Inverse Gamma operation 0h = GAMMAINVERSION is before Scaler. Only Horizontal Resize is possible in this mode 1h = GAMMAINVERSION is after Scaler. No restrictions on resizing

**Table 12-688. DSS0\_VID\_ATTRIBUTES Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
28	PREMULTIPLYALPHA	R/W	0h	The field configures the DISPC VID to process incoming data as premultiplied alpha data or non premultiplied alpha data. Default setting is non premultiplied alpha data  0h = Non premultiplied alpha data color component 1h = Premultiplied alpha data color component
27-25	RESERVED	R	0h	Reserved
24	SELFREFRESH	R/W	0h	Enables the self refresh of the video window, from its own DMA buffer only  0h = The video pipeline accesses the interconnect to fetch data from the system memory 1h = The video pipeline does not need any more to fetch data from memory. Only the DMA buffer associated with the video1 is used. It takes effect after the frame has been loaded in the DMA buffer
23	ARBITRATION	R/W	0h	Determines the priority of the video pipeline. The video pipeline is one of the high priority pipelines. The arbitration gives always the priority first to the high priority pipelines using round-robin between them. When there are only normal priority pipelines sending requests, the round-robin applies between the normal priority pipelines  0h = The video pipeline is one of the normal priority pipelines 1h = The video pipeline is one of the high priority pipelines
22	RESERVED	R	0h	Reserved
21	VERTICALTAPS	R/W	0h	Video Vertical Resize Tap Number. The vertical poly-phase filter can be configured in 3-tap or 5-tap configuration. According to the number of taps, the maximum input picture width is double while using 3-tap compared to 5-tap  0h = 3 taps are used for the vertical filtering logic. The 2 other taps are not used. The associated bit-fields for the 2 other taps coefficients do not need to be initialized 1h = 5 taps are used for the vertical filtering logic
20	RESERVED	R	0h	Reserved
19	BUFPRELOAD	R/W	0h	Video DSS0_VID_PRELOAD Value  0h = H/W prefetches pixels up to the DSS0_VID_PRELOAD value defined in the DSS0_VID_PRELOAD register 1h = H/W prefetches pixels up to high threshold value
18	RESERVED	R	0h	Write 0's for future compatibility. Reads return 0

**Table 12-688. DSS0\_VID\_ATTRIBUTES Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
17	SELFREFRESHAUTO	R/W	0h	Automatic self refresh mode  0h = The transition from SELFREFRESH disabled to enabled is controlled SW  1h = The transition from SELFREFRESH disabled to enabled is controlled only by HW
16-14	RESERVED	R	0h	Reserved
13	CROP	R/W	0h	Enables cropping operation at the output of Video Pipeline  0h = DSS0_VID_CROP feature is disabled  1h = DSS0_VID_CROP feature is enabled
12	FLIP	R/W	0h	Describes the frame buffer flip operation  0h = No Flip  1h = Frame Buffer is flipped
11	FULLRANGE	R/W	0h	Color Space Conversion full range setting  0h = Limited Range Selected  1h = Full Range Selected
10	NIBBLEMODE	R/W	0h	Video Nibble mode [only for 1-, 2- and 4-bpp]  0h = Nibble Mode Disabled  1h = Nibble Mode Enabled
9	COLORCONVENABLE	R/W	0h	Enable the color space conversion. The HW does not enable/disable the conversion based on the pixel format  0h = Color Space Conversion Disabled  1h = Color Space Conversion Enabled
8-7	RESIZEENABLE	R/W	0h	Video Resize Enable  0h = Disable both horizontal and vertical resizing  1h = Enable horizontal resizing  2h = Enable vertical resizing  3h = Enable both horizontal and vertical resizing



**Table 12-688. DSS0\_VID\_ATTRIBUTES Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6-1	FORMAT	R/W	0h	<p>Video Format.</p> <p>It defines the pixel format when fetching the video frame buffer</p> <p>00h = 0x00</p> <p>01h = 0x01</p> <p>02h = 0x02</p> <p>03h = 0x03</p> <p>04h = 0x04</p> <p>05h = 0x05</p> <p>06h = 0x06</p> <p>07h = 0x07</p> <p>08h = 0x08</p> <p>09h = 0x09</p> <p>0Ah = 0x0a</p> <p>0Bh = 0x0b</p> <p>0Ch = 0x0c</p> <p>0Eh = 0x0e</p> <p>0Fh = 0x0f</p> <p>10h = 0x10</p> <p>11h = 0x11</p> <p>12h = 0x12</p> <p>13h = 0x13</p> <p>14h = 0x14</p> <p>15h = 0x15</p> <p>16h = 0x16</p> <p>17h = 0x17</p> <p>20h = 0x20</p> <p>21h = 0x21</p> <p>22h = 0x22</p> <p>25h = 0x25</p> <p>26h = 0x26</p> <p>27h = 0x27</p> <p>28h = 0x28</p> <p>29h = 0x29</p> <p>2Ah = 0x2a</p> <p>2Eh = 0x2e</p> <p>2Fh = 0x2f</p> <p>30h = 0x30</p> <p>31h = 0x31</p>

**Table 12-688. DSS0\_VID\_ATTRIBUTES Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
				3Ch = 0x3c 3Dh = 0x3d 3Eh = 0x3e 3Fh = 0x3f
0	ENABLE	R/W	0h	Video pipeline Enable 0h = Video Pipe Disabled 1h = Video Pipe Enabled

### 12.6.4.11.13.2.10 DSS0\_VID\_ATTRIBUTES2 Register (Offset = 24h) [reset = X]

DSS0\_VID\_ATTRIBUTES2 is shown in [Figure 12-672](#) and described in [Table 12-690](#).

Return to [Summary Table](#).

The register configures the DSS0\_VID\_ATTRIBUTES of the video window. Shadow register

**Table 12-689. DSS0\_VID\_ATTRIBUTES2 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0024h
DSS0_VIDL2	04A3 0024h
DSS0_VID1	04A5 0024h
DSS0_VID2	04A6 0024h

**Figure 12-672. DSS0\_VID\_ATTRIBUTES2 Register**

31	30	29	28	27	26	25	24
RESERVED	TAGS					MPORTSEL	RESERVED
R-0h	R/W-Fh					R/W-0h	R-0h
23	22	21	20	19	18	17	16
RESERVED				RESERVED			
R-0h				R/W-X			
15	14	13	12	11	10	9	8
RESERVED					YUV_ALIGN	YUV_MODE	YUV_SIZE
R-0h					R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
YUV_SIZE	VC1_RANGE_CBCR			VC1_RANGE_Y			VC1ENABLE
R/W-0h	R/W-0h			R/W-0h			R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-690. DSS0\_VID\_ATTRIBUTES2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R	0h	Reserved
30-26	TAGS	R/W	Fh	Number of OCP TAGS to be used for the pipeline [from 0x0 to 0xF]. A value of 0x0 means only a single tag will be used. A value of 0xF means all 16 tags can be used
25	MPORTSEL	R/W	0h	Master-Port Selection. By default, use primary master port only 0h = Use Primary Master Port 1h = Use Secondary Master Port
24-20	RESERVED	R	0h	Reserved
19-16	RESERVED	R/W	X	
15-11	RESERVED	R	0h	Reserved
10	YUV_ALIGN	R/W	0h	Alignment [MSB or LSB align] for unpacked 10b/12b YUV data 0h = lsb aligned - unused msb 1h = msb aligned - unused lsb

**Table 12-690. DSS0\_VID\_ATTRIBUTES2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
9	YUV_MODE	R/W	0h	<p>Mode of packing for YUV data [only for 10b/12b formats]</p> <p>0h = YUV 10-bit formats have the same component packing order as 8-bit formats except that the packing is done across a multiple 32-bit word with 2 MSB in each 32-bit word not used. YUV 12-bit formats have the same component packing order as 8-bit formats except that the packing is done across a multiple 64-bit word with 4 MSB in each 64-bit word not used</p> <p>1h = YUV 10-bit/12-bit unpacked formats have the same component packing order as 8-bit formats except that each component is stored in a 16-bit container - with MSB or LSB bits within the 16-bit container not used depending on the MSB/LSB alignment</p>
8-7	YUV_SIZE	R/W	0h	<p>DSS0_VID_SIZE of YUV data 8b/10b/12b</p> <p>0h = 8b per component-default</p> <p>1h = 10b per component</p> <p>2h = 12b per component</p>
6-4	VC1_RANGE_CBCR	R/W	0h	Defines the VC1 range value for the CbCr component from 0 to 7
3-1	VC1_RANGE_Y	R/W	0h	Defines the VC1 range value for the Y component from 0 to 7
0	VC1ENABLE	R/W	0h	<p>Enable/disable the VC1 range mapping processing.</p> <p>The bit-field is ignored if the format is not one of the supported YUV formats</p> <p>0h = VC1 range mapping disabled</p> <p>1h = VC1 range mapping enabled</p>

#### 12.6.4.11.13.2.11 DSS0\_VID\_BA\_0 Register (Offset = 28h) [reset = 0h]

DSS0\_VID\_BA\_0 is shown in [Figure 12-673](#) and described in [Table 12-692](#).

Return to [Summary Table](#).

The register configures the base address of the single video buffer. In case of single plane ARGB or YUV, this is the BA. In case of two plane YUV, this is the BA\_Y. In case of two plane RGB565-A8, this is the BA\_Alpha. BA\_\_0 & BA\_\_1 for ping-pong mechanism with external trigger, based on the field polarity otherwise only BA\_\_0 is used. Shadow register

**Table 12-691. DSS0\_VID\_BA\_0 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0028h
DSS0_VIDL2	04A3 0028h
DSS0_VID1	04A5 0028h
DSS0_VID2	04A6 0028h

**Figure 12-673. DSS0\_VID\_BA\_0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BA																															
R/W-0h																															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-692. DSS0\_VID\_BA\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	BA	R/W	0h	Video base address. Base address of the video buffer [Aligned on pixel DSS0_VID_SIZE boundary except for the following. In case of RGB24 packed format, 4-pixel alignment is required. In case of YUV422, 2-pixel alignment is required. In case of YUV420, byte alignment is required]

### 12.6.4.11.13.2.12 DSS0\_VID\_BA\_1 Register (Offset = 2Ch) [reset = 0h]

DSS0\_VID\_BA\_1 is shown in [Figure 12-674](#) and described in [Table 12-694](#).

Return to [Summary Table](#).

The register configures the base address of the single video buffer. In case of single plane ARGB or YUV, this is the BA. In case of two plane YUV, this is the BA\_Y. In case of two plane RGB565-A8, this is the BA\_Alpha. BA\_\_0 & BA\_\_1 for ping-pong mechanism with external trigger, based on the field polarity otherwise only BA\_\_0 is used. Shadow register

**Table 12-693. DSS0\_VID\_BA\_1 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 002Ch
DSS0_VIDL2	04A3 002Ch
DSS0_VID1	04A5 002Ch
DSS0_VID2	04A6 002Ch

**Figure 12-674. DSS0\_VID\_BA\_1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BA																															
R/W-0h																															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-694. DSS0\_VID\_BA\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	BA	R/W	0h	Video base address. Base address of the video buffer [Aligned on pixel DSS0_VID_SIZE boundary except for the following. In case of RGB24 packed format, 4-pixel alignment is required. In case of YUV422, 2-pixel alignment is required. In case of YUV420, byte alignment is required]

#### 12.6.4.11.13.2.13 DSS0\_VID\_BA\_UV\_0 Register (Offset = 30h) [reset = 0h]

DSS0\_VID\_BA\_UV\_0 is shown in [Figure 12-675](#) and described in [Table 12-696](#).

Return to [Summary Table](#).

The register configures the base address of the UV buffer for two plane YUV or RGB buffer for two plane RGB565-A8, for the video window. BA\_UV\_\_0 & BA\_UV\_\_1 for ping-pong mechanism with external trigger, based on the field polarity otherwise only BA\_UV\_\_0 is used. Shadow register

**Table 12-695. DSS0\_VID\_BA\_UV\_0 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0030h
DSS0_VIDL2	04A3 0030h
DSS0_VID1	04A5 0030h
DSS0_VID2	04A6 0030h

**Figure 12-675. DSS0\_VID\_BA\_UV\_0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BA																															
R/W-0h																															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-696. DSS0\_VID\_BA\_UV\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	BA	R/W	0h	Video base address aligned on 16-bit boundary Base address of the UV video buffer used only in case of YUV420-NV12

### 12.6.4.11.13.2.14 DSS0\_VID\_BA\_UV\_1 Register (Offset = 34h) [reset = 0h]

DSS0\_VID\_BA\_UV\_1 is shown in [Figure 12-676](#) and described in [Table 12-698](#).

Return to [Summary Table](#).

The register configures the base address of the UV buffer for two plane YUV or RGB buffer for two plane RGB565-A8, for the video window. BA\_UV\_\_0 & BA\_UV\_\_1 for ping-pong mechanism with external trigger, based on the field polarity otherwise only BA\_UV\_\_0 is used. Shadow register

**Table 12-697. DSS0\_VID\_BA\_UV\_1 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0034h
DSS0_VIDL2	04A3 0034h
DSS0_VID1	04A5 0034h
DSS0_VID2	04A6 0034h

**Figure 12-676. DSS0\_VID\_BA\_UV\_1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BA																															
R/W-0h																															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-698. DSS0\_VID\_BA\_UV\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	BA	R/W	0h	Video base address aligned on 16-bit boundary Base address of the UV video buffer used only in case of YUV420-NV12



#### 12.6.4.11.13.2.15 DSS0\_VID\_BUF\_SIZE\_STATUS Register (Offset = 38h) [reset = 1000h]

DSS0\_VID\_BUF\_SIZE\_STATUS is shown in [Figure 12-677](#) and described in [Table 12-700](#).

Return to [Summary Table](#).

The register returns the Video buffer DSS0\_VID\_SIZE for the video pipeline

**Table 12-699. DSS0\_VID\_BUF\_SIZE\_STATUS Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0038h
DSS0_VIDL2	04A3 0038h
DSS0_VID1	04A5 0038h
DSS0_VID2	04A6 0038h

**Figure 12-677. DSS0\_VID\_BUF\_SIZE\_STATUS Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																BUFSIZE															
R-0h																R-1000h															

LEGEND: R = Read Only; -n = value after reset

**Table 12-700. DSS0\_VID\_BUF\_SIZE\_STATUS Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
15-0	BUFSIZE	R	1000h	Video DMA buffer DSS0_VID_SIZE in number of 128-bits

### 12.6.4.11.13.2.16 DSS0\_VID\_BUF\_THRESHOLD Register (Offset = 3Ch) [reset = 0FFF0FF8h]

DSS0\_VID\_BUF\_THRESHOLD is shown in [Figure 12-678](#) and described in [Table 12-702](#).

Return to [Summary Table](#).

The register configures the video buffer associated with the video pipeline. Shadow register

**Table 12-701. DSS0\_VID\_BUF\_THRESHOLD  
Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 003Ch
DSS0_VIDL2	04A3 003Ch
DSS0_VID1	04A5 003Ch
DSS0_VID2	04A6 003Ch

**Figure 12-678. DSS0\_VID\_BUF\_THRESHOLD Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUFHIGHTHRESHOLD																BUFLOWTHRESHOLD															
R/W-FFFh																R/W-FF8h															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-702. DSS0\_VID\_BUF\_THRESHOLD Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	BUFHIGHTHRESHOLD	R/W	FFFh	DMA buffer High Threshold. Number of 128-bits defining the threshold value
15-0	BUFLOWTHRESHOLD	R/W	FF8h	DMA buffer Low Threshold. Number of 128-bits defining the threshold value

### 12.6.4.11.13.2.17 DSS0\_VID\_CSC\_COEF0 Register (Offset = 40h) [reset = 0h]

DSS0\_VID\_CSC\_COEF0 is shown in [Figure 12-679](#) and described in [Table 12-704](#).

Return to [Summary Table](#).

The register configures the color space conversion matrix coefficients. Shadow register

**Table 12-703. DSS0\_VID\_CSC\_COEF0 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0040h
DSS0_VIDL2	04A3 0040h
DSS0_VID1	04A5 0040h
DSS0_VID2	04A6 0040h

**Figure 12-679. DSS0\_VID\_CSC\_COEF0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED								C01							
R-0h								R/W-0h							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								C00							
R-0h								R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-704. DSS0\_VID\_CSC\_COEF0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
26-16	C01	R/W	0h	C01 Coefficient. Encoded signed value [from -1024 to 1023]
15-11	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
10-0	C00	R/W	0h	C00 Coefficient. Encoded signed value [from -1024 to 1023]

### 12.6.4.11.13.2.18 DSS0\_VID\_CSC\_COEF1 Register (Offset = 44h) [reset = 0h]

DSS0\_VID\_CSC\_COEF1 is shown in [Figure 12-680](#) and described in [Table 12-706](#).

Return to [Summary Table](#).

The register configures the color space conversion matrix coefficients. Shadow register

**Table 12-705. DSS0\_VID\_CSC\_COEF1 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0044h
DSS0_VIDL2	04A3 0044h
DSS0_VID1	04A5 0044h
DSS0_VID2	04A6 0044h

**Figure 12-680. DSS0\_VID\_CSC\_COEF1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED								C10							
R-0h								R/W-0h							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								C02							
R-0h								R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-706. DSS0\_VID\_CSC\_COEF1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
26-16	C10	R/W	0h	C10 Coefficient. Encoded signed value [from -1024 to 1023]
15-11	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
10-0	C02	R/W	0h	C02 Coefficient. Encoded signed value [from -1024 to 1023]

### 12.6.4.11.13.2.19 DSS0\_VID\_CSC\_COEF2 Register (Offset = 48h) [reset = 0h]

DSS0\_VID\_CSC\_COEF2 is shown in [Figure 12-681](#) and described in [Table 12-708](#).

Return to [Summary Table](#).

The register configures the color space conversion matrix coefficients. Shadow register

**Table 12-707. DSS0\_VID\_CSC\_COEF2 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0048h
DSS0_VIDL2	04A3 0048h
DSS0_VID1	04A5 0048h
DSS0_VID2	04A6 0048h

**Figure 12-681. DSS0\_VID\_CSC\_COEF2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED								C12							
R-0h								R/W-0h							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								C11							
R-0h								R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-708. DSS0\_VID\_CSC\_COEF2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
26-16	C12	R/W	0h	C12 Coefficient. Encoded signed value [from -1024 to 1023]
15-11	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
10-0	C11	R/W	0h	C11 Coefficient. Encoded signed value [from -1024 to 1023]

### 12.6.4.11.13.2.20 DSS0\_VID\_CSC\_COEF3 Register (Offset = 4Ch) [reset = 0h]

DSS0\_VID\_CSC\_COEF3 is shown in [Figure 12-682](#) and described in [Table 12-710](#).

Return to [Summary Table](#).

The register configures the color space conversion matrix coefficients. Shadow register

**Table 12-709. DSS0\_VID\_CSC\_COEF3 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 004Ch
DSS0_VIDL2	04A3 004Ch
DSS0_VID1	04A5 004Ch
DSS0_VID2	04A6 004Ch

**Figure 12-682. DSS0\_VID\_CSC\_COEF3 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED								C21							
R-0h								R/W-0h							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								C20							
R-0h								R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-710. DSS0\_VID\_CSC\_COEF3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
26-16	C21	R/W	0h	C21 coefficient. Encoded signed value [from -1024 to 1023]
15-11	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
10-0	C20	R/W	0h	C20 coefficient. Encoded signed value [from -1024 to 1023]

### 12.6.4.11.13.2.21 DSS0\_VID\_CSC\_COEF4 Register (Offset = 50h) [reset = 0h]

DSS0\_VID\_CSC\_COEF4 is shown in [Figure 12-683](#) and described in [Table 12-712](#).

Return to [Summary Table](#).

The register configures the color space conversion matrix coefficients. Shadow register

**Table 12-711. DSS0\_VID\_CSC\_COEF4 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0050h
DSS0_VIDL2	04A3 0050h
DSS0_VID1	04A5 0050h
DSS0_VID2	04A6 0050h

**Figure 12-683. DSS0\_VID\_CSC\_COEF4 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
RESERVED																					C22																
R-0h																					R/W-0h																

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-712. DSS0\_VID\_CSC\_COEF4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-11	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
10-0	C22	R/W	0h	C22 Coefficient. Encoded signed value [from -1024 to 1023]

### 12.6.4.11.13.2.22 DSS0\_VID\_CSC\_COEF5 Register (Offset = 54h) [reset = 0h]

DSS0\_VID\_CSC\_COEF5 is shown in [Figure 12-684](#) and described in [Table 12-714](#).

Return to [Summary Table](#).

The register configures the color space conversion matrix coefficients. Shadow register

**Table 12-713. DSS0\_VID\_CSC\_COEF5 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0054h
DSS0_VIDL2	04A3 0054h
DSS0_VID1	04A5 0054h
DSS0_VID2	04A6 0054h

**Figure 12-684. DSS0\_VID\_CSC\_COEF5 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PREOFFSET2												RESERVED			
R/W-0h												R-0h			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PREOFFSET1												RESERVED			
R/W-0h												R-0h			

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-714. DSS0\_VID\_CSC\_COEF5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-19	PREOFFSET2	R/W	0h	Row-2 pre-offset. Encoded signed value [from -4096 to 4095]
18-16	RESERVED	R	0h	Reserved
15-3	PREOFFSET1	R/W	0h	Row1 pre-offset. Encoded signed value [from -4096 to 4095]
2-0	RESERVED	R	0h	Reserved



### 12.6.4.11.13.2.23 DSS0\_VID\_CSC\_COEF6 Register (Offset = 58h) [reset = 0h]

DSS0\_VID\_CSC\_COEF6 is shown in [Figure 12-685](#) and described in [Table 12-716](#).

Return to [Summary Table](#).

The register configures the color space conversion matrix coefficients. Shadow register

**Table 12-715. DSS0\_VID\_CSC\_COEF6 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0058h
DSS0_VIDL2	04A3 0058h
DSS0_VID1	04A5 0058h
DSS0_VID2	04A6 0058h

**Figure 12-685. DSS0\_VID\_CSC\_COEF6 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
POSTOFFSET1												RESERVED			
R/W-0h												R-0h			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PREOFFSET3												RESERVED			
R/W-0h												R-0h			

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-716. DSS0\_VID\_CSC\_COEF6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-19	POSTOFFSET1	R/W	0h	Row-1 post-offset. Encoded signed value [from -4096 to 4095]
18-16	RESERVED	R	0h	Reserved
15-3	PREOFFSET3	R/W	0h	Row-3 pre-offset. Encoded signed value [from -4096 to 4095]
2-0	RESERVED	R	0h	Reserved

#### 12.6.4.11.13.2.24 DSS0\_VID\_FIRH Register (Offset = 5Ch) [reset = 00200000h]

DSS0\_VID\_FIRH is shown in [Figure 12-686](#) and described in [Table 12-718](#).

Return to [Summary Table](#).

The register configures the resize factor for horizontal up/down-sampling of the video window. It is used for ARGB and Y setting. Shadow register

**Table 12-717. DSS0\_VID\_FIRH Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 005Ch
DSS0_VID2	04A6 005Ch

**Figure 12-686. DSS0\_VID\_FIRH Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								FIRHINC																							
R-0h								R/W-00200000h																							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-718. DSS0\_VID\_FIRH Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-0	FIRHINC	R/W	00200000h	Horizontal increment of the up/down-sampling filter. The value 0 is invalid

#### 12.6.4.11.13.2.25 DSS0\_VID\_FIRH2 Register (Offset = 60h) [reset = 00200000h]

DSS0\_VID\_FIRH2 is shown in [Figure 12-687](#) and described in [Table 12-720](#).

Return to [Summary Table](#).

The register configures the resize factor for horizontal up/down-sampling of the video window. It is used for U/V components for YUV 422 and 420 input formats. It is not used if input format is any RGB format. Shadow register

**Table 12-719. DSS0\_VID\_FIRH2 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0060h
DSS0_VID2	04A6 0060h

**Figure 12-687. DSS0\_VID\_FIRH2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								FIRHINC																							
R-0h								R/W-00200000h																							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-720. DSS0\_VID\_FIRH2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-0	FIRHINC	R/W	00200000h	Horizontal increment of the up/down-sampling filter for Cb and Cr. The value 0 is invalid

### 12.6.4.11.13.2.26 DSS0\_VID\_FIRV Register (Offset = 64h) [reset = 00200000h]

DSS0\_VID\_FIRV is shown in [Figure 12-688](#) and described in [Table 12-722](#).

Return to [Summary Table](#).

The register configures the resize factor for vertical up/down-sampling of the video window. It is used for ARGB and Y setting. Shadow register

**Table 12-721. DSS0\_VID\_FIRV Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0064h
DSS0_VID2	04A6 0064h

**Figure 12-688. DSS0\_VID\_FIRV Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								FIRVINC																							
R-0h								R/W-00200000h																							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-722. DSS0\_VID\_FIRV Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-0	FIRVINC	R/W	00200000h	Vertical increment of the up/down-sampling filter. The value 0 is invalid

#### 12.6.4.11.13.2.27 DSS0\_VID\_FIRV2 Register (Offset = 68h) [reset = 00200000h]

DSS0\_VID\_FIRV2 is shown in [Figure 12-689](#) and described in [Table 12-724](#).

Return to [Summary Table](#).

The register configures the resize factor for vertical up/down-sampling of the video window. It is used for U/V components for YUV420 input format. It is not used when the input format is any RGB format or YUV422 format. Shadow register.

**Table 12-723. DSS0\_VID\_FIRV2 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0068h
DSS0_VID2	04A6 0068h

**Figure 12-689. DSS0\_VID\_FIRV2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								FIRVINC																							
R-0h								R/W-00200000h																							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-724. DSS0\_VID\_FIRV2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-0	FIRVINC	R/W	00200000h	Vertical increment of the up/down-sampling filter for Cb and Cr. The value 0 is invalid

### 12.6.4.11.13.2.28 DSS0\_VID\_FIR\_COEF\_H0\_0 Register (Offset = 6Ch) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H0\_0 is shown in [Figure 12-690](#) and described in [Table 12-726](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-725. DSS0\_VID\_FIR\_COEF\_H0\_0 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 006Ch
DSS0_VID2	04A6 006Ch

**Figure 12-690. DSS0\_VID\_FIR\_COEF\_H0\_0 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-726. DSS0\_VID\_FIR\_COEF\_H0\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase 0

#### 12.6.4.11.13.2.29 DSS0\_VID\_FIR\_COEF\_H0\_1 Register (Offset = 70h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H0\_1 is shown in [Figure 12-691](#) and described in [Table 12-728](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-727. DSS0\_VID\_FIR\_COEF\_H0\_1 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0070h
DSS0_VID2	04A6 0070h

**Figure 12-691. DSS0\_VID\_FIR\_COEF\_H0\_1 Register**

31	30	29	28	27	26	25	24
RESERVED		RESERVED					
R-0h		R-0h					
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-728. DSS0\_VID\_FIR\_COEF\_H0\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase 1

### 12.6.4.11.13.2.30 DSS0\_VID\_FIR\_COEF\_H0\_2 Register (Offset = 74h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H0\_2 is shown in [Figure 12-692](#) and described in [Table 12-730](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-729. DSS0\_VID\_FIR\_COEF\_H0\_2 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0074h
DSS0_VID2	04A6 0074h

**Figure 12-692. DSS0\_VID\_FIR\_COEF\_H0\_2 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-730. DSS0\_VID\_FIR\_COEF\_H0\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase 2



#### 12.6.4.11.13.2.31 DSS0\_VID\_FIR\_COEF\_H0\_3 Register (Offset = 78h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H0\_3 is shown in [Figure 12-693](#) and described in [Table 12-732](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-731. DSS0\_VID\_FIR\_COEF\_H0\_3 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0078h
DSS0_VID2	04A6 0078h

**Figure 12-693. DSS0\_VID\_FIR\_COEF\_H0\_3 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-732. DSS0\_VID\_FIR\_COEF\_H0\_3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase 3

### 12.6.4.11.13.2.32 DSS0\_VID\_FIR\_COEF\_H0\_4 Register (Offset = 7Ch) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H0\_4 is shown in [Figure 12-694](#) and described in [Table 12-734](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-733. DSS0\_VID\_FIR\_COEF\_H0\_4 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 007Ch
DSS0_VID2	04A6 007Ch

**Figure 12-694. DSS0\_VID\_FIR\_COEF\_H0\_4 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-734. DSS0\_VID\_FIR\_COEF\_H0\_4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase 4

#### 12.6.4.11.13.2.33 DSS0\_VID\_FIR\_COEF\_H0\_5 Register (Offset = 80h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H0\_5 is shown in [Figure 12-695](#) and described in [Table 12-736](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-735. DSS0\_VID\_FIR\_COEF\_H0\_5 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0080h
DSS0_VID2	04A6 0080h

**Figure 12-695. DSS0\_VID\_FIR\_COEF\_H0\_5 Register**

31	30	29	28	27	26	25	24
RESERVED		RESERVED					
R-0h		R-0h					
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-736. DSS0\_VID\_FIR\_COEF\_H0\_5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase 5

### 12.6.4.11.13.2.34 DSS0\_VID\_FIR\_COEF\_H0\_6 Register (Offset = 84h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H0\_6 is shown in [Figure 12-696](#) and described in [Table 12-738](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-737. DSS0\_VID\_FIR\_COEF\_H0\_6 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0084h
DSS0_VID2	04A6 0084h

**Figure 12-696. DSS0\_VID\_FIR\_COEF\_H0\_6 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-738. DSS0\_VID\_FIR\_COEF\_H0\_6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase 6

#### 12.6.4.11.13.2.35 DSS0\_VID\_FIR\_COEF\_H0\_7 Register (Offset = 88h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H0\_7 is shown in [Figure 12-697](#) and described in [Table 12-740](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-739. DSS0\_VID\_FIR\_COEF\_H0\_7 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0088h
DSS0_VID2	04A6 0088h

**Figure 12-697. DSS0\_VID\_FIR\_COEF\_H0\_7 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-740. DSS0\_VID\_FIR\_COEF\_H0\_7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase 7

### 12.6.4.11.13.2.36 DSS0\_VID\_FIR\_COEF\_H0\_8 Register (Offset = 8Ch) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H0\_8 is shown in [Figure 12-698](#) and described in [Table 12-742](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-741. DSS0\_VID\_FIR\_COEF\_H0\_8 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 008Ch
DSS0_VID2	04A6 008Ch

**Figure 12-698. DSS0\_VID\_FIR\_COEF\_H0\_8 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-742. DSS0\_VID\_FIR\_COEF\_H0\_8 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase 8

#### 12.6.4.11.13.2.37 DSS0\_VID\_FIR\_COEF\_H0\_C\_0 Register (Offset = 90h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H0\_C\_0 is shown in [Figure 12-699](#) and described in [Table 12-744](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-743. DSS0\_VID\_FIR\_COEF\_H0\_C\_0  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0090h
DSS0_VID2	04A6 0090h

**Figure 12-699. DSS0\_VID\_FIR\_COEF\_H0\_C\_0 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-744. DSS0\_VID\_FIR\_COEF\_H0\_C\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase 0

### 12.6.4.11.13.2.38 DSS0\_VID\_FIR\_COEF\_H0\_C\_1 Register (Offset = 94h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H0\_C\_1 is shown in [Figure 12-700](#) and described in [Table 12-746](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-745. DSS0\_VID\_FIR\_COEF\_H0\_C\_1  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0094h
DSS0_VID2	04A6 0094h

**Figure 12-700. DSS0\_VID\_FIR\_COEF\_H0\_C\_1 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-746. DSS0\_VID\_FIR\_COEF\_H0\_C\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase 1



#### 12.6.4.11.13.2.39 DSS0\_VID\_FIR\_COEF\_H0\_C\_2 Register (Offset = 98h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H0\_C\_2 is shown in [Figure 12-701](#) and described in [Table 12-748](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-747. DSS0\_VID\_FIR\_COEF\_H0\_C\_2  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0098h
DSS0_VID2	04A6 0098h

**Figure 12-701. DSS0\_VID\_FIR\_COEF\_H0\_C\_2 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-748. DSS0\_VID\_FIR\_COEF\_H0\_C\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase 2

### 12.6.4.11.13.2.40 DSS0\_VID\_FIR\_COEF\_H0\_C\_3 Register (Offset = 9Ch) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H0\_C\_3 is shown in [Figure 12-702](#) and described in [Table 12-750](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-749. DSS0\_VID\_FIR\_COEF\_H0\_C\_3  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 009Ch
DSS0_VID2	04A6 009Ch

**Figure 12-702. DSS0\_VID\_FIR\_COEF\_H0\_C\_3 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-750. DSS0\_VID\_FIR\_COEF\_H0\_C\_3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase 3

#### 12.6.4.11.13.2.41 DSS0\_VID\_FIR\_COEF\_H0\_C\_4 Register (Offset = A0h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H0\_C\_4 is shown in [Figure 12-703](#) and described in [Table 12-752](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-751. DSS0\_VID\_FIR\_COEF\_H0\_C\_4  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 00A0h
DSS0_VID2	04A6 00A0h

**Figure 12-703. DSS0\_VID\_FIR\_COEF\_H0\_C\_4 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-752. DSS0\_VID\_FIR\_COEF\_H0\_C\_4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase 4

#### 12.6.4.11.13.2.42 DSS0\_VID\_FIR\_COEF\_H0\_C\_5 Register (Offset = A4h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H0\_C\_5 is shown in [Figure 12-704](#) and described in [Table 12-754](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-753. DSS0\_VID\_FIR\_COEF\_H0\_C\_5  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 00A4h
DSS0_VID2	04A6 00A4h

**Figure 12-704. DSS0\_VID\_FIR\_COEF\_H0\_C\_5 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-754. DSS0\_VID\_FIR\_COEF\_H0\_C\_5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase 5

#### 12.6.4.11.13.2.43 DSS0\_VID\_FIR\_COEF\_H0\_C\_6 Register (Offset = A8h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H0\_C\_6 is shown in [Figure 12-705](#) and described in [Table 12-756](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-755. DSS0\_VID\_FIR\_COEF\_H0\_C\_6  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 00A8h
DSS0_VID2	04A6 00A8h

**Figure 12-705. DSS0\_VID\_FIR\_COEF\_H0\_C\_6 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-756. DSS0\_VID\_FIR\_COEF\_H0\_C\_6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase 6

### 12.6.4.11.13.2.44 DSS0\_VID\_FIR\_COEF\_H0\_C\_7 Register (Offset = ACh) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H0\_C\_7 is shown in [Figure 12-706](#) and described in [Table 12-758](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-757. DSS0\_VID\_FIR\_COEF\_H0\_C\_7  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 00ACh
DSS0_VID2	04A6 00ACh

**Figure 12-706. DSS0\_VID\_FIR\_COEF\_H0\_C\_7 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-758. DSS0\_VID\_FIR\_COEF\_H0\_C\_7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase 7

#### 12.6.4.11.13.2.45 DSS0\_VID\_FIR\_COEF\_H0\_C\_8 Register (Offset = B0h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H0\_C\_8 is shown in [Figure 12-707](#) and described in [Table 12-760](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-759. DSS0\_VID\_FIR\_COEF\_H0\_C\_8 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 00B0h
DSS0_VID2	04A6 00B0h

**Figure 12-707. DSS0\_VID\_FIR\_COEF\_H0\_C\_8 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-760. DSS0\_VID\_FIR\_COEF\_H0\_C\_8 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase 8

### 12.6.4.11.13.2.46 DSS0\_VID\_FIR\_COEF\_H12\_0 Register (Offset = B4h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H12\_0 is shown in [Figure 12-708](#) and described in [Table 12-762](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-761. DSS0\_VID\_FIR\_COEF\_H12\_0  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 00B4h
DSS0_VID2	04A6 00B4h

**Figure 12-708. DSS0\_VID\_FIR\_COEF\_H12\_0 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-762. DSS0\_VID\_FIR\_COEF\_H12\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase 0
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase 0
9-0	RESERVED	R	0h	Reserved



#### 12.6.4.11.13.2.47 DSS0\_VID\_FIR\_COEF\_H12\_1 Register (Offset = B8h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H12\_1 is shown in [Figure 12-709](#) and described in [Table 12-764](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-763. DSS0\_VID\_FIR\_COEF\_H12\_1  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 00B8h
DSS0_VID2	04A6 00B8h

**Figure 12-709. DSS0\_VID\_FIR\_COEF\_H12\_1 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-764. DSS0\_VID\_FIR\_COEF\_H12\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase 1
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase 1
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.48 DSS0\_VID\_FIR\_COEF\_H12\_2 Register (Offset = BCh) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H12\_2 is shown in [Figure 12-710](#) and described in [Table 12-766](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-765. DSS0\_VID\_FIR\_COEF\_H12\_2  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 00BCh
DSS0_VID2	04A6 00BCh

**Figure 12-710. DSS0\_VID\_FIR\_COEF\_H12\_2 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-766. DSS0\_VID\_FIR\_COEF\_H12\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase 2
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase 2
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.49 DSS0\_VID\_FIR\_COEF\_H12\_3 Register (Offset = C0h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H12\_3 is shown in [Figure 12-711](#) and described in [Table 12-768](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-767. DSS0\_VID\_FIR\_COEF\_H12\_3  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 00C0h
DSS0_VID2	04A6 00C0h

**Figure 12-711. DSS0\_VID\_FIR\_COEF\_H12\_3 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-768. DSS0\_VID\_FIR\_COEF\_H12\_3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase 3
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase 3
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.50 DSS0\_VID\_FIR\_COEF\_H12\_4 Register (Offset = C4h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H12\_4 is shown in [Figure 12-712](#) and described in [Table 12-770](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-769. DSS0\_VID\_FIR\_COEF\_H12\_4  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 00C4h
DSS0_VID2	04A6 00C4h

**Figure 12-712. DSS0\_VID\_FIR\_COEF\_H12\_4 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-770. DSS0\_VID\_FIR\_COEF\_H12\_4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase 4
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase 4
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.51 DSS0\_VID\_FIR\_COEF\_H12\_5 Register (Offset = C8h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H12\_5 is shown in [Figure 12-713](#) and described in [Table 12-772](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-771. DSS0\_VID\_FIR\_COEF\_H12\_5  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 00C8h
DSS0_VID2	04A6 00C8h

**Figure 12-713. DSS0\_VID\_FIR\_COEF\_H12\_5 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-772. DSS0\_VID\_FIR\_COEF\_H12\_5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase 5
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase 5
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.52 DSS0\_VID\_FIR\_COEF\_H12\_6 Register (Offset = CCh) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H12\_6 is shown in [Figure 12-714](#) and described in [Table 12-774](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-773. DSS0\_VID\_FIR\_COEF\_H12\_6  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 00CCh
DSS0_VID2	04A6 00CCh

**Figure 12-714. DSS0\_VID\_FIR\_COEF\_H12\_6 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-774. DSS0\_VID\_FIR\_COEF\_H12\_6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase 6
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase 6
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.53 DSS0\_VID\_FIR\_COEF\_H12\_7 Register (Offset = D0h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H12\_7 is shown in [Figure 12-715](#) and described in [Table 12-776](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-775. DSS0\_VID\_FIR\_COEF\_H12\_7  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 00D0h
DSS0_VID2	04A6 00D0h

**Figure 12-715. DSS0\_VID\_FIR\_COEF\_H12\_7 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-776. DSS0\_VID\_FIR\_COEF\_H12\_7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase 7
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase 7
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.54 DSS0\_VID\_FIR\_COEF\_H12\_8 Register (Offset = D4h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H12\_8 is shown in [Figure 12-716](#) and described in [Table 12-778](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-777. DSS0\_VID\_FIR\_COEF\_H12\_8  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 00D4h
DSS0_VID2	04A6 00D4h

**Figure 12-716. DSS0\_VID\_FIR\_COEF\_H12\_8 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-778. DSS0\_VID\_FIR\_COEF\_H12\_8 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase 8
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase 8
9-0	RESERVED	R	0h	Reserved



### 12.6.4.11.13.2.55 DSS0\_VID\_FIR\_COEF\_H12\_9 Register (Offset = D8h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H12\_9 is shown in [Figure 12-717](#) and described in [Table 12-780](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-779. DSS0\_VID\_FIR\_COEF\_H12\_9  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 00D8h
DSS0_VID2	04A6 00D8h

**Figure 12-717. DSS0\_VID\_FIR\_COEF\_H12\_9 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-780. DSS0\_VID\_FIR\_COEF\_H12\_9 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase 9
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase 9
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.56 DSS0\_VID\_FIR\_COEF\_H12\_10 Register (Offset = DCh) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H12\_10 is shown in [Figure 12-718](#) and described in [Table 12-782](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-781. DSS0\_VID\_FIR\_COEF\_H12\_10  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 00DCh
DSS0_VID2	04A6 00DCh

**Figure 12-718. DSS0\_VID\_FIR\_COEF\_H12\_10 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-782. DSS0\_VID\_FIR\_COEF\_H12\_10 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase 10
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase 10
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.57 DSS0\_VID\_FIR\_COEF\_H12\_11 Register (Offset = E0h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H12\_11 is shown in [Figure 12-719](#) and described in [Table 12-784](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-783. DSS0\_VID\_FIR\_COEF\_H12\_11  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 00E0h
DSS0_VID2	04A6 00E0h

**Figure 12-719. DSS0\_VID\_FIR\_COEF\_H12\_11 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-784. DSS0\_VID\_FIR\_COEF\_H12\_11 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase 11
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase 11
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.58 DSS0\_VID\_FIR\_COEF\_H12\_12 Register (Offset = E4h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H12\_12 is shown in [Figure 12-720](#) and described in [Table 12-786](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-785. DSS0\_VID\_FIR\_COEF\_H12\_12  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 00E4h
DSS0_VID2	04A6 00E4h

**Figure 12-720. DSS0\_VID\_FIR\_COEF\_H12\_12 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-786. DSS0\_VID\_FIR\_COEF\_H12\_12 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase 12
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase 12
9-0	RESERVED	R	0h	Reserved

#### 12.6.4.11.13.2.59 DSS0\_VID\_FIR\_COEF\_H12\_13 Register (Offset = E8h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H12\_13 is shown in [Figure 12-721](#) and described in [Table 12-788](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-787. DSS0\_VID\_FIR\_COEF\_H12\_13  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 00E8h
DSS0_VID2	04A6 00E8h

**Figure 12-721. DSS0\_VID\_FIR\_COEF\_H12\_13 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-788. DSS0\_VID\_FIR\_COEF\_H12\_13 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase 13
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase 13
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.60 DSS0\_VID\_FIR\_COEF\_H12\_14 Register (Offset = ECh) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H12\_14 is shown in [Figure 12-722](#) and described in [Table 12-790](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-789. DSS0\_VID\_FIR\_COEF\_H12\_14  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 00ECh
DSS0_VID2	04A6 00ECh

**Figure 12-722. DSS0\_VID\_FIR\_COEF\_H12\_14 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-790. DSS0\_VID\_FIR\_COEF\_H12\_14 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase 14
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase 14
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.61 DSS0\_VID\_FIR\_COEF\_H12\_15 Register (Offset = F0h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H12\_15 is shown in [Figure 12-723](#) and described in [Table 12-792](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-791. DSS0\_VID\_FIR\_COEF\_H12\_15  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 00F0h
DSS0_VID2	04A6 00F0h

**Figure 12-723. DSS0\_VID\_FIR\_COEF\_H12\_15 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-792. DSS0\_VID\_FIR\_COEF\_H12\_15 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase 15
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase 15
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.62 DSS0\_VID\_FIR\_COEF\_H12\_C\_0 Register (Offset = F4h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H12\_C\_0 is shown in [Figure 12-724](#) and described in [Table 12-794](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-793. DSS0\_VID\_FIR\_COEF\_H12\_C\_0  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 00F4h
DSS0_VID2	04A6 00F4h

**Figure 12-724. DSS0\_VID\_FIR\_COEF\_H12\_C\_0 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-794. DSS0\_VID\_FIR\_COEF\_H12\_C\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase 0
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase 0
9-0	RESERVED	R	0h	Reserved



### 12.6.4.11.13.2.63 DSS0\_VID\_FIR\_COEF\_H12\_C\_1 Register (Offset = F8h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H12\_C\_1 is shown in [Figure 12-725](#) and described in [Table 12-796](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-795. DSS0\_VID\_FIR\_COEF\_H12\_C\_1  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 00F8h
DSS0_VID2	04A6 00F8h

**Figure 12-725. DSS0\_VID\_FIR\_COEF\_H12\_C\_1 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-796. DSS0\_VID\_FIR\_COEF\_H12\_C\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase 1
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase 1
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.64 DSS0\_VID\_FIR\_COEF\_H12\_C\_2 Register (Offset = FCh) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H12\_C\_2 is shown in [Figure 12-726](#) and described in [Table 12-798](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-797. DSS0\_VID\_FIR\_COEF\_H12\_C\_2  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 00FCh
DSS0_VID2	04A6 00FCh

**Figure 12-726. DSS0\_VID\_FIR\_COEF\_H12\_C\_2 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-798. DSS0\_VID\_FIR\_COEF\_H12\_C\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase 2
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase 2
9-0	RESERVED	R	0h	Reserved

#### 12.6.4.11.13.2.65 DSS0\_VID\_FIR\_COEF\_H12\_C\_3 Register (Offset = 100h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H12\_C\_3 is shown in [Figure 12-727](#) and described in [Table 12-800](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-799. DSS0\_VID\_FIR\_COEF\_H12\_C\_3  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0100h
DSS0_VID2	04A6 0100h

**Figure 12-727. DSS0\_VID\_FIR\_COEF\_H12\_C\_3 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-800. DSS0\_VID\_FIR\_COEF\_H12\_C\_3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase 3
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase 3
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.66 DSS0\_VID\_FIR\_COEF\_H12\_C\_4 Register (Offset = 104h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H12\_C\_4 is shown in [Figure 12-728](#) and described in [Table 12-802](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-801. DSS0\_VID\_FIR\_COEF\_H12\_C\_4  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0104h
DSS0_VID2	04A6 0104h

**Figure 12-728. DSS0\_VID\_FIR\_COEF\_H12\_C\_4 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-802. DSS0\_VID\_FIR\_COEF\_H12\_C\_4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase 4
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase 4
9-0	RESERVED	R	0h	Reserved

#### 12.6.4.11.13.2.67 DSS0\_VID\_FIR\_COEF\_H12\_C\_5 Register (Offset = 108h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H12\_C\_5 is shown in [Figure 12-729](#) and described in [Table 12-804](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-803. DSS0\_VID\_FIR\_COEF\_H12\_C\_5  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0108h
DSS0_VID2	04A6 0108h

**Figure 12-729. DSS0\_VID\_FIR\_COEF\_H12\_C\_5 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-804. DSS0\_VID\_FIR\_COEF\_H12\_C\_5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase 5
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase 5
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.68 DSS0\_VID\_FIR\_COEF\_H12\_C\_6 Register (Offset = 10Ch) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H12\_C\_6 is shown in [Figure 12-730](#) and described in [Table 12-806](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-805. DSS0\_VID\_FIR\_COEF\_H12\_C\_6  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 010Ch
DSS0_VID2	04A6 010Ch

**Figure 12-730. DSS0\_VID\_FIR\_COEF\_H12\_C\_6 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-806. DSS0\_VID\_FIR\_COEF\_H12\_C\_6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase 6
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase 6
9-0	RESERVED	R	0h	Reserved

#### 12.6.4.11.13.2.69 DSS0\_VID\_FIR\_COEF\_H12\_C\_7 Register (Offset = 110h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H12\_C\_7 is shown in [Figure 12-731](#) and described in [Table 12-808](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-807. DSS0\_VID\_FIR\_COEF\_H12\_C\_7  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0110h
DSS0_VID2	04A6 0110h

**Figure 12-731. DSS0\_VID\_FIR\_COEF\_H12\_C\_7 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-808. DSS0\_VID\_FIR\_COEF\_H12\_C\_7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase 7
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase 7
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.70 DSS0\_VID\_FIR\_COEF\_H12\_C\_8 Register (Offset = 114h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H12\_C\_8 is shown in [Figure 12-732](#) and described in [Table 12-810](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-809. DSS0\_VID\_FIR\_COEF\_H12\_C\_8  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0114h
DSS0_VID2	04A6 0114h

**Figure 12-732. DSS0\_VID\_FIR\_COEF\_H12\_C\_8 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-810. DSS0\_VID\_FIR\_COEF\_H12\_C\_8 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase 8
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase 8
9-0	RESERVED	R	0h	Reserved



#### 12.6.4.11.13.2.71 DSS0\_VID\_FIR\_COEF\_H12\_C\_9 Register (Offset = 118h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H12\_C\_9 is shown in [Figure 12-733](#) and described in [Table 12-812](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-811. DSS0\_VID\_FIR\_COEF\_H12\_C\_9  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0118h
DSS0_VID2	04A6 0118h

**Figure 12-733. DSS0\_VID\_FIR\_COEF\_H12\_C\_9 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-812. DSS0\_VID\_FIR\_COEF\_H12\_C\_9 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase 9
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase 9
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.72 DSS0\_VID\_FIR\_COEF\_H12\_C\_10 Register (Offset = 11Ch) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H12\_C\_10 is shown in [Figure 12-734](#) and described in [Table 12-814](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-813. DSS0\_VID\_FIR\_COEF\_H12\_C\_10 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 011Ch
DSS0_VID2	04A6 011Ch

**Figure 12-734. DSS0\_VID\_FIR\_COEF\_H12\_C\_10 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-814. DSS0\_VID\_FIR\_COEF\_H12\_C\_10 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase 10
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase 10
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.73 DSS0\_VID\_FIR\_COEF\_H12\_C\_11 Register (Offset = 120h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H12\_C\_11 is shown in [Figure 12-735](#) and described in [Table 12-816](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-815. DSS0\_VID\_FIR\_COEF\_H12\_C\_11 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0120h
DSS0_VID2	04A6 0120h

**Figure 12-735. DSS0\_VID\_FIR\_COEF\_H12\_C\_11 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-816. DSS0\_VID\_FIR\_COEF\_H12\_C\_11 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase 11
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase 11
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.74 DSS0\_VID\_FIR\_COEF\_H12\_C\_12 Register (Offset = 124h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H12\_C\_12 is shown in [Figure 12-736](#) and described in [Table 12-818](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-817. DSS0\_VID\_FIR\_COEF\_H12\_C\_12 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0124h
DSS0_VID2	04A6 0124h

**Figure 12-736. DSS0\_VID\_FIR\_COEF\_H12\_C\_12 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-818. DSS0\_VID\_FIR\_COEF\_H12\_C\_12 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase 12
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase 12
9-0	RESERVED	R	0h	Reserved

#### 12.6.4.11.13.2.75 DSS0\_VID\_FIR\_COEF\_H12\_C\_13 Register (Offset = 128h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H12\_C\_13 is shown in [Figure 12-737](#) and described in [Table 12-820](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-819. DSS0\_VID\_FIR\_COEF\_H12\_C\_13 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0128h
DSS0_VID2	04A6 0128h

**Figure 12-737. DSS0\_VID\_FIR\_COEF\_H12\_C\_13 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-820. DSS0\_VID\_FIR\_COEF\_H12\_C\_13 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase 13
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase 13
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.76 DSS0\_VID\_FIR\_COEF\_H12\_C\_14 Register (Offset = 12Ch) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H12\_C\_14 is shown in [Figure 12-738](#) and described in [Table 12-822](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-821. DSS0\_VID\_FIR\_COEF\_H12\_C\_14 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 012Ch
DSS0_VID2	04A6 012Ch

**Figure 12-738. DSS0\_VID\_FIR\_COEF\_H12\_C\_14 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-822. DSS0\_VID\_FIR\_COEF\_H12\_C\_14 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase 14
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase 14
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.77 DSS0\_VID\_FIR\_COEF\_H12\_C\_15 Register (Offset = 130h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_H12\_C\_15 is shown in [Figure 12-739](#) and described in [Table 12-824](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-823. DSS0\_VID\_FIR\_COEF\_H12\_C\_15  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0130h
DSS0_VID2	04A6 0130h

**Figure 12-739. DSS0\_VID\_FIR\_COEF\_H12\_C\_15 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-824. DSS0\_VID\_FIR\_COEF\_H12\_C\_15 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase 15
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase 15
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.78 DSS0\_VID\_FIR\_COEF\_V0\_0 Register (Offset = 134h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V0\_0 is shown in [Figure 12-740](#) and described in [Table 12-826](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-825. DSS0\_VID\_FIR\_COEF\_V0\_0 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0134h
DSS0_VID2	04A6 0134h

**Figure 12-740. DSS0\_VID\_FIR\_COEF\_V0\_0 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-826. DSS0\_VID\_FIR\_COEF\_V0\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase 0



### 12.6.4.11.13.2.79 DSS0\_VID\_FIR\_COEF\_V0\_1 Register (Offset = 138h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V0\_1 is shown in [Figure 12-741](#) and described in [Table 12-828](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-827. DSS0\_VID\_FIR\_COEF\_V0\_1 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0138h
DSS0_VID2	04A6 0138h

**Figure 12-741. DSS0\_VID\_FIR\_COEF\_V0\_1 Register**

31	30	29	28	27	26	25	24
RESERVED		RESERVED					
R-0h		R-0h					
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-828. DSS0\_VID\_FIR\_COEF\_V0\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase 1

### 12.6.4.11.13.2.80 DSS0\_VID\_FIR\_COEF\_V0\_2 Register (Offset = 13Ch) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V0\_2 is shown in [Figure 12-742](#) and described in [Table 12-830](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-829. DSS0\_VID\_FIR\_COEF\_V0\_2 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 013Ch
DSS0_VID2	04A6 013Ch

**Figure 12-742. DSS0\_VID\_FIR\_COEF\_V0\_2 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-830. DSS0\_VID\_FIR\_COEF\_V0\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase 2

#### 12.6.4.11.13.2.81 DSS0\_VID\_FIR\_COEF\_V0\_3 Register (Offset = 140h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V0\_3 is shown in [Figure 12-743](#) and described in [Table 12-832](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-831. DSS0\_VID\_FIR\_COEF\_V0\_3 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0140h
DSS0_VID2	04A6 0140h

**Figure 12-743. DSS0\_VID\_FIR\_COEF\_V0\_3 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-832. DSS0\_VID\_FIR\_COEF\_V0\_3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase 3

### 12.6.4.11.13.2.82 DSS0\_VID\_FIR\_COEF\_V0\_4 Register (Offset = 144h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V0\_4 is shown in [Figure 12-744](#) and described in [Table 12-834](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-833. DSS0\_VID\_FIR\_COEF\_V0\_4 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0144h
DSS0_VID2	04A6 0144h

**Figure 12-744. DSS0\_VID\_FIR\_COEF\_V0\_4 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-834. DSS0\_VID\_FIR\_COEF\_V0\_4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase 4

### 12.6.4.11.13.2.83 DSS0\_VID\_FIR\_COEF\_V0\_5 Register (Offset = 148h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V0\_5 is shown in [Figure 12-745](#) and described in [Table 12-836](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-835. DSS0\_VID\_FIR\_COEF\_V0\_5 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0148h
DSS0_VID2	04A6 0148h

**Figure 12-745. DSS0\_VID\_FIR\_COEF\_V0\_5 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-836. DSS0\_VID\_FIR\_COEF\_V0\_5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase 5

#### 12.6.4.11.13.2.84 DSS0\_VID\_FIR\_COEF\_V0\_6 Register (Offset = 14Ch) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V0\_6 is shown in [Figure 12-746](#) and described in [Table 12-838](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-837. DSS0\_VID\_FIR\_COEF\_V0\_6 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 014Ch
DSS0_VID2	04A6 014Ch

**Figure 12-746. DSS0\_VID\_FIR\_COEF\_V0\_6 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-838. DSS0\_VID\_FIR\_COEF\_V0\_6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase 6

#### 12.6.4.11.13.2.85 DSS0\_VID\_FIR\_COEF\_V0\_7 Register (Offset = 150h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V0\_7 is shown in [Figure 12-747](#) and described in [Table 12-840](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-839. DSS0\_VID\_FIR\_COEF\_V0\_7 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0150h
DSS0_VID2	04A6 0150h

**Figure 12-747. DSS0\_VID\_FIR\_COEF\_V0\_7 Register**

31	30	29	28	27	26	25	24
RESERVED		RESERVED					
R-0h		R-0h					
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-840. DSS0\_VID\_FIR\_COEF\_V0\_7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase 7

### 12.6.4.11.13.2.86 DSS0\_VID\_FIR\_COEF\_V0\_8 Register (Offset = 154h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V0\_8 is shown in [Figure 12-748](#) and described in [Table 12-842](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-841. DSS0\_VID\_FIR\_COEF\_V0\_8 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0154h
DSS0_VID2	04A6 0154h

**Figure 12-748. DSS0\_VID\_FIR\_COEF\_V0\_8 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-842. DSS0\_VID\_FIR\_COEF\_V0\_8 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase 8



#### 12.6.4.11.13.2.87 DSS0\_VID\_FIR\_COEF\_V0\_C\_0 Register (Offset = 158h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V0\_C\_0 is shown in [Figure 12-749](#) and described in [Table 12-844](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-843. DSS0\_VID\_FIR\_COEF\_V0\_C\_0  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0158h
DSS0_VID2	04A6 0158h

**Figure 12-749. DSS0\_VID\_FIR\_COEF\_V0\_C\_0 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-844. DSS0\_VID\_FIR\_COEF\_V0\_C\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase 0

### 12.6.4.11.13.2.88 DSS0\_VID\_FIR\_COEF\_V0\_C\_1 Register (Offset = 15Ch) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V0\_C\_1 is shown in [Figure 12-750](#) and described in [Table 12-846](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-845. DSS0\_VID\_FIR\_COEF\_V0\_C\_1 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 015Ch
DSS0_VID2	04A6 015Ch

**Figure 12-750. DSS0\_VID\_FIR\_COEF\_V0\_C\_1 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-846. DSS0\_VID\_FIR\_COEF\_V0\_C\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase 1

### 12.6.4.11.13.2.89 DSS0\_VID\_FIR\_COEF\_V0\_C\_2 Register (Offset = 160h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V0\_C\_2 is shown in [Figure 12-751](#) and described in [Table 12-848](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-847. DSS0\_VID\_FIR\_COEF\_V0\_C\_2  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0160h
DSS0_VID2	04A6 0160h

**Figure 12-751. DSS0\_VID\_FIR\_COEF\_V0\_C\_2 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-848. DSS0\_VID\_FIR\_COEF\_V0\_C\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase 2

### 12.6.4.11.13.2.90 DSS0\_VID\_FIR\_COEF\_V0\_C\_3 Register (Offset = 164h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V0\_C\_3 is shown in [Figure 12-752](#) and described in [Table 12-850](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-849. DSS0\_VID\_FIR\_COEF\_V0\_C\_3  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0164h
DSS0_VID2	04A6 0164h

**Figure 12-752. DSS0\_VID\_FIR\_COEF\_V0\_C\_3 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-850. DSS0\_VID\_FIR\_COEF\_V0\_C\_3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase 3

#### 12.6.4.11.13.2.91 DSS0\_VID\_FIR\_COEF\_V0\_C\_4 Register (Offset = 168h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V0\_C\_4 is shown in [Figure 12-753](#) and described in [Table 12-852](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-851. DSS0\_VID\_FIR\_COEF\_V0\_C\_4  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0168h
DSS0_VID2	04A6 0168h

**Figure 12-753. DSS0\_VID\_FIR\_COEF\_V0\_C\_4 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-852. DSS0\_VID\_FIR\_COEF\_V0\_C\_4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase 4

### 12.6.4.11.13.2.92 DSS0\_VID\_FIR\_COEF\_V0\_C\_5 Register (Offset = 16Ch) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V0\_C\_5 is shown in [Figure 12-754](#) and described in [Table 12-854](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-853. DSS0\_VID\_FIR\_COEF\_V0\_C\_5  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 016Ch
DSS0_VID2	04A6 016Ch

**Figure 12-754. DSS0\_VID\_FIR\_COEF\_V0\_C\_5 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-854. DSS0\_VID\_FIR\_COEF\_V0\_C\_5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase 5

#### 12.6.4.11.13.2.93 DSS0\_VID\_FIR\_COEF\_V0\_C\_6 Register (Offset = 170h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V0\_C\_6 is shown in [Figure 12-755](#) and described in [Table 12-856](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-855. DSS0\_VID\_FIR\_COEF\_V0\_C\_6  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0170h
DSS0_VID2	04A6 0170h

**Figure 12-755. DSS0\_VID\_FIR\_COEF\_V0\_C\_6 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-856. DSS0\_VID\_FIR\_COEF\_V0\_C\_6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase 6

### 12.6.4.11.13.2.94 DSS0\_VID\_FIR\_COEF\_V0\_C\_7 Register (Offset = 174h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V0\_C\_7 is shown in [Figure 12-756](#) and described in [Table 12-858](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-857. DSS0\_VID\_FIR\_COEF\_V0\_C\_7  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0174h
DSS0_VID2	04A6 0174h

**Figure 12-756. DSS0\_VID\_FIR\_COEF\_V0\_C\_7 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-858. DSS0\_VID\_FIR\_COEF\_V0\_C\_7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase 7



#### 12.6.4.11.13.2.95 DSS0\_VID\_FIR\_COEF\_V0\_C\_8 Register (Offset = 178h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V0\_C\_8 is shown in [Figure 12-757](#) and described in [Table 12-860](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-859. DSS0\_VID\_FIR\_COEF\_V0\_C\_8 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0178h
DSS0_VID2	04A6 0178h

**Figure 12-757. DSS0\_VID\_FIR\_COEF\_V0\_C\_8 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-860. DSS0\_VID\_FIR\_COEF\_V0\_C\_8 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase 8

### 12.6.4.11.13.2.96 DSS0\_VID\_FIR\_COEF\_V12\_0 Register (Offset = 17Ch) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V12\_0 is shown in [Figure 12-758](#) and described in [Table 12-862](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-861. DSS0\_VID\_FIR\_COEF\_V12\_0  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 017Ch
DSS0_VID2	04A6 017Ch

**Figure 12-758. DSS0\_VID\_FIR\_COEF\_V12\_0 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-862. DSS0\_VID\_FIR\_COEF\_V12\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase 0
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase 0
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.97 DSS0\_VID\_FIR\_COEF\_V12\_1 Register (Offset = 180h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V12\_1 is shown in [Figure 12-759](#) and described in [Table 12-864](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-863. DSS0\_VID\_FIR\_COEF\_V12\_1  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0180h
DSS0_VID2	04A6 0180h

**Figure 12-759. DSS0\_VID\_FIR\_COEF\_V12\_1 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-864. DSS0\_VID\_FIR\_COEF\_V12\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase 1
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase 1
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.98 DSS0\_VID\_FIR\_COEF\_V12\_2 Register (Offset = 184h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V12\_2 is shown in [Figure 12-760](#) and described in [Table 12-866](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-865. DSS0\_VID\_FIR\_COEF\_V12\_2  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0184h
DSS0_VID2	04A6 0184h

**Figure 12-760. DSS0\_VID\_FIR\_COEF\_V12\_2 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-866. DSS0\_VID\_FIR\_COEF\_V12\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase 2
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase 2
9-0	RESERVED	R	0h	Reserved

#### 12.6.4.11.13.2.99 DSS0\_VID\_FIR\_COEF\_V12\_3 Register (Offset = 188h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V12\_3 is shown in [Figure 12-761](#) and described in [Table 12-868](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-867. DSS0\_VID\_FIR\_COEF\_V12\_3  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0188h
DSS0_VID2	04A6 0188h

**Figure 12-761. DSS0\_VID\_FIR\_COEF\_V12\_3 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-868. DSS0\_VID\_FIR\_COEF\_V12\_3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase 3
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase 3
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.100 DSS0\_VID\_FIR\_COEF\_V12\_4 Register (Offset = 18Ch) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V12\_4 is shown in [Figure 12-762](#) and described in [Table 12-870](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-869. DSS0\_VID\_FIR\_COEF\_V12\_4  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 018Ch
DSS0_VID2	04A6 018Ch

**Figure 12-762. DSS0\_VID\_FIR\_COEF\_V12\_4 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-870. DSS0\_VID\_FIR\_COEF\_V12\_4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase 4
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase 4
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.101 DSS0\_VID\_FIR\_COEF\_V12\_5 Register (Offset = 190h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V12\_5 is shown in [Figure 12-763](#) and described in [Table 12-872](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-871. DSS0\_VID\_FIR\_COEF\_V12\_5  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0190h
DSS0_VID2	04A6 0190h

**Figure 12-763. DSS0\_VID\_FIR\_COEF\_V12\_5 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-872. DSS0\_VID\_FIR\_COEF\_V12\_5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase 5
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase 5
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.102 DSS0\_VID\_FIR\_COEF\_V12\_6 Register (Offset = 194h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V12\_6 is shown in [Figure 12-764](#) and described in [Table 12-874](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-873. DSS0\_VID\_FIR\_COEF\_V12\_6  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0194h
DSS0_VID2	04A6 0194h

**Figure 12-764. DSS0\_VID\_FIR\_COEF\_V12\_6 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-874. DSS0\_VID\_FIR\_COEF\_V12\_6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase 6
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase 6
9-0	RESERVED	R	0h	Reserved



### 12.6.4.11.13.2.103 DSS0\_VID\_FIR\_COEF\_V12\_7 Register (Offset = 198h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V12\_7 is shown in [Figure 12-765](#) and described in [Table 12-876](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-875. DSS0\_VID\_FIR\_COEF\_V12\_7  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 0198h
DSS0_VID2	04A6 0198h

**Figure 12-765. DSS0\_VID\_FIR\_COEF\_V12\_7 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-876. DSS0\_VID\_FIR\_COEF\_V12\_7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase 7
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase 7
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.104 DSS0\_VID\_FIR\_COEF\_V12\_8 Register (Offset = 19Ch) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V12\_8 is shown in [Figure 12-766](#) and described in [Table 12-878](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-877. DSS0\_VID\_FIR\_COEF\_V12\_8  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 019Ch
DSS0_VID2	04A6 019Ch

**Figure 12-766. DSS0\_VID\_FIR\_COEF\_V12\_8 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-878. DSS0\_VID\_FIR\_COEF\_V12\_8 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase 8
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase 8
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.105 DSS0\_VID\_FIR\_COEF\_V12\_9 Register (Offset = 1A0h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V12\_9 is shown in [Figure 12-767](#) and described in [Table 12-880](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-879. DSS0\_VID\_FIR\_COEF\_V12\_9  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 01A0h
DSS0_VID2	04A6 01A0h

**Figure 12-767. DSS0\_VID\_FIR\_COEF\_V12\_9 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-880. DSS0\_VID\_FIR\_COEF\_V12\_9 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase 9
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase 9
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.106 DSS0\_VID\_FIR\_COEF\_V12\_10 Register (Offset = 1A4h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V12\_10 is shown in [Figure 12-768](#) and described in [Table 12-882](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-881. DSS0\_VID\_FIR\_COEF\_V12\_10  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 01A4h
DSS0_VID2	04A6 01A4h

**Figure 12-768. DSS0\_VID\_FIR\_COEF\_V12\_10 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-882. DSS0\_VID\_FIR\_COEF\_V12\_10 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase 10
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase 10
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.107 DSS0\_VID\_FIR\_COEF\_V12\_11 Register (Offset = 1A8h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V12\_11 is shown in [Figure 12-769](#) and described in [Table 12-884](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-883. DSS0\_VID\_FIR\_COEF\_V12\_11  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 01A8h
DSS0_VID2	04A6 01A8h

**Figure 12-769. DSS0\_VID\_FIR\_COEF\_V12\_11 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-884. DSS0\_VID\_FIR\_COEF\_V12\_11 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase 11
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase 11
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.108 DSS0\_VID\_FIR\_COEF\_V12\_12 Register (Offset = 1ACh) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V12\_12 is shown in [Figure 12-770](#) and described in [Table 12-886](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-885. DSS0\_VID\_FIR\_COEF\_V12\_12  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 01ACh
DSS0_VID2	04A6 01ACh

**Figure 12-770. DSS0\_VID\_FIR\_COEF\_V12\_12 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-886. DSS0\_VID\_FIR\_COEF\_V12\_12 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase 12
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase 12
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.109 DSS0\_VID\_FIR\_COEF\_V12\_13 Register (Offset = 1B0h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V12\_13 is shown in [Figure 12-771](#) and described in [Table 12-888](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-887. DSS0\_VID\_FIR\_COEF\_V12\_13  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 01B0h
DSS0_VID2	04A6 01B0h

**Figure 12-771. DSS0\_VID\_FIR\_COEF\_V12\_13 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-888. DSS0\_VID\_FIR\_COEF\_V12\_13 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase 13
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase 13
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.110 DSS0\_VID\_FIR\_COEF\_V12\_14 Register (Offset = 1B4h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V12\_14 is shown in [Figure 12-772](#) and described in [Table 12-890](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-889. DSS0\_VID\_FIR\_COEF\_V12\_14  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 01B4h
DSS0_VID2	04A6 01B4h

**Figure 12-772. DSS0\_VID\_FIR\_COEF\_V12\_14 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-890. DSS0\_VID\_FIR\_COEF\_V12\_14 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase 14
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase 14
9-0	RESERVED	R	0h	Reserved



### 12.6.4.11.13.2.111 DSS0\_VID\_FIR\_COEF\_V12\_15 Register (Offset = 1B8h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V12\_15 is shown in [Figure 12-773](#) and described in [Table 12-892](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the 16 phases. It is used for ARGB and Y setting. Shadow register

**Table 12-891. DSS0\_VID\_FIR\_COEF\_V12\_15  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 01B8h
DSS0_VID2	04A6 01B8h

**Figure 12-773. DSS0\_VID\_FIR\_COEF\_V12\_15 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-892. DSS0\_VID\_FIR\_COEF\_V12\_15 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase 15
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase 15
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.112 DSS0\_VID\_FIR\_COEF\_V12\_C\_0 Register (Offset = 1BCh) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V12\_C\_0 is shown in [Figure 12-774](#) and described in [Table 12-894](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-893. DSS0\_VID\_FIR\_COEF\_V12\_C\_0  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 01BCh
DSS0_VID2	04A6 01BCh

**Figure 12-774. DSS0\_VID\_FIR\_COEF\_V12\_C\_0 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-894. DSS0\_VID\_FIR\_COEF\_V12\_C\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase 0
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase 0
9-0	RESERVED	R	0h	Reserved

#### 12.6.4.11.13.2.113 DSS0\_VID\_FIR\_COEF\_V12\_C\_1 Register (Offset = 1C0h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V12\_C\_1 is shown in [Figure 12-775](#) and described in [Table 12-896](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-895. DSS0\_VID\_FIR\_COEF\_V12\_C\_1  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 01C0h
DSS0_VID2	04A6 01C0h

**Figure 12-775. DSS0\_VID\_FIR\_COEF\_V12\_C\_1 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-896. DSS0\_VID\_FIR\_COEF\_V12\_C\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase 1
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase 1
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.114 DSS0\_VID\_FIR\_COEF\_V12\_C\_2 Register (Offset = 1C4h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V12\_C\_2 is shown in [Figure 12-776](#) and described in [Table 12-898](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-897. DSS0\_VID\_FIR\_COEF\_V12\_C\_2  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 01C4h
DSS0_VID2	04A6 01C4h

**Figure 12-776. DSS0\_VID\_FIR\_COEF\_V12\_C\_2 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-898. DSS0\_VID\_FIR\_COEF\_V12\_C\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase 2
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase 2
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.115 DSS0\_VID\_FIR\_COEF\_V12\_C\_3 Register (Offset = 1C8h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V12\_C\_3 is shown in [Figure 12-777](#) and described in [Table 12-900](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-899. DSS0\_VID\_FIR\_COEF\_V12\_C\_3  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 01C8h
DSS0_VID2	04A6 01C8h

**Figure 12-777. DSS0\_VID\_FIR\_COEF\_V12\_C\_3 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-900. DSS0\_VID\_FIR\_COEF\_V12\_C\_3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase 3
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase 3
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.116 DSS0\_VID\_FIR\_COEF\_V12\_C\_4 Register (Offset = 1CCh) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V12\_C\_4 is shown in [Figure 12-778](#) and described in [Table 12-902](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-901. DSS0\_VID\_FIR\_COEF\_V12\_C\_4 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 01CCh
DSS0_VID2	04A6 01CCh

**Figure 12-778. DSS0\_VID\_FIR\_COEF\_V12\_C\_4 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-902. DSS0\_VID\_FIR\_COEF\_V12\_C\_4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase 4
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase 4
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.117 DSS0\_VID\_FIR\_COEF\_V12\_C\_5 Register (Offset = 1D0h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V12\_C\_5 is shown in [Figure 12-779](#) and described in [Table 12-904](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-903. DSS0\_VID\_FIR\_COEF\_V12\_C\_5  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 01D0h
DSS0_VID2	04A6 01D0h

**Figure 12-779. DSS0\_VID\_FIR\_COEF\_V12\_C\_5 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-904. DSS0\_VID\_FIR\_COEF\_V12\_C\_5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase 5
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase 5
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.118 DSS0\_VID\_FIR\_COEF\_V12\_C\_6 Register (Offset = 1D4h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V12\_C\_6 is shown in [Figure 12-780](#) and described in [Table 12-906](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-905. DSS0\_VID\_FIR\_COEF\_V12\_C\_6  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 01D4h
DSS0_VID2	04A6 01D4h

**Figure 12-780. DSS0\_VID\_FIR\_COEF\_V12\_C\_6 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-906. DSS0\_VID\_FIR\_COEF\_V12\_C\_6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase 6
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase 6
9-0	RESERVED	R	0h	Reserved



### 12.6.4.11.13.2.119 DSS0\_VID\_FIR\_COEF\_V12\_C\_7 Register (Offset = 1D8h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V12\_C\_7 is shown in [Figure 12-781](#) and described in [Table 12-908](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-907. DSS0\_VID\_FIR\_COEF\_V12\_C\_7  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 01D8h
DSS0_VID2	04A6 01D8h

**Figure 12-781. DSS0\_VID\_FIR\_COEF\_V12\_C\_7 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-908. DSS0\_VID\_FIR\_COEF\_V12\_C\_7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase 7
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase 7
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.120 DSS0\_VID\_FIR\_COEF\_V12\_C\_8 Register (Offset = 1DCh) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V12\_C\_8 is shown in [Figure 12-782](#) and described in [Table 12-910](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-909. DSS0\_VID\_FIR\_COEF\_V12\_C\_8  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 01DCh
DSS0_VID2	04A6 01DCh

**Figure 12-782. DSS0\_VID\_FIR\_COEF\_V12\_C\_8 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-910. DSS0\_VID\_FIR\_COEF\_V12\_C\_8 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase 8
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase 8
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.121 DSS0\_VID\_FIR\_COEF\_V12\_C\_9 Register (Offset = 1E0h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V12\_C\_9 is shown in [Figure 12-783](#) and described in [Table 12-912](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-911. DSS0\_VID\_FIR\_COEF\_V12\_C\_9  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 01E0h
DSS0_VID2	04A6 01E0h

**Figure 12-783. DSS0\_VID\_FIR\_COEF\_V12\_C\_9 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-912. DSS0\_VID\_FIR\_COEF\_V12\_C\_9 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase 9
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase 9
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.122 DSS0\_VID\_FIR\_COEF\_V12\_C\_10 Register (Offset = 1E4h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V12\_C\_10 is shown in [Figure 12-784](#) and described in [Table 12-914](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-913. DSS0\_VID\_FIR\_COEF\_V12\_C\_10 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 01E4h
DSS0_VID2	04A6 01E4h

**Figure 12-784. DSS0\_VID\_FIR\_COEF\_V12\_C\_10 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-914. DSS0\_VID\_FIR\_COEF\_V12\_C\_10 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase 10
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase 10
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.123 DSS0\_VID\_FIR\_COEF\_V12\_C\_11 Register (Offset = 1E8h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V12\_C\_11 is shown in [Figure 12-785](#) and described in [Table 12-916](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-915. DSS0\_VID\_FIR\_COEF\_V12\_C\_11 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 01E8h
DSS0_VID2	04A6 01E8h

**Figure 12-785. DSS0\_VID\_FIR\_COEF\_V12\_C\_11 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-916. DSS0\_VID\_FIR\_COEF\_V12\_C\_11 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase 11
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase 11
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.124 DSS0\_VID\_FIR\_COEF\_V12\_C\_12 Register (Offset = 1ECh) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V12\_C\_12 is shown in [Figure 12-786](#) and described in [Table 12-918](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-917. DSS0\_VID\_FIR\_COEF\_V12\_C\_12 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 01ECh
DSS0_VID2	04A6 01ECh

**Figure 12-786. DSS0\_VID\_FIR\_COEF\_V12\_C\_12 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-918. DSS0\_VID\_FIR\_COEF\_V12\_C\_12 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase 12
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase 12
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.125 DSS0\_VID\_FIR\_COEF\_V12\_C\_13 Register (Offset = 1F0h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V12\_C\_13 is shown in [Figure 12-787](#) and described in [Table 12-920](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-919. DSS0\_VID\_FIR\_COEF\_V12\_C\_13 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 01F0h
DSS0_VID2	04A6 01F0h

**Figure 12-787. DSS0\_VID\_FIR\_COEF\_V12\_C\_13 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-920. DSS0\_VID\_FIR\_COEF\_V12\_C\_13 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase 13
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase 13
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.126 DSS0\_VID\_FIR\_COEF\_V12\_C\_14 Register (Offset = 1F4h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V12\_C\_14 is shown in [Figure 12-788](#) and described in [Table 12-922](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-921. DSS0\_VID\_FIR\_COEF\_V12\_C\_14 Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 01F4h
DSS0_VID2	04A6 01F4h

**Figure 12-788. DSS0\_VID\_FIR\_COEF\_V12\_C\_14 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-922. DSS0\_VID\_FIR\_COEF\_V12\_C\_14 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase 14
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase 14
9-0	RESERVED	R	0h	Reserved



### 12.6.4.11.13.2.127 DSS0\_VID\_FIR\_COEF\_V12\_C\_15 Register (Offset = 1F8h) [reset = 0h]

DSS0\_VID\_FIR\_COEF\_V12\_C\_15 is shown in [Figure 12-789](#) and described in [Table 12-924](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the vertical resize of the video picture associated with the video window for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-923. DSS0\_VID\_FIR\_COEF\_V12\_C\_15  
Instances**

Instance	Physical Address
DSS0_VIDL1	N/A
DSS0_VIDL2	N/A
DSS0_VID1	04A5 01F8h
DSS0_VID2	04A6 01F8h

**Figure 12-789. DSS0\_VID\_FIR\_COEF\_V12\_C\_15 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-924. DSS0\_VID\_FIR\_COEF\_V12\_C\_15 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase 15
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase 15
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.2.128 DSS0\_VID\_GLOBAL\_ALPHA Register (Offset = 1FCh) [reset = FFh]

DSS0\_VID\_GLOBAL\_ALPHA is shown in [Figure 12-790](#) and described in [Table 12-926](#).

Return to [Summary Table](#).

The register defines the global alpha value for the video pipeline. Shadow register

**Table 12-925. DSS0\_VID\_GLOBAL\_ALPHA Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 01FCh
DSS0_VIDL2	04A3 01FCh
DSS0_VID1	04A5 01FCh
DSS0_VID2	04A6 01FCh

**Figure 12-790. DSS0\_VID\_GLOBAL\_ALPHA Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								GLOBALALPHA							
R-0h								R/W-FFh							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-926. DSS0\_VID\_GLOBAL\_ALPHA Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-0	GLOBALALPHA	R/W	FFh	Global alpha value from 0 to 255. 0 corresponds to fully transparent and 255 corresponds to fully opaque

### 12.6.4.11.13.2.129 DSS0\_VID\_MFLAG\_THRESHOLD Register (Offset = 208h) [reset = 0h]

DSS0\_VID\_MFLAG\_THRESHOLD is shown in [Figure 12-791](#) and described in [Table 12-928](#).

Return to [Summary Table](#).

**Table 12-927. DSS0\_VID\_MFLAG\_THRESHOLD Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0208h
DSS0_VIDL2	04A3 0208h
DSS0_VID1	04A5 0208h
DSS0_VID2	04A6 0208h

**Figure 12-791. DSS0\_VID\_MFLAG\_THRESHOLD Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HT_MFLAG																LT_MFLAG															
R/W-0h																R/W-0h															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-928. DSS0\_VID\_MFLAG\_THRESHOLD Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	HT_MFLAG	R/W	0h	MFlag High Threshold
15-0	LT_MFLAG	R/W	0h	MFlag Low Threshold

### 12.6.4.11.13.2.130 DSS0\_VID\_PICTURE\_SIZE Register (Offset = 20Ch) [reset = X]

DSS0\_VID\_PICTURE\_SIZE is shown in [Figure 12-792](#) and described in [Table 12-930](#).

Return to [Summary Table](#).

The register configures the DSS0\_VID\_SIZE of the video picture associated with the video layer before up/down-scaling. Shadow register

**Table 12-929. DSS0\_VID\_PICTURE\_SIZE Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 020Ch
DSS0_VIDL2	04A3 020Ch
DSS0_VID1	04A5 020Ch
DSS0_VID2	04A6 020Ch

**Figure 12-792. DSS0\_VID\_PICTURE\_SIZE Register**

31	30	29	28	27	26	25	24
RESERVED				MEMSIZEY			
R/W-X				R/W-0h			
23	22	21	20	19	18	17	16
MEMSIZEY							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED				MEMSIZEX			
R/W-X				R/W-0h			
7	6	5	4	3	2	1	0
MEMSIZEX							
R/W-0h							

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-930. DSS0\_VID\_PICTURE\_SIZE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	X	
29-16	MEMSIZEY	R/W	0h	Number of lines of the video picture Encoded value [from 1 to 16384] to specify the number of lines of the video picture in memory [program to value minus one] When predecimation is set, the value represents the DSS0_VID_SIZE of the image after predecimation but the max DSS0_VID_SIZE of the unpredecimated image DSS0_VID_SIZE in memory is still bounded to 2exp[11]
15-14	RESERVED	R/W	X	

**Table 12-930. DSS0\_VID\_PICTURE\_SIZE Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13-0	MEMSIZE_X	R/W	0h	<p>Number of pixels of the video picture Encoded value [from 1 to 16384] to specify the number of pixels of the video picture in memory [program to value minus one].</p> <p>The DSS0_VID_SIZE is limited to the DSS0_VID_SIZE of the line buffer of the vertical sampling block in case the video picture is processed by the vertical filtering unit [program to value minus one].</p> <p>When predecimation is set, the value represents the DSS0_VID_SIZE of the image after predecimation but the max DSS0_VID_SIZE of the unpredecimated image DSS0_VID_SIZE in memory is still bounded to <math>2^{\text{exp}[11]}</math></p>

### 12.6.4.11.13.2.131 DSS0\_VID\_PIXEL\_INC Register (Offset = 210h) [reset = 1h]

DSS0\_VID\_PIXEL\_INC is shown in [Figure 12-793](#) and described in [Table 12-932](#).

Return to [Summary Table](#).

The register configures the number of bytes to increment between two pixels for the buffer associated with the video window. Shadow register

**Table 12-931. DSS0\_VID\_PIXEL\_INC Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0210h
DSS0_VIDL2	04A3 0210h
DSS0_VID1	04A5 0210h
DSS0_VID2	04A6 0210h

**Figure 12-793. DSS0\_VID\_PIXEL\_INC Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																PIXELINC															
R-0h																R/W-1h															

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-932. DSS0\_VID\_PIXEL\_INC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
7-0	PIXELINC	R/W	1h	Number of bytes to increment between two pixels Encoded unsigned value [from 1 to 255] to specify the number of bytes between two pixels in the video buffer. The value 0 is invalid The value 1 means next pixel The value 1+n*bpp means increment of n pixels For YUV420. Max supported value is 128

### 12.6.4.11.13.2.132 DSS0\_VID\_PRELOAD Register (Offset = 218h) [reset = 100h]

DSS0\_VID\_PRELOAD is shown in [Figure 12-794](#) and described in [Table 12-934](#).

Return to [Summary Table](#).

The register configures the DMA buffer of the video pipeline. Shadow register

**Table 12-933. DSS0\_VID\_PRELOAD Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0218h
DSS0_VIDL2	04A3 0218h
DSS0_VID1	04A5 0218h
DSS0_VID2	04A6 0218h

**Figure 12-794. DSS0\_VID\_PRELOAD Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
RESERVED																				PRELOAD															
R-0h																				R/W-100h															

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-934. DSS0\_VID\_PRELOAD Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
11-0	PRELOAD	R/W	100h	DMA buffer DSS0_VID_PRELOAD value Number of 128-bit words defining the DSS0_VID_PRELOAD value

### 12.6.4.11.13.2.133 DSS0\_VID\_ROW\_INC Register (Offset = 21Ch) [reset = 1h]

DSS0\_VID\_ROW\_INC is shown in [Figure 12-795](#) and described in [Table 12-936](#).

Return to [Summary Table](#).

The register configures the number of bytes to increment at the end of the row for the buffer associated with the video window. For YUV420 formats this corresponds to the Y Buffer. Shadow register

**Table 12-935. DSS0\_VID\_ROW\_INC Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 021Ch
DSS0_VIDL2	04A3 021Ch
DSS0_VID1	04A5 021Ch
DSS0_VID2	04A6 021Ch

**Figure 12-795. DSS0\_VID\_ROW\_INC Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ROWINC																															
R/W-1h																															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-936. DSS0\_VID\_ROW\_INC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	ROWINC	R/W	1h	Number of bytes to increment at the end of the row Encoded signed value [from $-2^{31}-1$ to $2^{31}$ ] to specify the number of bytes to increment at the end of the row in the video buffer. The value 0 is invalid. The value 1 means next pixel. The value $1+n*bpp$ means increment of n pixels The value $1-[n+1]*bpp$ means decrement of n pixels



#### 12.6.4.11.13.2.134 DSS0\_VID\_SIZE Register (Offset = 220h) [reset = X]

DSS0\_VID\_SIZE is shown in [Figure 12-796](#) and described in [Table 12-938](#).

Return to [Summary Table](#).

The register configures the DSS0\_VID\_SIZE of the video window. Shadow register

**Table 12-937. DSS0\_VID\_SIZE Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0220h
DSS0_VIDL2	04A3 0220h
DSS0_VID1	04A5 0220h
DSS0_VID2	04A6 0220h

**Figure 12-796. DSS0\_VID\_SIZE Register**

31	30	29	28	27	26	25	24
RESERVED				SIZEY			
R/W-X				R/W-0h			
23	22	21	20	19	18	17	16
SIZEY							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED				SIZEX			
R/W-X				R/W-0h			
7	6	5	4	3	2	1	0
SIZEX							
R/W-0h							

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-938. DSS0\_VID\_SIZE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	X	
29-16	SIZEY	R/W	0h	Number of lines of the video window Encoded value [from 1 to 16384] to specify the number of lines of the video window [program DSS0_VID_SIZE -1]
15-14	RESERVED	R/W	X	
13-0	SIZEX	R/W	0h	Number of pixels of the video window Encoded value [from 1 to 16384] to specify the number of pixels of the video window [program DSS0_VID_SIZE -1]

### 12.6.4.11.13.2.135 DSS0\_VID\_BA\_EXT\_0 Register (Offset = 22Ch) [reset = 0h]

DSS0\_VID\_BA\_EXT\_0 is shown in [Figure 12-797](#) and described in [Table 12-940](#).

Return to [Summary Table](#).

The register configures the 16-bit base address extension. It is the base-address of the single video buffer for single plane ARGB or YUV. For the Y buffer for two plane YUV. For the Alpha buffer for two plane RGB565-A8. 0 & 1 : For ping-pong mechanism with external trigger, based on the field polarity. Shadow register

**Table 12-939. DSS0\_VID\_BA\_EXT\_0 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 022Ch
DSS0_VIDL2	04A3 022Ch
DSS0_VID1	04A5 022Ch
DSS0_VID2	04A6 022Ch

**Figure 12-797. DSS0\_VID\_BA\_EXT\_0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																BA_EXT															
R-0h																R/W-0h															

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-940. DSS0\_VID\_BA\_EXT\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	BA_EXT	R/W	0h	Video base address extension [16 bits]. Addr extension to make the address space 48b wide

#### 12.6.4.11.13.2.136 DSS0\_VID\_BA\_EXT\_1 Register (Offset = 230h) [reset = 0h]

DSS0\_VID\_BA\_EXT\_1 is shown in [Figure 12-798](#) and described in [Table 12-942](#).

Return to [Summary Table](#).

The register configures the 16-bit base address extension. It is the base-address of the single video buffer for single plane ARGB or YUV. For the Y buffer for two plane YUV. For the Alpha buffer for two plane RGB565-A8. 0 & 1 : For ping-pong mechanism with external trigger, based on the field polarity. Shadow register

**Table 12-941. DSS0\_VID\_BA\_EXT\_1 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0230h
DSS0_VIDL2	04A3 0230h
DSS0_VID1	04A5 0230h
DSS0_VID2	04A6 0230h

**Figure 12-798. DSS0\_VID\_BA\_EXT\_1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																BA_EXT															
R-0h																R/W-0h															

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-942. DSS0\_VID\_BA\_EXT\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	BA_EXT	R/W	0h	Video base address extension [16 bits]. Addr extension to make the address space 48b wide

### 12.6.4.11.13.2.137 DSS0\_VID\_BA\_UV\_EXT\_0 Register (Offset = 234h) [reset = 0h]

DSS0\_VID\_BA\_UV\_EXT\_0 is shown in [Figure 12-799](#) and described in [Table 12-944](#).

Return to [Summary Table](#).

The register configures the 16-bit base address extension of the UV buffer for two plane YUV or the RGB buffer for two plane RGB565-A8. 0 & 1 : For ping-pong mechanism with external trigger, based on the field polarity. Shadow register

**Table 12-943. DSS0\_VID\_BA\_UV\_EXT\_0 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0234h
DSS0_VIDL2	04A3 0234h
DSS0_VID1	04A5 0234h
DSS0_VID2	04A6 0234h

**Figure 12-799. DSS0\_VID\_BA\_UV\_EXT\_0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																BA_UV_EXT															
R-0h																R/W-0h															

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-944. DSS0\_VID\_BA\_UV\_EXT\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	BA_UV_EXT	R/W	0h	Video base address extension [16 bits]. Addr extension to make the address space 48b wide

#### 12.6.4.11.13.2.138 DSS0\_VID\_BA\_UV\_EXT\_1 Register (Offset = 238h) [reset = 0h]

DSS0\_VID\_BA\_UV\_EXT\_1 is shown in [Figure 12-800](#) and described in [Table 12-946](#).

Return to [Summary Table](#).

The register configures the 16-bit base address extension of the UV buffer for two plane YUV or the RGB buffer for two plane RGB565-A8. 0 & 1 : For ping-pong mechanism with external trigger, based on the field polarity. Shadow register

**Table 12-945. DSS0\_VID\_BA\_UV\_EXT\_1 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0238h
DSS0_VIDL2	04A3 0238h
DSS0_VID1	04A5 0238h
DSS0_VID2	04A6 0238h

**Figure 12-800. DSS0\_VID\_BA\_UV\_EXT\_1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																BA_UV_EXT															
R-0h																R/W-0h															

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-946. DSS0\_VID\_BA\_UV\_EXT\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	BA_UV_EXT	R/W	0h	Video base address extension [16 bits]. Addr extension to make the address space 48b wide

### 12.6.4.11.13.2.139 DSS0\_VID\_CSC\_COEF7 Register (Offset = 23Ch) [reset = 0h]

DSS0\_VID\_CSC\_COEF7 is shown in [Figure 12-801](#) and described in [Table 12-948](#).

Return to [Summary Table](#).

The register configures the color space conversion matrix coefficients. Shadow register

**Table 12-947. DSS0\_VID\_CSC\_COEF7 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 023Ch
DSS0_VIDL2	04A3 023Ch
DSS0_VID1	04A5 023Ch
DSS0_VID2	04A6 023Ch

**Figure 12-801. DSS0\_VID\_CSC\_COEF7 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
POSTOFFSET3												RESERVED			
R/W-0h												R-0h			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POSTOFFSET2												RESERVED			
R/W-0h												R-0h			

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-948. DSS0\_VID\_CSC\_COEF7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-19	POSTOFFSET3	R/W	0h	Row-3 post-offset. Encoded signed value [from -4096 to 4095]
18-16	RESERVED	R	0h	Reserved
15-3	POSTOFFSET2	R/W	0h	Row-2 post-offset. Encoded signed value [from -4096 to 4095]
2-0	RESERVED	R	0h	Reserved

#### 12.6.4.11.13.2.140 DSS0\_VID\_ROW\_INC\_UV Register (Offset = 248h) [reset = 1h]

DSS0\_VID\_ROW\_INC\_UV is shown in [Figure 12-802](#) and described in [Table 12-950](#).

Return to [Summary Table](#).

The register configures the number of bytes to increment at the end of the row for the UV buffer associated with the video window for YUV420 formats. For non-YUV420 formats this register is unused. Shadow register

**Table 12-949. DSS0\_VID\_ROW\_INC\_UV Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0248h
DSS0_VIDL2	04A3 0248h
DSS0_VID1	04A5 0248h
DSS0_VID2	04A6 0248h

**Figure 12-802. DSS0\_VID\_ROW\_INC\_UV Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ROWINC																															
R/W-1h																															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-950. DSS0\_VID\_ROW\_INC\_UV Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	ROWINC	R/W	1h	Number of bytes to increment at the end of the row Encoded signed value [from $-2^{31}-1$ to $2^{31}$ ] to specify the number of bytes to increment at the end of the row in the video buffer. The value 0 is invalid The value 1 means next pixel. The value $1+n* bpp$ means increment of n pixels. The value $1- [n+1]* bpp$ means decrement of n pixels

### 12.6.4.11.13.2.141 DSS0\_VID\_TILE Register (Offset = 24Ch) [reset = X]

DSS0\_VID\_TILE is shown in [Figure 12-803](#) and described in [Table 12-952](#).

Return to [Summary Table](#).

Defines the characteristics of the position of the first pixel inside the compressed frame buffer. In case of non-compressed frame buffer, the register is not used.

**Table 12-951. DSS0\_VID\_TILE Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 024Ch
DSS0_VIDL2	04A3 024Ch
DSS0_VID1	04A5 024Ch
DSS0_VID2	04A6 024Ch

**Figure 12-803. DSS0\_VID\_TILE Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED										TILEINDEX																					
R/W-X										R/W-0h																					

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-952. DSS0\_VID\_TILE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-23	RESERVED	R/W	X	
22-0	TILEINDEX	R/W	0h	Defines the DSS0_VID_TILE number for the first DSS0_VID_TILE of the frame buffer: -0 means that the first DSS0_VID_TILE is accessed otherwise some tiles are skipped to support cropping of the frame buffer



### 12.6.4.11.13.2.142 DSS0\_VID\_TILE2 Register (Offset = 250h) [reset = 0h]

DSS0\_VID\_TILE2 is shown in [Figure 12-804](#) and described in [Table 12-954](#).

Return to [Summary Table](#).

Defines the number of tiles in the frame buffer. In case of non-compressed frame buffer, the register is not used.

**Table 12-953. DSS0\_VID\_TILE2 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0250h
DSS0_VIDL2	04A3 0250h
DSS0_VID1	04A5 0250h
DSS0_VID2	04A6 0250h

**Figure 12-804. DSS0\_VID\_TILE2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED									NUM_TILES																						
R-0h									R/W-0h																						

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-954. DSS0\_VID\_TILE2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-23	RESERVED	R	0h	Reserved
22-0	NUM_TILES	R/W	0h	Defines the total number of tiles in the compressed frame buffer

### 12.6.4.11.13.2.143 DSS0\_VID\_FBDC\_ATTRIBUTES Register (Offset = 254h) [reset = X]

DSS0\_VID\_FBDC\_ATTRIBUTES is shown in [Figure 12-805](#) and described in [Table 12-956](#).

Return to [Summary Table](#).

Defines the DSS0\_VID\_ATTRIBUTES for the compression engine -FBDC

**Table 12-955. DSS0\_VID\_FBDC\_ATTRIBUTES  
Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0254h
DSS0_VIDL2	04A3 0254h
DSS0_VID1	04A5 0254h
DSS0_VID2	04A6 0254h

**Figure 12-805. DSS0\_VID\_FBDC\_ATTRIBUTES Register**

31	30	29	28	27	26	25	24
RESERVED							
R/W-X							
23	22	21	20	19	18	17	16
RESERVED							
R/W-X							
15	14	13	12	11	10	9	8
RESERVED						TILETYPE	
R/W-X						R/W-0h	
7	6	5	4	3	2	1	0
FORMAT						ENABLE	
R/W-0h						R/W-0h	

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-956. DSS0\_VID\_FBDC\_ATTRIBUTES Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-10	RESERVED	R/W	X	
9-8	TILETYPE	R/W	0h	FBDC DSS0_VID_TILE-type 2h = 16x4 DSS0_VID_TILE 3h = 32x2 DSS0_VID_TILE
7-1	FORMAT	R/W	0h	FBDC format 0Ch = U8U8U8U8 0Eh = A2R10B10G10
0	ENABLE	R/W	0h	Frame Buffer Compression is Enabled. Transactions shall use secondary master port

### 12.6.4.11.13.2.144 DSS0\_VID\_FBDC\_CLEAR\_COLOR Register (Offset = 258h) [reset = 0h]

DSS0\_VID\_FBDC\_CLEAR\_COLOR is shown in [Figure 12-806](#) and described in [Table 12-958](#).

Return to [Summary Table](#).

Defines the Clear Color value to be used for the channel in FBDC

**Table 12-957. DSS0\_VID\_FBDC\_CLEAR\_COLOR Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0258h
DSS0_VIDL2	04A3 0258h
DSS0_VID1	04A5 0258h
DSS0_VID2	04A6 0258h

**Figure 12-806. DSS0\_VID\_FBDC\_CLEAR\_COLOR Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLEARCOLOR																															
R/W-0h																															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-958. DSS0\_VID\_FBDC\_CLEAR\_COLOR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	CLEARCOLOR	R/W	0h	Defines the Clear Color value to be used for the channel in FBDC

### 12.6.4.11.13.2.145 DSS0\_VID\_CLUT\_0 Register (Offset = 260h) [reset = X]

DSS0\_VID\_CLUT\_0 is shown in [Figure 12-807](#) and described in [Table 12-960](#).

Return to [Summary Table](#).

The register configures the Color Look Up Table CLUT for VID pipeline. CLUT is used in conjunction with bitmap formats

**Table 12-959. DSS0\_VID\_CLUT\_0 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0260h
DSS0_VIDL2	04A3 0260h
DSS0_VID1	04A5 0260h
DSS0_VID2	04A6 0260h

**Figure 12-807. DSS0\_VID\_CLUT\_0 Register**

31	30	29	28	27	26	25	24
INDEX	RESERVED	VALUE_R					
W-0h	W-X	W-0h					
23	22	21	20	19	18	17	16
VALUE_R				VALUE_G			
W-0h				W-0h			
15	14	13	12	11	10	9	8
VALUE_G						VALUE_B	
W-0h						W-0h	
7	6	5	4	3	2	1	0
VALUE_B							
W-0h							

LEGEND: W = Write Only; -n = value after reset

**Table 12-960. DSS0\_VID\_CLUT\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	INDEX	W	0h	Write 1 to reset the index
30	RESERVED	W	X	
29-20	VALUE_R	W	0h	10-bit R-value to store at the location in the table defined by the incrementing INDEX
19-10	VALUE_G	W	0h	10-bit G-value to store at the location in the table defined by the incrementing INDEX
9-0	VALUE_B	W	0h	10-bit B-value to store at the location in the table defined by the incrementing INDEX

### 12.6.4.11.13.2.146 DSS0\_VID\_CLUT\_1 Register (Offset = 264h) [reset = X]

DSS0\_VID\_CLUT\_1 is shown in [Figure 12-808](#) and described in [Table 12-962](#).

Return to [Summary Table](#).

The register configures the Color Look Up Table CLUT for VID pipeline. CLUT is used in conjunction with bitmap formats

**Table 12-961. DSS0\_VID\_CLUT\_1 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0264h
DSS0_VIDL2	04A3 0264h
DSS0_VID1	04A5 0264h
DSS0_VID2	04A6 0264h

**Figure 12-808. DSS0\_VID\_CLUT\_1 Register**

31	30	29	28	27	26	25	24
INDEX	RESERVED	VALUE_R					
W-0h	W-X	W-0h					
23	22	21	20	19	18	17	16
VALUE_R				VALUE_G			
W-0h				W-0h			
15	14	13	12	11	10	9	8
VALUE_G						VALUE_B	
W-0h						W-0h	
7	6	5	4	3	2	1	0
VALUE_B							
W-0h							

LEGEND: W = Write Only; -n = value after reset

**Table 12-962. DSS0\_VID\_CLUT\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	INDEX	W	0h	Write 1 to reset the index
30	RESERVED	W	X	
29-20	VALUE_R	W	0h	10-bit R-value to store at the location in the table defined by the incrementing INDEX
19-10	VALUE_G	W	0h	10-bit G-value to store at the location in the table defined by the incrementing INDEX
9-0	VALUE_B	W	0h	10-bit B-value to store at the location in the table defined by the incrementing INDEX

### 12.6.4.11.13.2.147 DSS0\_VID\_CLUT\_2 Register (Offset = 268h) [reset = X]

DSS0\_VID\_CLUT\_2 is shown in [Figure 12-809](#) and described in [Table 12-964](#).

Return to [Summary Table](#).

The register configures the Color Look Up Table CLUT for VID pipeline. CLUT is used in conjunction with bitmap formats

**Table 12-963. DSS0\_VID\_CLUT\_2 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0268h
DSS0_VIDL2	04A3 0268h
DSS0_VID1	04A5 0268h
DSS0_VID2	04A6 0268h

**Figure 12-809. DSS0\_VID\_CLUT\_2 Register**

31	30	29	28	27	26	25	24
INDEX	RESERVED	VALUE_R					
W-0h	W-X	W-0h					
23	22	21	20	19	18	17	16
VALUE_R				VALUE_G			
W-0h				W-0h			
15	14	13	12	11	10	9	8
VALUE_G						VALUE_B	
W-0h						W-0h	
7	6	5	4	3	2	1	0
VALUE_B							
W-0h							

LEGEND: W = Write Only; -n = value after reset

**Table 12-964. DSS0\_VID\_CLUT\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	INDEX	W	0h	Write 1 to reset the index
30	RESERVED	W	X	
29-20	VALUE_R	W	0h	10-bit R-value to store at the location in the table defined by the incrementing INDEX
19-10	VALUE_G	W	0h	10-bit G-value to store at the location in the table defined by the incrementing INDEX
9-0	VALUE_B	W	0h	10-bit B-value to store at the location in the table defined by the incrementing INDEX

### 12.6.4.11.13.2.148 DSS0\_VID\_CLUT\_3 Register (Offset = 26Ch) [reset = X]

DSS0\_VID\_CLUT\_3 is shown in [Figure 12-810](#) and described in [Table 12-966](#).

Return to [Summary Table](#).

The register configures the Color Look Up Table CLUT for VID pipeline. CLUT is used in conjunction with bitmap formats

**Table 12-965. DSS0\_VID\_CLUT\_3 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 026Ch
DSS0_VIDL2	04A3 026Ch
DSS0_VID1	04A5 026Ch
DSS0_VID2	04A6 026Ch

**Figure 12-810. DSS0\_VID\_CLUT\_3 Register**

31	30	29	28	27	26	25	24
INDEX	RESERVED	VALUE_R					
W-0h	W-X	W-0h					
23	22	21	20	19	18	17	16
VALUE_R				VALUE_G			
W-0h				W-0h			
15	14	13	12	11	10	9	8
VALUE_G						VALUE_B	
W-0h						W-0h	
7	6	5	4	3	2	1	0
VALUE_B							
W-0h							

LEGEND: W = Write Only; -n = value after reset

**Table 12-966. DSS0\_VID\_CLUT\_3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	INDEX	W	0h	Write 1 to reset the index
30	RESERVED	W	X	
29-20	VALUE_R	W	0h	10-bit R-value to store at the location in the table defined by the incrementing INDEX
19-10	VALUE_G	W	0h	10-bit G-value to store at the location in the table defined by the incrementing INDEX
9-0	VALUE_B	W	0h	10-bit B-value to store at the location in the table defined by the incrementing INDEX

### 12.6.4.11.13.2.149 DSS0\_VID\_CLUT\_4 Register (Offset = 270h) [reset = X]

DSS0\_VID\_CLUT\_4 is shown in [Figure 12-811](#) and described in [Table 12-968](#).

Return to [Summary Table](#).

The register configures the Color Look Up Table CLUT for VID pipeline. CLUT is used in conjunction with bitmap formats

**Table 12-967. DSS0\_VID\_CLUT\_4 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0270h
DSS0_VIDL2	04A3 0270h
DSS0_VID1	04A5 0270h
DSS0_VID2	04A6 0270h

**Figure 12-811. DSS0\_VID\_CLUT\_4 Register**

31	30	29	28	27	26	25	24
INDEX	RESERVED	VALUE_R					
W-0h	W-X	W-0h					
23	22	21	20	19	18	17	16
VALUE_R				VALUE_G			
W-0h				W-0h			
15	14	13	12	11	10	9	8
VALUE_G						VALUE_B	
W-0h						W-0h	
7	6	5	4	3	2	1	0
VALUE_B							
W-0h							

LEGEND: W = Write Only; -n = value after reset

**Table 12-968. DSS0\_VID\_CLUT\_4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	INDEX	W	0h	Write 1 to reset the index
30	RESERVED	W	X	
29-20	VALUE_R	W	0h	10-bit R-value to store at the location in the table defined by the incrementing INDEX
19-10	VALUE_G	W	0h	10-bit G-value to store at the location in the table defined by the incrementing INDEX
9-0	VALUE_B	W	0h	10-bit B-value to store at the location in the table defined by the incrementing INDEX



### 12.6.4.11.13.2.150 DSS0\_VID\_CLUT\_5 Register (Offset = 274h) [reset = X]

DSS0\_VID\_CLUT\_5 is shown in [Figure 12-812](#) and described in [Table 12-970](#).

Return to [Summary Table](#).

The register configures the Color Look Up Table CLUT for VID pipeline. CLUT is used in conjunction with bitmap formats

**Table 12-969. DSS0\_VID\_CLUT\_5 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0274h
DSS0_VIDL2	04A3 0274h
DSS0_VID1	04A5 0274h
DSS0_VID2	04A6 0274h

**Figure 12-812. DSS0\_VID\_CLUT\_5 Register**

31	30	29	28	27	26	25	24
INDEX	RESERVED	VALUE_R					
W-0h	W-X	W-0h					
23	22	21	20	19	18	17	16
VALUE_R				VALUE_G			
W-0h				W-0h			
15	14	13	12	11	10	9	8
VALUE_G						VALUE_B	
W-0h						W-0h	
7	6	5	4	3	2	1	0
VALUE_B							
W-0h							

LEGEND: W = Write Only; -n = value after reset

**Table 12-970. DSS0\_VID\_CLUT\_5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	INDEX	W	0h	Write 1 to reset the index
30	RESERVED	W	X	
29-20	VALUE_R	W	0h	10-bit R-value to store at the location in the table defined by the incrementing INDEX
19-10	VALUE_G	W	0h	10-bit G-value to store at the location in the table defined by the incrementing INDEX
9-0	VALUE_B	W	0h	10-bit B-value to store at the location in the table defined by the incrementing INDEX

### 12.6.4.11.13.2.151 DSS0\_VID\_CLUT\_6 Register (Offset = 278h) [reset = X]

DSS0\_VID\_CLUT\_6 is shown in [Figure 12-813](#) and described in [Table 12-972](#).

Return to [Summary Table](#).

The register configures the Color Look Up Table CLUT for VID pipeline. CLUT is used in conjunction with bitmap formats

**Table 12-971. DSS0\_VID\_CLUT\_6 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0278h
DSS0_VIDL2	04A3 0278h
DSS0_VID1	04A5 0278h
DSS0_VID2	04A6 0278h

**Figure 12-813. DSS0\_VID\_CLUT\_6 Register**

31	30	29	28	27	26	25	24
INDEX	RESERVED	VALUE_R					
W-0h	W-X	W-0h					
23	22	21	20	19	18	17	16
VALUE_R				VALUE_G			
W-0h				W-0h			
15	14	13	12	11	10	9	8
VALUE_G						VALUE_B	
W-0h						W-0h	
7	6	5	4	3	2	1	0
VALUE_B							
W-0h							

LEGEND: W = Write Only; -n = value after reset

**Table 12-972. DSS0\_VID\_CLUT\_6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	INDEX	W	0h	Write 1 to reset the index
30	RESERVED	W	X	
29-20	VALUE_R	W	0h	10-bit R-value to store at the location in the table defined by the incrementing INDEX
19-10	VALUE_G	W	0h	10-bit G-value to store at the location in the table defined by the incrementing INDEX
9-0	VALUE_B	W	0h	10-bit B-value to store at the location in the table defined by the incrementing INDEX

### 12.6.4.11.13.2.152 DSS0\_VID\_CLUT\_7 Register (Offset = 27Ch) [reset = X]

DSS0\_VID\_CLUT\_7 is shown in [Figure 12-814](#) and described in [Table 12-974](#).

Return to [Summary Table](#).

The register configures the Color Look Up Table CLUT for VID pipeline. CLUT is used in conjunction with bitmap formats

**Table 12-973. DSS0\_VID\_CLUT\_7 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 027Ch
DSS0_VIDL2	04A3 027Ch
DSS0_VID1	04A5 027Ch
DSS0_VID2	04A6 027Ch

**Figure 12-814. DSS0\_VID\_CLUT\_7 Register**

31	30	29	28	27	26	25	24
INDEX	RESERVED	VALUE_R					
W-0h	W-X	W-0h					
23	22	21	20	19	18	17	16
VALUE_R				VALUE_G			
W-0h				W-0h			
15	14	13	12	11	10	9	8
VALUE_G						VALUE_B	
W-0h						W-0h	
7	6	5	4	3	2	1	0
VALUE_B							
W-0h							

LEGEND: W = Write Only; -n = value after reset

**Table 12-974. DSS0\_VID\_CLUT\_7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	INDEX	W	0h	Write 1 to reset the index
30	RESERVED	W	X	
29-20	VALUE_R	W	0h	10-bit R-value to store at the location in the table defined by the incrementing INDEX
19-10	VALUE_G	W	0h	10-bit G-value to store at the location in the table defined by the incrementing INDEX
9-0	VALUE_B	W	0h	10-bit B-value to store at the location in the table defined by the incrementing INDEX

### 12.6.4.11.13.2.153 DSS0\_VID\_CLUT\_8 Register (Offset = 280h) [reset = X]

DSS0\_VID\_CLUT\_8 is shown in [Figure 12-815](#) and described in [Table 12-976](#).

Return to [Summary Table](#).

The register configures the Color Look Up Table CLUT for VID pipeline. CLUT is used in conjunction with bitmap formats

**Table 12-975. DSS0\_VID\_CLUT\_8 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0280h
DSS0_VIDL2	04A3 0280h
DSS0_VID1	04A5 0280h
DSS0_VID2	04A6 0280h

**Figure 12-815. DSS0\_VID\_CLUT\_8 Register**

31	30	29	28	27	26	25	24
INDEX	RESERVED	VALUE_R					
W-0h	W-X	W-0h					
23	22	21	20	19	18	17	16
VALUE_R				VALUE_G			
W-0h				W-0h			
15	14	13	12	11	10	9	8
VALUE_G						VALUE_B	
W-0h						W-0h	
7	6	5	4	3	2	1	0
VALUE_B							
W-0h							

LEGEND: W = Write Only; -n = value after reset

**Table 12-976. DSS0\_VID\_CLUT\_8 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	INDEX	W	0h	Write 1 to reset the index
30	RESERVED	W	X	
29-20	VALUE_R	W	0h	10-bit R-value to store at the location in the table defined by the incrementing INDEX
19-10	VALUE_G	W	0h	10-bit G-value to store at the location in the table defined by the incrementing INDEX
9-0	VALUE_B	W	0h	10-bit B-value to store at the location in the table defined by the incrementing INDEX

#### 12.6.4.11.13.2.154 DSS0\_VID\_CLUT\_9 Register (Offset = 284h) [reset = X]

DSS0\_VID\_CLUT\_9 is shown in [Figure 12-816](#) and described in [Table 12-978](#).

Return to [Summary Table](#).

The register configures the Color Look Up Table CLUT for VID pipeline. CLUT is used in conjunction with bitmap formats

**Table 12-977. DSS0\_VID\_CLUT\_9 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0284h
DSS0_VIDL2	04A3 0284h
DSS0_VID1	04A5 0284h
DSS0_VID2	04A6 0284h

**Figure 12-816. DSS0\_VID\_CLUT\_9 Register**

31	30	29	28	27	26	25	24
INDEX	RESERVED	VALUE_R					
W-0h	W-X	W-0h					
23	22	21	20	19	18	17	16
VALUE_R				VALUE_G			
W-0h				W-0h			
15	14	13	12	11	10	9	8
VALUE_G						VALUE_B	
W-0h						W-0h	
7	6	5	4	3	2	1	0
VALUE_B							
W-0h							

LEGEND: W = Write Only; -n = value after reset

**Table 12-978. DSS0\_VID\_CLUT\_9 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	INDEX	W	0h	Write 1 to reset the index
30	RESERVED	W	X	
29-20	VALUE_R	W	0h	10-bit R-value to store at the location in the table defined by the incrementing INDEX
19-10	VALUE_G	W	0h	10-bit G-value to store at the location in the table defined by the incrementing INDEX
9-0	VALUE_B	W	0h	10-bit B-value to store at the location in the table defined by the incrementing INDEX

### 12.6.4.11.13.2.155 DSS0\_VID\_CLUT\_10 Register (Offset = 288h) [reset = X]

DSS0\_VID\_CLUT\_10 is shown in [Figure 12-817](#) and described in [Table 12-980](#).

Return to [Summary Table](#).

The register configures the Color Look Up Table CLUT for VID pipeline. CLUT is used in conjunction with bitmap formats

**Table 12-979. DSS0\_VID\_CLUT\_10 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0288h
DSS0_VIDL2	04A3 0288h
DSS0_VID1	04A5 0288h
DSS0_VID2	04A6 0288h

**Figure 12-817. DSS0\_VID\_CLUT\_10 Register**

31	30	29	28	27	26	25	24
INDEX	RESERVED	VALUE_R					
W-0h	W-X	W-0h					
23	22	21	20	19	18	17	16
VALUE_R				VALUE_G			
W-0h				W-0h			
15	14	13	12	11	10	9	8
VALUE_G						VALUE_B	
W-0h						W-0h	
7	6	5	4	3	2	1	0
VALUE_B							
W-0h							

LEGEND: W = Write Only; -n = value after reset

**Table 12-980. DSS0\_VID\_CLUT\_10 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	INDEX	W	0h	Write 1 to reset the index
30	RESERVED	W	X	
29-20	VALUE_R	W	0h	10-bit R-value to store at the location in the table defined by the incrementing INDEX
19-10	VALUE_G	W	0h	10-bit G-value to store at the location in the table defined by the incrementing INDEX
9-0	VALUE_B	W	0h	10-bit B-value to store at the location in the table defined by the incrementing INDEX

### 12.6.4.11.13.2.156 DSS0\_VID\_CLUT\_11 Register (Offset = 28Ch) [reset = X]

DSS0\_VID\_CLUT\_11 is shown in [Figure 12-818](#) and described in [Table 12-982](#).

Return to [Summary Table](#).

The register configures the Color Look Up Table CLUT for VID pipeline. CLUT is used in conjunction with bitmap formats

**Table 12-981. DSS0\_VID\_CLUT\_11 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 028Ch
DSS0_VIDL2	04A3 028Ch
DSS0_VID1	04A5 028Ch
DSS0_VID2	04A6 028Ch

**Figure 12-818. DSS0\_VID\_CLUT\_11 Register**

31	30	29	28	27	26	25	24
INDEX	RESERVED	VALUE_R					
W-0h	W-X	W-0h					
23	22	21	20	19	18	17	16
VALUE_R				VALUE_G			
W-0h				W-0h			
15	14	13	12	11	10	9	8
VALUE_G						VALUE_B	
W-0h						W-0h	
7	6	5	4	3	2	1	0
VALUE_B							
W-0h							

LEGEND: W = Write Only; -n = value after reset

**Table 12-982. DSS0\_VID\_CLUT\_11 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	INDEX	W	0h	Write 1 to reset the index
30	RESERVED	W	X	
29-20	VALUE_R	W	0h	10-bit R-value to store at the location in the table defined by the incrementing INDEX
19-10	VALUE_G	W	0h	10-bit G-value to store at the location in the table defined by the incrementing INDEX
9-0	VALUE_B	W	0h	10-bit B-value to store at the location in the table defined by the incrementing INDEX

### 12.6.4.11.13.2.157 DSS0\_VID\_CLUT\_12 Register (Offset = 290h) [reset = X]

DSS0\_VID\_CLUT\_12 is shown in [Figure 12-819](#) and described in [Table 12-984](#).

Return to [Summary Table](#).

The register configures the Color Look Up Table CLUT for VID pipeline. CLUT is used in conjunction with bitmap formats

**Table 12-983. DSS0\_VID\_CLUT\_12 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0290h
DSS0_VIDL2	04A3 0290h
DSS0_VID1	04A5 0290h
DSS0_VID2	04A6 0290h

**Figure 12-819. DSS0\_VID\_CLUT\_12 Register**

31	30	29	28	27	26	25	24
INDEX	RESERVED	VALUE_R					
W-0h	W-X	W-0h					
23	22	21	20	19	18	17	16
VALUE_R				VALUE_G			
W-0h				W-0h			
15	14	13	12	11	10	9	8
VALUE_G						VALUE_B	
W-0h						W-0h	
7	6	5	4	3	2	1	0
VALUE_B							
W-0h							

LEGEND: W = Write Only; -n = value after reset

**Table 12-984. DSS0\_VID\_CLUT\_12 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	INDEX	W	0h	Write 1 to reset the index
30	RESERVED	W	X	
29-20	VALUE_R	W	0h	10-bit R-value to store at the location in the table defined by the incrementing INDEX
19-10	VALUE_G	W	0h	10-bit G-value to store at the location in the table defined by the incrementing INDEX
9-0	VALUE_B	W	0h	10-bit B-value to store at the location in the table defined by the incrementing INDEX



### 12.6.4.11.13.2.158 DSS0\_VID\_CLUT\_13 Register (Offset = 294h) [reset = X]

DSS0\_VID\_CLUT\_13 is shown in [Figure 12-820](#) and described in [Table 12-986](#).

Return to [Summary Table](#).

The register configures the Color Look Up Table CLUT for VID pipeline. CLUT is used in conjunction with bitmap formats

**Table 12-985. DSS0\_VID\_CLUT\_13 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0294h
DSS0_VIDL2	04A3 0294h
DSS0_VID1	04A5 0294h
DSS0_VID2	04A6 0294h

**Figure 12-820. DSS0\_VID\_CLUT\_13 Register**

31	30	29	28	27	26	25	24
INDEX	RESERVED	VALUE_R					
W-0h	W-X	W-0h					
23	22	21	20	19	18	17	16
VALUE_R				VALUE_G			
W-0h				W-0h			
15	14	13	12	11	10	9	8
VALUE_G						VALUE_B	
W-0h						W-0h	
7	6	5	4	3	2	1	0
VALUE_B							
W-0h							

LEGEND: W = Write Only; -n = value after reset

**Table 12-986. DSS0\_VID\_CLUT\_13 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	INDEX	W	0h	Write 1 to reset the index
30	RESERVED	W	X	
29-20	VALUE_R	W	0h	10-bit R-value to store at the location in the table defined by the incrementing INDEX
19-10	VALUE_G	W	0h	10-bit G-value to store at the location in the table defined by the incrementing INDEX
9-0	VALUE_B	W	0h	10-bit B-value to store at the location in the table defined by the incrementing INDEX

### 12.6.4.11.13.2.159 DSS0\_VID\_CLUT\_14 Register (Offset = 298h) [reset = X]

DSS0\_VID\_CLUT\_14 is shown in [Figure 12-821](#) and described in [Table 12-988](#).

Return to [Summary Table](#).

The register configures the Color Look Up Table CLUT for VID pipeline. CLUT is used in conjunction with bitmap formats

**Table 12-987. DSS0\_VID\_CLUT\_14 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 0298h
DSS0_VIDL2	04A3 0298h
DSS0_VID1	04A5 0298h
DSS0_VID2	04A6 0298h

**Figure 12-821. DSS0\_VID\_CLUT\_14 Register**

31	30	29	28	27	26	25	24
INDEX	RESERVED	VALUE_R					
W-0h	W-X	W-0h					
23	22	21	20	19	18	17	16
VALUE_R				VALUE_G			
W-0h				W-0h			
15	14	13	12	11	10	9	8
VALUE_G						VALUE_B	
W-0h						W-0h	
7	6	5	4	3	2	1	0
VALUE_B							
W-0h							

LEGEND: W = Write Only; -n = value after reset

**Table 12-988. DSS0\_VID\_CLUT\_14 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	INDEX	W	0h	Write 1 to reset the index
30	RESERVED	W	X	
29-20	VALUE_R	W	0h	10-bit R-value to store at the location in the table defined by the incrementing INDEX
19-10	VALUE_G	W	0h	10-bit G-value to store at the location in the table defined by the incrementing INDEX
9-0	VALUE_B	W	0h	10-bit B-value to store at the location in the table defined by the incrementing INDEX

### 12.6.4.11.13.2.160 DSS0\_VID\_CLUT\_15 Register (Offset = 29Ch) [reset = X]

DSS0\_VID\_CLUT\_15 is shown in [Figure 12-822](#) and described in [Table 12-990](#).

Return to [Summary Table](#).

The register configures the Color Look Up Table CLUT for VID pipeline. CLUT is used in conjunction with bitmap formats

**Table 12-989. DSS0\_VID\_CLUT\_15 Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 029Ch
DSS0_VIDL2	04A3 029Ch
DSS0_VID1	04A5 029Ch
DSS0_VID2	04A6 029Ch

**Figure 12-822. DSS0\_VID\_CLUT\_15 Register**

31	30	29	28	27	26	25	24
INDEX	RESERVED	VALUE_R					
W-0h	W-X	W-0h					
23	22	21	20	19	18	17	16
VALUE_R				VALUE_G			
W-0h				W-0h			
15	14	13	12	11	10	9	8
VALUE_G						VALUE_B	
W-0h						W-0h	
7	6	5	4	3	2	1	0
VALUE_B							
W-0h							

LEGEND: W = Write Only; -n = value after reset

**Table 12-990. DSS0\_VID\_CLUT\_15 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	INDEX	W	0h	Write 1 to reset the index
30	RESERVED	W	X	
29-20	VALUE_R	W	0h	10-bit R-value to store at the location in the table defined by the incrementing INDEX
19-10	VALUE_G	W	0h	10-bit G-value to store at the location in the table defined by the incrementing INDEX
9-0	VALUE_B	W	0h	10-bit B-value to store at the location in the table defined by the incrementing INDEX

### 12.6.4.11.13.2.161 DSS0\_VID\_SAFETY\_ATTRIBUTES Register (Offset = 2A0h) [reset = 0h]

DSS0\_VID\_SAFETY\_ATTRIBUTES is shown in [Figure 12-823](#) and described in [Table 12-992](#).

Return to [Summary Table](#).

The register configures the safety sub-region. Shadow register

**Table 12-991. DSS0\_VID\_SAFETY\_ATTRIBUTES Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 02A0h
DSS0_VIDL2	04A3 02A0h
DSS0_VID1	04A5 02A0h
DSS0_VID2	04A6 02A0h

**Figure 12-823. DSS0\_VID\_SAFETY\_ATTRIBUTES Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED			FRAMESKIP		THRESHOLD		
R-0h			R/W-0h		R/W-0h		
7	6	5	4	3	2	1	0
THRESHOLD					SEEDSELECT	CAPTUREMODE	ENABLE
R/W-0h					R/W-0h	R/W-0h	R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-992. DSS0\_VID\_SAFETY\_ATTRIBUTES Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-13	RESERVED	R	0h	Reserved
12-11	FRAMESKIP	R/W	0h	Indicates which frames to be skipped while doing FRAMEFREEZE or DATACHECK [Useful for interlaced displays]. 0x 0: No frames are skipped, 0x 1: Even Frames are skipped starting from second frame after ENABLE, 0x 2: Odd Frames are skipped starting from first frame after ENABLE, 0x 3: Reserved  0h = No frames are skipped 1h = Even Frames are skipped starting from second frame after ENABLE 2h = Odd Frames are skipped starting from first frame after ENABLE 3h = Reserved

**Table 12-992. DSS0\_VID\_SAFETY\_ATTRIBUTES Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
10-3	THRESHOLD	R/W	0h	Allowed maximum number of frames with the same frame signature. When the freeze frame counter reaches 1 over this value [freeze_frame_thold+1], a freeze frame detection will occur. Note: The freeze frame counter is cleared on reset -OR- MISR not enabled -OR- terminal count reached -OR- compare == no match
2	SEEDSELECT	R/W	0h	Initial seed selection control  0h = Initial seed is always 0xFFFF_FFFF 1h = Initial seed is defined by SAFETY_LFSR_START.SEED
1	CAPTUREMODE	R/W	0h	Mode of operation of the safety check module  0h = Frame freeze detect enabled 1h = Data correctness check enabled
0	ENABLE	R/W	0h	Safety check Enable for the region. Note: Transition from 0 to 1 clears the signature register

### 12.6.4.11.13.2.162 DSS0\_VID\_SAFETY\_CAPT\_SIGNATURE Register (Offset = 2A4h) [reset = 0h]

DSS0\_VID\_SAFETY\_CAPT\_SIGNATURE is shown in [Figure 12-824](#) and described in [Table 12-994](#).

Return to [Summary Table](#).

The register captures the signature from the MISR of the safety sub-region. Shadow register

**Table 12-993.**

**DSS0\_VID\_SAFETY\_CAPT\_SIGNATURE Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 02A4h
DSS0_VIDL2	04A3 02A4h
DSS0_VID1	04A5 02A4h
DSS0_VID2	04A6 02A4h

**Figure 12-824. DSS0\_VID\_SAFETY\_CAPT\_SIGNATURE Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGNATURE																															
R-0h																															

LEGEND: R = Read Only; -n = value after reset

**Table 12-994. DSS0\_VID\_SAFETY\_CAPT\_SIGNATURE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SIGNATURE	R	0h	The register configures the reference signature of the safety sub-region. Shadow register

### 12.6.4.11.13.2.163 DSS0\_VID\_SAFETY\_POSITION Register (Offset = 2A8h) [reset = X]

DSS0\_VID\_SAFETY\_POSITION is shown in [Figure 12-825](#) and described in [Table 12-996](#).

Return to [Summary Table](#).

The register configures the position of the safety sub-region. Shadow register

**Table 12-995. DSS0\_VID\_SAFETY\_POSITION  
Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 02A8h
DSS0_VIDL2	04A3 02A8h
DSS0_VID1	04A5 02A8h
DSS0_VID2	04A6 02A8h

**Figure 12-825. DSS0\_VID\_SAFETY\_POSITION Register**

31	30	29	28	27	26	25	24
RESERVED				POSY			
R/W-X				R/W-0h			
23	22	21	20	19	18	17	16
				POSY			
				R/W-0h			
15	14	13	12	11	10	9	8
RESERVED				POSX			
R/W-X				R/W-0h			
7	6	5	4	3	2	1	0
				POSX			
				R/W-0h			

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-996. DSS0\_VID\_SAFETY\_POSITION Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	X	
29-16	POSY	R/W	0h	Y position of the safety sub-region. Encoded value [from 0 to 16383] to specify the Y position of the sub-region on the screen. The first line on the top of the screen has the Y-position 0
15-14	RESERVED	R/W	X	
13-0	POSX	R/W	0h	X position of the safety sub-region. Encoded value [from 0 to 16383] to specify the X position of the sub-region on the screen. The first pixel on the left of the screen has the X-position 0

### 12.6.4.11.13.2.164 DSS0\_VID\_SAFETY\_REF\_SIGNATURE Register (Offset = 2ACh) [reset = 0h]

DSS0\_VID\_SAFETY\_REF\_SIGNATURE is shown in [Figure 12-826](#) and described in [Table 12-998](#).

Return to [Summary Table](#).

The register configures the reference signature of the safety sub-region. Shadow register

**Table 12-997. DSS0\_VID\_SAFETY\_REF\_SIGNATURE Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 02ACh
DSS0_VIDL2	04A3 02ACh
DSS0_VID1	04A5 02ACh
DSS0_VID2	04A6 02ACh

**Figure 12-826. DSS0\_VID\_SAFETY\_REF\_SIGNATURE Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGNATURE																															
R/W-0h																															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-998. DSS0\_VID\_SAFETY\_REF\_SIGNATURE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SIGNATURE	R/W	0h	The register configures the reference signature of the safety sub-region. Shadow register



### 12.6.4.11.13.2.165 DSS0\_VID\_SAFETY\_SIZE Register (Offset = 2B0h) [reset = X]

DSS0\_VID\_SAFETY\_SIZE is shown in [Figure 12-827](#) and described in [Table 12-1000](#).

Return to [Summary Table](#).

The register configures the DSS0\_VID\_SIZE of the safety sub-region. Shadow register

**Table 12-999. DSS0\_VID\_SAFETY\_SIZE Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 02B0h
DSS0_VIDL2	04A3 02B0h
DSS0_VID1	04A5 02B0h
DSS0_VID2	04A6 02B0h

**Figure 12-827. DSS0\_VID\_SAFETY\_SIZE Register**

31	30	29	28	27	26	25	24
RESERVED				SIZEY			
R/W-X				R/W-0h			
23	22	21	20	19	18	17	16
SIZEY							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED				SIZEX			
R/W-X				R/W-0h			
7	6	5	4	3	2	1	0
SIZEX							
R/W-0h							

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1000. DSS0\_VID\_SAFETY\_SIZE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	X	
29-16	SIZEY	R/W	0h	Height of the safety sub-region. Encoded value [from 0 to 16383] to specify the height of the sub-region on the screen. One line height region has value of 0
15-14	RESERVED	R/W	X	
13-0	SIZEX	R/W	0h	Width of the safety sub-region. Encoded value [from 0 to 16383] to specify the width of the sub-region on the screen. One pixel wide region has value of 0

### 12.6.4.11.13.2.166 DSS0\_VID\_SAFETY\_LFSR\_SEED Register (Offset = 2B4h) [reset = 0h]

DSS0\_VID\_SAFETY\_LFSR\_SEED is shown in [Figure 12-828](#) and described in [Table 12-1002](#).

Return to [Summary Table](#).

The register configures the seed [initial value] of the MISR. Otherwise, the MISR is initialized with 0xFFFF\_FFFF. Shadow register

**Table 12-1001. DSS0\_VID\_SAFETY\_LFSR\_SEED Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 02B4h
DSS0_VIDL2	04A3 02B4h
DSS0_VID1	04A5 02B4h
DSS0_VID2	04A6 02B4h

**Figure 12-828. DSS0\_VID\_SAFETY\_LFSR\_SEED Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEED																															
R/W-0h																															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1002. DSS0\_VID\_SAFETY\_LFSR\_SEED Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SEED	R/W	0h	The register configures the seed [initial value] of the MISR. Otherwise, the MISR is initialized with 0xFFFF_FFFF. Shadow register

### 12.6.4.11.13.2.167 DSS0\_VID\_LUMAKEY Register (Offset = 2B8h) [reset = 0h]

DSS0\_VID\_LUMAKEY is shown in [Figure 12-829](#) and described in [Table 12-1004](#).

Return to [Summary Table](#).

The register configures the LUMA KEY transparency min and max values. Shadow register

**Table 12-1003. DSS0\_VID\_LUMAKEY Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 02B8h
DSS0_VIDL2	04A3 02B8h
DSS0_VID1	04A5 02B8h
DSS0_VID2	04A6 02B8h

**Figure 12-829. DSS0\_VID\_LUMAKEY Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED				LUMAKEYMAX											
R-0h				R/W-0h											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				LUMAKEYMIN											
R-0h				R/W-0h											

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1004. DSS0\_VID\_LUMAKEY Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
27-16	LUMAKEYMAX	R/W	0h	12b luma_key_max value
15-12	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
11-0	LUMAKEYMIN	R/W	0h	12b luma_key_min value

### 12.6.4.11.13.2.168 DSS0\_VID\_DMA\_BUFSIZE Register (Offset = 2BCh) [reset = 4h]

DSS0\_VID\_DMA\_BUFSIZE is shown in [Figure 12-830](#) and described in [Table 12-1006](#).

Return to [Summary Table](#).

The register configures the DMA buffer DSS0\_VID\_SIZE allocated to the pipeline - New Shared memory feature

**Table 12-1005. DSS0\_VID\_DMA\_BUFSIZE Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 02BCh
DSS0_VIDL2	04A3 02BCh
DSS0_VID1	04A5 02BCh
DSS0_VID2	04A6 02BCh

**Figure 12-830. DSS0\_VID\_DMA\_BUFSIZE Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											BUFSIZE				
R-0h											R/W-4h				

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1006. DSS0\_VID\_DMA\_BUFSIZE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reserved
4-0	BUFSIZE	R/W	4h	DMA buffer DSS0_VID_SIZE, if VID pipe is enabled.If the value programmed is n, then the allocated buffer DSS0_VID_SIZE is 16KB*n. Default:64KB

### 12.6.4.11.13.2.169 DSS0\_VID\_CROP Register (Offset = 2C0h) [reset = X]

DSS0\_VID\_CROP is shown in [Figure 12-831](#) and described in [Table 12-1008](#).

Return to [Summary Table](#).

Defines the DSS0\_VID\_ATTRIBUTES for the output cropping in Video Pipe

**Table 12-1007. DSS0\_VID\_CROP Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 02C0h
DSS0_VIDL2	04A3 02C0h
DSS0_VID1	04A5 02C0h
DSS0_VID2	04A6 02C0h

**Figure 12-831. DSS0\_VID\_CROP Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED				CROPBOTTOM				RESERVED				CROPTOP			
R/W-X				R/W-0h				R/W-X				R/W-0h			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				CROPRIGHT				RESERVED				CROPLEFT			
R/W-X				R/W-0h				R/W-X				R/W-0h			

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1008. DSS0\_VID\_CROP Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-29	RESERVED	R/W	X	
28-24	CROPBOTTOM	R/W	0h	DSS0_VID_CROP Bottom in Lines. Values from 0-31
23-21	RESERVED	R/W	X	
20-16	CROPTOP	R/W	0h	DSS0_VID_CROP Top in Lines. Values from 0-31
15-13	RESERVED	R/W	X	
12-8	CROPRIGHT	R/W	0h	DSS0_VID_CROP Right in Pixels. Values from 0-31
7-5	RESERVED	R/W	X	
4-0	CROPLEFT	R/W	0h	DSS0_VID_CROP Left in Pixels. Values from 0-31

### 12.6.4.11.13.2.170 DSS0\_VID\_SECURE Register (Offset = 2C4h) [reset = 0h]

DSS0\_VID\_SECURE is shown in [Figure 12-832](#) and described in [Table 12-1010](#).

Return to [Summary Table](#).

Security bit settings for the sub-module

**Table 12-1009. DSS0\_VID\_SECURE Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 02C4h
DSS0_VIDL2	04A3 02C4h
DSS0_VID1	04A5 02C4h
DSS0_VID2	04A6 02C4h

**Figure 12-832. DSS0\_VID\_SECURE Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							SECURE
R-0h							R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1010. DSS0\_VID\_SECURE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	SECURE	R/W	0h	DSS0_VID_SECURE bit 0h = DSS0_VID_SECURE bit is reset 1h = DSS0_VID_SECURE bit is set

### 12.6.4.11.13.2.171 DSS0\_VID\_PIPE\_GO Register (Offset = 2C8h) [reset = 0h]

DSS0\_VID\_PIPE\_GO is shown in [Figure 12-833](#) and described in [Table 12-1012](#).

Return to [Summary Table](#).

PIPE GO bit settings

**Table 12-1011. DSS0\_VID\_PIPE\_GO Instances**

Instance	Physical Address
DSS0_VIDL1	04A2 02C8h
DSS0_VIDL2	04A3 02C8h
DSS0_VID1	04A5 02C8h
DSS0_VID2	04A6 02C8h

**Figure 12-833. DSS0\_VID\_PIPE\_GO Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							GOBIT
R-0h							R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1012. DSS0\_VID\_PIPE\_GO Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	GOBIT	R/W	0h	Go bit  0h = The hardware has finished the synchronization  1h = Software has requested for synchronization after register updates and the hardware has not finished the synchronization

### 12.6.4.11.13.3 DSS\_OVR Registers

Table 12-1014 lists the memory-mapped registers for the DSS\_OVR. All register offset addresses not listed in Table 12-1014 should be considered as reserved locations and the register contents should not be modified.

OVR Registers

**Table 12-1013. DSS\_OVR Instances**

Instance	Base Address
DSS0_OVR1	04A7 0000h
DSS0_OVR2	04A9 0000h
DSS0_OVR3	04AB 0000h
DSS0_OVR4	04AD 0000h

**Table 12-1014. DSS\_OVR Registers**

Offset	Acronym	Register Name	DSS0_OVR1 Physical Address	DSS0_OVR2 Physical Address	DSS0_OVR3 Physical Address	DSS0_OVR4 Physical Address
00h	<a href="#">DSS0_OVR_CONFIG</a>		04A7 0000h	04A9 0000h	04AB 0000h	04AD 0000h
04h	<a href="#">DSS0_OVR_VIRTUALVP</a>		04A7 0004h	04A9 0004h	04AB 0004h	04AD 0004h
08h	<a href="#">DSS0_OVR_DEFAULT_COLOR</a>		04A7 0008h	04A9 0008h	04AB 0008h	04AD 0008h
0Ch	<a href="#">DSS0_OVR_DEFAULT_COLOR2</a>		04A7 000Ch	04A9 000Ch	04AB 000Ch	04AD 000Ch
10h	<a href="#">DSS0_OVR_TRANS_COLOR_MAX</a>		04A7 0010h	04A9 0010h	04AB 0010h	04AD 0010h
14h	<a href="#">DSS0_OVR_TRANS_COLOR_MAX2</a>		04A7 0014h	04A9 0014h	04AB 0014h	04AD 0014h
18h	<a href="#">DSS0_OVR_TRANS_COLOR_MIN</a>		04A7 0018h	04A9 0018h	04AB 0018h	04AD 0018h
1Ch	<a href="#">DSS0_OVR_TRANS_COLOR_MIN2</a>		04A7 001Ch	04A9 001Ch	04AB 001Ch	04AD 001Ch
20h	<a href="#">DSS0_OVR_ATTRIBUTES_0</a>		04A7 0020h	04A9 0020h	04AB 0020h	04AD 0020h
24h	<a href="#">DSS0_OVR_ATTRIBUTES_1</a>		04A7 0024h	04A9 0024h	04AB 0024h	04AD 0024h
28h	<a href="#">DSS0_OVR_ATTRIBUTES_2</a>		04A7 0028h	04A9 0028h	04AB 0028h	04AD 0028h
2Ch	<a href="#">DSS0_OVR_ATTRIBUTES_3</a>		04A7 002Ch	04A9 002Ch	04AB 002Ch	04AD 002Ch
30h	<a href="#">DSS0_OVR_ATTRIBUTES_4</a>		04A7 0030h	04A9 0030h	04AB 0030h	04AD 0030h
34h	<a href="#">DSS0_OVR_ATTRIBUTES2_0</a>		04A7 0034h	04A9 0034h	04AB 0034h	04AD 0034h
38h	<a href="#">DSS0_OVR_ATTRIBUTES2_1</a>		04A7 0038h	04A9 0038h	04AB 0038h	04AD 0038h
3Ch	<a href="#">DSS0_OVR_ATTRIBUTES2_2</a>		04A7 003Ch	04A9 003Ch	04AB 003Ch	04AD 003Ch
40h	<a href="#">DSS0_OVR_ATTRIBUTES2_3</a>		04A7 0040h	04A9 0040h	04AB 0040h	04AD 0040h
44h	<a href="#">DSS0_OVR_ATTRIBUTES2_4</a>		04A7 0044h	04A9 0044h	04AB 0044h	04AD 0044h
48h	<a href="#">DSS0_OVR_SECURE</a>		04A7 0048h	04A9 0048h	04AB 0048h	04AD 0048h



### 12.6.4.11.13.3.1 DSS0\_OVR\_CONFIG Register (Offset = 00h) [reset = 0h]

DSS0\_OVR\_CONFIG is shown in [Figure 12-834](#) and described in [Table 12-1016](#).

Return to [Summary Table](#).

The control register configures the Display Controller module for the VP output. Shadow register

**Table 12-1015. DSS0\_OVR\_CONFIG Instances**

Instance	Physical Address
DSS0_OVR1	04A7 0000h
DSS0_OVR2	04A9 0000h
DSS0_OVR3	04AB 0000h
DSS0_OVR4	04AD 0000h

**Figure 12-834. DSS0\_OVR\_CONFIG Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED		RESERVED	RESERVED	TCKLCDSELECTION	TCKLCDENABLE	RESERVED	
R-0h		R-0h	R-0h	R/W-0h	R/W-0h	R-0h	
7	6	5	4	3	2	1	0
RESERVED						COLORBAREN	RESERVED
R-0h						R/W-0h	R-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1016. DSS0\_OVR\_CONFIG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-14	RESERVED	R	0h	
13	RESERVED	R	0h	
12	RESERVED	R	0h	
11	TCKLCDSELECTION	R/W	0h	Transparency Color Key Selection 0h = Destination transparency color key selected 1h = Source transparency color key selected
10	TCKLCDENABLE	R/W	0h	Transparency Color Key Enable 0h = Disable the transparency color key for the LCD 1h = Enable the transparency color key for the LCD
9-2	RESERVED	R	0h	
1	COLORBAREN	R/W	0h	Enable the Color-Bar 0h = Disabled 1h = Enabled

**Table 12-1016. DSS0\_OVR\_CONFIG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
0	RESERVED	R	0h	

### 12.6.4.11.13.3.2 DSS0\_OVR\_VIRTUALVP Register (Offset = 04h) [reset = X]

DSS0\_OVR\_VIRTUALVP is shown in [Figure 12-835](#) and described in [Table 12-1018](#).

Return to [Summary Table](#).

Configures the new VIRTUAL VP operation. Shadow register

**Table 12-1017. DSS0\_OVR\_VIRTUALVP Instances**

Instance	Physical Address
DSS0_OVR1	04A7 0004h
DSS0_OVR2	04A9 0004h
DSS0_OVR3	04AB 0004h
DSS0_OVR4	04AD 0004h

**Figure 12-835. DSS0\_OVR\_VIRTUALVP Register**

31	30	29	28	27	26	25	24
ENABLE	RESERVED	LPP					
R/W-0h	R/W-X	R/W-0h					
23	22	21	20	19	18	17	16
LPP							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED	PPL						
R/W-X	R/W-0h						
7	6	5	4	3	2	1	0
PPL							
R/W-0h							

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1018. DSS0\_OVR\_VIRTUALVP Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	ENABLE	R/W	0h	Enable the Virtual VP Operation
30	RESERVED	R/W	X	
29-16	LPP	R/W	0h	Lines per panel Encoded value [from 1 to 16384] to specify the number of lines on the Virtual VP [program to value minus 1]
15-14	RESERVED	R/W	X	
13-0	PPL	R/W	0h	Pixels per line Encoded value [from 1 to 16384] to specify the number of pixels contains within each line on the Virtual VP [program to value minus 1]

### 12.6.4.11.13.3.3 DSS0\_OVR\_DEFAULT\_COLOR Register (Offset = 08h) [reset = 0h]

DSS0\_OVR\_DEFAULT\_COLOR is shown in [Figure 12-836](#) and described in [Table 12-1020](#).

Return to [Summary Table](#).

The control register configures the default solid background color LSB[31:0]. Shadow register

**Table 12-1019. DSS0\_OVR\_DEFAULT\_COLOR  
Instances**

Instance	Physical Address
DSS0_OVR1	04A7 0008h
DSS0_OVR2	04A9 0008h
DSS0_OVR3	04AB 0008h
DSS0_OVR4	04AD 0008h

**Figure 12-836. DSS0\_OVR\_DEFAULT\_COLOR Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEFAULTCOLOR																															
R/W-0h																															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1020. DSS0\_OVR\_DEFAULT\_COLOR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	DEFAULTCOLOR	R/W	0h	32-bit LSB of ARGB background color

#### 12.6.4.11.13.3.4 DSS0\_OVR\_DEFAULT\_COLOR2 Register (Offset = 0Ch) [reset = 0h]

DSS0\_OVR\_DEFAULT\_COLOR2 is shown in [Figure 12-837](#) and described in [Table 12-1022](#).

Return to [Summary Table](#).

The control register configures the default solid background color MSB[47:32]. Shadow register

**Table 12-1021. DSS0\_OVR\_DEFAULT\_COLOR2  
Instances**

Instance	Physical Address
DSS0_OVR1	04A7 000Ch
DSS0_OVR2	04A9 000Ch
DSS0_OVR3	04AB 000Ch
DSS0_OVR4	04AD 000Ch

**Figure 12-837. DSS0\_OVR\_DEFAULT\_COLOR2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																DEFAULTCOLOR															
R-0h																R/W-0h															

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1022. DSS0\_OVR\_DEFAULT\_COLOR2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	
15-0	DEFAULTCOLOR	R/W	0h	16-bit MSB of ARGB background color

### 12.6.4.11.13.3.5 DSS0\_OVR\_TRANS\_COLOR\_MAX Register (Offset = 10h) [reset = 0h]

DSS0\_OVR\_TRANS\_COLOR\_MAX is shown in [Figure 12-838](#) and described in [Table 12-1024](#).

Return to [Summary Table](#).

The register sets the max transparency color value for the overlays. Shadow register

**Table 12-1023. DSS0\_OVR\_TRANS\_COLOR\_MAX Instances**

Instance	Physical Address
DSS0_OVR1	04A7 0010h
DSS0_OVR2	04A9 0010h
DSS0_OVR3	04AB 0010h
DSS0_OVR4	04AD 0010h

**Figure 12-838. DSS0\_OVR\_TRANS\_COLOR\_MAX Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRANSCOLORKEY																															
R/W-0h																															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1024. DSS0\_OVR\_TRANS\_COLOR\_MAX Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	TRANSCOLORKEY	R/W	0h	LSB[31:0]. Transparency Color Key Value in 36-bit RGB format

### 12.6.4.11.13.3.6 DSS0\_OVR\_TRANS\_COLOR\_MAX2 Register (Offset = 14h) [reset = 0h]

DSS0\_OVR\_TRANS\_COLOR\_MAX2 is shown in [Figure 12-839](#) and described in [Table 12-1026](#).

Return to [Summary Table](#).

The register sets the max transparency color value for the overlays. Shadow register

**Table 12-1025. DSS0\_OVR\_TRANS\_COLOR\_MAX2 Instances**

Instance	Physical Address
DSS0_OVR1	04A7 0014h
DSS0_OVR2	04A9 0014h
DSS0_OVR3	04AB 0014h
DSS0_OVR4	04AD 0014h

**Figure 12-839. DSS0\_OVR\_TRANS\_COLOR\_MAX2 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				TRANSCOLORKEY			
R-0h				R/W-0h			

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1026. DSS0\_OVR\_TRANS\_COLOR\_MAX2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	
3-0	TRANSCOLORKEY	R/W	0h	MSB [35:32]. Transparency Color Key Value in 36-bit RGB format

### 12.6.4.11.13.3.7 DSS0\_OVR\_TRANS\_COLOR\_MIN Register (Offset = 18h) [reset = 0h]

DSS0\_OVR\_TRANS\_COLOR\_MIN is shown in [Figure 12-840](#) and described in [Table 12-1028](#).

Return to [Summary Table](#).

The register sets the min transparency color value for the overlays. Shadow register

**Table 12-1027. DSS0\_OVR\_TRANS\_COLOR\_MIN Instances**

Instance	Physical Address
DSS0_OVR1	04A7 0018h
DSS0_OVR2	04A9 0018h
DSS0_OVR3	04AB 0018h
DSS0_OVR4	04AD 0018h

**Figure 12-840. DSS0\_OVR\_TRANS\_COLOR\_MIN Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRANSCOLORKEY																															
R/W-0h																															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1028. DSS0\_OVR\_TRANS\_COLOR\_MIN Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	TRANSCOLORKEY	R/W	0h	LSB[31:0]. Transparency Color Key Value in 36-bit RGB format



### 12.6.4.11.13.3.8 DSS0\_OVR\_TRANS\_COLOR\_MIN2 Register (Offset = 1Ch) [reset = 0h]

DSS0\_OVR\_TRANS\_COLOR\_MIN2 is shown in [Figure 12-841](#) and described in [Table 12-1030](#).

Return to [Summary Table](#).

The register sets the min transparency color value for the overlays. Shadow register

**Table 12-1029. DSS0\_OVR\_TRANS\_COLOR\_MIN2 Instances**

Instance	Physical Address
DSS0_OVR1	04A7 001Ch
DSS0_OVR2	04A9 001Ch
DSS0_OVR3	04AB 001Ch
DSS0_OVR4	04AD 001Ch

**Figure 12-841. DSS0\_OVR\_TRANS\_COLOR\_MIN2 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				TRANSCOLORKEY			
R-0h				R/W-0h			

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1030. DSS0\_OVR\_TRANS\_COLOR\_MIN2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	
3-0	TRANSCOLORKEY	R/W	0h	MSB [35:32]. Transparency Color Key Value in 36-bit RGB format

### 12.6.4.11.13.3.9 DSS0\_OVR\_ATTRIBUTES\_0 Register (Offset = 20h) [reset = X]

DSS0\_OVR\_ATTRIBUTES\_0 is shown in [Figure 12-842](#) and described in [Table 12-1032](#).

Return to [Summary Table](#).

The register configures the attributes of layer-0, ZORDER= 0, of the Overlay manager. Shadow register

**Table 12-1031. DSS0\_OVR\_ATTRIBUTES\_0  
Instances**

Instance	Physical Address
DSS0_OVR1	04A7 0020h
DSS0_OVR2	04A9 0020h
DSS0_OVR3	04AB 0020h
DSS0_OVR4	04AD 0020h

**Figure 12-842. DSS0\_OVR\_ATTRIBUTES\_0 Register**

31	30	29	28	27	26	25	24
RESERVED							
R/W-X							
23	22	21	20	19	18	17	16
RESERVED							
R/W-X							
15	14	13	12	11	10	9	8
RESERVED							
R/W-X							
7	6	5	4	3	2	1	0
RESERVED				CHANNELIN			ENABLE
R/W-X				R/W-0h			R/W-0h

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1032. DSS0\_OVR\_ATTRIBUTES\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	RESERVED	R/W	X	
4-1	CHANNELIN	R/W	0h	Input channel connected to Layer 0h = VID is connected to the layer 1h = VIDL-1 is connected to the layer 2h = VID-2 is connected to the layer 3h = VIDL-2 is connected to the layer 4h = Virtual Channel, from another dispcc instance, is connected to the layer
0	ENABLE	R/W	0h	Layer Enable

### 12.6.4.11.13.3.10 DSS0\_OVR\_ATTRIBUTES\_1 Register (Offset = 24h) [reset = X]

DSS0\_OVR\_ATTRIBUTES\_1 is shown in [Figure 12-843](#) and described in [Table 12-1034](#).

Return to [Summary Table](#).

The register configures the attributes of layer-1, ZORDER= 1, of the Overlay manager. Shadow register

**Table 12-1033. DSS0\_OVR\_ATTRIBUTES\_1  
Instances**

Instance	Physical Address
DSS0_OVR1	04A7 0024h
DSS0_OVR2	04A9 0024h
DSS0_OVR3	04AB 0024h
DSS0_OVR4	04AD 0024h

**Figure 12-843. DSS0\_OVR\_ATTRIBUTES\_1 Register**

31	30	29	28	27	26	25	24
RESERVED							
R/W-X							
23	22	21	20	19	18	17	16
RESERVED							
R/W-X							
15	14	13	12	11	10	9	8
RESERVED							
R/W-X							
7	6	5	4	3	2	1	0
RESERVED				CHANNELIN			ENABLE
R/W-X				R/W-0h			R/W-0h

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1034. DSS0\_OVR\_ATTRIBUTES\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	RESERVED	R/W	X	
4-1	CHANNELIN	R/W	0h	Input channel connected to Layer 0h = VID is connected to the layer 1h = VIDL-1 is connected to the layer 2h = VID-2 is connected to the layer 3h = VIDL-2 is connected to the layer 4h = Virtual Channel, from another dispcc instance, is connected to the layer
0	ENABLE	R/W	0h	Layer Enable

### 12.6.4.11.13.3.11 DSS0\_OVR\_ATTRIBUTES\_2 Register (Offset = 28h) [reset = X]

DSS0\_OVR\_ATTRIBUTES\_2 is shown in [Figure 12-844](#) and described in [Table 12-1036](#).

Return to [Summary Table](#).

The register configures the attributes of layer-2, ZORDER= 2, of the Overlay manager. Shadow register

**Table 12-1035. DSS0\_OVR\_ATTRIBUTES\_2  
Instances**

Instance	Physical Address
DSS0_OVR1	04A7 0028h
DSS0_OVR2	04A9 0028h
DSS0_OVR3	04AB 0028h
DSS0_OVR4	04AD 0028h

**Figure 12-844. DSS0\_OVR\_ATTRIBUTES\_2 Register**

31	30	29	28	27	26	25	24
RESERVED							
R/W-X							
23	22	21	20	19	18	17	16
RESERVED							
R/W-X							
15	14	13	12	11	10	9	8
RESERVED							
R/W-X							
7	6	5	4	3	2	1	0
RESERVED				CHANNELIN			ENABLE
R/W-X				R/W-0h			R/W-0h

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1036. DSS0\_OVR\_ATTRIBUTES\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	RESERVED	R/W	X	
4-1	CHANNELIN	R/W	0h	Input channel connected to Layer 0h = VID is connected to the layer 1h = VIDL-1 is connected to the layer 2h = VID-2 is connected to the layer 3h = VIDL-2 is connected to the layer 4h = Virtual Channel, from another dispcc instance, is connected to the layer
0	ENABLE	R/W	0h	Layer Enable

### 12.6.4.11.13.3.12 DSS0\_OVR\_ATTRIBUTES\_3 Register (Offset = 2Ch) [reset = X]

DSS0\_OVR\_ATTRIBUTES\_3 is shown in [Figure 12-845](#) and described in [Table 12-1038](#).

Return to [Summary Table](#).

The register configures the attributes of layer-3, ZORDER= 3, of the Overlay manager. Shadow register

**Table 12-1037. DSS0\_OVR\_ATTRIBUTES\_3  
Instances**

Instance	Physical Address
DSS0_OVR1	04A7 002Ch
DSS0_OVR2	04A9 002Ch
DSS0_OVR3	04AB 002Ch
DSS0_OVR4	04AD 002Ch

**Figure 12-845. DSS0\_OVR\_ATTRIBUTES\_3 Register**

31	30	29	28	27	26	25	24
RESERVED							
R/W-X							
23	22	21	20	19	18	17	16
RESERVED							
R/W-X							
15	14	13	12	11	10	9	8
RESERVED							
R/W-X							
7	6	5	4	3	2	1	0
RESERVED				CHANNELIN			ENABLE
R/W-X				R/W-0h			R/W-0h

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1038. DSS0\_OVR\_ATTRIBUTES\_3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	RESERVED	R/W	X	
4-1	CHANNELIN	R/W	0h	Input channel connected to Layer 0h = VID is connected to the layer 1h = VIDL-1 is connected to the layer 2h = VID-2 is connected to the layer 3h = VIDL-2 is connected to the layer 4h = Virtual Channel, from another dispcc instance, is connected to the layer
0	ENABLE	R/W	0h	Layer Enable

### 12.6.4.11.13.3.13 DSS0\_OVR\_ATTRIBUTES\_4 Register (Offset = 30h) [reset = X]

DSS0\_OVR\_ATTRIBUTES\_4 is shown in [Figure 12-846](#) and described in [Table 12-1040](#).

Return to [Summary Table](#).

The register configures the attributes of layer-4, ZORDER= 4, of the Overlay manager. Shadow register

**Table 12-1039. DSS0\_OVR\_ATTRIBUTES\_4  
Instances**

Instance	Physical Address
DSS0_OVR1	04A7 0030h
DSS0_OVR2	04A9 0030h
DSS0_OVR3	04AB 0030h
DSS0_OVR4	04AD 0030h

**Figure 12-846. DSS0\_OVR\_ATTRIBUTES\_4 Register**

31	30	29	28	27	26	25	24
RESERVED							
R/W-X							
23	22	21	20	19	18	17	16
RESERVED							
R/W-X							
15	14	13	12	11	10	9	8
RESERVED							
R/W-X							
7	6	5	4	3	2	1	0
RESERVED				CHANNELIN			ENABLE
R/W-X				R/W-0h			R/W-0h

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1040. DSS0\_OVR\_ATTRIBUTES\_4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	RESERVED	R/W	X	
4-1	CHANNELIN	R/W	0h	Input channel connected to Layer 0h = VID is connected to the layer 1h = VIDL-1 is connected to the layer 2h = VID-2 is connected to the layer 3h = VIDL-2 is connected to the layer 4h = Virtual Channel, from another dispcc instance, is connected to the layer
0	ENABLE	R/W	0h	Layer Enable

### 12.6.4.11.13.3.14 DSS0\_OVR\_ATTRIBUTES2\_0 Register (Offset = 34h) [reset = X]

DSS0\_OVR\_ATTRIBUTES2\_0 is shown in [Figure 12-847](#) and described in [Table 12-1042](#).

Return to [Summary Table](#).

The register configures the additional attributes of layer-0, ZORDER= 0, of the Overlay manager. Shadow register

**Table 12-1041. DSS0\_OVR\_ATTRIBUTES2\_0 Instances**

Instance	Physical Address
DSS0_OVR1	04A7 0034h
DSS0_OVR2	04A9 0034h
DSS0_OVR3	04AB 0034h
DSS0_OVR4	04AD 0034h

**Figure 12-847. DSS0\_OVR\_ATTRIBUTES2\_0 Register**

31	30	29	28	27	26	25	24
RESERVED				POSY			
R/W-X				R/W-0h			
23	22	21	20	19	18	17	16
				POSY			
				R/W-0h			
15	14	13	12	11	10	9	8
RESERVED				POSX			
R/W-X				R/W-0h			
7	6	5	4	3	2	1	0
				POSX			
				R/W-0h			

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1042. DSS0\_OVR\_ATTRIBUTES2\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	X	
29-16	POSY	R/W	0h	Y position of the layer. Encoded value [from 0 to 16383] to specify the Y position of the layer on the screen. The line at the top has the Y-position 0
15-14	RESERVED	R/W	X	
13-0	POSX	R/W	0h	X position of the layer. Encoded value [from 0 to 16383] to specify the X position of the layer on the screen. The first pixel on the left of the screen has the X-position 0

### 12.6.4.11.13.3.15 DSS0\_OVR\_ATTRIBUTES2\_1 Register (Offset = 38h) [reset = X]

DSS0\_OVR\_ATTRIBUTES2\_1 is shown in [Figure 12-848](#) and described in [Table 12-1044](#).

Return to [Summary Table](#).

The register configures the additional attributes of layer-1, ZORDER= 1, of the Overlay manager. Shadow register

**Table 12-1043. DSS0\_OVR\_ATTRIBUTES2\_1 Instances**

Instance	Physical Address
DSS0_OVR1	04A7 0038h
DSS0_OVR2	04A9 0038h
DSS0_OVR3	04AB 0038h
DSS0_OVR4	04AD 0038h

**Figure 12-848. DSS0\_OVR\_ATTRIBUTES2\_1 Register**

31	30	29	28	27	26	25	24
RESERVED				POSY			
R/W-X				R/W-0h			
23	22	21	20	19	18	17	16
				POSY			
				R/W-0h			
15	14	13	12	11	10	9	8
RESERVED				POSX			
R/W-X				R/W-0h			
7	6	5	4	3	2	1	0
				POSX			
				R/W-0h			

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1044. DSS0\_OVR\_ATTRIBUTES2\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	X	
29-16	POSY	R/W	0h	Y position of the layer. Encoded value [from 0 to 16383] to specify the Y position of the layer on the screen. The line at the top has the Y-position 0
15-14	RESERVED	R/W	X	
13-0	POSX	R/W	0h	X position of the layer. Encoded value [from 0 to 16383] to specify the X position of the layer on the screen. The first pixel on the left of the screen has the X-position 0



### 12.6.4.11.13.3.16 DSS0\_OVR\_ATTRIBUTES2\_2 Register (Offset = 3Ch) [reset = X]

DSS0\_OVR\_ATTRIBUTES2\_2 is shown in [Figure 12-849](#) and described in [Table 12-1046](#).

Return to [Summary Table](#).

The register configures the additional attributes of layer-2, ZORDER= 2, of the Overlay manager. Shadow register

**Table 12-1045. DSS0\_OVR\_ATTRIBUTES2\_2 Instances**

Instance	Physical Address
DSS0_OVR1	04A7 003Ch
DSS0_OVR2	04A9 003Ch
DSS0_OVR3	04AB 003Ch
DSS0_OVR4	04AD 003Ch

**Figure 12-849. DSS0\_OVR\_ATTRIBUTES2\_2 Register**

31	30	29	28	27	26	25	24
RESERVED				POSY			
R/W-X				R/W-0h			
23	22	21	20	19	18	17	16
				POSY			
				R/W-0h			
15	14	13	12	11	10	9	8
RESERVED				POSX			
R/W-X				R/W-0h			
7	6	5	4	3	2	1	0
				POSX			
				R/W-0h			

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1046. DSS0\_OVR\_ATTRIBUTES2\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	X	
29-16	POSY	R/W	0h	Y position of the layer. Encoded value [from 0 to 16383] to specify the Y position of the layer on the screen. The line at the top has the Y-position 0
15-14	RESERVED	R/W	X	
13-0	POSX	R/W	0h	X position of the layer. Encoded value [from 0 to 16383] to specify the X position of the layer on the screen. The first pixel on the left of the screen has the X-position 0

### 12.6.4.11.13.3.17 DSS0\_OVR\_ATTRIBUTES2\_3 Register (Offset = 40h) [reset = X]

DSS0\_OVR\_ATTRIBUTES2\_3 is shown in [Figure 12-850](#) and described in [Table 12-1048](#).

Return to [Summary Table](#).

The register configures the additional attributes of layer-3, ZORDER= 3, of the Overlay manager. Shadow register

**Table 12-1047. DSS0\_OVR\_ATTRIBUTES2\_3 Instances**

Instance	Physical Address
DSS0_OVR1	04A7 0040h
DSS0_OVR2	04A9 0040h
DSS0_OVR3	04AB 0040h
DSS0_OVR4	04AD 0040h

**Figure 12-850. DSS0\_OVR\_ATTRIBUTES2\_3 Register**

31	30	29	28	27	26	25	24
RESERVED				POSY			
R/W-X				R/W-0h			
23	22	21	20	19	18	17	16
				POSY			
				R/W-0h			
15	14	13	12	11	10	9	8
RESERVED				POSX			
R/W-X				R/W-0h			
7	6	5	4	3	2	1	0
				POSX			
				R/W-0h			

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1048. DSS0\_OVR\_ATTRIBUTES2\_3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	X	
29-16	POSY	R/W	0h	Y position of the layer. Encoded value [from 0 to 16383] to specify the Y position of the layer on the screen. The line at the top has the Y-position 0
15-14	RESERVED	R/W	X	
13-0	POSX	R/W	0h	X position of the layer. Encoded value [from 0 to 16383] to specify the X position of the layer on the screen. The first pixel on the left of the screen has the X-position 0

### 12.6.4.11.13.3.18 DSS0\_OVR\_ATTRIBUTES2\_4 Register (Offset = 44h) [reset = X]

DSS0\_OVR\_ATTRIBUTES2\_4 is shown in [Figure 12-851](#) and described in [Table 12-1050](#).

Return to [Summary Table](#).

The register configures the additional attributes of layer-4, ZORDER= 4, of the Overlay manager. Shadow register

**Table 12-1049. DSS0\_OVR\_ATTRIBUTES2\_4 Instances**

Instance	Physical Address
DSS0_OVR1	04A7 0044h
DSS0_OVR2	04A9 0044h
DSS0_OVR3	04AB 0044h
DSS0_OVR4	04AD 0044h

**Figure 12-851. DSS0\_OVR\_ATTRIBUTES2\_4 Register**

31	30	29	28	27	26	25	24
RESERVED				POSY			
R/W-X				R/W-0h			
23	22	21	20	19	18	17	16
				POSY			
				R/W-0h			
15	14	13	12	11	10	9	8
RESERVED				POSX			
R/W-X				R/W-0h			
7	6	5	4	3	2	1	0
				POSX			
				R/W-0h			

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1050. DSS0\_OVR\_ATTRIBUTES2\_4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	X	
29-16	POSY	R/W	0h	Y position of the layer. Encoded value [from 0 to 16383] to specify the Y position of the layer on the screen. The line at the top has the Y-position 0
15-14	RESERVED	R/W	X	
13-0	POSX	R/W	0h	X position of the layer. Encoded value [from 0 to 16383] to specify the X position of the layer on the screen. The first pixel on the left of the screen has the X-position 0

### 12.6.4.11.13.3.19 DSS0\_OVR\_SECURE Register (Offset = 48h) [reset = 0h]

DSS0\_OVR\_SECURE is shown in [Figure 12-852](#) and described in [Table 12-1052](#).

Return to [Summary Table](#).

Security bit settings for the sub-module

**Table 12-1051. DSS0\_OVR\_SECURE Instances**

Instance	Physical Address
DSS0_OVR1	04A7 0048h
DSS0_OVR2	04A9 0048h
DSS0_OVR3	04AB 0048h
DSS0_OVR4	04AD 0048h

**Figure 12-852. DSS0\_OVR\_SECURE Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							SECURE
R-0h							R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1052. DSS0\_OVR\_SECURE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	SECURE	R/W	0h	DSS0_OVR_SECURE bit 0h = DSS0_OVR_SECURE bit is reset 1h = DSS0_OVR_SECURE bit is set

#### 12.6.4.11.13.4 DSS\_VP Registers

Table 12-1054 lists the memory-mapped registers for the DSS\_VP. All register offset addresses not listed in Table 12-1054 should be considered as reserved locations and the register contents should not be modified.

VP Registers

**Table 12-1053. DSS\_VP Instances**

Instance	Base Address
DSS0_VP1	04A8 0000h
DSS0_VP2	04AA 0000h
DSS0_VP3	04AC 0000h
DSS0_VP4	04AE 0000h

**Table 12-1054. DSS\_VP Registers**

Offset	Acronym	Register Name	DSS0_VP1 Physical Address	DSS0_VP2 Physical Address	DSS0_VP3 Physical Address	DSS0_VP4 Physical Address
0h	<a href="#">DSS0_VP_CONFIG</a>		04A8 0000h	04AA 0000h	04AC 0000h	04AE 0000h
4h	<a href="#">DSS0_VP_CONTROL</a>		04A8 0004h	04AA 0004h	04AC 0004h	04AE 0004h
8h	<a href="#">DSS0_VP_CSC_COEF0</a>		04A8 0008h	04AA 0008h	04AC 0008h	04AE 0008h
Ch	<a href="#">DSS0_VP_CSC_COEF1</a>		04A8 000Ch	04AA 000Ch	04AC 000Ch	04AE 000Ch
10h	<a href="#">DSS0_VP_CSC_COEF2</a>		04A8 0010h	04AA 0010h	04AC 0010h	04AE 0010h
14h	<a href="#">DSS0_VP_DATA_CYCLE_0</a>		04A8 0014h	04AA 0014h	04AC 0014h	04AE 0014h
18h	<a href="#">DSS0_VP_DATA_CYCLE_1</a>		04A8 0018h	04AA 0018h	04AC 0018h	04AE 0018h
1Ch	<a href="#">DSS0_VP_DATA_CYCLE_2</a>		04A8 001Ch	04AA 001Ch	04AC 001Ch	04AE 001Ch
44h	<a href="#">DSS0_VP_LINE_NUMBER</a>		04A8 0044h	04AA 0044h	04AC 0044h	04AE 0044h
4Ch	<a href="#">DSS0_VP_POL_FREQ</a>		04A8 004Ch	04AA 004Ch	04AC 004Ch	04AE 004Ch
50h	<a href="#">DSS0_VP_SIZE_SCREEN</a>		04A8 0050h	04AA 0050h	04AC 0050h	04AE 0050h
54h	<a href="#">DSS0_VP_TIMING_H</a>		04A8 0054h	04AA 0054h	04AC 0054h	04AE 0054h
58h	<a href="#">DSS0_VP_TIMING_V</a>		04A8 0058h	04AA 0058h	04AC 0058h	04AE 0058h
5Ch	<a href="#">DSS0_VP_CSC_COEF3</a>		04A8 005Ch	04AA 005Ch	04AC 005Ch	04AE 005Ch
60h	<a href="#">DSS0_VP_CSC_COEF4</a>		04A8 0060h	04AA 0060h	04AC 0060h	04AE 0060h
64h	<a href="#">DSS0_VP_CSC_COEF5</a>		04A8 0064h	04AA 0064h	04AC 0064h	04AE 0064h
68h	<a href="#">DSS0_VP_CSC_COEF6</a>		04A8 0068h	04AA 0068h	04AC 0068h	04AE 0068h
6Ch	<a href="#">DSS0_VP_CSC_COEF7</a>		04A8 006Ch	04AA 006Ch	04AC 006Ch	04AE 006Ch
70h	<a href="#">DSS0_VP_SAFETY_ATTRIBUTES_0</a>		04A8 0070h	04AA 0070h	04AC 0070h	04AE 0070h
74h	<a href="#">DSS0_VP_SAFETY_ATTRIBUTES_1</a>		04A8 0074h	04AA 0074h	04AC 0074h	04AE 0074h
78h	<a href="#">DSS0_VP_SAFETY_ATTRIBUTES_2</a>		04A8 0078h	04AA 0078h	04AC 0078h	04AE 0078h
7Ch	<a href="#">DSS0_VP_SAFETY_ATTRIBUTES_3</a>		04A8 007Ch	04AA 007Ch	04AC 007Ch	04AE 007Ch
80h	<a href="#">DSS0_VP_SAFETY_ATTRIBUTES_4</a>		04A8 0080h	04AA 0080h	04AC 0080h	04AE 0080h
84h	<a href="#">DSS0_VP_SAFETY_ATTRIBUTES_5</a>		04A8 0084h	04AA 0084h	04AC 0084h	04AE 0084h
88h	<a href="#">DSS0_VP_SAFETY_ATTRIBUTES_6</a>		04A8 0088h	04AA 0088h	04AC 0088h	04AE 0088h
8Ch	<a href="#">DSS0_VP_SAFETY_ATTRIBUTES_7</a>		04A8 008Ch	04AA 008Ch	04AC 008Ch	04AE 008Ch
90h	<a href="#">DSS0_VP_SAFETY_CAPT_SIGNATURE_0</a>		04A8 0090h	04AA 0090h	04AC 0090h	04AE 0090h
94h	<a href="#">DSS0_VP_SAFETY_CAPT_SIGNATURE_1</a>		04A8 0094h	04AA 0094h	04AC 0094h	04AE 0094h
98h	<a href="#">DSS0_VP_SAFETY_CAPT_SIGNATURE_2</a>		04A8 0098h	04AA 0098h	04AC 0098h	04AE 0098h
9Ch	<a href="#">DSS0_VP_SAFETY_CAPT_SIGNATURE_3</a>		04A8 009Ch	04AA 009Ch	04AC 009Ch	04AE 009Ch
A0h	<a href="#">DSS0_VP_SAFETY_CAPT_SIGNATURE_4</a>		04A8 00A0h	04AA 00A0h	04AC 00A0h	04AE 00A0h
A4h	<a href="#">DSS0_VP_SAFETY_CAPT_SIGNATURE_5</a>		04A8 00A4h	04AA 00A4h	04AC 00A4h	04AE 00A4h

**Table 12-1054. DSS\_VP Registers (continued)**

Offset	Acronym	Register Name	DSS0_VP1 Physical Address	DSS0_VP2 Physical Address	DSS0_VP3 Physical Address	DSS0_VP4 Physical Address
A8h	<a href="#">DSS0_VP_SAFETY_CAPT_SIGNATURE_6</a>		04A8 00A8h	04AA 00A8h	04AC 00A8h	04AE 00A8h
ACh	<a href="#">DSS0_VP_SAFETY_CAPT_SIGNATURE_7</a>		04A8 00ACh	04AA 00ACh	04AC 00ACh	04AE 00ACh
B0h	<a href="#">DSS0_VP_SAFETY_POSITION_0</a>		04A8 00B0h	04AA 00B0h	04AC 00B0h	04AE 00B0h
B4h	<a href="#">DSS0_VP_SAFETY_POSITION_1</a>		04A8 00B4h	04AA 00B4h	04AC 00B4h	04AE 00B4h
B8h	<a href="#">DSS0_VP_SAFETY_POSITION_2</a>		04A8 00B8h	04AA 00B8h	04AC 00B8h	04AE 00B8h
BCh	<a href="#">DSS0_VP_SAFETY_POSITION_3</a>		04A8 00BCh	04AA 00BCh	04AC 00BCh	04AE 00BCh
C0h	<a href="#">DSS0_VP_SAFETY_POSITION_4</a>		04A8 00C0h	04AA 00C0h	04AC 00C0h	04AE 00C0h
C4h	<a href="#">DSS0_VP_SAFETY_POSITION_5</a>		04A8 00C4h	04AA 00C4h	04AC 00C4h	04AE 00C4h
C8h	<a href="#">DSS0_VP_SAFETY_POSITION_6</a>		04A8 00C8h	04AA 00C8h	04AC 00C8h	04AE 00C8h
CCh	<a href="#">DSS0_VP_SAFETY_POSITION_7</a>		04A8 00CCh	04AA 00CCh	04AC 00CCh	04AE 00CCh
D0h	<a href="#">DSS0_VP_SAFETY_REF_SIGNATURE_0</a>		04A8 00D0h	04AA 00D0h	04AC 00D0h	04AE 00D0h
D4h	<a href="#">DSS0_VP_SAFETY_REF_SIGNATURE_1</a>		04A8 00D4h	04AA 00D4h	04AC 00D4h	04AE 00D4h
D8h	<a href="#">DSS0_VP_SAFETY_REF_SIGNATURE_2</a>		04A8 00D8h	04AA 00D8h	04AC 00D8h	04AE 00D8h
DCh	<a href="#">DSS0_VP_SAFETY_REF_SIGNATURE_3</a>		04A8 00DCh	04AA 00DCh	04AC 00DCh	04AE 00DCh
E0h	<a href="#">DSS0_VP_SAFETY_REF_SIGNATURE_4</a>		04A8 00E0h	04AA 00E0h	04AC 00E0h	04AE 00E0h
E4h	<a href="#">DSS0_VP_SAFETY_REF_SIGNATURE_5</a>		04A8 00E4h	04AA 00E4h	04AC 00E4h	04AE 00E4h
E8h	<a href="#">DSS0_VP_SAFETY_REF_SIGNATURE_6</a>		04A8 00E8h	04AA 00E8h	04AC 00E8h	04AE 00E8h
ECh	<a href="#">DSS0_VP_SAFETY_REF_SIGNATURE_7</a>		04A8 00ECh	04AA 00ECh	04AC 00ECh	04AE 00ECh
F0h	<a href="#">DSS0_VP_SAFETY_SIZE_0</a>		04A8 00F0h	04AA 00F0h	04AC 00F0h	04AE 00F0h
F4h	<a href="#">DSS0_VP_SAFETY_SIZE_1</a>		04A8 00F4h	04AA 00F4h	04AC 00F4h	04AE 00F4h
F8h	<a href="#">DSS0_VP_SAFETY_SIZE_2</a>		04A8 00F8h	04AA 00F8h	04AC 00F8h	04AE 00F8h
FCh	<a href="#">DSS0_VP_SAFETY_SIZE_3</a>		04A8 00FCh	04AA 00FCh	04AC 00FCh	04AE 00FCh
100h	<a href="#">DSS0_VP_SAFETY_SIZE_4</a>		04A8 0100h	04AA 0100h	04AC 0100h	04AE 0100h
104h	<a href="#">DSS0_VP_SAFETY_SIZE_5</a>		04A8 0104h	04AA 0104h	04AC 0104h	04AE 0104h
108h	<a href="#">DSS0_VP_SAFETY_SIZE_6</a>		04A8 0108h	04AA 0108h	04AC 0108h	04AE 0108h
10Ch	<a href="#">DSS0_VP_SAFETY_SIZE_7</a>		04A8 010Ch	04AA 010Ch	04AC 010Ch	04AE 010Ch
110h	<a href="#">DSS0_VP_SAFETY_LFSR_SEED</a>		04A8 0110h	04AA 0110h	04AC 0110h	04AE 0110h
120h	<a href="#">DSS0_VP_GAMMA_TABLE_0</a>		04A8 0120h	04AA 0120h	04AC 0120h	04AE 0120h
124h	<a href="#">DSS0_VP_GAMMA_TABLE_1</a>		04A8 0124h	04AA 0124h	04AC 0124h	04AE 0124h
128h	<a href="#">DSS0_VP_GAMMA_TABLE_2</a>		04A8 0128h	04AA 0128h	04AC 0128h	04AE 0128h
12Ch	<a href="#">DSS0_VP_GAMMA_TABLE_3</a>		04A8 012Ch	04AA 012Ch	04AC 012Ch	04AE 012Ch
130h	<a href="#">DSS0_VP_GAMMA_TABLE_4</a>		04A8 0130h	04AA 0130h	04AC 0130h	04AE 0130h
134h	<a href="#">DSS0_VP_GAMMA_TABLE_5</a>		04A8 0134h	04AA 0134h	04AC 0134h	04AE 0134h
138h	<a href="#">DSS0_VP_GAMMA_TABLE_6</a>		04A8 0138h	04AA 0138h	04AC 0138h	04AE 0138h
13Ch	<a href="#">DSS0_VP_GAMMA_TABLE_7</a>		04A8 013Ch	04AA 013Ch	04AC 013Ch	04AE 013Ch
140h	<a href="#">DSS0_VP_GAMMA_TABLE_8</a>		04A8 0140h	04AA 0140h	04AC 0140h	04AE 0140h
144h	<a href="#">DSS0_VP_GAMMA_TABLE_9</a>		04A8 0144h	04AA 0144h	04AC 0144h	04AE 0144h
148h	<a href="#">DSS0_VP_GAMMA_TABLE_10</a>		04A8 0148h	04AA 0148h	04AC 0148h	04AE 0148h
14Ch	<a href="#">DSS0_VP_GAMMA_TABLE_11</a>		04A8 014Ch	04AA 014Ch	04AC 014Ch	04AE 014Ch
150h	<a href="#">DSS0_VP_GAMMA_TABLE_12</a>		04A8 0150h	04AA 0150h	04AC 0150h	04AE 0150h
154h	<a href="#">DSS0_VP_GAMMA_TABLE_13</a>		04A8 0154h	04AA 0154h	04AC 0154h	04AE 0154h
158h	<a href="#">DSS0_VP_GAMMA_TABLE_14</a>		04A8 0158h	04AA 0158h	04AC 0158h	04AE 0158h
15Ch	<a href="#">DSS0_VP_GAMMA_TABLE_15</a>		04A8 015Ch	04AA 015Ch	04AC 015Ch	04AE 015Ch

**Table 12-1054. DSS\_VP Registers (continued)**

Offset	Acronym	Register Name	DSS0_VP1 Physical Address	DSS0_VP2 Physical Address	DSS0_VP3 Physical Address	DSS0_VP4 Physical Address
178h	<a href="#">DSS0_VP_SECURE</a>		04A8 0178h	04AA 0178h	04AC 0178h	04AE 0178h

#### 12.6.4.11.3.4.1 DSS0\_VP\_CONFIG Register (Offset = 0h) [reset = 0h]

DSS0\_VP\_CONFIG is shown in [Figure 12-853](#) and described in [Table 12-1056](#).

Return to [Summary Table](#).

The DSS0\_VP\_CONTROL register configures the Display Controller module for the VP output. Shadow register.

**Table 12-1055. DSS0\_VP\_CONFIG Instances**

Instance	Physical Address
DSS0_VP1	04A8 0000h
DSS0_VP2	04AA 0000h
DSS0_VP3	04AC 0000h
DSS0_VP4	04AE 0000h

**Figure 12-853. DSS0\_VP\_CONFIG Register**

31	30	29	28	27	26	25	24
RESERVED					COLORCONV OS	FULLRANGE	COLORCONVE NABLE
R-0h					R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
FIDFIRST	OUTPUTMODE ENABLE	BT1120ENABL E	BT656ENABLE	RESERVED			BUFFERHAND SHAKE
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R-0h			R/W-0h
15	14	13	12	11	10	9	8
CPR	RESERVED						EXTERNALSY NCEN
R/W-0h	R-0h						R/W-0h
7	6	5	4	3	2	1	0
VSYNCGATED	HSYNCGATED	PIXELCLOCKG ATED	PIXELDATAGA TED	HDMIMODE	GAMMAENABL E	DATAENABLEG ATED	PIXELGATED
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1056. DSS0\_VP\_CONFIG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	
26	COLORCONVPOS	R/W	0h	Determines the position of the COLORCONV module 0h = CSC block is after GAMMA correction 1h = CSC block is before GAMMA correction
25	FULLRANGE	R/W	0h	Color Space Conversion full range setting 0h = Limited range selected. 1h = Full range selected.
24	COLORCONVENABLE	R/W	0h	Enable the color space conversion. The coefficients and offsets used are all programmable and controlled by CPR_COEFF_* and CPR_OFFSET_* registers 0h = Disable Color Space Conversion RGB to YUV 1h = Enable Color Space Conversion RGB to YUV



**Table 12-1056. DSS0\_VP\_CONFIG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
23	FIDFIRST	R/W	0h	Selects the first field to output in case of interlace mode. In case of progressive mode, the value is not used  0h = First field is even. 1h = Odd field is first.
22	OUTPUTMODEENABLE	R/W	0h	Selects between progressive and interlace mode for the VP output  0h = Progressive mode selected. 1h = Interlace mode selected.
21	BT1120ENABLE	R/W	0h	Selects BT-1120 format on the VP output. It is not possible to enable BT656 and BT1120 at the same time one the same LCD output  0h = BT-1120 is disabled. 1h = BT-1120 is enabled.
20	BT656ENABLE	R/W	0h	Selects BT-656 format on the VP output. It is not possible to enable BT656 and BT1120 at the same time one the same LCD output  0h = BT-656 is disabled. 1h = BT-656 is enabled.
19-17	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
16	BUFFERHANDSHAKE	R/W	0h	Deprecated. Always write 0
15	CPR	R/W	0h	Deprecated. Always write 0
14-9	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
8	EXTERNALSYNCEN	R/W	0h	Deprecated. Always write 0
7	VSYNCGATED	R/W	0h	VSYNC Gated Enabled [VP output]. Shadow bit-field  0h = VSYNC Gated Disabled 1h = VSYNC Gated Enabled
6	HSYNCGATED	R/W	0h	HSYNC Gated Enabled [VP output]. Shadow bit-field  0h = HSYNC Gated Disabled 1h = HSYNC Gated Enabled
5	PIXELCLOCKGATED	R/W	0h	Pixel Clock Gated Enabled [VP output]. Shadow bit-field  0h = Pixel Clock Gated Disabled 1h = Pixel Clock Gated Enabled

**Table 12-1056. DSS0\_VP\_CONFIG Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	PIXELDATAGATED	R/W	0h	Pixel Data Gated Enabled [VP output]. Shadow bit-field  0h = Pixel Data Gated Disabled 1h = Pixel Data Gated Enabled
3	HDMIMODE	R/W	0h	Deprecated. Always write 0
2	GAMMAENABLE	R/W	0h	Enable the gamma Shadow bit-field  0h = Gamma disabled 1h = Gamma enabled
1	DATAENABLEGATED	R/W	0h	DE Gated Enable Shadow bit-field  0h = DE signal is not gated 1h = DE signal is gated.
0	PIXELGATED	R/W	0h	Pixel Gated Enable. Shadow bit-field  0h = Pixel clock always toggles - only in TFT mode 1h = Pixel clock only toggles when there is valid data to display -only in TFT mode

#### 12.6.4.11.13.4.2 DSS0\_VP\_CONTROL Register (Offset = 4h) [reset = 40h]

DSS0\_VP\_CONTROL is shown in [Figure 12-854](#) and described in [Table 12-1058](#).

Return to [Summary Table](#).

The DSS0\_VP\_CONTROL register configures the Display Controller module for the VP output

**Table 12-1057. DSS0\_VP\_CONTROL Instances**

Instance	Physical Address
DSS0_VP1	04A8 0004h
DSS0_VP2	04AA 0004h
DSS0_VP3	04AC 0004h
DSS0_VP4	04AE 0004h

**Figure 12-854. DSS0\_VP\_CONTROL Register**

31	30	29	28	27	26	25	24
SPATIALTEMPORALDITHERING FRAMES			RESERVED			TDMUNUSEDBITS	TDMCYCLEFO RMA
R/W-0h			R-0h			R/W-0h	
23	22	21	20	19	18	17	16
TDMCYCLEFO RMA	TDMPARALLELMODE		TDMENABLE	RESERVED			HT
R/W-0h		R/W-0h		R/W-0h		R-0h	
15	14	13	12	11	10	9	8
HT		RESERVED	STALLMODE YPE	STALLMODE	DATA		
R/W-0h		R-0h		R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
STDITHERENA BLE	DPIENABLE	GOBIT	M8B	STN	MONOCOLOR	VPPROGLINEN UMBERMODUL O	ENABLE
R/W-0h		R/W-1h		R/W-0h		R/W-0h	

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1058. DSS0\_VP\_CONTROL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	SPATIALTEMPORALDITH ERINGFRAMES	R/W	0h	Spatial/Temporal dithering number of frames for the VP output Shadow bit-field  0h = Spatial only  1h = Spatial and temporal over 2 frames  2h = Spatial and temporal over 4 frames  3h = Reserved
29-27	RESERVED	R	0h	Write 0's for future compatibility Reads return 0

**Table 12-1058. DSS0\_VP\_CONTROL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
26-25	TDMUNUSEDBITS	R/W	0h	State of unused bits [TDM mode only] for the VP output Shadow bit-field  0h = low level  1h = high level  2h = unchanged from previous state  3h = reserved
24-23	TDMCYCLEFORMAT	R/W	0h	Cycle format [TDM mode only] for the VP output Shadow bit-field  0h = 1 cycle for 1 pixel  1h = 2 cycles for 1 pixel  2h = 3 cycles for 1 pixel  3h = 3 cycles for 2 pixels
22-21	TDMPARALLELMODE	R/W	0h	Output Interface width [TDM mode only] for the VP output Shadow bit-field  0h = 8-bit parallel output interface selected  1h = 9-bit parallel output interface selected  2h = 12-bit parallel output interface selected  3h = 16-bit parallel output interface selected
20	TDMENABLE	R/W	0h	Enable the multiple cycle format for the VP output Shadow bit-field  0h = TDM disabled  1h = TDM enabled
19-17	RESERVED	R	0h	
16-14	HT	R/W	0h	Hold Time for output. Shadow bit-field. Encoded value [from 1 to 8] to specify the number of external digital clock periods to hold the data [programmed value = value minus one]
13	RESERVED	R	0h	
12	STALLMODETYPE	R/W	0h	The type of transfer in STALLMODE - If STALLMODE is enabled  0h = Command Mode over STALL interface  1h = Video Mode over STALL interface
11	STALLMODE	R/W	0h	Enable the STALLMODE on DPI output  0h = STALL on DPI output is disabled  1h = STALL on DPI output is enabled

**Table 12-1058. DSS0\_VP\_CONTROL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
10-8	DATALINES	R/W	0h	Width of the data bus on VP output Shadow bit-field 0h = 12-bit output aligned on the LSB of the pixel data interface 1h = 16-bit output aligned on the LSB of the pixel data interface 2h = 18-bit output aligned on the LSB of the pixel data interface 3h = 24-bit output aligned on the LSB of the pixel data interface 4h = 30-bit output aligned on the LSB of the pixel data interface 5h = 36-bit output aligned on the LSB of the pixel data interface
7	STDITHERENABLE	R/W	0h	Spatial Temporal dithering enable for the VP output Shadow bit-field 0h = Spatial/Temporal dithering logic disabled 1h = Spatial/Temporal dithering logic enabled
6	DPIENABLE	R/W	1h	Enable the DPI output. wr:immediate 0h = DPI output disabled 1h = DPI output enabled
5	GOBIT	R/W	0h	GO Command for the VP output. It is used to synchronize the pipelines associated with the VP output wr:immediate 0h = The hardware has finished the synchronization 1h = Software has requested for synchronization after register updates and the hardware has not finished the synchronization
4	M8B	R/W	0h	Deprecated. Always write 0
3	STN	R/W	0h	Deprecated. Always write 0
2	MONOCOLOR	R/W	0h	Deprecated. Always write 0
1	VPPROGLINENUMBERMODULO	R/W	0h	Enable the modulo of the line number interrupt generation 0h = Disable modulo 1h = Enable Modulo
0	ENABLE	R/W	0h	Enable the video port output. wr:immediate 0h = LCD output disabled-at the end of the frame when the bit is reset 1h = LCD output enabled

### 12.6.4.11.13.4.3 DSS0\_VP\_CSC\_COEF0 Register (Offset = 8h) [reset = 0h]

DSS0\_VP\_CSC\_COEF0 is shown in [Figure 12-855](#) and described in [Table 12-1060](#).

Return to [Summary Table](#).

The register configures the color space conversion matrix coefficients. Shadow register

**Table 12-1059. DSS0\_VP\_CSC\_COEF0 Instances**

Instance	Physical Address
DSS0_VP1	04A8 0008h
DSS0_VP2	04AA 0008h
DSS0_VP3	04AC 0008h
DSS0_VP4	04AE 0008h

**Figure 12-855. DSS0\_VP\_CSC\_COEF0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED					C01										
R-0h					R/W-0h										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED					C00										
R-0h					R/W-0h										

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1060. DSS0\_VP\_CSC\_COEF0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Write 0's for future compatibility. Reads return 0
26-16	C01	R/W	0h	C01 Coefficient. Encoded signed value [from -1024 to 1023]
15-11	RESERVED	R	0h	Write 0's for future compatibility. Reads return 0
10-0	C00	R/W	0h	C00 Coefficient. Encoded signed value [from -1024 to 1023]

#### 12.6.4.11.13.4.4 DSS0\_VP\_CSC\_COEF1 Register (Offset = Ch) [reset = 0h]

DSS0\_VP\_CSC\_COEF1 is shown in [Figure 12-856](#) and described in [Table 12-1062](#).

Return to [Summary Table](#).

The register configures the color space conversion matrix coefficients. Shadow register

**Table 12-1061. DSS0\_VP\_CSC\_COEF1 Instances**

Instance	Physical Address
DSS0_VP1	04A8 000Ch
DSS0_VP2	04AA 000Ch
DSS0_VP3	04AC 000Ch
DSS0_VP4	04AE 000Ch

**Figure 12-856. DSS0\_VP\_CSC\_COEF1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED					C10										
R-0h					R/W-0h										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED					C02										
R-0h					R/W-0h										

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1062. DSS0\_VP\_CSC\_COEF1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Write 0's for future compatibility. Reads return 0
26-16	C10	R/W	0h	C10 Coefficient. Encoded signed value [from -1024 to 1023]
15-11	RESERVED	R	0h	Write 0's for future compatibility. Reads return 0
10-0	C02	R/W	0h	C02 Coefficient. Encoded signed value [from -1024 to 1023]

#### 12.6.4.11.13.4.5 DSS0\_VP\_CSC\_COEF2 Register (Offset = 10h) [reset = 0h]

DSS0\_VP\_CSC\_COEF2 is shown in [Figure 12-857](#) and described in [Table 12-1064](#).

Return to [Summary Table](#).

The register configures the color space conversion matrix coefficients. Shadow register

**Table 12-1063. DSS0\_VP\_CSC\_COEF2 Instances**

Instance	Physical Address
DSS0_VP1	04A8 0010h
DSS0_VP2	04AA 0010h
DSS0_VP3	04AC 0010h
DSS0_VP4	04AE 0010h

**Figure 12-857. DSS0\_VP\_CSC\_COEF2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED					C12										
R-0h					R/W-0h										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED					C11										
R-0h					R/W-0h										

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1064. DSS0\_VP\_CSC\_COEF2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
26-16	C12	R/W	0h	C12 Coefficient. Encoded signed value [from -1024 to 1023]
15-11	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
10-0	C11	R/W	0h	C11 Coefficient. Encoded signed value [from -1024 to 1023]



#### 12.6.4.11.13.4.6 DSS0\_VP\_DATA\_CYCLE\_0 Register (Offset = 14h) [reset = 0h]

DSS0\_VP\_DATA\_CYCLE\_0 is shown in [Figure 12-858](#) and described in [Table 12-1066](#).

Return to [Summary Table](#).

The DSS0\_VP\_CONTROL register configures the output data format over up to 3 cycles. Shadow register

**Table 12-1065. DSS0\_VP\_DATA\_CYCLE\_0 Instances**

Instance	Physical Address
DSS0_VP1	04A8 0014h
DSS0_VP2	04AA 0014h
DSS0_VP3	04AC 0014h
DSS0_VP4	04AE 0014h

**Figure 12-858. DSS0\_VP\_DATA\_CYCLE\_0 Register**

31	30	29	28	27	26	25	24
RESERVED				BITALIGNMENTPIXEL2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
RESERVED				NBBITSPIXEL2			
R-0h				R/W-0h			
15	14	13	12	11	10	9	8
RESERVED				BITALIGNMENTPIXEL1			
R-0h				R/W-0h			
7	6	5	4	3	2	1	0
RESERVED				NBBITSPIXEL1			
R-0h				R/W-0h			

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1066. DSS0\_VP\_DATA\_CYCLE\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
27-24	BITALIGNMENTPIXEL2	R/W	0h	Bit alignment Alignment of the bits from pixel 2 on the output interface
23-21	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
20-16	NBBITSPIXEL2	R/W	0h	Number of bits Number of bits from the pixel 2 [value from 0 to 16 bits]. The values from 17 to 31 are invalid
15-12	RESERVED	R	0h	Write 0's for future compatibility. Reads return 0
11-8	BITALIGNMENTPIXEL1	R/W	0h	Bit alignment Alignment of the bits from pixel 1 on the output interface
7-5	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
4-0	NBBITSPIXEL1	R/W	0h	Number of bits Number of bits from the pixel 1 [value from 0 to 16 bits]. The values from 17 to 31 are invalid

#### 12.6.4.11.13.4.7 DSS0\_VP\_DATA\_CYCLE\_1 Register (Offset = 18h) [reset = 0h]

DSS0\_VP\_DATA\_CYCLE\_1 is shown in [Figure 12-859](#) and described in [Table 12-1068](#).

Return to [Summary Table](#).

The DSS0\_VP\_CONTROL register configures the output data format over up to 3 cycles. Shadow register

**Table 12-1067. DSS0\_VP\_DATA\_CYCLE\_1 Instances**

Instance	Physical Address
DSS0_VP1	04A8 0018h
DSS0_VP2	04AA 0018h
DSS0_VP3	04AC 0018h
DSS0_VP4	04AE 0018h

**Figure 12-859. DSS0\_VP\_DATA\_CYCLE\_1 Register**

31	30	29	28	27	26	25	24
RESERVED				BITALIGNMENTPIXEL2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
RESERVED				NBBITSPIXEL2			
R-0h				R/W-0h			
15	14	13	12	11	10	9	8
RESERVED				BITALIGNMENTPIXEL1			
R-0h				R/W-0h			
7	6	5	4	3	2	1	0
RESERVED				NBBITSPIXEL1			
R-0h				R/W-0h			

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1068. DSS0\_VP\_DATA\_CYCLE\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
27-24	BITALIGNMENTPIXEL2	R/W	0h	Bit alignment Alignment of the bits from pixel 2 on the output interface
23-21	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
20-16	NBBITSPIXEL2	R/W	0h	Number of bits Number of bits from the pixel 2 [value from 0 to 16 bits]. The values from 17 to 31 are invalid
15-12	RESERVED	R	0h	Write 0's for future compatibility. Reads return 0
11-8	BITALIGNMENTPIXEL1	R/W	0h	Bit alignment Alignment of the bits from pixel 1 on the output interface
7-5	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
4-0	NBBITSPIXEL1	R/W	0h	Number of bits Number of bits from the pixel 1 [value from 0 to 16 bits]. The values from 17 to 31 are invalid

#### 12.6.4.11.13.4.8 DSS0\_VP\_DATA\_CYCLE\_2 Register (Offset = 1Ch) [reset = 0h]

DSS0\_VP\_DATA\_CYCLE\_2 is shown in [Figure 12-860](#) and described in [Table 12-1070](#).

Return to [Summary Table](#).

The DSS0\_VP\_CONTROL register configures the output data format over up to 3 cycles. Shadow register

**Table 12-1069. DSS0\_VP\_DATA\_CYCLE\_2 Instances**

Instance	Physical Address
DSS0_VP1	04A8 001Ch
DSS0_VP2	04AA 001Ch
DSS0_VP3	04AC 001Ch
DSS0_VP4	04AE 001Ch

**Figure 12-860. DSS0\_VP\_DATA\_CYCLE\_2 Register**

31	30	29	28	27	26	25	24
RESERVED				BITALIGNMENTPIXEL2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
RESERVED				NBBITSPIXEL2			
R-0h				R/W-0h			
15	14	13	12	11	10	9	8
RESERVED				BITALIGNMENTPIXEL1			
R-0h				R/W-0h			
7	6	5	4	3	2	1	0
RESERVED				NBBITSPIXEL1			
R-0h				R/W-0h			

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1070. DSS0\_VP\_DATA\_CYCLE\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
27-24	BITALIGNMENTPIXEL2	R/W	0h	Bit alignment Alignment of the bits from pixel 2 on the output interface
23-21	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
20-16	NBBITSPIXEL2	R/W	0h	Number of bits Number of bits from the pixel 2 [value from 0 to 16 bits]. The values from 17 to 31 are invalid
15-12	RESERVED	R	0h	Write 0's for future compatibility. Reads return 0
11-8	BITALIGNMENTPIXEL1	R/W	0h	Bit alignment Alignment of the bits from pixel 1 on the output interface
7-5	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
4-0	NBBITSPIXEL1	R/W	0h	Number of bits Number of bits from the pixel 1 [value from 0 to 16 bits]. The values from 17 to 31 are invalid

#### 12.6.4.11.13.4.9 DSS0\_VP\_LINE\_NUMBER Register (Offset = 44h) [reset = 0h]

DSS0\_VP\_LINE\_NUMBER is shown in [Figure 12-861](#) and described in [Table 12-1072](#).

Return to [Summary Table](#).

The DSS0\_VP\_CONTROL register indicates the panel display line number for the interrupt and the DMA request. Shadow register

**Table 12-1071. DSS0\_VP\_LINE\_NUMBER Instances**

Instance	Physical Address
DSS0_VP1	04A8 0044h
DSS0_VP2	04AA 0044h
DSS0_VP3	04AC 0044h
DSS0_VP4	04AE 0044h

**Figure 12-861. DSS0\_VP\_LINE\_NUMBER Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																		LINENUMBER													
R-0h																		R/W-0h													

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1072. DSS0\_VP\_LINE\_NUMBER Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-14	RESERVED	R	0h	
13-0	LINENUMBER	R/W	0h	LCD panel line number programming LCD line number defines the line on which the programmable interrupt is generated and the DMA request occurs

#### 12.6.4.11.13.4.10 DSS0\_VP\_POL\_FREQ Register (Offset = 4Ch) [reset = 0h]

DSS0\_VP\_POL\_FREQ is shown in [Figure 12-862](#) and described in [Table 12-1074](#).

Return to [Summary Table](#).

The register configures the signal configuration. Shadow register

**Table 12-1073. DSS0\_VP\_POL\_FREQ Instances**

Instance	Physical Address
DSS0_VP1	04A8 004Ch
DSS0_VP2	04AA 004Ch
DSS0_VP3	04AC 004Ch
DSS0_VP4	04AE 004Ch

**Figure 12-862. DSS0\_VP\_POL\_FREQ Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED					ALIGN	ONOFF	RF
R-0h					R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
IEO	IPC	IHS	IVS	ACBI			
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h			
7	6	5	4	3	2	1	0
ACB							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1074. DSS0\_VP\_POL\_FREQ Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-19	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
18	ALIGN	R/W	0h	Defines the alignment between HSYNC and VSYNC assertion 0h = VSYNC and HSYNC are not aligned 1h = VSYNC and HSYNC assertions are aligned.
17	ONOFF	R/W	0h	HSYNC/VSYNC Pixel clock DSS0_VP_CONTROL On/Off 0h = HSYNC and VSYNC are driven on opposite edges of pixel clock than pixel data 1h = HSYNC and VSYNC are driven according to bit 16
16	RF	R/W	0h	Program HSYNC/VSYNC Rise or Fall 0h = HSYNC and VSYNC are driven on falling edge of pixel clock -if bit 17 set to 1 1h = HSYNC and VSYNC are driven on rising edge of pixel clock -if bit 17 set to 1

**Table 12-1074. DSS0\_VP\_POL\_FREQ Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
15	IEO	R/W	0h	Invert output enable 0h = DE is active high 1h = DE is active low
14	IPC	R/W	0h	Invert pixel clock 0h = Data is driven on the LCD data lines on the rising-edge of the pixel clock 1h = Data is driven on the LCD data lines on the falling-edge of the pixel clock
13	IHS	R/W	0h	Invert HSYNC 0h = Hsync pin is active high and inactive low 1h = Hsync pin is active low and inactive high
12	IVS	R/W	0h	Invert VSYNC 0h = Vsync pin is active high and inactive low 1h = Vsync pin is active low and inactive high
11-8	ACBI	R/W	0h	AC Bias Pin transitions per interrupt Value [from 0 to 15] used to specify the number of AC Bias pin transitions
7-0	ACB	R/W	0h	AC Bias Pin Frequency Value [from 0 to 255] used to specify the number of line clocks to count before transitioning the AC Bias pin. This pin is used to periodically invert the polarity of the power supply to prevent DC charge build-up within the display

#### 12.6.4.11.13.4.11 DSS0\_VP\_SIZE\_SCREEN Register (Offset = 50h) [reset = X]

DSS0\_VP\_SIZE\_SCREEN is shown in [Figure 12-863](#) and described in [Table 12-1076](#).

Return to [Summary Table](#).

The register configures the panel size horizontal and vertical. Shadow register. A delta value is used to indicate if the odd field has same vertical size as the even field or +/- one line.

**Table 12-1075. DSS0\_VP\_SIZE\_SCREEN Instances**

Instance	Physical Address
DSS0_VP1	04A8 0050h
DSS0_VP2	04AA 0050h
DSS0_VP3	04AC 0050h
DSS0_VP4	04AE 0050h

**Figure 12-863. DSS0\_VP\_SIZE\_SCREEN Register**

31	30	29	28	27	26	25	24
RESERVED				LPP			
R/W-X				R/W-0h			
23	22	21	20	19	18	17	16
LPP							
R/W-0h							
15	14	13	12	11	10	9	8
DELTA_LPP				PPL			
R/W-0h				R/W-0h			
7	6	5	4	3	2	1	0
PPL							
R/W-0h							

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1076. DSS0\_VP\_SIZE\_SCREEN Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	X	
29-16	LPP	R/W	0h	Lines per panel Encoded value [from 1 to 16384] to specify the number of lines per panel [program to value minus one]
15-14	DELTA_LPP	R/W	0h	Indicates the delta size value of the odd field compared to the even field  0h = Same size 1h = Odd size is even size plus 1 2h = Odd size is even size minus 1
13-0	PPL	R/W	0h	Pixels per line Encoded value [from 1 to 16384] to specify the number of pixels contains within each line on the display [program to value minus one]. In STALL mode, any value is valid In non-STALL mode, only values multiple of 8 pixels are valid

#### 12.6.4.11.13.4.12 DSS0\_VP\_TIMING\_H Register (Offset = 54h) [reset = 0h]

DSS0\_VP\_TIMING\_H is shown in [Figure 12-864](#) and described in [Table 12-1078](#).

Return to [Summary Table](#).

The register configures the timing logic for the HSYNC signal. Shadow register

**Table 12-1077. DSS0\_VP\_TIMING\_H Instances**

Instance	Physical Address
DSS0_VP1	04A8 0054h
DSS0_VP2	04AA 0054h
DSS0_VP3	04AC 0054h
DSS0_VP4	04AE 0054h

**Figure 12-864. DSS0\_VP\_TIMING\_H Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HBP												HFP												HSW							
R/W-0h												R/W-0h												R/W-0h							

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1078. DSS0\_VP\_TIMING\_H Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	HBP	R/W	0h	Horizontal Back Porch Encoded value [from 1 to 4096] to specify the number of pixel clock periods to add to the beginning of a line transmission before the first set of pixels is output to the display [program to value minus one] When in BT mode and interlaced, this field corresponds to the vertical field blanking No 2 for Even Field
19-8	HFP	R/W	0h	Horizontal front porch Encoded value [from 1 to 4096] to specify the number of pixel clock periods to add to the end of a line transmission before line clock is asserted display [program to value minus one] When in BT mode and interlaced, this field corresponds to the vertical field blanking No 1 for Even Field
7-0	HSW	R/W	0h	Horizontal synchronization pulse width Encoded value [from 1 to 256] to specify the number of pixel clock periods to pulse the line clock at the end of each line display [program to value minus one] When in BT mode, this field corresponds to the LSB 8-bits of the 12-bit horizontal blanking[BT_HBLANK [11:0]= {VSW [3:0],HSW [7:0]]}



#### 12.6.4.11.13.4.13 DSS0\_VP\_TIMING\_V Register (Offset = 58h) [reset = 0h]

DSS0\_VP\_TIMING\_V is shown in [Figure 12-865](#) and described in [Table 12-1080](#).

Return to [Summary Table](#).

The register configures the timing logic for the VSYNC signal. Shadow register

**Table 12-1079. DSS0\_VP\_TIMING\_V Instances**

Instance	Physical Address
DSS0_VP1	04A8 0058h
DSS0_VP2	04AA 0058h
DSS0_VP3	04AC 0058h
DSS0_VP4	04AE 0058h

**Figure 12-865. DSS0\_VP\_TIMING\_V Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VBP												VFP												VSW							
R/W-0h												R/W-0h												R/W-0h							

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1080. DSS0\_VP\_TIMING\_V Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	VBP	R/W	0h	Vertical back porch Encoded value [from 0 to 4095] to specify the number of line clock periods to add to the beginning of a frame When in BT mode and interlaced, this field corresponds to the vertical field blanking No 2 for Odd Field When in BT and in progressive mode, this field corresponds to the Vertical frame blanking No 2 before the first set of pixels is output to the display
19-8	VFP	R/W	0h	Vertical front porch Encoded value [from 0 to 4095] to specify the number of line clock periods to add to the end of each frame When in BT mode and interlaced, this field corresponds to the vertical field blanking No 1 for Odd Field When in BT and in progressive mode, this field corresponds to the Vertical frame blanking No 2
7-0	VSW	R/W	0h	Vertical synchronization pulse width Encoded value [from 1 to 256] to specify the number of line clock periods to pulse the frame clock [VSYNC] pin at the end of each frame after the end of frame wait [VFP] period elapses Frame clock uses as VSYNC signal in active mode When in BT mode, the lsb 4-bits of this field [VSW [3:0]] corresponds to the MSB 4-bits of the 12-bit horizontal blanking[BT_HBLANK= {VSW [3:0],HSW [7:0]]}

#### 12.6.4.11.13.4.14 DSS0\_VP\_CSC\_COEF3 Register (Offset = 5Ch) [reset = 0h]

DSS0\_VP\_CSC\_COEF3 is shown in [Figure 12-866](#) and described in [Table 12-1082](#).

Return to [Summary Table](#).

The register configures the color space conversion matrix coefficients. Shadow register

**Table 12-1081. DSS0\_VP\_CSC\_COEF3 Instances**

Instance	Physical Address
DSS0_VP1	04A8 005Ch
DSS0_VP2	04AA 005Ch
DSS0_VP3	04AC 005Ch
DSS0_VP4	04AE 005Ch

**Figure 12-866. DSS0\_VP\_CSC\_COEF3 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED					C21										
R-0h					R/W-0h										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED					C20										
R-0h					R/W-0h										

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1082. DSS0\_VP\_CSC\_COEF3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Write 0's for future compatibility. Reads return 0
26-16	C21	R/W	0h	C21 coefficient. Encoded signed value [from -1024 to 1023]
15-11	RESERVED	R	0h	Write 0's for future compatibility. Reads return 0
10-0	C20	R/W	0h	C20 coefficient. Encoded signed value [from -1024 to 1023]

#### 12.6.4.11.13.4.15 DSS0\_VP\_CSC\_COEF4 Register (Offset = 60h) [reset = 0h]

DSS0\_VP\_CSC\_COEF4 is shown in [Figure 12-867](#) and described in [Table 12-1084](#).

Return to [Summary Table](#).

The register configures the color space conversion matrix coefficients. Shadow register

**Table 12-1083. DSS0\_VP\_CSC\_COEF4 Instances**

Instance	Physical Address
DSS0_VP1	04A8 0060h
DSS0_VP2	04AA 0060h
DSS0_VP3	04AC 0060h
DSS0_VP4	04AE 0060h

**Figure 12-867. DSS0\_VP\_CSC\_COEF4 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																					C22										
R-0h																					R/W-0h										

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1084. DSS0\_VP\_CSC\_COEF4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-11	RESERVED	R	0h	Write 0's for future compatibility. Reads return 0
10-0	C22	R/W	0h	C22 Coefficient. Encoded signed value [from -1024 to 1023]

#### 12.6.4.11.13.4.16 DSS0\_VP\_CSC\_COEF5 Register (Offset = 64h) [reset = 0h]

DSS0\_VP\_CSC\_COEF5 is shown in [Figure 12-868](#) and described in [Table 12-1086](#).

Return to [Summary Table](#).

The register configures the color space conversion matrix coefficients. Shadow register

**Table 12-1085. DSS0\_VP\_CSC\_COEF5 Instances**

Instance	Physical Address
DSS0_VP1	04A8 0064h
DSS0_VP2	04AA 0064h
DSS0_VP3	04AC 0064h
DSS0_VP4	04AE 0064h

**Figure 12-868. DSS0\_VP\_CSC\_COEF5 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PREOFFSET2												RESERVED			
R/W-0h												R-0h			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PREOFFSET1												RESERVED			
R/W-0h												R-0h			

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1086. DSS0\_VP\_CSC\_COEF5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-19	PREOFFSET2	R/W	0h	Row-2 pre-offset. Encoded signed value [from -4096 to 4095]
18-16	RESERVED	R	0h	
15-3	PREOFFSET1	R/W	0h	Row1 pre-offset. Encoded signed value [from -4096 to 4095]
2-0	RESERVED	R	0h	

#### 12.6.4.11.13.4.17 DSS0\_VP\_CSC\_COEF6 Register (Offset = 68h) [reset = 0h]

DSS0\_VP\_CSC\_COEF6 is shown in [Figure 12-869](#) and described in [Table 12-1088](#).

Return to [Summary Table](#).

The register configures the color space conversion matrix coefficients. Shadow register

**Table 12-1087. DSS0\_VP\_CSC\_COEF6 Instances**

Instance	Physical Address
DSS0_VP1	04A8 0068h
DSS0_VP2	04AA 0068h
DSS0_VP3	04AC 0068h
DSS0_VP4	04AE 0068h

**Figure 12-869. DSS0\_VP\_CSC\_COEF6 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
POSTOFFSET1													RESERVED		
R/W-0h													R-0h		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PREOFFSET3													RESERVED		
R/W-0h													R-0h		

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1088. DSS0\_VP\_CSC\_COEF6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-19	POSTOFFSET1	R/W	0h	Row-1 post-offset. Encoded signed value [from -4096 to 4095]
18-16	RESERVED	R	0h	
15-3	PREOFFSET3	R/W	0h	Row-3 pre-offset. Encoded signed value [from -4096 to 4095]
2-0	RESERVED	R	0h	

#### 12.6.4.11.13.4.18 DSS0\_VP\_CSC\_COEF7 Register (Offset = 6Ch) [reset = 0h]

DSS0\_VP\_CSC\_COEF7 is shown in [Figure 12-870](#) and described in [Table 12-1090](#).

Return to [Summary Table](#).

The register configures the color space conversion matrix coefficients. Shadow register

**Table 12-1089. DSS0\_VP\_CSC\_COEF7 Instances**

Instance	Physical Address
DSS0_VP1	04A8 006Ch
DSS0_VP2	04AA 006Ch
DSS0_VP3	04AC 006Ch
DSS0_VP4	04AE 006Ch

**Figure 12-870. DSS0\_VP\_CSC\_COEF7 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
POSTOFFSET3												RESERVED			
R/W-0h												R-0h			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POSTOFFSET2												RESERVED			
R/W-0h												R-0h			

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1090. DSS0\_VP\_CSC\_COEF7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-19	POSTOFFSET3	R/W	0h	Row-3 post-offset. Encoded signed value [from -4096 to 4095]
18-16	RESERVED	R	0h	
15-3	POSTOFFSET2	R/W	0h	Row-2 post-offset. Encoded signed value [from -4096 to 4095]
2-0	RESERVED	R	0h	

#### 12.6.4.11.13.4.19 DSS0\_VP\_SAFETY\_ATTRIBUTES\_0 Register (Offset = 70h) [reset = 0h]

DSS0\_VP\_SAFETY\_ATTRIBUTES\_0 is shown in [Figure 12-871](#) and described in [Table 12-1092](#).

Return to [Summary Table](#).

The register configures the safety sub-region n. Shadow register

**Table 12-1091. DSS0\_VP\_SAFETY\_ATTRIBUTES\_0 Instances**

Instance	Physical Address
DSS0_VP1	04A8 0070h
DSS0_VP2	04AA 0070h
DSS0_VP3	04AC 0070h
DSS0_VP4	04AE 0070h

**Figure 12-871. DSS0\_VP\_SAFETY\_ATTRIBUTES\_0 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED				FRAMESKIP		THRESHOLD	
R-0h				R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
THRESHOLD					SEEDSELECT	CAPTUREMODE	ENABLE
R/W-0h					R/W-0h	R/W-0h	R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1092. DSS0\_VP\_SAFETY\_ATTRIBUTES\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-13	RESERVED	R	0h	
12-11	FRAMESKIP	R/W	0h	Indicates which frames to be skipped while doing FRAMEFREEZE or DATACHECK [Useful for interlaced displays]. 0x 0: No frames are skipped, 0x 1: Even Frames are skipped starting from second frame after ENABLE, 0x 2: Odd Frames are skipped starting from first frame after ENABLE, 0x 3: Reserved  0h = No frames are skipped 1h = Even Frames are skipped starting from second frame after ENABLE 2h = Odd Frames are skipped starting from first frame after ENABLE 3h = Reserved

**Table 12-1092. DSS0\_VP\_SAFETY\_ATTRIBUTES\_0 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
10-3	THRESHOLD	R/W	0h	Allowed maximum number of frames with the same frame signature When the freeze frame counter reaches 1 over this value [freeze_frame_thold+1], a freeze frame detection will occur Note: The freeze frame counter is cleared on reset -OR- MISR not enabled -OR- terminal count reached -OR- compare == no match
2	SEEDSELECT	R/W	0h	Initial seed selection DSS0_VP_CONTROL 0h = Initial seed is always 0xFFFF_FFFF 1h = Initial seed is defined by SAFETY_LFSR_START.SEED
1	CAPTUREMODE	R/W	0h	Mode of operation of the safety check module 0h = Frame freeze detect enabled 1h = Data correctness check enabled
0	ENABLE	R/W	0h	Safety check Enable for the region Note: Transition from 0 to 1 clears the signature register



#### 12.6.4.11.13.4.20 DSS0\_VP\_SAFETY\_ATTRIBUTES\_1 Register (Offset = 74h) [reset = 0h]

DSS0\_VP\_SAFETY\_ATTRIBUTES\_1 is shown in [Figure 12-872](#) and described in [Table 12-1094](#).

Return to [Summary Table](#).

The register configures the safety sub-region n. Shadow register

**Table 12-1093. DSS0\_VP\_SAFETY\_ATTRIBUTES\_1 Instances**

Instance	Physical Address
DSS0_VP1	04A8 0074h
DSS0_VP2	04AA 0074h
DSS0_VP3	04AC 0074h
DSS0_VP4	04AE 0074h

**Figure 12-872. DSS0\_VP\_SAFETY\_ATTRIBUTES\_1 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED				FRAMESKIP		THRESHOLD	
R-0h				R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
THRESHOLD					SEEDSELECT	CAPTUREMODE	ENABLE
R/W-0h					R/W-0h	R/W-0h	R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1094. DSS0\_VP\_SAFETY\_ATTRIBUTES\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-13	RESERVED	R	0h	
12-11	FRAMESKIP	R/W	0h	Indicates which frames to be skipped while doing FRAMEFREEZE or DATACHECK [Useful for interlaced displays]. 0x 0: No frames are skipped, 0x 1: Even Frames are skipped starting from second frame after ENABLE, 0x 2: Odd Frames are skipped starting from first frame after ENABLE, 0x 3: Reserved  0h = No frames are skipped 1h = Even Frames are skipped starting from second frame after ENABLE 2h = Odd Frames are skipped starting from first frame after ENABLE 3h = Reserved

**Table 12-1094. DSS0\_VP\_SAFETY\_ATTRIBUTES\_1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
10-3	THRESHOLD	R/W	0h	Allowed maximum number of frames with the same frame signature When the freeze frame counter reaches 1 over this value [freeze_frame_thold+1], a freeze frame detection will occur Note: The freeze frame counter is cleared on reset -OR- MISR not enabled -OR- terminal count reached -OR- compare == no match
2	SEEDSELECT	R/W	0h	Initial seed selection DSS0_VP_CONTROL 0h = Initial seed is always 0xFFFF_FFFF 1h = Initial seed is defined by SAFETY_LFSR_START.SEED
1	CAPTUREMODE	R/W	0h	Mode of operation of the safety check module 0h = Frame freeze detect enabled 1h = Data correctness check enabled
0	ENABLE	R/W	0h	Safety check Enable for the region Note: Transition from 0 to 1 clears the signature register

#### 12.6.4.11.13.4.21 DSS0\_VP\_SAFETY\_ATTRIBUTES\_2 Register (Offset = 78h) [reset = 0h]

DSS0\_VP\_SAFETY\_ATTRIBUTES\_2 is shown in [Figure 12-873](#) and described in [Table 12-1096](#).

Return to [Summary Table](#).

The register configures the safety sub-region n. Shadow register

**Table 12-1095. DSS0\_VP\_SAFETY\_ATTRIBUTES\_2 Instances**

Instance	Physical Address
DSS0_VP1	04A8 0078h
DSS0_VP2	04AA 0078h
DSS0_VP3	04AC 0078h
DSS0_VP4	04AE 0078h

**Figure 12-873. DSS0\_VP\_SAFETY\_ATTRIBUTES\_2 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED				FRAMESKIP		THRESHOLD	
R-0h				R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
THRESHOLD					SEEDSELECT	CAPTUREMODE	ENABLE
R/W-0h					R/W-0h	R/W-0h	R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1096. DSS0\_VP\_SAFETY\_ATTRIBUTES\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-13	RESERVED	R	0h	
12-11	FRAMESKIP	R/W	0h	Indicates which frames to be skipped while doing FRAMEFREEZE or DATACHECK [Useful for interlaced displays]. 0x 0: No frames are skipped, 0x 1: Even Frames are skipped starting from second frame after ENABLE, 0x 2: Odd Frames are skipped starting from first frame after ENABLE, 0x 3: Reserved  0h = No frames are skipped 1h = Even Frames are skipped starting from second frame after ENABLE 2h = Odd Frames are skipped starting from first frame after ENABLE 3h = Reserved

**Table 12-1096. DSS0\_VP\_SAFETY\_ATTRIBUTES\_2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
10-3	THRESHOLD	R/W	0h	Allowed maximum number of frames with the same frame signature When the freeze frame counter reaches 1 over this value [freeze_frame_thold+1], a freeze frame detection will occur Note: The freeze frame counter is cleared on reset -OR- MISR not enabled -OR- terminal count reached -OR- compare == no match
2	SEEDSELECT	R/W	0h	Initial seed selection DSS0_VP_CONTROL 0h = Initial seed is always 0xFFFF_FFFF 1h = Initial seed is defined by SAFETY_LFSR_START.SEED
1	CAPTUREMODE	R/W	0h	Mode of operation of the safety check module 0h = Frame freeze detect enabled 1h = Data correctness check enabled
0	ENABLE	R/W	0h	Safety check Enable for the region Note: Transition from 0 to 1 clears the signature register

#### 12.6.4.11.13.4.22 DSS0\_VP\_SAFETY\_ATTRIBUTES\_3 Register (Offset = 7Ch) [reset = 0h]

DSS0\_VP\_SAFETY\_ATTRIBUTES\_3 is shown in [Figure 12-874](#) and described in [Table 12-1098](#).

Return to [Summary Table](#).

The register configures the safety sub-region n. Shadow register

**Table 12-1097. DSS0\_VP\_SAFETY\_ATTRIBUTES\_3 Instances**

Instance	Physical Address
DSS0_VP1	04A8 007Ch
DSS0_VP2	04AA 007Ch
DSS0_VP3	04AC 007Ch
DSS0_VP4	04AE 007Ch

**Figure 12-874. DSS0\_VP\_SAFETY\_ATTRIBUTES\_3 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED				FRAMESKIP		THRESHOLD	
R-0h				R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
THRESHOLD					SEEDSELECT	CAPTUREMODE	ENABLE
R/W-0h					R/W-0h	R/W-0h	R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1098. DSS0\_VP\_SAFETY\_ATTRIBUTES\_3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-13	RESERVED	R	0h	
12-11	FRAMESKIP	R/W	0h	Indicates which frames to be skipped while doing FRAMEFREEZE or DATACHECK [Useful for interlaced displays]. 0x 0: No frames are skipped, 0x 1: Even Frames are skipped starting from second frame after ENABLE, 0x 2: Odd Frames are skipped starting from first frame after ENABLE, 0x 3: Reserved 0h = No frames are skipped 1h = Even Frames are skipped starting from second frame after ENABLE 2h = Odd Frames are skipped starting from first frame after ENABLE 3h = Reserved

**Table 12-1098. DSS0\_VP\_SAFETY\_ATTRIBUTES\_3 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
10-3	THRESHOLD	R/W	0h	Allowed maximum number of frames with the same frame signature When the freeze frame counter reaches 1 over this value [freeze_frame_thold+1], a freeze frame detection will occur Note: The freeze frame counter is cleared on reset -OR- MISR not enabled -OR- terminal count reached -OR- compare == no match
2	SEEDSELECT	R/W	0h	Initial seed selection DSS0_VP_CONTROL 0h = Initial seed is always 0xFFFF_FFFF 1h = Initial seed is defined by SAFETY_LFSR_START.SEED
1	CAPTUREMODE	R/W	0h	Mode of operation of the safety check module 0h = Frame freeze detect enabled 1h = Data correctness check enabled
0	ENABLE	R/W	0h	Safety check Enable for the region Note: Transition from 0 to 1 clears the signature register

#### 12.6.4.11.13.4.23 DSS0\_VP\_SAFETY\_ATTRIBUTES\_4 Register (Offset = 80h) [reset = 0h]

DSS0\_VP\_SAFETY\_ATTRIBUTES\_4 is shown in [Figure 12-875](#) and described in [Table 12-1100](#).

Return to [Summary Table](#).

The register configures the safety sub-region n. Shadow register

**Table 12-1099. DSS0\_VP\_SAFETY\_ATTRIBUTES\_4 Instances**

Instance	Physical Address
DSS0_VP1	04A8 0080h
DSS0_VP2	04AA 0080h
DSS0_VP3	04AC 0080h
DSS0_VP4	04AE 0080h

**Figure 12-875. DSS0\_VP\_SAFETY\_ATTRIBUTES\_4 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED				FRAMESKIP		THRESHOLD	
R-0h				R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
THRESHOLD					SEEDSELECT	CAPTUREMODE	ENABLE
R/W-0h					R/W-0h	R/W-0h	R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1100. DSS0\_VP\_SAFETY\_ATTRIBUTES\_4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-13	RESERVED	R	0h	
12-11	FRAMESKIP	R/W	0h	Indicates which frames to be skipped while doing FRAMEFREEZE or DATACHECK [Useful for interlaced displays]. 0x 0: No frames are skipped, 0x 1: Even Frames are skipped starting from second frame after ENABLE, 0x 2: Odd Frames are skipped starting from first frame after ENABLE, 0x 3: Reserved  0h = No frames are skipped 1h = Even Frames are skipped starting from second frame after ENABLE 2h = Odd Frames are skipped starting from first frame after ENABLE 3h = Reserved

**Table 12-1100. DSS0\_VP\_SAFETY\_ATTRIBUTES\_4 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
10-3	THRESHOLD	R/W	0h	Allowed maximum number of frames with the same frame signature When the freeze frame counter reaches 1 over this value [freeze_frame_thold+1], a freeze frame detection will occur Note: The freeze frame counter is cleared on reset -OR- MISR not enabled -OR- terminal count reached -OR- compare == no match
2	SEEDSELECT	R/W	0h	Initial seed selection DSS0_VP_CONTROL 0h = Initial seed is always 0xFFFF_FFFF 1h = Initial seed is defined by SAFETY_LFSR_START.SEED
1	CAPTUREMODE	R/W	0h	Mode of operation of the safety check module 0h = Frame freeze detect enabled 1h = Data correctness check enabled
0	ENABLE	R/W	0h	Safety check Enable for the region Note: Transition from 0 to 1 clears the signature register



#### 12.6.4.11.13.4.24 DSS0\_VP\_SAFETY\_ATTRIBUTES\_5 Register (Offset = 84h) [reset = 0h]

DSS0\_VP\_SAFETY\_ATTRIBUTES\_5 is shown in [Figure 12-876](#) and described in [Table 12-1102](#).

Return to [Summary Table](#).

The register configures the safety sub-region n. Shadow register

**Table 12-1101. DSS0\_VP\_SAFETY\_ATTRIBUTES\_5 Instances**

Instance	Physical Address
DSS0_VP1	04A8 0084h
DSS0_VP2	04AA 0084h
DSS0_VP3	04AC 0084h
DSS0_VP4	04AE 0084h

**Figure 12-876. DSS0\_VP\_SAFETY\_ATTRIBUTES\_5 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED				FRAMESKIP		THRESHOLD	
R-0h				R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
THRESHOLD					SEEDSELECT	CAPTUREMODE	ENABLE
R/W-0h					R/W-0h	R/W-0h	R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1102. DSS0\_VP\_SAFETY\_ATTRIBUTES\_5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-13	RESERVED	R	0h	
12-11	FRAMESKIP	R/W	0h	Indicates which frames to be skipped while doing FRAMEFREEZE or DATACHECK [Useful for interlaced displays]. 0x 0: No frames are skipped, 0x 1: Even Frames are skipped starting from second frame after ENABLE, 0x 2: Odd Frames are skipped starting from first frame after ENABLE, 0x 3: Reserved  0h = No frames are skipped 1h = Even Frames are skipped starting from second frame after ENABLE 2h = Odd Frames are skipped starting from first frame after ENABLE 3h = Reserved

**Table 12-1102. DSS0\_VP\_SAFETY\_ATTRIBUTES\_5 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
10-3	THRESHOLD	R/W	0h	Allowed maximum number of frames with the same frame signature When the freeze frame counter reaches 1 over this value [freeze_frame_thold+1], a freeze frame detection will occur Note: The freeze frame counter is cleared on reset -OR- MISR not enabled -OR- terminal count reached -OR- compare == no match
2	SEEDSELECT	R/W	0h	Initial seed selection DSS0_VP_CONTROL 0h = Initial seed is always 0xFFFF_FFFF 1h = Initial seed is defined by SAFETY_LFSR_START.SEED
1	CAPTUREMODE	R/W	0h	Mode of operation of the safety check module 0h = Frame freeze detect enabled 1h = Data correctness check enabled
0	ENABLE	R/W	0h	Safety check Enable for the region Note: Transition from 0 to 1 clears the signature register

#### 12.6.4.11.13.4.25 DSS0\_VP\_SAFETY\_ATTRIBUTES\_6 Register (Offset = 88h) [reset = 0h]

DSS0\_VP\_SAFETY\_ATTRIBUTES\_6 is shown in [Figure 12-877](#) and described in [Table 12-1104](#).

Return to [Summary Table](#).

The register configures the safety sub-region n. Shadow register

**Table 12-1103. DSS0\_VP\_SAFETY\_ATTRIBUTES\_6 Instances**

Instance	Physical Address
DSS0_VP1	04A8 0088h
DSS0_VP2	04AA 0088h
DSS0_VP3	04AC 0088h
DSS0_VP4	04AE 0088h

**Figure 12-877. DSS0\_VP\_SAFETY\_ATTRIBUTES\_6 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED				FRAMESKIP		THRESHOLD	
R-0h				R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
THRESHOLD					SEEDSELECT	CAPTUREMODE	ENABLE
R/W-0h					R/W-0h	R/W-0h	R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1104. DSS0\_VP\_SAFETY\_ATTRIBUTES\_6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-13	RESERVED	R	0h	
12-11	FRAMESKIP	R/W	0h	Indicates which frames to be skipped while doing FRAMEFREEZE or DATACHECK [Useful for interlaced displays]. 0x 0: No frames are skipped, 0x 1: Even Frames are skipped starting from second frame after ENABLE, 0x 2: Odd Frames are skipped starting from first frame after ENABLE, 0x 3: Reserved 0h = No frames are skipped 1h = Even Frames are skipped starting from second frame after ENABLE 2h = Odd Frames are skipped starting from first frame after ENABLE 3h = Reserved

**Table 12-1104. DSS0\_VP\_SAFETY\_ATTRIBUTES\_6 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
10-3	THRESHOLD	R/W	0h	Allowed maximum number of frames with the same frame signature When the freeze frame counter reaches 1 over this value [freeze_frame_thold+1], a freeze frame detection will occur Note: The freeze frame counter is cleared on reset -OR- MISR not enabled -OR- terminal count reached -OR- compare == no match
2	SEEDSELECT	R/W	0h	Initial seed selection DSS0_VP_CONTROL 0h = Initial seed is always 0xFFFF_FFFF 1h = Initial seed is defined by SAFETY_LFSR_START.SEED
1	CAPTUREMODE	R/W	0h	Mode of operation of the safety check module 0h = Frame freeze detect enabled 1h = Data correctness check enabled
0	ENABLE	R/W	0h	Safety check Enable for the region Note: Transition from 0 to 1 clears the signature register

### 12.6.4.11.13.4.26 DSS0\_VP\_SAFETY\_ATTRIBUTES\_7 Register (Offset = 8Ch) [reset = 0h]

DSS0\_VP\_SAFETY\_ATTRIBUTES\_7 is shown in [Figure 12-878](#) and described in [Table 12-1106](#).

Return to [Summary Table](#).

The register configures the safety sub-region n. Shadow register

**Table 12-1105. DSS0\_VP\_SAFETY\_ATTRIBUTES\_7 Instances**

Instance	Physical Address
DSS0_VP1	04A8 008Ch
DSS0_VP2	04AA 008Ch
DSS0_VP3	04AC 008Ch
DSS0_VP4	04AE 008Ch

**Figure 12-878. DSS0\_VP\_SAFETY\_ATTRIBUTES\_7 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED				FRAMESKIP		THRESHOLD	
R-0h				R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
THRESHOLD					SEEDSELECT	CAPTUREMODE	ENABLE
R/W-0h					R/W-0h	R/W-0h	R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1106. DSS0\_VP\_SAFETY\_ATTRIBUTES\_7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-13	RESERVED	R	0h	
12-11	FRAMESKIP	R/W	0h	Indicates which frames to be skipped while doing FRAMEFREEZE or DATACHECK [Useful for interlaced displays]. 0x 0: No frames are skipped, 0x 1: Even Frames are skipped starting from second frame after ENABLE, 0x 2: Odd Frames are skipped starting from first frame after ENABLE, 0x 3: Reserved 0h = No frames are skipped 1h = Even Frames are skipped starting from second frame after ENABLE 2h = Odd Frames are skipped starting from first frame after ENABLE 3h = Reserved

**Table 12-1106. DSS0\_VP\_SAFETY\_ATTRIBUTES\_7 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
10-3	THRESHOLD	R/W	0h	Allowed maximum number of frames with the same frame signature When the freeze frame counter reaches 1 over this value [freeze_frame_thold+1], a freeze frame detection will occur Note: The freeze frame counter is cleared on reset -OR- MISR not enabled -OR- terminal count reached -OR- compare == no match
2	SEEDSELECT	R/W	0h	Initial seed selection DSS0_VP_CONTROL 0h = Initial seed is always 0xFFFF_FFFF 1h = Initial seed is defined by SAFETY_LFSR_START.SEED
1	CAPTUREMODE	R/W	0h	Mode of operation of the safety check module 0h = Frame freeze detect enabled 1h = Data correctness check enabled
0	ENABLE	R/W	0h	Safety check Enable for the region Note: Transition from 0 to 1 clears the signature register

### 12.6.4.11.13.4.27 DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_0 Register (Offset = 90h) [reset = 0h]

DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_0 is shown in [Figure 12-879](#) and described in [Table 12-1108](#).

Return to [Summary Table](#).

The register captures the signature from the MISR of the safety sub-region n. Shadow register

**Table 12-1107.**

**DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_0 Instances**

Instance	Physical Address
DSS0_VP1	04A8 0090h
DSS0_VP2	04AA 0090h
DSS0_VP3	04AC 0090h
DSS0_VP4	04AE 0090h

**Figure 12-879. DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGNATURE																															
R-0h																															

LEGEND: R = Read Only; -n = value after reset

**Table 12-1108. DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SIGNATURE	R	0h	The register configures the reference signature of the safety sub-region n Shadow register

#### 12.6.4.11.13.4.28 DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_1 Register (Offset = 94h) [reset = 0h]

DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_1 is shown in [Figure 12-880](#) and described in [Table 12-1110](#).

Return to [Summary Table](#).

The register captures the signature from the MISR of the safety sub-region n. Shadow register

**Table 12-1109.**

**DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_1 Instances**

Instance	Physical Address
DSS0_VP1	04A8 0094h
DSS0_VP2	04AA 0094h
DSS0_VP3	04AC 0094h
DSS0_VP4	04AE 0094h

**Figure 12-880. DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGNATURE																															
R-0h																															

LEGEND: R = Read Only; -n = value after reset

**Table 12-1110. DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SIGNATURE	R	0h	The register configures the reference signature of the safety sub-region n Shadow register



#### 12.6.4.11.13.4.29 DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_2 Register (Offset = 98h) [reset = 0h]

DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_2 is shown in [Figure 12-881](#) and described in [Table 12-1112](#).

Return to [Summary Table](#).

The register captures the signature from the MISR of the safety sub-region n. Shadow register

**Table 12-1111.**

**DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_2 Instances**

Instance	Physical Address
DSS0_VP1	04A8 0098h
DSS0_VP2	04AA 0098h
DSS0_VP3	04AC 0098h
DSS0_VP4	04AE 0098h

**Figure 12-881. DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGNATURE																															
R-0h																															

LEGEND: R = Read Only; -n = value after reset

**Table 12-1112. DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SIGNATURE	R	0h	The register configures the reference signature of the safety sub-region n Shadow register

#### 12.6.4.11.13.4.30 DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_3 Register (Offset = 9Ch) [reset = 0h]

DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_3 is shown in [Figure 12-882](#) and described in [Table 12-1114](#).

Return to [Summary Table](#).

The register captures the signature from the MISR of the safety sub-region n. Shadow register

**Table 12-1113.**

**DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_3 Instances**

Instance	Physical Address
DSS0_VP1	04A8 009Ch
DSS0_VP2	04AA 009Ch
DSS0_VP3	04AC 009Ch
DSS0_VP4	04AE 009Ch

**Figure 12-882. DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_3 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGNATURE																															
R-0h																															

LEGEND: R = Read Only; -n = value after reset

**Table 12-1114. DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SIGNATURE	R	0h	The register configures the reference signature of the safety sub-region n Shadow register

#### 12.6.4.11.13.4.31 DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_4 Register (Offset = A0h) [reset = 0h]

DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_4 is shown in [Figure 12-883](#) and described in [Table 12-1116](#).

Return to [Summary Table](#).

The register captures the signature from the MISR of the safety sub-region n. Shadow register

**Table 12-1115.**

**DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_4 Instances**

Instance	Physical Address
DSS0_VP1	04A8 00A0h
DSS0_VP2	04AA 00A0h
DSS0_VP3	04AC 00A0h
DSS0_VP4	04AE 00A0h

**Figure 12-883. DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_4 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGNATURE																															
R-0h																															

LEGEND: R = Read Only; -n = value after reset

**Table 12-1116. DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SIGNATURE	R	0h	The register configures the reference signature of the safety sub-region n Shadow register

#### 12.6.4.11.13.4.32 DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_5 Register (Offset = A4h) [reset = 0h]

DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_5 is shown in [Figure 12-884](#) and described in [Table 12-1118](#).

Return to [Summary Table](#).

The register captures the signature from the MISR of the safety sub-region n. Shadow register

**Table 12-1117.**

**DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_5 Instances**

Instance	Physical Address
DSS0_VP1	04A8 00A4h
DSS0_VP2	04AA 00A4h
DSS0_VP3	04AC 00A4h
DSS0_VP4	04AE 00A4h

**Figure 12-884. DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_5 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGNATURE																															
R-0h																															

LEGEND: R = Read Only; -n = value after reset

**Table 12-1118. DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SIGNATURE	R	0h	The register configures the reference signature of the safety sub-region n Shadow register

### 12.6.4.11.13.4.33 DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_6 Register (Offset = A8h) [reset = 0h]

DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_6 is shown in [Figure 12-885](#) and described in [Table 12-1120](#).

Return to [Summary Table](#).

The register captures the signature from the MISR of the safety sub-region n. Shadow register

**Table 12-1119.**

**DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_6 Instances**

Instance	Physical Address
DSS0_VP1	04A8 00A8h
DSS0_VP2	04AA 00A8h
DSS0_VP3	04AC 00A8h
DSS0_VP4	04AE 00A8h

**Figure 12-885. DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_6 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGNATURE																															
R-0h																															

LEGEND: R = Read Only; -n = value after reset

**Table 12-1120. DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SIGNATURE	R	0h	The register configures the reference signature of the safety sub-region n Shadow register

#### 12.6.4.11.13.4.34 DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_7 Register (Offset = ACh) [reset = 0h]

DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_7 is shown in [Figure 12-886](#) and described in [Table 12-1122](#).

Return to [Summary Table](#).

The register captures the signature from the MISR of the safety sub-region n. Shadow register

**Table 12-1121.**

**DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_7 Instances**

Instance	Physical Address
DSS0_VP1	04A8 00ACh
DSS0_VP2	04AA 00ACh
DSS0_VP3	04AC 00ACh
DSS0_VP4	04AE 00ACh

**Figure 12-886. DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_7 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGNATURE																															
R-0h																															

LEGEND: R = Read Only; -n = value after reset

**Table 12-1122. DSS0\_VP\_SAFETY\_CAPT\_SIGNATURE\_7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SIGNATURE	R	0h	The register configures the reference signature of the safety sub-region n Shadow register

#### 12.6.4.11.13.4.35 DSS0\_VP\_SAFETY\_POSITION\_0 Register (Offset = B0h) [reset = X]

DSS0\_VP\_SAFETY\_POSITION\_0 is shown in [Figure 12-887](#) and described in [Table 12-1124](#).

Return to [Summary Table](#).

The register configures the position of the safety sub-region n. Shadow register

**Table 12-1123. DSS0\_VP\_SAFETY\_POSITION\_0  
Instances**

Instance	Physical Address
DSS0_VP1	04A8 00B0h
DSS0_VP2	04AA 00B0h
DSS0_VP3	04AC 00B0h
DSS0_VP4	04AE 00B0h

**Figure 12-887. DSS0\_VP\_SAFETY\_POSITION\_0 Register**

31	30	29	28	27	26	25	24
RESERVED				POSY			
R/W-X				R/W-0h			
23	22	21	20	19	18	17	16
				POSY			
				R/W-0h			
15	14	13	12	11	10	9	8
RESERVED				POSX			
R/W-X				R/W-0h			
7	6	5	4	3	2	1	0
				POSX			
				R/W-0h			

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1124. DSS0\_VP\_SAFETY\_POSITION\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	X	
29-16	POSY	R/W	0h	Y position of the safety sub-region n. Encoded value [from 0 to 16383] to specify the Y position of the sub-region on the screen The first line on the top of the screen has the Y-position 0
15-14	RESERVED	R/W	X	
13-0	POSX	R/W	0h	X position of the safety sub-region n. Encoded value [from 0 to 16383] to specify the X position of the sub-region on the screen The first pixel on the left of the screen has the X-position 0

### 12.6.4.11.13.4.36 DSS0\_VP\_SAFETY\_POSITION\_1 Register (Offset = B4h) [reset = X]

DSS0\_VP\_SAFETY\_POSITION\_1 is shown in [Figure 12-888](#) and described in [Table 12-1126](#).

Return to [Summary Table](#).

The register configures the position of the safety sub-region n. Shadow register

**Table 12-1125. DSS0\_VP\_SAFETY\_POSITION\_1 Instances**

Instance	Physical Address
DSS0_VP1	04A8 00B4h
DSS0_VP2	04AA 00B4h
DSS0_VP3	04AC 00B4h
DSS0_VP4	04AE 00B4h

**Figure 12-888. DSS0\_VP\_SAFETY\_POSITION\_1 Register**

31	30	29	28	27	26	25	24
RESERVED				POSY			
R/W-X				R/W-0h			
23	22	21	20	19	18	17	16
				POSY			
				R/W-0h			
15	14	13	12	11	10	9	8
RESERVED				POSX			
R/W-X				R/W-0h			
7	6	5	4	3	2	1	0
				POSX			
				R/W-0h			

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1126. DSS0\_VP\_SAFETY\_POSITION\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	X	
29-16	POSY	R/W	0h	Y position of the safety sub-region n. Encoded value [from 0 to 16383] to specify the Y position of the sub-region on the screen The first line on the top of the screen has the Y-position 0
15-14	RESERVED	R/W	X	
13-0	POSX	R/W	0h	X position of the safety sub-region n. Encoded value [from 0 to 16383] to specify the X position of the sub-region on the screen The first pixel on the left of the screen has the X-position 0



### 12.6.4.11.13.4.37 DSS0\_VP\_SAFETY\_POSITION\_2 Register (Offset = B8h) [reset = X]

DSS0\_VP\_SAFETY\_POSITION\_2 is shown in [Figure 12-889](#) and described in [Table 12-1128](#).

Return to [Summary Table](#).

The register configures the position of the safety sub-region n. Shadow register

**Table 12-1127. DSS0\_VP\_SAFETY\_POSITION\_2  
Instances**

Instance	Physical Address
DSS0_VP1	04A8 00B8h
DSS0_VP2	04AA 00B8h
DSS0_VP3	04AC 00B8h
DSS0_VP4	04AE 00B8h

**Figure 12-889. DSS0\_VP\_SAFETY\_POSITION\_2 Register**

31	30	29	28	27	26	25	24
RESERVED				POSY			
R/W-X				R/W-0h			
23	22	21	20	19	18	17	16
				POSY			
				R/W-0h			
15	14	13	12	11	10	9	8
RESERVED				POSX			
R/W-X				R/W-0h			
7	6	5	4	3	2	1	0
				POSX			
				R/W-0h			

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1128. DSS0\_VP\_SAFETY\_POSITION\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	X	
29-16	POSY	R/W	0h	Y position of the safety sub-region n. Encoded value [from 0 to 16383] to specify the Y position of the sub-region on the screen The first line on the top of the screen has the Y-position 0
15-14	RESERVED	R/W	X	
13-0	POSX	R/W	0h	X position of the safety sub-region n. Encoded value [from 0 to 16383] to specify the X position of the sub-region on the screen The first pixel on the left of the screen has the X-position 0

### 12.6.4.11.13.4.38 DSS0\_VP\_SAFETY\_POSITION\_3 Register (Offset = BCh) [reset = X]

DSS0\_VP\_SAFETY\_POSITION\_3 is shown in [Figure 12-890](#) and described in [Table 12-1130](#).

Return to [Summary Table](#).

The register configures the position of the safety sub-region n. Shadow register

**Table 12-1129. DSS0\_VP\_SAFETY\_POSITION\_3  
Instances**

Instance	Physical Address
DSS0_VP1	04A8 00BCh
DSS0_VP2	04AA 00BCh
DSS0_VP3	04AC 00BCh
DSS0_VP4	04AE 00BCh

**Figure 12-890. DSS0\_VP\_SAFETY\_POSITION\_3 Register**

31	30	29	28	27	26	25	24
RESERVED				POSY			
R/W-X				R/W-0h			
23	22	21	20	19	18	17	16
POSY							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED				POSX			
R/W-X				R/W-0h			
7	6	5	4	3	2	1	0
POSX							
R/W-0h							

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1130. DSS0\_VP\_SAFETY\_POSITION\_3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	X	
29-16	POSY	R/W	0h	Y position of the safety sub-region n. Encoded value [from 0 to 16383] to specify the Y position of the sub-region on the screen The first line on the top of the screen has the Y-position 0
15-14	RESERVED	R/W	X	
13-0	POSX	R/W	0h	X position of the safety sub-region n. Encoded value [from 0 to 16383] to specify the X position of the sub-region on the screen The first pixel on the left of the screen has the X-position 0

#### 12.6.4.11.13.4.39 DSS0\_VP\_SAFETY\_POSITION\_4 Register (Offset = C0h) [reset = X]

DSS0\_VP\_SAFETY\_POSITION\_4 is shown in [Figure 12-891](#) and described in [Table 12-1132](#).

Return to [Summary Table](#).

The register configures the position of the safety sub-region n. Shadow register

**Table 12-1131. DSS0\_VP\_SAFETY\_POSITION\_4 Instances**

Instance	Physical Address
DSS0_VP1	04A8 00C0h
DSS0_VP2	04AA 00C0h
DSS0_VP3	04AC 00C0h
DSS0_VP4	04AE 00C0h

**Figure 12-891. DSS0\_VP\_SAFETY\_POSITION\_4 Register**

31	30	29	28	27	26	25	24
RESERVED				POSY			
R/W-X				R/W-0h			
23	22	21	20	19	18	17	16
				POSY			
				R/W-0h			
15	14	13	12	11	10	9	8
RESERVED				POSX			
R/W-X				R/W-0h			
7	6	5	4	3	2	1	0
				POSX			
				R/W-0h			

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1132. DSS0\_VP\_SAFETY\_POSITION\_4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	X	
29-16	POSY	R/W	0h	Y position of the safety sub-region n. Encoded value [from 0 to 16383] to specify the Y position of the sub-region on the screen The first line on the top of the screen has the Y-position 0
15-14	RESERVED	R/W	X	
13-0	POSX	R/W	0h	X position of the safety sub-region n. Encoded value [from 0 to 16383] to specify the X position of the sub-region on the screen The first pixel on the left of the screen has the X-position 0

#### 12.6.4.11.13.4.40 DSS0\_VP\_SAFETY\_POSITION\_5 Register (Offset = C4h) [reset = X]

DSS0\_VP\_SAFETY\_POSITION\_5 is shown in [Figure 12-892](#) and described in [Table 12-1134](#).

Return to [Summary Table](#).

The register configures the position of the safety sub-region n. Shadow register

**Table 12-1133. DSS0\_VP\_SAFETY\_POSITION\_5  
Instances**

Instance	Physical Address
DSS0_VP1	04A8 00C4h
DSS0_VP2	04AA 00C4h
DSS0_VP3	04AC 00C4h
DSS0_VP4	04AE 00C4h

**Figure 12-892. DSS0\_VP\_SAFETY\_POSITION\_5 Register**

31	30	29	28	27	26	25	24
RESERVED				POSY			
R/W-X				R/W-0h			
23	22	21	20	19	18	17	16
				POSY			
				R/W-0h			
15	14	13	12	11	10	9	8
RESERVED				POSX			
R/W-X				R/W-0h			
7	6	5	4	3	2	1	0
				POSX			
				R/W-0h			

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1134. DSS0\_VP\_SAFETY\_POSITION\_5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	X	
29-16	POSY	R/W	0h	Y position of the safety sub-region n. Encoded value [from 0 to 16383] to specify the Y position of the sub-region on the screen The first line on the top of the screen has the Y-position 0
15-14	RESERVED	R/W	X	
13-0	POSX	R/W	0h	X position of the safety sub-region n. Encoded value [from 0 to 16383] to specify the X position of the sub-region on the screen The first pixel on the left of the screen has the X-position 0

#### 12.6.4.11.13.4.41 DSS0\_VP\_SAFETY\_POSITION\_6 Register (Offset = C8h) [reset = X]

DSS0\_VP\_SAFETY\_POSITION\_6 is shown in [Figure 12-893](#) and described in [Table 12-1136](#).

Return to [Summary Table](#).

The register configures the position of the safety sub-region n. Shadow register

**Table 12-1135. DSS0\_VP\_SAFETY\_POSITION\_6  
Instances**

Instance	Physical Address
DSS0_VP1	04A8 00C8h
DSS0_VP2	04AA 00C8h
DSS0_VP3	04AC 00C8h
DSS0_VP4	04AE 00C8h

**Figure 12-893. DSS0\_VP\_SAFETY\_POSITION\_6 Register**

31	30	29	28	27	26	25	24
RESERVED				POSY			
R/W-X				R/W-0h			
23	22	21	20	19	18	17	16
				POSY			
				R/W-0h			
15	14	13	12	11	10	9	8
RESERVED				POSX			
R/W-X				R/W-0h			
7	6	5	4	3	2	1	0
				POSX			
				R/W-0h			

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1136. DSS0\_VP\_SAFETY\_POSITION\_6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	X	
29-16	POSY	R/W	0h	Y position of the safety sub-region n. Encoded value [from 0 to 16383] to specify the Y position of the sub-region on the screen The first line on the top of the screen has the Y-position 0
15-14	RESERVED	R/W	X	
13-0	POSX	R/W	0h	X position of the safety sub-region n. Encoded value [from 0 to 16383] to specify the X position of the sub-region on the screen The first pixel on the left of the screen has the X-position 0

#### 12.6.4.11.13.4.42 DSS0\_VP\_SAFETY\_POSITION\_7 Register (Offset = CCh) [reset = X]

DSS0\_VP\_SAFETY\_POSITION\_7 is shown in [Figure 12-894](#) and described in [Table 12-1138](#).

Return to [Summary Table](#).

The register configures the position of the safety sub-region n. Shadow register

**Table 12-1137. DSS0\_VP\_SAFETY\_POSITION\_7 Instances**

Instance	Physical Address
DSS0_VP1	04A8 00CCh
DSS0_VP2	04AA 00CCh
DSS0_VP3	04AC 00CCh
DSS0_VP4	04AE 00CCh

**Figure 12-894. DSS0\_VP\_SAFETY\_POSITION\_7 Register**

31	30	29	28	27	26	25	24
RESERVED				POSY			
R/W-X				R/W-0h			
23	22	21	20	19	18	17	16
POSY							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED				POSX			
R/W-X				R/W-0h			
7	6	5	4	3	2	1	0
POSX							
R/W-0h							

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1138. DSS0\_VP\_SAFETY\_POSITION\_7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	X	
29-16	POSY	R/W	0h	Y position of the safety sub-region n. Encoded value [from 0 to 16383] to specify the Y position of the sub-region on the screen The first line on the top of the screen has the Y-position 0
15-14	RESERVED	R/W	X	
13-0	POSX	R/W	0h	X position of the safety sub-region n. Encoded value [from 0 to 16383] to specify the X position of the sub-region on the screen The first pixel on the left of the screen has the X-position 0

#### 12.6.4.11.13.4.43 DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_0 Register (Offset = D0h) [reset = 0h]

DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_0 is shown in [Figure 12-895](#) and described in [Table 12-1140](#).

Return to [Summary Table](#).

The register configures the reference signature of the safety sub-region n. Shadow register

**Table 12-1139.**

**DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_0 Instances**

Instance	Physical Address
DSS0_VP1	04A8 00D0h
DSS0_VP2	04AA 00D0h
DSS0_VP3	04AC 00D0h
DSS0_VP4	04AE 00D0h

**Figure 12-895. DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGNATURE																															
R/W-0h																															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1140. DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SIGNATURE	R/W	0h	The register configures the reference signature of the safety sub-region n. Shadow register

#### 12.6.4.11.13.4.44 DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_1 Register (Offset = D4h) [reset = 0h]

DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_1 is shown in [Figure 12-896](#) and described in [Table 12-1142](#).

Return to [Summary Table](#).

The register configures the reference signature of the safety sub-region n. Shadow register

**Table 12-1141.**

**DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_1 Instances**

Instance	Physical Address
DSS0_VP1	04A8 00D4h
DSS0_VP2	04AA 00D4h
DSS0_VP3	04AC 00D4h
DSS0_VP4	04AE 00D4h

**Figure 12-896. DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGNATURE																															
R/W-0h																															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1142. DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SIGNATURE	R/W	0h	The register configures the reference signature of the safety sub-region n. Shadow register



#### 12.6.4.11.13.4.45 DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_2 Register (Offset = D8h) [reset = 0h]

DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_2 is shown in [Figure 12-897](#) and described in [Table 12-1144](#).

Return to [Summary Table](#).

The register configures the reference signature of the safety sub-region n. Shadow register

**Table 12-1143.**

**DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_2 Instances**

Instance	Physical Address
DSS0_VP1	04A8 00D8h
DSS0_VP2	04AA 00D8h
DSS0_VP3	04AC 00D8h
DSS0_VP4	04AE 00D8h

**Figure 12-897. DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGNATURE																															
R/W-0h																															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1144. DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SIGNATURE	R/W	0h	The register configures the reference signature of the safety sub-region n. Shadow register

#### 12.6.4.11.13.4.46 DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_3 Register (Offset = DCh) [reset = 0h]

DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_3 is shown in [Figure 12-898](#) and described in [Table 12-1146](#).

Return to [Summary Table](#).

The register configures the reference signature of the safety sub-region n. Shadow register

**Table 12-1145.**

**DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_3 Instances**

Instance	Physical Address
DSS0_VP1	04A8 00DCh
DSS0_VP2	04AA 00DCh
DSS0_VP3	04AC 00DCh
DSS0_VP4	04AE 00DCh

**Figure 12-898. DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_3 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGNATURE																															
R/W-0h																															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1146. DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SIGNATURE	R/W	0h	The register configures the reference signature of the safety sub-region n. Shadow register

#### 12.6.4.11.13.4.47 DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_4 Register (Offset = E0h) [reset = 0h]

DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_4 is shown in [Figure 12-899](#) and described in [Table 12-1148](#).

Return to [Summary Table](#).

The register configures the reference signature of the safety sub-region n. Shadow register

**Table 12-1147.**

**DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_4 Instances**

Instance	Physical Address
DSS0_VP1	04A8 00E0h
DSS0_VP2	04AA 00E0h
DSS0_VP3	04AC 00E0h
DSS0_VP4	04AE 00E0h

**Figure 12-899. DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_4 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGNATURE																															
R/W-0h																															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1148. DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SIGNATURE	R/W	0h	The register configures the reference signature of the safety sub-region n. Shadow register

#### 12.6.4.11.13.4.48 DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_5 Register (Offset = E4h) [reset = 0h]

DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_5 is shown in [Figure 12-900](#) and described in [Table 12-1150](#).

Return to [Summary Table](#).

The register configures the reference signature of the safety sub-region n. Shadow register

**Table 12-1149.**

**DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_5 Instances**

Instance	Physical Address
DSS0_VP1	04A8 00E4h
DSS0_VP2	04AA 00E4h
DSS0_VP3	04AC 00E4h
DSS0_VP4	04AE 00E4h

**Figure 12-900. DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_5 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGNATURE																															
R/W-0h																															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1150. DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SIGNATURE	R/W	0h	The register configures the reference signature of the safety sub-region n. Shadow register

#### 12.6.4.11.13.4.49 DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_6 Register (Offset = E8h) [reset = 0h]

DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_6 is shown in [Figure 12-901](#) and described in [Table 12-1152](#).

Return to [Summary Table](#).

The register configures the reference signature of the safety sub-region n. Shadow register

**Table 12-1151.**

**DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_6 Instances**

Instance	Physical Address
DSS0_VP1	04A8 00E8h
DSS0_VP2	04AA 00E8h
DSS0_VP3	04AC 00E8h
DSS0_VP4	04AE 00E8h

**Figure 12-901. DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_6 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGNATURE																															
R/W-0h																															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1152. DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SIGNATURE	R/W	0h	The register configures the reference signature of the safety sub-region n. Shadow register

#### 12.6.4.11.13.4.50 DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_7 Register (Offset = ECh) [reset = 0h]

DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_7 is shown in [Figure 12-902](#) and described in [Table 12-1154](#).

Return to [Summary Table](#).

The register configures the reference signature of the safety sub-region n. Shadow register

**Table 12-1153.**

**DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_7 Instances**

Instance	Physical Address
DSS0_VP1	04A8 00ECh
DSS0_VP2	04AA 00ECh
DSS0_VP3	04AC 00ECh
DSS0_VP4	04AE 00ECh

**Figure 12-902. DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_7 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIGNATURE																															
R/W-0h																															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1154. DSS0\_VP\_SAFETY\_REF\_SIGNATURE\_7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SIGNATURE	R/W	0h	The register configures the reference signature of the safety sub-region n. Shadow register

### 12.6.4.11.13.4.51 DSS0\_VP\_SAFETY\_SIZE\_0 Register (Offset = F0h) [reset = X]

DSS0\_VP\_SAFETY\_SIZE\_0 is shown in [Figure 12-903](#) and described in [Table 12-1156](#).

Return to [Summary Table](#).

The register configures the size of the safety sub-region n Shadow register.

**Table 12-1155. DSS0\_VP\_SAFETY\_SIZE\_0 Instances**

Instance	Physical Address
DSS0_VP1	04A8 00F0h
DSS0_VP2	04AA 00F0h
DSS0_VP3	04AC 00F0h
DSS0_VP4	04AE 00F0h

**Figure 12-903. DSS0\_VP\_SAFETY\_SIZE\_0 Register**

31	30	29	28	27	26	25	24
RESERVED				SIZEY			
R/W-X				R/W-0h			
23	22	21	20	19	18	17	16
SIZEY							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED				SIZEX			
R/W-X				R/W-0h			
7	6	5	4	3	2	1	0
SIZEX							
R/W-0h							

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1156. DSS0\_VP\_SAFETY\_SIZE\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	X	
29-16	SIZEY	R/W	0h	Height of the safety sub-region n Encoded value [from 0 to 16383] to specify the height of the sub-region on the screen One line height region has value of 0
15-14	RESERVED	R/W	X	
13-0	SIZEX	R/W	0h	Width of the safety sub-region n Encoded value [from 0 to 16383] to specify the width of the sub-region on the screen One pixel wide region has value of 0

#### 12.6.4.11.13.4.52 DSS0\_VP\_SAFETY\_SIZE\_1 Register (Offset = F4h) [reset = X]

DSS0\_VP\_SAFETY\_SIZE\_1 is shown in [Figure 12-904](#) and described in [Table 12-1158](#).

Return to [Summary Table](#).

The register configures the size of the safety sub-region n Shadow register.

**Table 12-1157. DSS0\_VP\_SAFETY\_SIZE\_1 Instances**

Instance	Physical Address
DSS0_VP1	04A8 00F4h
DSS0_VP2	04AA 00F4h
DSS0_VP3	04AC 00F4h
DSS0_VP4	04AE 00F4h

**Figure 12-904. DSS0\_VP\_SAFETY\_SIZE\_1 Register**

31	30	29	28	27	26	25	24
RESERVED				SIZEY			
R/W-X				R/W-0h			
23	22	21	20	19	18	17	16
SIZEY							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED				SIZEX			
R/W-X				R/W-0h			
7	6	5	4	3	2	1	0
SIZEX							
R/W-0h							

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1158. DSS0\_VP\_SAFETY\_SIZE\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	X	
29-16	SIZEY	R/W	0h	Height of the safety sub-region n Encoded value [from 0 to 16383] to specify the height of the sub-region on the screen One line height region has value of 0
15-14	RESERVED	R/W	X	
13-0	SIZEX	R/W	0h	Width of the safety sub-region n Encoded value [from 0 to 16383] to specify the width of the sub-region on the screen One pixel wide region has value of 0



#### 12.6.4.11.13.4.53 DSS0\_VP\_SAFETY\_SIZE\_2 Register (Offset = F8h) [reset = X]

DSS0\_VP\_SAFETY\_SIZE\_2 is shown in [Figure 12-905](#) and described in [Table 12-1160](#).

Return to [Summary Table](#).

The register configures the size of the safety sub-region n Shadow register.

**Table 12-1159. DSS0\_VP\_SAFETY\_SIZE\_2 Instances**

Instance	Physical Address
DSS0_VP1	04A8 00F8h
DSS0_VP2	04AA 00F8h
DSS0_VP3	04AC 00F8h
DSS0_VP4	04AE 00F8h

**Figure 12-905. DSS0\_VP\_SAFETY\_SIZE\_2 Register**

31	30	29	28	27	26	25	24
RESERVED				SIZEY			
R/W-X				R/W-0h			
23	22	21	20	19	18	17	16
SIZEY							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED				SIZEX			
R/W-X				R/W-0h			
7	6	5	4	3	2	1	0
SIZEX							
R/W-0h							

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1160. DSS0\_VP\_SAFETY\_SIZE\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	X	
29-16	SIZEY	R/W	0h	Height of the safety sub-region n Encoded value [from 0 to 16383] to specify the height of the sub-region on the screen One line height region has value of 0
15-14	RESERVED	R/W	X	
13-0	SIZEX	R/W	0h	Width of the safety sub-region n Encoded value [from 0 to 16383] to specify the width of the sub-region on the screen One pixel wide region has value of 0

#### 12.6.4.11.13.4.54 DSS0\_VP\_SAFETY\_SIZE\_3 Register (Offset = FCh) [reset = X]

DSS0\_VP\_SAFETY\_SIZE\_3 is shown in [Figure 12-906](#) and described in [Table 12-1162](#).

Return to [Summary Table](#).

The register configures the size of the safety sub-region n Shadow register.

**Table 12-1161. DSS0\_VP\_SAFETY\_SIZE\_3 Instances**

Instance	Physical Address
DSS0_VP1	04A8 00FCh
DSS0_VP2	04AA 00FCh
DSS0_VP3	04AC 00FCh
DSS0_VP4	04AE 00FCh

**Figure 12-906. DSS0\_VP\_SAFETY\_SIZE\_3 Register**

31	30	29	28	27	26	25	24
RESERVED				SIZEY			
R/W-X				R/W-0h			
23	22	21	20	19	18	17	16
SIZEY							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED				SIZEX			
R/W-X				R/W-0h			
7	6	5	4	3	2	1	0
SIZEX							
R/W-0h							

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1162. DSS0\_VP\_SAFETY\_SIZE\_3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	X	
29-16	SIZEY	R/W	0h	Height of the safety sub-region n Encoded value [from 0 to 16383] to specify the height of the sub-region on the screen One line height region has value of 0
15-14	RESERVED	R/W	X	
13-0	SIZEX	R/W	0h	Width of the safety sub-region n Encoded value [from 0 to 16383] to specify the width of the sub-region on the screen One pixel wide region has value of 0

#### 12.6.4.11.13.4.55 DSS0\_VP\_SAFETY\_SIZE\_4 Register (Offset = 100h) [reset = X]

DSS0\_VP\_SAFETY\_SIZE\_4 is shown in [Figure 12-907](#) and described in [Table 12-1164](#).

Return to [Summary Table](#).

The register configures the size of the safety sub-region n Shadow register.

**Table 12-1163. DSS0\_VP\_SAFETY\_SIZE\_4 Instances**

Instance	Physical Address
DSS0_VP1	04A8 0100h
DSS0_VP2	04AA 0100h
DSS0_VP3	04AC 0100h
DSS0_VP4	04AE 0100h

**Figure 12-907. DSS0\_VP\_SAFETY\_SIZE\_4 Register**

31	30	29	28	27	26	25	24
RESERVED				SIZEY			
R/W-X				R/W-0h			
23	22	21	20	19	18	17	16
SIZEY							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED				SIZEX			
R/W-X				R/W-0h			
7	6	5	4	3	2	1	0
SIZEX							
R/W-0h							

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1164. DSS0\_VP\_SAFETY\_SIZE\_4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	X	
29-16	SIZEY	R/W	0h	Height of the safety sub-region n Encoded value [from 0 to 16383] to specify the height of the sub-region on the screen One line height region has value of 0
15-14	RESERVED	R/W	X	
13-0	SIZEX	R/W	0h	Width of the safety sub-region n Encoded value [from 0 to 16383] to specify the width of the sub-region on the screen One pixel wide region has value of 0

### 12.6.4.11.13.4.56 DSS0\_VP\_SAFETY\_SIZE\_5 Register (Offset = 104h) [reset = X]

DSS0\_VP\_SAFETY\_SIZE\_5 is shown in [Figure 12-908](#) and described in [Table 12-1166](#).

Return to [Summary Table](#).

The register configures the size of the safety sub-region n Shadow register.

**Table 12-1165. DSS0\_VP\_SAFETY\_SIZE\_5 Instances**

Instance	Physical Address
DSS0_VP1	04A8 0104h
DSS0_VP2	04AA 0104h
DSS0_VP3	04AC 0104h
DSS0_VP4	04AE 0104h

**Figure 12-908. DSS0\_VP\_SAFETY\_SIZE\_5 Register**

31	30	29	28	27	26	25	24
RESERVED				SIZEY			
R/W-X				R/W-0h			
23	22	21	20	19	18	17	16
SIZEY							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED				SIZEX			
R/W-X				R/W-0h			
7	6	5	4	3	2	1	0
SIZEX							
R/W-0h							

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1166. DSS0\_VP\_SAFETY\_SIZE\_5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	X	
29-16	SIZEY	R/W	0h	Height of the safety sub-region n Encoded value [from 0 to 16383] to specify the height of the sub-region on the screen One line height region has value of 0
15-14	RESERVED	R/W	X	
13-0	SIZEX	R/W	0h	Width of the safety sub-region n Encoded value [from 0 to 16383] to specify the width of the sub-region on the screen One pixel wide region has value of 0

#### 12.6.4.11.13.4.57 DSS0\_VP\_SAFETY\_SIZE\_6 Register (Offset = 108h) [reset = X]

DSS0\_VP\_SAFETY\_SIZE\_6 is shown in [Figure 12-909](#) and described in [Table 12-1168](#).

Return to [Summary Table](#).

The register configures the size of the safety sub-region n Shadow register.

**Table 12-1167. DSS0\_VP\_SAFETY\_SIZE\_6 Instances**

Instance	Physical Address
DSS0_VP1	04A8 0108h
DSS0_VP2	04AA 0108h
DSS0_VP3	04AC 0108h
DSS0_VP4	04AE 0108h

**Figure 12-909. DSS0\_VP\_SAFETY\_SIZE\_6 Register**

31	30	29	28	27	26	25	24
RESERVED				SIZEY			
R/W-X				R/W-0h			
23	22	21	20	19	18	17	16
SIZEY							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED				SIZEX			
R/W-X				R/W-0h			
7	6	5	4	3	2	1	0
SIZEX							
R/W-0h							

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1168. DSS0\_VP\_SAFETY\_SIZE\_6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	X	
29-16	SIZEY	R/W	0h	Height of the safety sub-region n Encoded value [from 0 to 16383] to specify the height of the sub-region on the screen One line height region has value of 0
15-14	RESERVED	R/W	X	
13-0	SIZEX	R/W	0h	Width of the safety sub-region n Encoded value [from 0 to 16383] to specify the width of the sub-region on the screen One pixel wide region has value of 0

### 12.6.4.11.13.4.58 DSS0\_VP\_SAFETY\_SIZE\_7 Register (Offset = 10Ch) [reset = X]

DSS0\_VP\_SAFETY\_SIZE\_7 is shown in [Figure 12-910](#) and described in [Table 12-1170](#).

Return to [Summary Table](#).

The register configures the size of the safety sub-region n Shadow register.

**Table 12-1169. DSS0\_VP\_SAFETY\_SIZE\_7 Instances**

Instance	Physical Address
DSS0_VP1	04A8 010Ch
DSS0_VP2	04AA 010Ch
DSS0_VP3	04AC 010Ch
DSS0_VP4	04AE 010Ch

**Figure 12-910. DSS0\_VP\_SAFETY\_SIZE\_7 Register**

31	30	29	28	27	26	25	24
RESERVED				SIZEY			
R/W-X				R/W-0h			
23	22	21	20	19	18	17	16
SIZEY							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED				SIZEX			
R/W-X				R/W-0h			
7	6	5	4	3	2	1	0
SIZEX							
R/W-0h							

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1170. DSS0\_VP\_SAFETY\_SIZE\_7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	X	
29-16	SIZEY	R/W	0h	Height of the safety sub-region n Encoded value [from 0 to 16383] to specify the height of the sub-region on the screen One line height region has value of 0
15-14	RESERVED	R/W	X	
13-0	SIZEX	R/W	0h	Width of the safety sub-region n Encoded value [from 0 to 16383] to specify the width of the sub-region on the screen One pixel wide region has value of 0

#### 12.6.4.11.13.4.59 DSS0\_VP\_SAFETY\_LFSR\_SEED Register (Offset = 110h) [reset = 0h]

DSS0\_VP\_SAFETY\_LFSR\_SEED is shown in [Figure 12-911](#) and described in [Table 12-1172](#).

Return to [Summary Table](#).

The register configures the seed initial signature value of MISRs that are to be initialized with a user programmed initial value. Otherwise, the MISR is initialized with 0xFFFF\_FFFF. Shadow register.

**Table 12-1171. DSS0\_VP\_SAFETY\_LFSR\_SEED Instances**

Instance	Physical Address
DSS0_VP1	04A8 0110h
DSS0_VP2	04AA 0110h
DSS0_VP3	04AC 0110h
DSS0_VP4	04AE 0110h

**Figure 12-911. DSS0\_VP\_SAFETY\_LFSR\_SEED Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEED																															
R/W-0h																															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1172. DSS0\_VP\_SAFETY\_LFSR\_SEED Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SEED	R/W	0h	The register configures the seed [initial signature value] of MISRs that are to be initialized with a user programmed initial value Otherwise, the MISR is initialized with 0xFFFF_FFFF Shadow register

#### 12.6.4.11.13.4.60 DSS0\_VP\_GAMMA\_TABLE\_0 Register (Offset = 120h) [reset = X]

DSS0\_VP\_GAMMA\_TABLE\_0 is shown in [Figure 12-912](#) and described in [Table 12-1174](#).

Return to [Summary Table](#).

The register configures the gamma table on VP output.

**Table 12-1173. DSS0\_VP\_GAMMA\_TABLE\_0  
Instances**

Instance	Physical Address
DSS0_VP1	04A8 0120h
DSS0_VP2	04AA 0120h
DSS0_VP3	04AC 0120h
DSS0_VP4	04AE 0120h

**Figure 12-912. DSS0\_VP\_GAMMA\_TABLE\_0 Register**

31	30	29	28	27	26	25	24
INDEX	RESERVED	VALUE_R					
W-0h	W-X	W-0h					
23	22	21	20	19	18	17	16
VALUE_R				VALUE_G			
W-0h				W-0h			
15	14	13	12	11	10	9	8
VALUE_G						VALUE_B	
W-0h						W-0h	
7	6	5	4	3	2	1	0
VALUE_B							
W-0h							

LEGEND: W = Write Only; -n = value after reset

**Table 12-1174. DSS0\_VP\_GAMMA\_TABLE\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	INDEX	W	0h	Write 1 to reset the index
30	RESERVED	W	X	
29-20	VALUE_R	W	0h	10-bit R value to be stored in the gamma table
19-10	VALUE_G	W	0h	10-bit G value to be stored in the gamma table
9-0	VALUE_B	W	0h	10-bit B value to be stored in the gamma table



#### 12.6.4.11.13.4.61 DSS0\_VP\_GAMMA\_TABLE\_1 Register (Offset = 124h) [reset = X]

DSS0\_VP\_GAMMA\_TABLE\_1 is shown in [Figure 12-913](#) and described in [Table 12-1176](#).

Return to [Summary Table](#).

The register configures the gamma table on VP output.

**Table 12-1175. DSS0\_VP\_GAMMA\_TABLE\_1  
Instances**

Instance	Physical Address
DSS0_VP1	04A8 0124h
DSS0_VP2	04AA 0124h
DSS0_VP3	04AC 0124h
DSS0_VP4	04AE 0124h

**Figure 12-913. DSS0\_VP\_GAMMA\_TABLE\_1 Register**

31	30	29	28	27	26	25	24
INDEX	RESERVED	VALUE_R					
W-0h	W-X	W-0h					
23	22	21	20	19	18	17	16
VALUE_R				VALUE_G			
W-0h				W-0h			
15	14	13	12	11	10	9	8
VALUE_G						VALUE_B	
W-0h						W-0h	
7	6	5	4	3	2	1	0
VALUE_B							
W-0h							

LEGEND: W = Write Only; -n = value after reset

**Table 12-1176. DSS0\_VP\_GAMMA\_TABLE\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	INDEX	W	0h	Write 1 to reset the index
30	RESERVED	W	X	
29-20	VALUE_R	W	0h	10-bit R value to be stored in the gamma table
19-10	VALUE_G	W	0h	10-bit G value to be stored in the gamma table
9-0	VALUE_B	W	0h	10-bit B value to be stored in the gamma table

#### 12.6.4.11.13.4.62 DSS0\_VP\_GAMMA\_TABLE\_2 Register (Offset = 128h) [reset = X]

DSS0\_VP\_GAMMA\_TABLE\_2 is shown in [Figure 12-914](#) and described in [Table 12-1178](#).

Return to [Summary Table](#).

The register configures the gamma table on VP output.

**Table 12-1177. DSS0\_VP\_GAMMA\_TABLE\_2  
Instances**

Instance	Physical Address
DSS0_VP1	04A8 0128h
DSS0_VP2	04AA 0128h
DSS0_VP3	04AC 0128h
DSS0_VP4	04AE 0128h

**Figure 12-914. DSS0\_VP\_GAMMA\_TABLE\_2 Register**

31	30	29	28	27	26	25	24
INDEX	RESERVED	VALUE_R					
W-0h	W-X	W-0h					
23	22	21	20	19	18	17	16
VALUE_R				VALUE_G			
W-0h				W-0h			
15	14	13	12	11	10	9	8
VALUE_G						VALUE_B	
W-0h						W-0h	
7	6	5	4	3	2	1	0
VALUE_B							
W-0h							

LEGEND: W = Write Only; -n = value after reset

**Table 12-1178. DSS0\_VP\_GAMMA\_TABLE\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	INDEX	W	0h	Write 1 to reset the index
30	RESERVED	W	X	
29-20	VALUE_R	W	0h	10-bit R value to be stored in the gamma table
19-10	VALUE_G	W	0h	10-bit G value to be stored in the gamma table
9-0	VALUE_B	W	0h	10-bit B value to be stored in the gamma table

#### 12.6.4.11.13.4.63 DSS0\_VP\_GAMMA\_TABLE\_3 Register (Offset = 12Ch) [reset = X]

DSS0\_VP\_GAMMA\_TABLE\_3 is shown in [Figure 12-915](#) and described in [Table 12-1180](#).

Return to [Summary Table](#).

The register configures the gamma table on VP output.

**Table 12-1179. DSS0\_VP\_GAMMA\_TABLE\_3  
Instances**

Instance	Physical Address
DSS0_VP1	04A8 012Ch
DSS0_VP2	04AA 012Ch
DSS0_VP3	04AC 012Ch
DSS0_VP4	04AE 012Ch

**Figure 12-915. DSS0\_VP\_GAMMA\_TABLE\_3 Register**

31	30	29	28	27	26	25	24
INDEX	RESERVED	VALUE_R					
W-0h	W-X	W-0h					
23	22	21	20	19	18	17	16
VALUE_R				VALUE_G			
W-0h				W-0h			
15	14	13	12	11	10	9	8
VALUE_G						VALUE_B	
W-0h						W-0h	
7	6	5	4	3	2	1	0
VALUE_B							
W-0h							

LEGEND: W = Write Only; -n = value after reset

**Table 12-1180. DSS0\_VP\_GAMMA\_TABLE\_3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	INDEX	W	0h	Write 1 to reset the index
30	RESERVED	W	X	
29-20	VALUE_R	W	0h	10-bit R value to be stored in the gamma table
19-10	VALUE_G	W	0h	10-bit G value to be stored in the gamma table
9-0	VALUE_B	W	0h	10-bit B value to be stored in the gamma table

#### 12.6.4.11.13.4.64 DSS0\_VP\_GAMMA\_TABLE\_4 Register (Offset = 130h) [reset = X]

DSS0\_VP\_GAMMA\_TABLE\_4 is shown in [Figure 12-916](#) and described in [Table 12-1182](#).

Return to [Summary Table](#).

The register configures the gamma table on VP output.

**Table 12-1181. DSS0\_VP\_GAMMA\_TABLE\_4  
Instances**

Instance	Physical Address
DSS0_VP1	04A8 0130h
DSS0_VP2	04AA 0130h
DSS0_VP3	04AC 0130h
DSS0_VP4	04AE 0130h

**Figure 12-916. DSS0\_VP\_GAMMA\_TABLE\_4 Register**

31	30	29	28	27	26	25	24
INDEX	RESERVED	VALUE_R					
W-0h	W-X	W-0h					
23	22	21	20	19	18	17	16
VALUE_R				VALUE_G			
W-0h				W-0h			
15	14	13	12	11	10	9	8
VALUE_G						VALUE_B	
W-0h						W-0h	
7	6	5	4	3	2	1	0
VALUE_B							
W-0h							

LEGEND: W = Write Only; -n = value after reset

**Table 12-1182. DSS0\_VP\_GAMMA\_TABLE\_4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	INDEX	W	0h	Write 1 to reset the index
30	RESERVED	W	X	
29-20	VALUE_R	W	0h	10-bit R value to be stored in the gamma table
19-10	VALUE_G	W	0h	10-bit G value to be stored in the gamma table
9-0	VALUE_B	W	0h	10-bit B value to be stored in the gamma table

#### 12.6.4.11.13.4.65 DSS0\_VP\_GAMMA\_TABLE\_5 Register (Offset = 134h) [reset = X]

DSS0\_VP\_GAMMA\_TABLE\_5 is shown in [Figure 12-917](#) and described in [Table 12-1184](#).

Return to [Summary Table](#).

The register configures the gamma table on VP output.

**Table 12-1183. DSS0\_VP\_GAMMA\_TABLE\_5  
Instances**

Instance	Physical Address
DSS0_VP1	04A8 0134h
DSS0_VP2	04AA 0134h
DSS0_VP3	04AC 0134h
DSS0_VP4	04AE 0134h

**Figure 12-917. DSS0\_VP\_GAMMA\_TABLE\_5 Register**

31	30	29	28	27	26	25	24
INDEX	RESERVED	VALUE_R					
W-0h	W-X	W-0h					
23	22	21	20	19	18	17	16
VALUE_R				VALUE_G			
W-0h				W-0h			
15	14	13	12	11	10	9	8
VALUE_G						VALUE_B	
W-0h						W-0h	
7	6	5	4	3	2	1	0
VALUE_B							
W-0h							

LEGEND: W = Write Only; -n = value after reset

**Table 12-1184. DSS0\_VP\_GAMMA\_TABLE\_5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	INDEX	W	0h	Write 1 to reset the index
30	RESERVED	W	X	
29-20	VALUE_R	W	0h	10-bit R value to be stored in the gamma table
19-10	VALUE_G	W	0h	10-bit G value to be stored in the gamma table
9-0	VALUE_B	W	0h	10-bit B value to be stored in the gamma table

### 12.6.4.11.13.4.66 DSS0\_VP\_GAMMA\_TABLE\_6 Register (Offset = 138h) [reset = X]

DSS0\_VP\_GAMMA\_TABLE\_6 is shown in [Figure 12-918](#) and described in [Table 12-1186](#).

Return to [Summary Table](#).

The register configures the gamma table on VP output.

**Table 12-1185. DSS0\_VP\_GAMMA\_TABLE\_6  
Instances**

Instance	Physical Address
DSS0_VP1	04A8 0138h
DSS0_VP2	04AA 0138h
DSS0_VP3	04AC 0138h
DSS0_VP4	04AE 0138h

**Figure 12-918. DSS0\_VP\_GAMMA\_TABLE\_6 Register**

31	30	29	28	27	26	25	24
INDEX	RESERVED	VALUE_R					
W-0h	W-X	W-0h					
23	22	21	20	19	18	17	16
VALUE_R				VALUE_G			
W-0h				W-0h			
15	14	13	12	11	10	9	8
VALUE_G						VALUE_B	
W-0h						W-0h	
7	6	5	4	3	2	1	0
VALUE_B							
W-0h							

LEGEND: W = Write Only; -n = value after reset

**Table 12-1186. DSS0\_VP\_GAMMA\_TABLE\_6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	INDEX	W	0h	Write 1 to reset the index
30	RESERVED	W	X	
29-20	VALUE_R	W	0h	10-bit R value to be stored in the gamma table
19-10	VALUE_G	W	0h	10-bit G value to be stored in the gamma table
9-0	VALUE_B	W	0h	10-bit B value to be stored in the gamma table

### 12.6.4.11.13.4.67 DSS0\_VP\_GAMMA\_TABLE\_7 Register (Offset = 13Ch) [reset = X]

DSS0\_VP\_GAMMA\_TABLE\_7 is shown in [Figure 12-919](#) and described in [Table 12-1188](#).

Return to [Summary Table](#).

The register configures the gamma table on VP output.

**Table 12-1187. DSS0\_VP\_GAMMA\_TABLE\_7  
Instances**

Instance	Physical Address
DSS0_VP1	04A8 013Ch
DSS0_VP2	04AA 013Ch
DSS0_VP3	04AC 013Ch
DSS0_VP4	04AE 013Ch

**Figure 12-919. DSS0\_VP\_GAMMA\_TABLE\_7 Register**

31	30	29	28	27	26	25	24
INDEX	RESERVED	VALUE_R					
W-0h	W-X	W-0h					
23	22	21	20	19	18	17	16
VALUE_R				VALUE_G			
W-0h				W-0h			
15	14	13	12	11	10	9	8
VALUE_G						VALUE_B	
W-0h						W-0h	
7	6	5	4	3	2	1	0
VALUE_B							
W-0h							

LEGEND: W = Write Only; -n = value after reset

**Table 12-1188. DSS0\_VP\_GAMMA\_TABLE\_7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	INDEX	W	0h	Write 1 to reset the index
30	RESERVED	W	X	
29-20	VALUE_R	W	0h	10-bit R value to be stored in the gamma table
19-10	VALUE_G	W	0h	10-bit G value to be stored in the gamma table
9-0	VALUE_B	W	0h	10-bit B value to be stored in the gamma table

### 12.6.4.11.13.4.68 DSS0\_VP\_GAMMA\_TABLE\_8 Register (Offset = 140h) [reset = X]

DSS0\_VP\_GAMMA\_TABLE\_8 is shown in [Figure 12-920](#) and described in [Table 12-1190](#).

Return to [Summary Table](#).

The register configures the gamma table on VP output.

**Table 12-1189. DSS0\_VP\_GAMMA\_TABLE\_8  
Instances**

Instance	Physical Address
DSS0_VP1	04A8 0140h
DSS0_VP2	04AA 0140h
DSS0_VP3	04AC 0140h
DSS0_VP4	04AE 0140h

**Figure 12-920. DSS0\_VP\_GAMMA\_TABLE\_8 Register**

31	30	29	28	27	26	25	24
INDEX	RESERVED	VALUE_R					
W-0h	W-X	W-0h					
23	22	21	20	19	18	17	16
VALUE_R				VALUE_G			
W-0h				W-0h			
15	14	13	12	11	10	9	8
VALUE_G						VALUE_B	
W-0h						W-0h	
7	6	5	4	3	2	1	0
VALUE_B							
W-0h							

LEGEND: W = Write Only; -n = value after reset

**Table 12-1190. DSS0\_VP\_GAMMA\_TABLE\_8 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	INDEX	W	0h	Write 1 to reset the index
30	RESERVED	W	X	
29-20	VALUE_R	W	0h	10-bit R value to be stored in the gamma table
19-10	VALUE_G	W	0h	10-bit G value to be stored in the gamma table
9-0	VALUE_B	W	0h	10-bit B value to be stored in the gamma table



#### 12.6.4.11.13.4.69 DSS0\_VP\_GAMMA\_TABLE\_9 Register (Offset = 144h) [reset = X]

DSS0\_VP\_GAMMA\_TABLE\_9 is shown in [Figure 12-921](#) and described in [Table 12-1192](#).

Return to [Summary Table](#).

The register configures the gamma table on VP output.

**Table 12-1191. DSS0\_VP\_GAMMA\_TABLE\_9  
Instances**

Instance	Physical Address
DSS0_VP1	04A8 0144h
DSS0_VP2	04AA 0144h
DSS0_VP3	04AC 0144h
DSS0_VP4	04AE 0144h

**Figure 12-921. DSS0\_VP\_GAMMA\_TABLE\_9 Register**

31	30	29	28	27	26	25	24
INDEX	RESERVED	VALUE_R					
W-0h	W-X	W-0h					
23	22	21	20	19	18	17	16
VALUE_R				VALUE_G			
W-0h				W-0h			
15	14	13	12	11	10	9	8
VALUE_G						VALUE_B	
W-0h						W-0h	
7	6	5	4	3	2	1	0
VALUE_B							
W-0h							

LEGEND: W = Write Only; -n = value after reset

**Table 12-1192. DSS0\_VP\_GAMMA\_TABLE\_9 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	INDEX	W	0h	Write 1 to reset the index
30	RESERVED	W	X	
29-20	VALUE_R	W	0h	10-bit R value to be stored in the gamma table
19-10	VALUE_G	W	0h	10-bit G value to be stored in the gamma table
9-0	VALUE_B	W	0h	10-bit B value to be stored in the gamma table

#### 12.6.4.11.13.4.70 DSS0\_VP\_GAMMA\_TABLE\_10 Register (Offset = 148h) [reset = X]

DSS0\_VP\_GAMMA\_TABLE\_10 is shown in [Figure 12-922](#) and described in [Table 12-1194](#).

Return to [Summary Table](#).

The register configures the gamma table on VP output.

**Table 12-1193. DSS0\_VP\_GAMMA\_TABLE\_10  
Instances**

Instance	Physical Address
DSS0_VP1	04A8 0148h
DSS0_VP2	04AA 0148h
DSS0_VP3	04AC 0148h
DSS0_VP4	04AE 0148h

**Figure 12-922. DSS0\_VP\_GAMMA\_TABLE\_10 Register**

31	30	29	28	27	26	25	24
INDEX	RESERVED	VALUE_R					
W-0h	W-X	W-0h					
23	22	21	20	19	18	17	16
VALUE_R				VALUE_G			
W-0h				W-0h			
15	14	13	12	11	10	9	8
VALUE_G						VALUE_B	
W-0h						W-0h	
7	6	5	4	3	2	1	0
VALUE_B							
W-0h							

LEGEND: W = Write Only; -n = value after reset

**Table 12-1194. DSS0\_VP\_GAMMA\_TABLE\_10 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	INDEX	W	0h	Write 1 to reset the index
30	RESERVED	W	X	
29-20	VALUE_R	W	0h	10-bit R value to be stored in the gamma table
19-10	VALUE_G	W	0h	10-bit G value to be stored in the gamma table
9-0	VALUE_B	W	0h	10-bit B value to be stored in the gamma table

### 12.6.4.11.13.4.71 DSS0\_VP\_GAMMA\_TABLE\_11 Register (Offset = 14Ch) [reset = X]

DSS0\_VP\_GAMMA\_TABLE\_11 is shown in [Figure 12-923](#) and described in [Table 12-1196](#).

Return to [Summary Table](#).

The register configures the gamma table on VP output.

**Table 12-1195. DSS0\_VP\_GAMMA\_TABLE\_11  
Instances**

Instance	Physical Address
DSS0_VP1	04A8 014Ch
DSS0_VP2	04AA 014Ch
DSS0_VP3	04AC 014Ch
DSS0_VP4	04AE 014Ch

**Figure 12-923. DSS0\_VP\_GAMMA\_TABLE\_11 Register**

31	30	29	28	27	26	25	24
INDEX	RESERVED	VALUE_R					
W-0h	W-X	W-0h					
23	22	21	20	19	18	17	16
VALUE_R				VALUE_G			
W-0h				W-0h			
15	14	13	12	11	10	9	8
VALUE_G						VALUE_B	
W-0h						W-0h	
7	6	5	4	3	2	1	0
VALUE_B							
W-0h							

LEGEND: W = Write Only; -n = value after reset

**Table 12-1196. DSS0\_VP\_GAMMA\_TABLE\_11 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	INDEX	W	0h	Write 1 to reset the index
30	RESERVED	W	X	
29-20	VALUE_R	W	0h	10-bit R value to be stored in the gamma table
19-10	VALUE_G	W	0h	10-bit G value to be stored in the gamma table
9-0	VALUE_B	W	0h	10-bit B value to be stored in the gamma table

### 12.6.4.11.13.4.72 DSS0\_VP\_GAMMA\_TABLE\_12 Register (Offset = 150h) [reset = X]

DSS0\_VP\_GAMMA\_TABLE\_12 is shown in [Figure 12-924](#) and described in [Table 12-1198](#).

Return to [Summary Table](#).

The register configures the gamma table on VP output.

**Table 12-1197. DSS0\_VP\_GAMMA\_TABLE\_12  
Instances**

Instance	Physical Address
DSS0_VP1	04A8 0150h
DSS0_VP2	04AA 0150h
DSS0_VP3	04AC 0150h
DSS0_VP4	04AE 0150h

**Figure 12-924. DSS0\_VP\_GAMMA\_TABLE\_12 Register**

31	30	29	28	27	26	25	24
INDEX	RESERVED	VALUE_R					
W-0h	W-X	W-0h					
23	22	21	20	19	18	17	16
VALUE_R				VALUE_G			
W-0h				W-0h			
15	14	13	12	11	10	9	8
VALUE_G						VALUE_B	
W-0h						W-0h	
7	6	5	4	3	2	1	0
VALUE_B							
W-0h							

LEGEND: W = Write Only; -n = value after reset

**Table 12-1198. DSS0\_VP\_GAMMA\_TABLE\_12 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	INDEX	W	0h	Write 1 to reset the index
30	RESERVED	W	X	
29-20	VALUE_R	W	0h	10-bit R value to be stored in the gamma table
19-10	VALUE_G	W	0h	10-bit G value to be stored in the gamma table
9-0	VALUE_B	W	0h	10-bit B value to be stored in the gamma table

#### 12.6.4.11.13.4.73 DSS0\_VP\_GAMMA\_TABLE\_13 Register (Offset = 154h) [reset = X]

DSS0\_VP\_GAMMA\_TABLE\_13 is shown in [Figure 12-925](#) and described in [Table 12-1200](#).

Return to [Summary Table](#).

The register configures the gamma table on VP output.

**Table 12-1199. DSS0\_VP\_GAMMA\_TABLE\_13  
Instances**

Instance	Physical Address
DSS0_VP1	04A8 0154h
DSS0_VP2	04AA 0154h
DSS0_VP3	04AC 0154h
DSS0_VP4	04AE 0154h

**Figure 12-925. DSS0\_VP\_GAMMA\_TABLE\_13 Register**

31	30	29	28	27	26	25	24
INDEX	RESERVED	VALUE_R					
W-0h	W-X	W-0h					
23	22	21	20	19	18	17	16
VALUE_R				VALUE_G			
W-0h				W-0h			
15	14	13	12	11	10	9	8
VALUE_G						VALUE_B	
W-0h						W-0h	
7	6	5	4	3	2	1	0
VALUE_B							
W-0h							

LEGEND: W = Write Only; -n = value after reset

**Table 12-1200. DSS0\_VP\_GAMMA\_TABLE\_13 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	INDEX	W	0h	Write 1 to reset the index
30	RESERVED	W	X	
29-20	VALUE_R	W	0h	10-bit R value to be stored in the gamma table
19-10	VALUE_G	W	0h	10-bit G value to be stored in the gamma table
9-0	VALUE_B	W	0h	10-bit B value to be stored in the gamma table

#### 12.6.4.11.13.4.74 DSS0\_VP\_GAMMA\_TABLE\_14 Register (Offset = 158h) [reset = X]

DSS0\_VP\_GAMMA\_TABLE\_14 is shown in [Figure 12-926](#) and described in [Table 12-1202](#).

Return to [Summary Table](#).

The register configures the gamma table on VP output.

**Table 12-1201. DSS0\_VP\_GAMMA\_TABLE\_14  
Instances**

Instance	Physical Address
DSS0_VP1	04A8 0158h
DSS0_VP2	04AA 0158h
DSS0_VP3	04AC 0158h
DSS0_VP4	04AE 0158h

**Figure 12-926. DSS0\_VP\_GAMMA\_TABLE\_14 Register**

31	30	29	28	27	26	25	24
INDEX	RESERVED	VALUE_R					
W-0h	W-X	W-0h					
23	22	21	20	19	18	17	16
VALUE_R				VALUE_G			
W-0h				W-0h			
15	14	13	12	11	10	9	8
VALUE_G						VALUE_B	
W-0h						W-0h	
7	6	5	4	3	2	1	0
VALUE_B							
W-0h							

LEGEND: W = Write Only; -n = value after reset

**Table 12-1202. DSS0\_VP\_GAMMA\_TABLE\_14 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	INDEX	W	0h	Write 1 to reset the index
30	RESERVED	W	X	
29-20	VALUE_R	W	0h	10-bit R value to be stored in the gamma table
19-10	VALUE_G	W	0h	10-bit G value to be stored in the gamma table
9-0	VALUE_B	W	0h	10-bit B value to be stored in the gamma table

#### 12.6.4.11.13.4.75 DSS0\_VP\_GAMMA\_TABLE\_15 Register (Offset = 15Ch) [reset = X]

DSS0\_VP\_GAMMA\_TABLE\_15 is shown in [Figure 12-927](#) and described in [Table 12-1204](#).

Return to [Summary Table](#).

The register configures the gamma table on VP output.

**Table 12-1203. DSS0\_VP\_GAMMA\_TABLE\_15  
Instances**

Instance	Physical Address
DSS0_VP1	04A8 015Ch
DSS0_VP2	04AA 015Ch
DSS0_VP3	04AC 015Ch
DSS0_VP4	04AE 015Ch

**Figure 12-927. DSS0\_VP\_GAMMA\_TABLE\_15 Register**

31	30	29	28	27	26	25	24
INDEX	RESERVED	VALUE_R					
W-0h	W-X	W-0h					
23	22	21	20	19	18	17	16
VALUE_R				VALUE_G			
W-0h				W-0h			
15	14	13	12	11	10	9	8
VALUE_G						VALUE_B	
W-0h						W-0h	
7	6	5	4	3	2	1	0
VALUE_B							
W-0h							

LEGEND: W = Write Only; -n = value after reset

**Table 12-1204. DSS0\_VP\_GAMMA\_TABLE\_15 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	INDEX	W	0h	Write 1 to reset the index
30	RESERVED	W	X	
29-20	VALUE_R	W	0h	10-bit R value to be stored in the gamma table
19-10	VALUE_G	W	0h	10-bit G value to be stored in the gamma table
9-0	VALUE_B	W	0h	10-bit B value to be stored in the gamma table

### 12.6.4.11.13.4.76 DSS0\_VP\_SECURE Register (Offset = 178h) [reset = 0h]

DSS0\_VP\_SECURE is shown in [Figure 12-928](#) and described in [Table 12-1206](#).

Return to [Summary Table](#).

Security bit settings for the sub-module

**Table 12-1205. DSS0\_VP\_SECURE Instances**

Instance	Physical Address
DSS0_VP1	04A8 0178h
DSS0_VP2	04AA 0178h
DSS0_VP3	04AC 0178h
DSS0_VP4	04AE 0178h

**Figure 12-928. DSS0\_VP\_SECURE Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							SECURE
R-0h							R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1206. DSS0\_VP\_SECURE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	SECURE	R/W	0h	DSS0_VP_SECURE bit 0h = DSS0_VP_SECURE bit is reset 1h = DSS0_VP_SECURE bit is set



#### 12.6.4.11.13.5 DSS\_WB Registers

Table 12-1208 lists the memory-mapped registers for the DSS\_WB registers. All register offset addresses not listed in Table 12-1208 should be considered as reserved locations and the register contents should not be modified.

WB Registers

**Table 12-1207. DSS\_WB Instances**

Instance	Base Address
DSS0_WB	04AF 0000h

**Table 12-1208. DSS\_WB Registers**

Offset	Acronym	Register Name	DSS0_WB Physical Address
0h	<a href="#">DSS0_WB_ACCUH_0</a>		04AF 0000h
4h	<a href="#">DSS0_WB_ACCUH_1</a>		04AF 0004h
8h	<a href="#">DSS0_WB_ACCUH2_0</a>		04AF 0008h
Ch	<a href="#">DSS0_WB_ACCUH2_1</a>		04AF 000Ch
10h	<a href="#">DSS0_WB_ACCUV_0</a>		04AF 0010h
14h	<a href="#">DSS0_WB_ACCUV_1</a>		04AF 0014h
18h	<a href="#">DSS0_WB_ACCUV2_0</a>		04AF 0018h
1Ch	<a href="#">DSS0_WB_ACCUV2_1</a>		04AF 001Ch
20h	<a href="#">DSS0_WB_ATTRIBUTES</a>		04AF 0020h
24h	<a href="#">DSS0_WB_ATTRIBUTES2</a>		04AF 0024h
28h	<a href="#">DSS0_WB_BA_0</a>		04AF 0028h
2Ch	<a href="#">DSS0_WB_BA_1</a>		04AF 002Ch
30h	<a href="#">DSS0_WB_BA_UV_0</a>		04AF 0030h
34h	<a href="#">DSS0_WB_BA_UV_1</a>		04AF 0034h
38h	<a href="#">DSS0_WB_BUF_SIZE_STATUS</a>		04AF 0038h
3Ch	<a href="#">DSS0_WB_BUF_THRESHOLD</a>		04AF 003Ch
40h	<a href="#">DSS0_WB_CSC_COEF0</a>		04AF 0040h
44h	<a href="#">DSS0_WB_CSC_COEF1</a>		04AF 0044h
48h	<a href="#">DSS0_WB_CSC_COEF2</a>		04AF 0048h
4Ch	<a href="#">DSS0_WB_CSC_COEF3</a>		04AF 004Ch
50h	<a href="#">DSS0_WB_CSC_COEF4</a>		04AF 0050h
54h	<a href="#">DSS0_WB_CSC_COEF5</a>		04AF 0054h
58h	<a href="#">DSS0_WB_CSC_COEF6</a>		04AF 0058h
5Ch	<a href="#">DSS0_WB_FIRH</a>		04AF 005Ch
60h	<a href="#">DSS0_WB_FIRH2</a>		04AF 0060h
64h	<a href="#">DSS0_WB_FIRV</a>		04AF 0064h
68h	<a href="#">DSS0_WB_FIRV2</a>		04AF 0068h
6Ch	<a href="#">DSS0_WB_FIR_COEF_H0_0</a>		04AF 006Ch
70h	<a href="#">DSS0_WB_FIR_COEF_H0_1</a>		04AF 0070h
74h	<a href="#">DSS0_WB_FIR_COEF_H0_2</a>		04AF 0074h
78h	<a href="#">DSS0_WB_FIR_COEF_H0_3</a>		04AF 0078h
7Ch	<a href="#">DSS0_WB_FIR_COEF_H0_4</a>		04AF 007Ch
80h	<a href="#">DSS0_WB_FIR_COEF_H0_5</a>		04AF 0080h
84h	<a href="#">DSS0_WB_FIR_COEF_H0_6</a>		04AF 0084h
88h	<a href="#">DSS0_WB_FIR_COEF_H0_7</a>		04AF 0088h

**Table 12-1208. DSS\_WB Registers (continued)**

Offset	Acronym	Register Name	DSS0_WB Physical Address
8Ch	<a href="#">DSS0_WB_FIR_COEF_H0_8</a>		04AF 008Ch
90h	<a href="#">DSS0_WB_FIR_COEF_H0_C_0</a>		04AF 0090h
94h	<a href="#">DSS0_WB_FIR_COEF_H0_C_1</a>		04AF 0094h
98h	<a href="#">DSS0_WB_FIR_COEF_H0_C_2</a>		04AF 0098h
9Ch	<a href="#">DSS0_WB_FIR_COEF_H0_C_3</a>		04AF 009Ch
A0h	<a href="#">DSS0_WB_FIR_COEF_H0_C_4</a>		04AF 00A0h
A4h	<a href="#">DSS0_WB_FIR_COEF_H0_C_5</a>		04AF 00A4h
A8h	<a href="#">DSS0_WB_FIR_COEF_H0_C_6</a>		04AF 00A8h
ACh	<a href="#">DSS0_WB_FIR_COEF_H0_C_7</a>		04AF 00ACh
B0h	<a href="#">DSS0_WB_FIR_COEF_H0_C_8</a>		04AF 00B0h
B4h	<a href="#">DSS0_WB_FIR_COEF_H12_0</a>		04AF 00B4h
B8h	<a href="#">DSS0_WB_FIR_COEF_H12_1</a>		04AF 00B8h
BCh	<a href="#">DSS0_WB_FIR_COEF_H12_2</a>		04AF 00BCh
C0h	<a href="#">DSS0_WB_FIR_COEF_H12_3</a>		04AF 00C0h
C4h	<a href="#">DSS0_WB_FIR_COEF_H12_4</a>		04AF 00C4h
C8h	<a href="#">DSS0_WB_FIR_COEF_H12_5</a>		04AF 00C8h
CCh	<a href="#">DSS0_WB_FIR_COEF_H12_6</a>		04AF 00CCh
D0h	<a href="#">DSS0_WB_FIR_COEF_H12_7</a>		04AF 00D0h
D4h	<a href="#">DSS0_WB_FIR_COEF_H12_8</a>		04AF 00D4h
D8h	<a href="#">DSS0_WB_FIR_COEF_H12_9</a>		04AF 00D8h
DCh	<a href="#">DSS0_WB_FIR_COEF_H12_10</a>		04AF 00DCh
E0h	<a href="#">DSS0_WB_FIR_COEF_H12_11</a>		04AF 00E0h
E4h	<a href="#">DSS0_WB_FIR_COEF_H12_12</a>		04AF 00E4h
E8h	<a href="#">DSS0_WB_FIR_COEF_H12_13</a>		04AF 00E8h
ECh	<a href="#">DSS0_WB_FIR_COEF_H12_14</a>		04AF 00ECh
F0h	<a href="#">DSS0_WB_FIR_COEF_H12_15</a>		04AF 00F0h
F4h	<a href="#">DSS0_WB_FIR_COEF_H12_C_0</a>		04AF 00F4h
F8h	<a href="#">DSS0_WB_FIR_COEF_H12_C_1</a>		04AF 00F8h
FCh	<a href="#">DSS0_WB_FIR_COEF_H12_C_2</a>		04AF 00FCh
100h	<a href="#">DSS0_WB_FIR_COEF_H12_C_3</a>		04AF 0100h
104h	<a href="#">DSS0_WB_FIR_COEF_H12_C_4</a>		04AF 0104h
108h	<a href="#">DSS0_WB_FIR_COEF_H12_C_5</a>		04AF 0108h
10Ch	<a href="#">DSS0_WB_FIR_COEF_H12_C_6</a>		04AF 010Ch
110h	<a href="#">DSS0_WB_FIR_COEF_H12_C_7</a>		04AF 0110h
114h	<a href="#">DSS0_WB_FIR_COEF_H12_C_8</a>		04AF 0114h
118h	<a href="#">DSS0_WB_FIR_COEF_H12_C_9</a>		04AF 0118h
11Ch	<a href="#">DSS0_WB_FIR_COEF_H12_C_10</a>		04AF 011Ch
120h	<a href="#">DSS0_WB_FIR_COEF_H12_C_11</a>		04AF 0120h
124h	<a href="#">DSS0_WB_FIR_COEF_H12_C_12</a>		04AF 0124h
128h	<a href="#">DSS0_WB_FIR_COEF_H12_C_13</a>		04AF 0128h
12Ch	<a href="#">DSS0_WB_FIR_COEF_H12_C_14</a>		04AF 012Ch
130h	<a href="#">DSS0_WB_FIR_COEF_H12_C_15</a>		04AF 0130h
134h	<a href="#">DSS0_WB_FIR_COEF_V0_0</a>		04AF 0134h
138h	<a href="#">DSS0_WB_FIR_COEF_V0_1</a>		04AF 0138h

**Table 12-1208. DSS\_WB Registers (continued)**

Offset	Acronym	Register Name	DSS0_WB Physical Address
13Ch	<a href="#">DSS0_WB_FIR_COEF_V0_2</a>		04AF 013Ch
140h	<a href="#">DSS0_WB_FIR_COEF_V0_3</a>		04AF 0140h
144h	<a href="#">DSS0_WB_FIR_COEF_V0_4</a>		04AF 0144h
148h	<a href="#">DSS0_WB_FIR_COEF_V0_5</a>		04AF 0148h
14Ch	<a href="#">DSS0_WB_FIR_COEF_V0_6</a>		04AF 014Ch
150h	<a href="#">DSS0_WB_FIR_COEF_V0_7</a>		04AF 0150h
154h	<a href="#">DSS0_WB_FIR_COEF_V0_8</a>		04AF 0154h
158h	<a href="#">DSS0_WB_FIR_COEF_V0_C_0</a>		04AF 0158h
15Ch	<a href="#">DSS0_WB_FIR_COEF_V0_C_1</a>		04AF 015Ch
160h	<a href="#">DSS0_WB_FIR_COEF_V0_C_2</a>		04AF 0160h
164h	<a href="#">DSS0_WB_FIR_COEF_V0_C_3</a>		04AF 0164h
168h	<a href="#">DSS0_WB_FIR_COEF_V0_C_4</a>		04AF 0168h
16Ch	<a href="#">DSS0_WB_FIR_COEF_V0_C_5</a>		04AF 016Ch
170h	<a href="#">DSS0_WB_FIR_COEF_V0_C_6</a>		04AF 0170h
174h	<a href="#">DSS0_WB_FIR_COEF_V0_C_7</a>		04AF 0174h
178h	<a href="#">DSS0_WB_FIR_COEF_V0_C_8</a>		04AF 0178h
17Ch	<a href="#">DSS0_WB_FIR_COEF_V12_0</a>		04AF 017Ch
180h	<a href="#">DSS0_WB_FIR_COEF_V12_1</a>		04AF 0180h
184h	<a href="#">DSS0_WB_FIR_COEF_V12_2</a>		04AF 0184h
188h	<a href="#">DSS0_WB_FIR_COEF_V12_3</a>		04AF 0188h
18Ch	<a href="#">DSS0_WB_FIR_COEF_V12_4</a>		04AF 018Ch
190h	<a href="#">DSS0_WB_FIR_COEF_V12_5</a>		04AF 0190h
194h	<a href="#">DSS0_WB_FIR_COEF_V12_6</a>		04AF 0194h
198h	<a href="#">DSS0_WB_FIR_COEF_V12_7</a>		04AF 0198h
19Ch	<a href="#">DSS0_WB_FIR_COEF_V12_8</a>		04AF 019Ch
1A0h	<a href="#">DSS0_WB_FIR_COEF_V12_9</a>		04AF 01A0h
1A4h	<a href="#">DSS0_WB_FIR_COEF_V12_10</a>		04AF 01A4h
1A8h	<a href="#">DSS0_WB_FIR_COEF_V12_11</a>		04AF 01A8h
1ACh	<a href="#">DSS0_WB_FIR_COEF_V12_12</a>		04AF 01ACh
1B0h	<a href="#">DSS0_WB_FIR_COEF_V12_13</a>		04AF 01B0h
1B4h	<a href="#">DSS0_WB_FIR_COEF_V12_14</a>		04AF 01B4h
1B8h	<a href="#">DSS0_WB_FIR_COEF_V12_15</a>		04AF 01B8h
1BCh	<a href="#">DSS0_WB_FIR_COEF_V12_C_0</a>		04AF 01BCh
1C0h	<a href="#">DSS0_WB_FIR_COEF_V12_C_1</a>		04AF 01C0h
1C4h	<a href="#">DSS0_WB_FIR_COEF_V12_C_2</a>		04AF 01C4h
1C8h	<a href="#">DSS0_WB_FIR_COEF_V12_C_3</a>		04AF 01C8h
1CCh	<a href="#">DSS0_WB_FIR_COEF_V12_C_4</a>		04AF 01CCh
1D0h	<a href="#">DSS0_WB_FIR_COEF_V12_C_5</a>		04AF 01D0h
1D4h	<a href="#">DSS0_WB_FIR_COEF_V12_C_6</a>		04AF 01D4h
1D8h	<a href="#">DSS0_WB_FIR_COEF_V12_C_7</a>		04AF 01D8h
1DCh	<a href="#">DSS0_WB_FIR_COEF_V12_C_8</a>		04AF 01DCh
1E0h	<a href="#">DSS0_WB_FIR_COEF_V12_C_9</a>		04AF 01E0h
1E4h	<a href="#">DSS0_WB_FIR_COEF_V12_C_10</a>		04AF 01E4h
1E8h	<a href="#">DSS0_WB_FIR_COEF_V12_C_11</a>		04AF 01E8h

**Table 12-1208. DSS\_WB Registers (continued)**

Offset	Acronym	Register Name	DSS0_WB Physical Address
1ECh	<a href="#">DSS0_WB_FIR_COEF_V12_C_12</a>		04AF 01ECh
1F0h	<a href="#">DSS0_WB_FIR_COEF_V12_C_13</a>		04AF 01F0h
1F4h	<a href="#">DSS0_WB_FIR_COEF_V12_C_14</a>		04AF 01F4h
1F8h	<a href="#">DSS0_WB_FIR_COEF_V12_C_15</a>		04AF 01F8h
204h	<a href="#">DSS0_WB_MFLAG_THRESHOLD</a>		04AF 0204h
208h	<a href="#">DSS0_WB_PICTURE_SIZE</a>		04AF 0208h
210h	<a href="#">DSS0_WB_SIZE</a>		04AF 0210h
214h	<a href="#">DSS0_WB_POSITION</a>		04AF 0214h
21Ch	<a href="#">DSS0_WB_CSC_COEF7</a>		04AF 021Ch
224h	<a href="#">DSS0_WB_ROW_INC</a>		04AF 0224h
228h	<a href="#">DSS0_WB_ROW_INC_UV</a>		04AF 0228h
22Ch	<a href="#">DSS0_WB_BA_EXT_0</a>		04AF 022Ch
230h	<a href="#">DSS0_WB_BA_EXT_1</a>		04AF 0230h
234h	<a href="#">DSS0_WB_BA_UV_EXT_0</a>		04AF 0234h
238h	<a href="#">DSS0_WB_BA_UV_EXT_1</a>		04AF 0238h
248h	<a href="#">DSS0_WB_SECURE</a>		04AF 0248h

### 12.6.4.11.13.5.1 DSS0\_WB\_ACCUH\_0 Register (Offset = 0h) [reset = 0h]

DSS0\_WB\_ACCUH\_0 is shown in [Figure 12-929](#) and described in [Table 12-1210](#).

Return to [Summary Table](#).

The register configures the resize accumulator init values for horizontal up/down-sampling of the write-back window. It is used for ARGB and Y setting. Shadow register

**Table 12-1209. DSS0\_WB\_ACCUH\_0 Instances**

Instance	Physical Address
DSS0_WB	04AF 0000h

**Figure 12-929. DSS0\_WB\_ACCUH\_0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								HORIZONTALACCU																							
R-0h								R/W-0h																							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1210. DSS0\_WB\_ACCUH\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-0	HORIZONTALACCU	R/W	0h	Horizontal initialization accu signed value

### 12.6.4.11.13.5.2 DSS0\_WB\_ACCUH\_1 Register (Offset = 4h) [reset = 0h]

DSS0\_WB\_ACCUH\_1 is shown in [Figure 12-930](#) and described in [Table 12-1212](#).

Return to [Summary Table](#).

The register configures the resize accumulator init values for horizontal up/down-sampling of the write-back window. It is used for ARGB and Y setting. Shadow register

**Table 12-1211. DSS0\_WB\_ACCUH\_1 Instances**

Instance	Physical Address
DSS0_WB	04AF 0004h

**Figure 12-930. DSS0\_WB\_ACCUH\_1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								HORIZONTALACCU																							
R-0h								R/W-0h																							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1212. DSS0\_WB\_ACCUH\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-0	HORIZONTALACCU	R/W	0h	Horizontal initialization accu signed value

### 12.6.4.11.13.5.3 DSS0\_WB\_ACCUH2\_0 Register (Offset = 8h) [reset = 0h]

DSS0\_WB\_ACCUH2\_0 is shown in [Figure 12-931](#) and described in [Table 12-1214](#).

Return to [Summary Table](#).

The register configures the resize accumulator init value for horizontal up/down-sampling of the write-back window. It is used for Cb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB, all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter as the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1213. DSS0\_WB\_ACCUH2\_0 Instances**

Instance	Physical Address
DSS0_WB	04AF 0008h

**Figure 12-931. DSS0\_WB\_ACCUH2\_0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								HORIZONTALACCU																							
R-0h								R/W-0h																							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1214. DSS0\_WB\_ACCUH2\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-0	HORIZONTALACCU	R/W	0h	Horizontal initialization accu signed value

#### 12.6.4.11.13.5.4 DSS0\_WB\_ACCUH2\_1 Register (Offset = Ch) [reset = 0h]

DSS0\_WB\_ACCUH2\_1 is shown in [Figure 12-932](#) and described in [Table 12-1216](#).

Return to [Summary Table](#).

The register configures the resize accumulator init value for horizontal up/down-sampling of the write-back window. It is used for Cb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB, all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter as the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1215. DSS0\_WB\_ACCUH2\_1 Instances**

Instance	Physical Address
DSS0_WB	04AF 000Ch

**Figure 12-932. DSS0\_WB\_ACCUH2\_1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								HORIZONTALACCU																							
R-0h								R/W-0h																							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1216. DSS0\_WB\_ACCUH2\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-0	HORIZONTALACCU	R/W	0h	Horizontal initialization accu signed value



#### 12.6.4.11.13.5.5 DSS0\_WB\_ACCUV\_0 Register (Offset = 10h) [reset = 0h]

DSS0\_WB\_ACCUV\_0 is shown in [Figure 12-933](#) and described in [Table 12-1218](#).

Return to [Summary Table](#).

The register configures the resize accumulator init value for vertical up/down-sampling of the write-back window. It is used for ARGB and Y setting. Shadow register

**Table 12-1217. DSS0\_WB\_ACCUV\_0 Instances**

Instance	Physical Address
DSS0_WB	04AF 0010h

**Figure 12-933. DSS0\_WB\_ACCUV\_0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								VERTICALACCU																							
R-0h								R/W-0h																							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1218. DSS0\_WB\_ACCUV\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-0	VERTICALACCU	R/W	0h	Vertical initialization accu signed value

### 12.6.4.11.13.5.6 DSS0\_WB\_ACCUV\_1 Register (Offset = 14h) [reset = 0h]

DSS0\_WB\_ACCUV\_1 is shown in [Figure 12-934](#) and described in [Table 12-1220](#).

Return to [Summary Table](#).

The register configures the resize accumulator init value for vertical up/down-sampling of the write-back window. It is used for ARGB and Y setting. Shadow register

**Table 12-1219. DSS0\_WB\_ACCUV\_1 Instances**

Instance	Physical Address
DSS0_WB	04AF 0014h

**Figure 12-934. DSS0\_WB\_ACCUV\_1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								VERTICALACCU																							
R-0h								R/W-0h																							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1220. DSS0\_WB\_ACCUV\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-0	VERTICALACCU	R/W	0h	Vertical initialization accu signed value

#### 12.6.4.11.13.5.7 DSS0\_WB\_ACCUV2\_0 Register (Offset = 18h) [reset = 0h]

DSS0\_WB\_ACCUV2\_0 is shown in [Figure 12-935](#) and described in [Table 12-1222](#).

Return to [Summary Table](#).

The register configures the resize accumulator init value for vertical up/down-sampling of the write-back window. It is used for Cb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB, all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter as the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1221. DSS0\_WB\_ACCUV2\_0 Instances**

Instance	Physical Address
DSS0_WB	04AF 0018h

**Figure 12-935. DSS0\_WB\_ACCUV2\_0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								VERTICALACCU																							
R-0h								R/W-0h																							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1222. DSS0\_WB\_ACCUV2\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-0	VERTICALACCU	R/W	0h	Vertical initialization accu signed value

### 12.6.4.11.13.5.8 DSS0\_WB\_ACCUV2\_1 Register (Offset = 1Ch) [reset = 0h]

DSS0\_WB\_ACCUV2\_1 is shown in [Figure 12-936](#) and described in [Table 12-1224](#).

Return to [Summary Table](#).

The register configures the resize accumulator init value for vertical up/down-sampling of the write-back window. It is used for Cb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB, all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter as the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1223. DSS0\_WB\_ACCUV2\_1 Instances**

Instance	Physical Address
DSS0_WB	04AF 001Ch

**Figure 12-936. DSS0\_WB\_ACCUV2\_1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								VERTICALACCU																							
R-0h								R/W-0h																							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1224. DSS0\_WB\_ACCUV2\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-0	VERTICALACCU	R/W	0h	Vertical initialization accu signed value

### 12.6.4.11.13.5.9 DSS0\_WB\_ATTRIBUTES Register (Offset = 20h) [reset = 0h]

DSS0\_WB\_ATTRIBUTES is shown in [Figure 12-937](#) and described in [Table 12-1226](#).

Return to [Summary Table](#).

The register configures the [DSS0\\_WB\\_ATTRIBUTES](#) of the write back pipeline. Shadow register

**Table 12-1225. DSS0\_WB\_ATTRIBUTES Instances**

Instance	Physical Address
DSS0_WB	04AF 0020h

**Figure 12-937. DSS0\_WB\_ATTRIBUTES Register**

31	30	29	28	27	26	25	24
IDLENUMBER				IDLESIZE	CAPTUREMODE		
R/W-0h				R/W-0h	R/W-0h		
23	22	21	20	19	18	17	16
ARBITRATION	RESERVED	VERTICALTAP S	GOBIT	WRITEBACKM ODE	RESERVED		
R/W-0h	R-0h	R/W-0h	R/W-0h	R/W-0h	R-0h		
15	14	13	12	11	10	9	8
RESERVED		RESERVED	FULLRANGE	COLORCONVE NABLE	RESERVED	ALPHAENABLE	RESIZEENABL E
R-0h		R-0h	R/W-0h	R/W-0h	R-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RESIZEENABL E	FORMAT						ENABLE
R/W-0h				R/W-0h			R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1226. DSS0\_WB\_ATTRIBUTES Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	IDLENUMBER	R/W	0h	Determines the number of idles between requests on the L3 interconnect. It is only used when the write-back pipeline does data transfer from memory to memory. When the output of an overlay is stored in memory through the write-back pipeline in capture mode, the bit-field IDLENUMBER is ignored since a timing generator is used to time the transfer The number of IDLE cycles is IDLENUMBER [from 0 to 15] if IDLESIZE=0 The number of IDLE cycles is IDLENUMBERx8 [from 0 to 120] if IDLESIZE=1
27	IDLESIZE	R/W	0h	Determines if the IDLENUMBER corresponds to a number of bursts or singles  0h = The number of idles between requests is defined by IDLENUMBER as number of cycles  1h = The number of idles between requests is defined by IDLENUMBERx8 as number of cycles

**Table 12-1226. DSS0\_WB\_ATTRIBUTES Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
26-24	CAPTUREMODE	R/W	0h	<p>Defines the frame rate capture</p> <p>0h = All frames are captures until the write-back channel is disabled or there is no more data generated by the overlay or the pipeline attached to the write-back channel</p> <p>1h = Only one frame is captured</p> <p>2h = Only one out of two frames is captured. The first one is captured then the second one is skipped and so on</p> <p>3h = Only one out of three frames is captured. The first one is captured then the second one is skipped and so on</p> <p>4h = Only one out of four frames is captured. The first one is captured then the second one is skipped and so on</p> <p>5h = Only one out of five frames is captured. The first one is captured then the second one is skipped and so on</p> <p>6h = Only one out of six frames is captured. The first one is captured then the second one is skipped and so on</p> <p>7h = Only one out of seven frames is captured. The first one is captured then the second one is skipped and so on</p>
23	ARBITRATION	R/W	0h	<p>Determines the priority of the write-back pipeline.</p> <p>The write-back pipeline is one of the high priority pipelines. The arbitration gives always the priority first to the high priority pipelines using round-robin between them When there are only normal priority pipelines sending requests, the round-robin applies between them</p> <p>0h = The write-back pipeline is one of the normal priority pipelines.</p> <p>1h = The write-back pipeline is one of the high priority pipelines.</p>
22	RESERVED	R	0h	Reserved
21	VERTICALTAPS	R/W	0h	<p>Video Vertical Resize Tap Number</p> <p>0h = 3 taps are used for the vertical filtering logic. The 2 other taps are not used.</p> <p>1h = 5 taps are used for the vertical filtering logic.</p>
20	GOBIT	R/W	0h	<p>GO Command for the WB output.</p> <p>It is used to synchronize the pipelines associated with the WB output wr:immediate</p> <p>0h = The hardware has finished updating the internal shadow registers of the pipeline</p> <p>1h = The user has finished to program the shadow registers of the pipeline</p>
19	WRITEBACKMODE	R/W	0h	<p>When connected to the overlay output of a channel the write back can operate as a simple transfer from memory to memory [composition engine] or as a capture channel</p> <p>0h = Capture mode</p> <p>1h = Memory to memory mode</p>

**Table 12-1226. DSS0\_WB\_ATTRIBUTES Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
18-14	RESERVED	R	0h	Reserved
13	RESERVED	R	0h	Reserved
12	FULLRANGE	R/W	0h	Color Space Conversion full range setting  0h = Limited range selected: 16 subtracted from Y before color space conversion  1h = Full range selected: Y is not modified before the color space conversion
11	COLORCONVENABLE	R/W	0h	Enable the color space conversion. The HW does not enable/disable the conversion based on the pixel format. The bit-field shall be reset when the format is not YUV  0h = Disable Color Space Conversion RGB to YUV  1h = Enable Color Space Conversion RGB to YUV
10	RESERVED	R	0h	Reserved
9	ALPHAENABLE	R/W	0h	Alpha enable on WB output  0h = Alpha out is disabled  1h = Alpha out is enabled
8-7	RESIZEENABLE	R/W	0h	Resize Enable  0h = Disable the resize processing  1h = Enable the horizontal resize processing  2h = Enable the vertical resize processing  3h = Enable both horizontal and vertical resize processing

**Table 12-1226. DSS0\_WB\_ATTRIBUTES Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
6-1	FORMAT	R/W	0h	<p>Write-back Format.</p> <p>It defines the pixel format when storing the write-back picture into memory</p> <p>00h = 0x00</p> <p>01h = 0x01</p> <p>02h = 0x02</p> <p>03h = 0x03</p> <p>04h = 0x04</p> <p>05h = 0x05</p> <p>06h = 0x06</p> <p>07h = 0x07</p> <p>08h = 0x08</p> <p>09h = 0x09</p> <p>0Ah = 0x0A</p> <p>0Bh = 0x0B</p> <p>10h = 0x10</p> <p>11h = 0x11</p> <p>16h = 0x16</p> <p>17h = 0x17</p> <p>20h = 0x20</p> <p>21h = 0x21</p> <p>22h = 0x22</p> <p>25h = 0x25</p> <p>26h = 0x26</p> <p>27h = 0x27</p> <p>28h = 0x28</p> <p>29h = 0x29</p> <p>2Ah = 0x2A</p> <p>30h = 0x30</p> <p>31h = 0x31</p> <p>3Ch = 0x3C</p> <p>3Dh = 0x3D</p> <p>3Eh = 0x3E</p> <p>3Fh = 0x3F</p>
0	ENABLE	R/W	0h	<p>Write-back Enable wr: immediate</p> <p>0h = Write-back disabled</p> <p>1h = Write-back enabled</p>



### 12.6.4.11.13.5.10 DSS0\_WB\_ATTRIBUTES2 Register (Offset = 24h) [reset = 3C000000h]

DSS0\_WB\_ATTRIBUTES2 is shown in [Figure 12-938](#) and described in [Table 12-1228](#).

Return to [Summary Table](#).

The register configures the [DSS0\\_WB\\_ATTRIBUTES](#) of the write back pipeline. Shadow register

**Table 12-1227. DSS0\_WB\_ATTRIBUTES2 Instances**

Instance	Physical Address
DSS0_WB	04AF 0024h

**Figure 12-938. DSS0\_WB\_ATTRIBUTES2 Register**

31	30	29	28	27	26	25	24
RESERVED	TAGS					RESERVED	
R-0h	R/W-Fh					R-0h	
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED					YUV_ALIGN	YUV_MODE	YUV_SIZE
R-0h					R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
YUV_SIZE	RESERVED						RESERVED
R/W-0h	R-0h					R/W-0h	

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1228. DSS0\_WB\_ATTRIBUTES2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	RESERVED	R	0h	Reserved
30-26	TAGS	R/W	Fh	Number of OCP TAGS to be used for the pipeline [0x0 to 0xF]. A value of '0' means a single tag will be used. A value of 'F' means all 16 tags can be used
25-11	RESERVED	R	0h	Reserved
10	YUV_ALIGN	R/W	0h	Alignment [MSB or LSB align] for unpacked 10b/12b YUV data 0h = lsb aligned - unused msb 1h = msb aligned - unused lsb
9	YUV_MODE	R/W	0h	Mode of packing for YUV data [only for 10b/12b formats] 0h = YUV 10-bit formats have the same component packing order as 8-bit formats except that the packing is done across a multiple 32-bit word with 2 MSB in each 32-bit word not used. YUV 12-bit formats have the same component packing order as 8-bit formats except that the packing is done across a multiple 64-bit word with 4 MSB in each 64-bit word not used 1h = YUV 10-bit/12-bit unpacked formats have the same component packing order as 8-bit formats except that each component is stored in a 16-bit container - with MSB or LSB bits within the 16-bit container not used depending on the MSB/LSB alignment

**Table 12-1228. DSS0\_WB\_ATTRIBUTES2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
8-7	YUV_SIZE	R/W	0h	DSS0_WB_SIZE of YUV data 8b/10b/12b  0h = 8b per component-default  1h = 10b per component  2h = 12b per component
6-1	RESERVED	R	0h	Reserved
0	RESERVED	R/W	0h	Reserved

#### 12.6.4.11.13.5.11 DSS0\_WB\_BA\_0 Register (Offset = 28h) [reset = 0h]

DSS0\_WB\_BA\_0 is shown in [Figure 12-939](#) and described in [Table 12-1230](#).

Return to [Summary Table](#).

The register configures the base address of the WB buffer. DISPC\_WB\_BA\_\_0 & DISPC\_WB\_BA\_\_1 for ping-pong mechanism with external trigger, based on the field polarity, otherwise only DISPC\_WB\_BA\_\_0 is used. Shadow register

**Table 12-1229. DSS0\_WB\_BA\_0 Instances**

Instance	Physical Address
DSS0_WB	04AF 0028h

**Figure 12-939. DSS0\_WB\_BA\_0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BA																															
R/W-0h																															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1230. DSS0\_WB\_BA\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	BA	R/W	0h	Write-back base address Base address of the WB buffer [aligned on pixel DSS0_WB_SIZE boundary except in case of RGB24 packed format, where 4-pixel alignment is required. In case of YUV422, 2-pixel alignment is required, and YUV420, byte alignment is supported]] In case of YUV 4:2:0 format, it indicates the base address of the Y buffer

### 12.6.4.11.13.5.12 DSS0\_WB\_BA\_1 Register (Offset = 2Ch) [reset = 0h]

DSS0\_WB\_BA\_1 is shown in [Figure 12-940](#) and described in [Table 12-1232](#).

Return to [Summary Table](#).

The register configures the base address of the WB buffer. DISPC\_WB\_BA\_\_0 & DISPC\_WB\_BA\_\_1 for ping-pong mechanism with external trigger, based on the field polarity, otherwise only DISPC\_WB\_BA\_\_0 is used. Shadow register

**Table 12-1231. DSS0\_WB\_BA\_1 Instances**

Instance	Physical Address
DSS0_WB	04AF 002Ch

**Figure 12-940. DSS0\_WB\_BA\_1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BA																															
R/W-0h																															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1232. DSS0\_WB\_BA\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	BA	R/W	0h	Write-back base address Base address of the WB buffer [aligned on pixel DSS0_WB_SIZE boundary except in case of RGB24 packed format, where 4-pixel alignment is required. In case of YUV422, 2-pixel alignment is required, and YUV420, byte alignment is supported]] In case of YUV 4:2:0 format, it indicates the base address of the Y buffer

#### 12.6.4.11.13.5.13 DSS0\_WB\_BA\_UV\_0 Register (Offset = 30h) [reset = 0h]

DSS0\_WB\_BA\_UV\_0 is shown in [Figure 12-941](#) and described in [Table 12-1234](#).

Return to [Summary Table](#).

The register configures the base address of the UV buffer for the write-back pipeline. DISPC\_WB\_BA\_UV\_\_0 & DISPC\_WB\_BA\_UV\_\_1 for ping-pong mechanism with external trigger, based on the field polarity, otherwise only DISPC\_WB\_BA\_UV\_\_0 is used. The register is also used to configure the RGB plane BA for RGB565A8 format. Shadow register

**Table 12-1233. DSS0\_WB\_BA\_UV\_0 Instances**

Instance	Physical Address
DSS0_WB	04AF 0030h

**Figure 12-941. DSS0\_WB\_BA\_UV\_0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BA																R/W-0h															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1234. DSS0\_WB\_BA\_UV\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	BA	R/W	0h	WB base address aligned on 16-bit boundary. Base address of the UV WB buffer

### 12.6.4.11.13.5.14 DSS0\_WB\_BA\_UV\_1 Register (Offset = 34h) [reset = 0h]

DSS0\_WB\_BA\_UV\_1 is shown in [Figure 12-942](#) and described in [Table 12-1236](#).

Return to [Summary Table](#).

The register configures the base address of the UV buffer for the write-back pipeline. DISPC\_WB\_BA\_UV\_\_0 & DISPC\_WB\_BA\_UV\_\_1 for ping-pong mechanism with external trigger, based on the field polarity, otherwise only DISPC\_WB\_BA\_UV\_\_0 is used. The register is also used to configure the RGB plane BA for RGB565A8 format. Shadow register

**Table 12-1235. DSS0\_WB\_BA\_UV\_1 Instances**

Instance	Physical Address
DSS0_WB	04AF 0034h

**Figure 12-942. DSS0\_WB\_BA\_UV\_1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BA																															
R/W-0h																															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1236. DSS0\_WB\_BA\_UV\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	BA	R/W	0h	WB base address aligned on 16-bit boundary. Base address of the UV WB buffer

#### 12.6.4.11.13.5.15 DSS0\_WB\_BUF\_SIZE\_STATUS Register (Offset = 38h) [reset = 1000h]

DSS0\_WB\_BUF\_SIZE\_STATUS is shown in [Figure 12-943](#) and described in [Table 12-1238](#).

Return to [Summary Table](#).

The register defines the DMA buffer DSS0\_WB\_SIZE for the write back pipeline

**Table 12-1237. DSS0\_WB\_BUF\_SIZE\_STATUS  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0038h

**Figure 12-943. DSS0\_WB\_BUF\_SIZE\_STATUS Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																BUFSIZE															
R-0h																R-1000h															

LEGEND: R = Read Only; -n = value after reset

**Table 12-1238. DSS0\_WB\_BUF\_SIZE\_STATUS Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Write 0's for future compatibility. Reads return 0
15-0	BUFSIZE	R	1000h	DMA buffer DSS0_WB_SIZE in number of 128-bits

### 12.6.4.11.13.5.16 DSS0\_WB\_BUF\_THRESHOLD Register (Offset = 3Ch) [reset = 0FFF0FF8h]

DSS0\_WB\_BUF\_THRESHOLD is shown in [Figure 12-944](#) and described in [Table 12-1240](#).

Return to [Summary Table](#).

The register configures the DMA buffer associated with the write-back pipeline. Shadow register

**Table 12-1239. DSS0\_WB\_BUF\_THRESHOLD Instances**

Instance	Physical Address
DSS0_WB	04AF 003Ch

**Figure 12-944. DSS0\_WB\_BUF\_THRESHOLD Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUFHIGHTHRESHOLD																BUFLOWTHRESHOLD															
R/W-FFFh																R/W-FF8h															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1240. DSS0\_WB\_BUF\_THRESHOLD Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	BUFHIGHTHRESHOLD	R/W	FFFh	DMA buffer High Threshold Number of 128-bits defining the threshold value
15-0	BUFLOWTHRESHOLD	R/W	FF8h	DMA buffer High Threshold Number of 128-bits defining the threshold value



### 12.6.4.11.13.5.17 DSS0\_WB\_CSC\_COEF0 Register (Offset = 40h) [reset = 0h]

DSS0\_WB\_CSC\_COEF0 is shown in [Figure 12-945](#) and described in [Table 12-1242](#).

Return to [Summary Table](#).

The register configures the color space conversion matrix coefficients. Shadow register

**Table 12-1241. DSS0\_WB\_CSC\_COEF0 Instances**

Instance	Physical Address
DSS0_WB	04AF 0040h

**Figure 12-945. DSS0\_WB\_CSC\_COEF0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED					C01										
R-0h					R/W-0h										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED					C00										
R-0h					R/W-0h										

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1242. DSS0\_WB\_CSC\_COEF0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
26-16	C01	R/W	0h	C01 Coefficient. Encoded signed value [from -1024 to 1023]
15-11	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
10-0	C00	R/W	0h	C00 Coefficient. Encoded signed value [from -1024 to 1023]

### 12.6.4.11.13.5.18 DSS0\_WB\_CSC\_COEF1 Register (Offset = 44h) [reset = 0h]

DSS0\_WB\_CSC\_COEF1 is shown in [Figure 12-946](#) and described in [Table 12-1244](#).

Return to [Summary Table](#).

The register configures the color space conversion matrix coefficients. Shadow register

**Table 12-1243. DSS0\_WB\_CSC\_COEF1 Instances**

Instance	Physical Address
DSS0_WB	04AF 0044h

**Figure 12-946. DSS0\_WB\_CSC\_COEF1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED								C10							
R-0h								R/W-0h							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								C02							
R-0h								R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1244. DSS0\_WB\_CSC\_COEF1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
26-16	C10	R/W	0h	C10 Coefficient. Encoded signed value [from -1024 to 1023]
15-11	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
10-0	C02	R/W	0h	C02 Coefficient. Encoded signed value [from -1024 to 1023]

### 12.6.4.11.13.5.19 DSS0\_WB\_CSC\_COEF2 Register (Offset = 48h) [reset = 0h]

DSS0\_WB\_CSC\_COEF2 is shown in [Figure 12-947](#) and described in [Table 12-1246](#).

Return to [Summary Table](#).

The register configures the color space conversion matrix coefficients. Shadow register

**Table 12-1245. DSS0\_WB\_CSC\_COEF2 Instances**

Instance	Physical Address
DSS0_WB	04AF 0048h

**Figure 12-947. DSS0\_WB\_CSC\_COEF2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED								C12							
R-0h								R/W-0h							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								C11							
R-0h								R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1246. DSS0\_WB\_CSC\_COEF2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
26-16	C12	R/W	0h	C12 Coefficient. Encoded signed value [from -1024 to 1023]
15-11	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
10-0	C11	R/W	0h	C11 Coefficient. Encoded signed value [from -1024 to 1023]

### 12.6.4.11.13.5.20 DSS0\_WB\_CSC\_COEF3 Register (Offset = 4Ch) [reset = 0h]

DSS0\_WB\_CSC\_COEF3 is shown in [Figure 12-948](#) and described in [Table 12-1248](#).

Return to [Summary Table](#).

The register configures the color space conversion matrix coefficients. Shadow register

**Table 12-1247. DSS0\_WB\_CSC\_COEF3 Instances**

Instance	Physical Address
DSS0_WB	04AF 004Ch

**Figure 12-948. DSS0\_WB\_CSC\_COEF3 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED					C21										
R-0h					R/W-0h										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED					C20										
R-0h					R/W-0h										

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1248. DSS0\_WB\_CSC\_COEF3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
26-16	C21	R/W	0h	C21 coefficient. Encoded signed value [from -1024 to 1023]
15-11	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
10-0	C20	R/W	0h	C20 coefficient. Encoded signed value [from -1024 to 1023]

### 12.6.4.11.13.5.21 DSS0\_WB\_CSC\_COEF4 Register (Offset = 50h) [reset = 0h]

DSS0\_WB\_CSC\_COEF4 is shown in [Figure 12-949](#) and described in [Table 12-1250](#).

Return to [Summary Table](#).

The register configures the color space conversion matrix coefficients. Shadow register

**Table 12-1249. DSS0\_WB\_CSC\_COEF4 Instances**

Instance	Physical Address
DSS0_WB	04AF 0050h

**Figure 12-949. DSS0\_WB\_CSC\_COEF4 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																					C22										
R-0h																					R/W-0h										

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1250. DSS0\_WB\_CSC\_COEF4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-11	RESERVED	R	0h	Write 0's for future compatibility Reads return 0
10-0	C22	R/W	0h	C22 Coefficient. Encoded signed value [from -1024 to 1023]

### 12.6.4.11.13.5.22 DSS0\_WB\_CSC\_COEF5 Register (Offset = 54h) [reset = 0h]

DSS0\_WB\_CSC\_COEF5 is shown in [Figure 12-950](#) and described in [Table 12-1252](#).

Return to [Summary Table](#).

The register configures the color space conversion matrix coefficients. Shadow register

**Table 12-1251. DSS0\_WB\_CSC\_COEF5 Instances**

Instance	Physical Address
DSS0_WB	04AF 0054h

**Figure 12-950. DSS0\_WB\_CSC\_COEF5 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PREOFFSET2												RESERVED			
R/W-0h												R-0h			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PREOFFSET1												RESERVED			
R/W-0h												R-0h			

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1252. DSS0\_WB\_CSC\_COEF5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-19	PREOFFSET2	R/W	0h	Row-2 pre-offset. Encoded signed value [from -4096 to 4095]
18-16	RESERVED	R	0h	Reserved
15-3	PREOFFSET1	R/W	0h	Row1 pre-offset. Encoded signed value [from -4096 to 4095]
2-0	RESERVED	R	0h	Reserved

#### 12.6.4.11.13.5.23 DSS0\_WB\_CSC\_COEF6 Register (Offset = 58h) [reset = 0h]

DSS0\_WB\_CSC\_COEF6 is shown in [Figure 12-951](#) and described in [Table 12-1254](#).

Return to [Summary Table](#).

The register configures the color space conversion matrix coefficients. Shadow register

**Table 12-1253. DSS0\_WB\_CSC\_COEF6 Instances**

Instance	Physical Address
DSS0_WB	04AF 0058h

**Figure 12-951. DSS0\_WB\_CSC\_COEF6 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
POSTOFFSET1												RESERVED			
R/W-0h												R-0h			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PREOFFSET3												RESERVED			
R/W-0h												R-0h			

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1254. DSS0\_WB\_CSC\_COEF6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-19	POSTOFFSET1	R/W	0h	Row-1 post-offset. Encoded signed value [from -4096 to 4095]
18-16	RESERVED	R	0h	Reserved
15-3	PREOFFSET3	R/W	0h	Row-3 pre-offset. Encoded signed value [from -4096 to 4095]
2-0	RESERVED	R	0h	Reserved

#### 12.6.4.11.13.5.24 DSS0\_WB\_FIRH Register (Offset = 5Ch) [reset = 00200000h]

DSS0\_WB\_FIRH is shown in [Figure 12-952](#) and described in [Table 12-1256](#).

Return to [Summary Table](#).

The register configures the resize factor for horizontal up/down-sampling of the write-back window. It is used for ARGB and Y setting. Shadow register

**Table 12-1255. DSS0\_WB\_FIRH Instances**

Instance	Physical Address
DSS0_WB	04AF 005Ch

**Figure 12-952. DSS0\_WB\_FIRH Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								FIRHINC																							
R-0h								R/W-00200000h																							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1256. DSS0\_WB\_FIRH Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-0	FIRHINC	R/W	00200000h	Horizontal increment of the up/down-sampling filter. The value 0 is invalid



#### 12.6.4.11.13.5.25 DSS0\_WB\_FIRH2 Register (Offset = 60h) [reset = 00200000h]

DSS0\_WB\_FIRH2 is shown in [Figure 12-953](#) and described in [Table 12-1258](#).

Return to [Summary Table](#).

The register configures the resize factor for horizontal up/down-sampling of the write-back window. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1257. DSS0\_WB\_FIRH2 Instances**

Instance	Physical Address
DSS0_WB	04AF 0060h

**Figure 12-953. DSS0\_WB\_FIRH2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								FIRHINC																							
R-0h								R/W-00200000h																							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1258. DSS0\_WB\_FIRH2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-0	FIRHINC	R/W	00200000h	Horizontal increment of the up/down-sampling filter for Cb and Cr. The value 0 is invalid

### 12.6.4.11.13.5.26 DSS0\_WB\_FIRV Register (Offset = 64h) [reset = 00200000h]

DSS0\_WB\_FIRV is shown in [Figure 12-954](#) and described in [Table 12-1260](#).

Return to [Summary Table](#).

The register configures the resize factor for vertical up/down-sampling of the write-back window. It is used for ARGB and Y setting. Shadow register

**Table 12-1259. DSS0\_WB\_FIRV Instances**

Instance	Physical Address
DSS0_WB	04AF 0064h

**Figure 12-954. DSS0\_WB\_FIRV Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								FIRVINC																							
R-0h								R/W-00200000h																							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1260. DSS0\_WB\_FIRV Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-0	FIRVINC	R/W	00200000h	Vertical increment of the up/down-sampling filter. The value 0 is invalid

#### 12.6.4.11.13.5.27 DSS0\_WB\_FIRV2 Register (Offset = 68h) [reset = 00200000h]

DSS0\_WB\_FIRV2 is shown in [Figure 12-955](#) and described in [Table 12-1262](#).

Return to [Summary Table](#).

The register configures the resize factor for vertical up/down-sampling of the write-back window. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1261. DSS0\_WB\_FIRV2 Instances**

Instance	Physical Address
DSS0_WB	04AF 0068h

**Figure 12-955. DSS0\_WB\_FIRV2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								FIRVINC																							
R-0h								R/W-00200000h																							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1262. DSS0\_WB\_FIRV2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reserved
23-0	FIRVINC	R/W	00200000h	Vertical increment of the up/down-sampling filter for Cb and Cr. The value 0 is invalid

### 12.6.4.11.13.5.28 DSS0\_WB\_FIR\_COEF\_H0\_0 Register (Offset = 6Ch) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H0\_0 is shown in [Figure 12-956](#) and described in [Table 12-1264](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1263. DSS0\_WB\_FIR\_COEF\_H0\_0  
Instances**

Instance	Physical Address
DSS0_WB	04AF 006Ch

**Figure 12-956. DSS0\_WB\_FIR\_COEF\_H0\_0 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1264. DSS0\_WB\_FIR\_COEF\_H0\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase n

### 12.6.4.11.13.5.29 DSS0\_WB\_FIR\_COEF\_H0\_1 Register (Offset = 70h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H0\_1 is shown in [Figure 12-957](#) and described in [Table 12-1266](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1265. DSS0\_WB\_FIR\_COEF\_H0\_1  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0070h

**Figure 12-957. DSS0\_WB\_FIR\_COEF\_H0\_1 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1266. DSS0\_WB\_FIR\_COEF\_H0\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase n

### 12.6.4.11.13.5.30 DSS0\_WB\_FIR\_COEF\_H0\_2 Register (Offset = 74h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H0\_2 is shown in [Figure 12-958](#) and described in [Table 12-1268](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1267. DSS0\_WB\_FIR\_COEF\_H0\_2  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0074h

**Figure 12-958. DSS0\_WB\_FIR\_COEF\_H0\_2 Register**

31	30	29	28	27	26	25	24
RESERVED		RESERVED					
R-0h		R-0h					
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1268. DSS0\_WB\_FIR\_COEF\_H0\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase n

### 12.6.4.11.13.5.31 DSS0\_WB\_FIR\_COEF\_H0\_3 Register (Offset = 78h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H0\_3 is shown in [Figure 12-959](#) and described in [Table 12-1270](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1269. DSS0\_WB\_FIR\_COEF\_H0\_3  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0078h

**Figure 12-959. DSS0\_WB\_FIR\_COEF\_H0\_3 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1270. DSS0\_WB\_FIR\_COEF\_H0\_3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase n

### 12.6.4.11.13.5.32 DSS0\_WB\_FIR\_COEF\_H0\_4 Register (Offset = 7Ch) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H0\_4 is shown in [Figure 12-960](#) and described in [Table 12-1272](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1271. DSS0\_WB\_FIR\_COEF\_H0\_4  
Instances**

Instance	Physical Address
DSS0_WB	04AF 007Ch

**Figure 12-960. DSS0\_WB\_FIR\_COEF\_H0\_4 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1272. DSS0\_WB\_FIR\_COEF\_H0\_4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase n



### 12.6.4.11.13.5.33 DSS0\_WB\_FIR\_COEF\_H0\_5 Register (Offset = 80h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H0\_5 is shown in [Figure 12-961](#) and described in [Table 12-1274](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1273. DSS0\_WB\_FIR\_COEF\_H0\_5  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0080h

**Figure 12-961. DSS0\_WB\_FIR\_COEF\_H0\_5 Register**

31	30	29	28	27	26	25	24
RESERVED		RESERVED					
R-0h		R-0h					
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1274. DSS0\_WB\_FIR\_COEF\_H0\_5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase n

### 12.6.4.11.13.5.34 DSS0\_WB\_FIR\_COEF\_H0\_6 Register (Offset = 84h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H0\_6 is shown in [Figure 12-962](#) and described in [Table 12-1276](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1275. DSS0\_WB\_FIR\_COEF\_H0\_6  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0084h

**Figure 12-962. DSS0\_WB\_FIR\_COEF\_H0\_6 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1276. DSS0\_WB\_FIR\_COEF\_H0\_6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase n

#### 12.6.4.11.13.5.35 DSS0\_WB\_FIR\_COEF\_H0\_7 Register (Offset = 88h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H0\_7 is shown in [Figure 12-963](#) and described in [Table 12-1278](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1277. DSS0\_WB\_FIR\_COEF\_H0\_7  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0088h

**Figure 12-963. DSS0\_WB\_FIR\_COEF\_H0\_7 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1278. DSS0\_WB\_FIR\_COEF\_H0\_7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase n

### 12.6.4.11.13.5.36 DSS0\_WB\_FIR\_COEF\_H0\_8 Register (Offset = 8Ch) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H0\_8 is shown in [Figure 12-964](#) and described in [Table 12-1280](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1279. DSS0\_WB\_FIR\_COEF\_H0\_8  
Instances**

Instance	Physical Address
DSS0_WB	04AF 008Ch

**Figure 12-964. DSS0\_WB\_FIR\_COEF\_H0\_8 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1280. DSS0\_WB\_FIR\_COEF\_H0\_8 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase n

#### 12.6.4.11.13.5.37 DSS0\_WB\_FIR\_COEF\_H0\_C\_0 Register (Offset = 90h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H0\_C\_0 is shown in [Figure 12-965](#) and described in [Table 12-1282](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1281. DSS0\_WB\_FIR\_COEF\_H0\_C\_0  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0090h

**Figure 12-965. DSS0\_WB\_FIR\_COEF\_H0\_C\_0 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1282. DSS0\_WB\_FIR\_COEF\_H0\_C\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase n

### 12.6.4.11.13.5.38 DSS0\_WB\_FIR\_COEF\_H0\_C\_1 Register (Offset = 94h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H0\_C\_1 is shown in [Figure 12-966](#) and described in [Table 12-1284](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter. If the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1283. DSS0\_WB\_FIR\_COEF\_H0\_C\_1  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0094h

**Figure 12-966. DSS0\_WB\_FIR\_COEF\_H0\_C\_1 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1284. DSS0\_WB\_FIR\_COEF\_H0\_C\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase n

### 12.6.4.11.13.5.39 DSS0\_WB\_FIR\_COEF\_H0\_C\_2 Register (Offset = 98h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H0\_C\_2 is shown in [Figure 12-967](#) and described in [Table 12-1286](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1285. DSS0\_WB\_FIR\_COEF\_H0\_C\_2  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0098h

**Figure 12-967. DSS0\_WB\_FIR\_COEF\_H0\_C\_2 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1286. DSS0\_WB\_FIR\_COEF\_H0\_C\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase n

### 12.6.4.11.13.5.40 DSS0\_WB\_FIR\_COEF\_H0\_C\_3 Register (Offset = 9Ch) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H0\_C\_3 is shown in [Figure 12-968](#) and described in [Table 12-1288](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter. If the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1287. DSS0\_WB\_FIR\_COEF\_H0\_C\_3  
Instances**

Instance	Physical Address
DSS0_WB	04AF 009Ch

**Figure 12-968. DSS0\_WB\_FIR\_COEF\_H0\_C\_3 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1288. DSS0\_WB\_FIR\_COEF\_H0\_C\_3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase n



#### 12.6.4.11.13.5.41 DSS0\_WB\_FIR\_COEF\_H0\_C\_4 Register (Offset = A0h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H0\_C\_4 is shown in [Figure 12-969](#) and described in [Table 12-1290](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1289. DSS0\_WB\_FIR\_COEF\_H0\_C\_4  
Instances**

Instance	Physical Address
DSS0_WB	04AF 00A0h

**Figure 12-969. DSS0\_WB\_FIR\_COEF\_H0\_C\_4 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1290. DSS0\_WB\_FIR\_COEF\_H0\_C\_4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase n

#### 12.6.4.11.13.5.42 DSS0\_WB\_FIR\_COEF\_H0\_C\_5 Register (Offset = A4h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H0\_C\_5 is shown in [Figure 12-970](#) and described in [Table 12-1292](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter. If the pixel format at the input of the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1291. DSS0\_WB\_FIR\_COEF\_H0\_C\_5  
Instances**

Instance	Physical Address
DSS0_WB	04AF 00A4h

**Figure 12-970. DSS0\_WB\_FIR\_COEF\_H0\_C\_5 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1292. DSS0\_WB\_FIR\_COEF\_H0\_C\_5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase n

#### 12.6.4.11.13.5.43 DSS0\_WB\_FIR\_COEF\_H0\_C\_6 Register (Offset = A8h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H0\_C\_6 is shown in [Figure 12-971](#) and described in [Table 12-1294](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1293. DSS0\_WB\_FIR\_COEF\_H0\_C\_6  
Instances**

Instance	Physical Address
DSS0_WB	04AF 00A8h

**Figure 12-971. DSS0\_WB\_FIR\_COEF\_H0\_C\_6 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1294. DSS0\_WB\_FIR\_COEF\_H0\_C\_6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase n

#### 12.6.4.11.13.5.44 DSS0\_WB\_FIR\_COEF\_H0\_C\_7 Register (Offset = ACh) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H0\_C\_7 is shown in [Figure 12-972](#) and described in [Table 12-1296](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter. If the pixel format at the input of the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1295. DSS0\_WB\_FIR\_COEF\_H0\_C\_7  
Instances**

Instance	Physical Address
DSS0_WB	04AF 00ACh

**Figure 12-972. DSS0\_WB\_FIR\_COEF\_H0\_C\_7 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1296. DSS0\_WB\_FIR\_COEF\_H0\_C\_7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase n

#### 12.6.4.11.13.5.45 DSS0\_WB\_FIR\_COEF\_H0\_C\_8 Register (Offset = B0h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H0\_C\_8 is shown in [Figure 12-973](#) and described in [Table 12-1298](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1297. DSS0\_WB\_FIR\_COEF\_H0\_C\_8  
Instances**

Instance	Physical Address
DSS0_WB	04AF 00B0h

**Figure 12-973. DSS0\_WB\_FIR\_COEF\_H0\_C\_8 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRHC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRHC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1298. DSS0\_WB\_FIR\_COEF\_H0\_C\_8 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRHC0	R/W	0h	Unsigned coefficient C0 for the horizontal up/down-scaling with the phase n

### 12.6.4.11.13.5.46 DSS0\_WB\_FIR\_COEF\_H12\_0 Register (Offset = B4h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H12\_0 is shown in [Figure 12-974](#) and described in [Table 12-1300](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1299. DSS0\_WB\_FIR\_COEF\_H12\_0  
Instances**

Instance	Physical Address
DSS0_WB	04AF 00B4h

**Figure 12-974. DSS0\_WB\_FIR\_COEF\_H12\_0 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1300. DSS0\_WB\_FIR\_COEF\_H12\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase n
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

#### 12.6.4.11.13.5.47 DSS0\_WB\_FIR\_COEF\_H12\_1 Register (Offset = B8h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H12\_1 is shown in [Figure 12-975](#) and described in [Table 12-1302](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1301. DSS0\_WB\_FIR\_COEF\_H12\_1  
Instances**

Instance	Physical Address
DSS0_WB	04AF 00B8h

**Figure 12-975. DSS0\_WB\_FIR\_COEF\_H12\_1 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1302. DSS0\_WB\_FIR\_COEF\_H12\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase n
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.48 DSS0\_WB\_FIR\_COEF\_H12\_2 Register (Offset = BCh) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H12\_2 is shown in [Figure 12-976](#) and described in [Table 12-1304](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1303. DSS0\_WB\_FIR\_COEF\_H12\_2  
Instances**

Instance	Physical Address
DSS0_WB	04AF 00BCh

**Figure 12-976. DSS0\_WB\_FIR\_COEF\_H12\_2 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1304. DSS0\_WB\_FIR\_COEF\_H12\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase n
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved



### 12.6.4.11.13.5.49 DSS0\_WB\_FIR\_COEF\_H12\_3 Register (Offset = C0h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H12\_3 is shown in [Figure 12-977](#) and described in [Table 12-1306](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1305. DSS0\_WB\_FIR\_COEF\_H12\_3  
Instances**

Instance	Physical Address
DSS0_WB	04AF 00C0h

**Figure 12-977. DSS0\_WB\_FIR\_COEF\_H12\_3 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1306. DSS0\_WB\_FIR\_COEF\_H12\_3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase n
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.50 DSS0\_WB\_FIR\_COEF\_H12\_4 Register (Offset = C4h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H12\_4 is shown in [Figure 12-978](#) and described in [Table 12-1308](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1307. DSS0\_WB\_FIR\_COEF\_H12\_4  
Instances**

Instance	Physical Address
DSS0_WB	04AF 00C4h

**Figure 12-978. DSS0\_WB\_FIR\_COEF\_H12\_4 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1308. DSS0\_WB\_FIR\_COEF\_H12\_4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase n
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

#### 12.6.4.11.13.5.51 DSS0\_WB\_FIR\_COEF\_H12\_5 Register (Offset = C8h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H12\_5 is shown in [Figure 12-979](#) and described in [Table 12-1310](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1309. DSS0\_WB\_FIR\_COEF\_H12\_5  
Instances**

Instance	Physical Address
DSS0_WB	04AF 00C8h

**Figure 12-979. DSS0\_WB\_FIR\_COEF\_H12\_5 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1310. DSS0\_WB\_FIR\_COEF\_H12\_5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase n
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.52 DSS0\_WB\_FIR\_COEF\_H12\_6 Register (Offset = CCh) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H12\_6 is shown in [Figure 12-980](#) and described in [Table 12-1312](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1311. DSS0\_WB\_FIR\_COEF\_H12\_6  
Instances**

Instance	Physical Address
DSS0_WB	04AF 00CCh

**Figure 12-980. DSS0\_WB\_FIR\_COEF\_H12\_6 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1312. DSS0\_WB\_FIR\_COEF\_H12\_6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase n
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.53 DSS0\_WB\_FIR\_COEF\_H12\_7 Register (Offset = D0h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H12\_7 is shown in [Figure 12-981](#) and described in [Table 12-1314](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1313. DSS0\_WB\_FIR\_COEF\_H12\_7  
Instances**

Instance	Physical Address
DSS0_WB	04AF 00D0h

**Figure 12-981. DSS0\_WB\_FIR\_COEF\_H12\_7 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1314. DSS0\_WB\_FIR\_COEF\_H12\_7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase n
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.54 DSS0\_WB\_FIR\_COEF\_H12\_8 Register (Offset = D4h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H12\_8 is shown in [Figure 12-982](#) and described in [Table 12-1316](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1315. DSS0\_WB\_FIR\_COEF\_H12\_8  
Instances**

Instance	Physical Address
DSS0_WB	04AF 00D4h

**Figure 12-982. DSS0\_WB\_FIR\_COEF\_H12\_8 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1316. DSS0\_WB\_FIR\_COEF\_H12\_8 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase n
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

#### 12.6.4.11.13.5.55 DSS0\_WB\_FIR\_COEF\_H12\_9 Register (Offset = D8h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H12\_9 is shown in [Figure 12-983](#) and described in [Table 12-1318](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1317. DSS0\_WB\_FIR\_COEF\_H12\_9  
Instances**

Instance	Physical Address
DSS0_WB	04AF 00D8h

**Figure 12-983. DSS0\_WB\_FIR\_COEF\_H12\_9 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1318. DSS0\_WB\_FIR\_COEF\_H12\_9 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase n
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.56 DSS0\_WB\_FIR\_COEF\_H12\_10 Register (Offset = DCh) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H12\_9 is shown in [Figure 12-984](#) and described in [Table 12-1320](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1319. DSS0\_WB\_FIR\_COEF\_H12\_10  
Instances**

Instance	Physical Address
DSS0_WB	04AF 00DCh

**Figure 12-984. DSS0\_WB\_FIR\_COEF\_H12\_10 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1320. DSS0\_WB\_FIR\_COEF\_H12\_10 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase n
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved



#### 12.6.4.11.13.5.57 DSS0\_WB\_FIR\_COEF\_H12\_11 Register (Offset = E0h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H12\_11 is shown in [Figure 12-985](#) and described in [Table 12-1322](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1321. DSS0\_WB\_FIR\_COEF\_H12\_11  
Instances**

Instance	Physical Address
DSS0_WB	04AF 00E0h

**Figure 12-985. DSS0\_WB\_FIR\_COEF\_H12\_11 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1322. DSS0\_WB\_FIR\_COEF\_H12\_11 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase n
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.58 DSS0\_WB\_FIR\_COEF\_H12\_12 Register (Offset = E4h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H12\_12 is shown in [Figure 12-986](#) and described in [Table 12-1324](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1323. DSS0\_WB\_FIR\_COEF\_H12\_12  
Instances**

Instance	Physical Address
DSS0_WB	04AF 00E4h

**Figure 12-986. DSS0\_WB\_FIR\_COEF\_H12\_12 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1324. DSS0\_WB\_FIR\_COEF\_H12\_12 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase n
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

#### 12.6.4.11.13.5.59 DSS0\_WB\_FIR\_COEF\_H12\_13 Register (Offset = E8h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H12\_13 is shown in [Figure 12-987](#) and described in [Table 12-1326](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1325. DSS0\_WB\_FIR\_COEF\_H12\_13  
Instances**

Instance	Physical Address
DSS0_WB	04AF 00E8h

**Figure 12-987. DSS0\_WB\_FIR\_COEF\_H12\_13 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1326. DSS0\_WB\_FIR\_COEF\_H12\_13 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase n
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.60 DSS0\_WB\_FIR\_COEF\_H12\_14 Register (Offset = ECh) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H12\_14 is shown in [Figure 12-988](#) and described in [Table 12-1328](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1327. DSS0\_WB\_FIR\_COEF\_H12\_14  
Instances**

Instance	Physical Address
DSS0_WB	04AF 00ECh

**Figure 12-988. DSS0\_WB\_FIR\_COEF\_H12\_14 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1328. DSS0\_WB\_FIR\_COEF\_H12\_14 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase n
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.61 DSS0\_WB\_FIR\_COEF\_H12\_15 Register (Offset = F0h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H12\_15 is shown in [Figure 12-989](#) and described in [Table 12-1330](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1329. DSS0\_WB\_FIR\_COEF\_H12\_15  
Instances**

Instance	Physical Address
DSS0_WB	04AF 00F0h

**Figure 12-989. DSS0\_WB\_FIR\_COEF\_H12\_15 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1330. DSS0\_WB\_FIR\_COEF\_H12\_15 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase n
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.62 DSS0\_WB\_FIR\_COEF\_H12\_C\_0 Register (Offset = F4h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H12\_C\_0 is shown in [Figure 12-990](#) and described in [Table 12-1332](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1331. DSS0\_WB\_FIR\_COEF\_H12\_C\_0  
Instances**

Instance	Physical Address
DSS0_WB	04AF 00F4h

**Figure 12-990. DSS0\_WB\_FIR\_COEF\_H12\_C\_0 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1332. DSS0\_WB\_FIR\_COEF\_H12\_C\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase n
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.63 DSS0\_WB\_FIR\_COEF\_H12\_C\_1 Register (Offset = F8h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H12\_C\_1 is shown in [Figure 12-991](#) and described in [Table 12-1334](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1333. DSS0\_WB\_FIR\_COEF\_H12\_C\_1  
Instances**

Instance	Physical Address
DSS0_WB	04AF 00F8h

**Figure 12-991. DSS0\_WB\_FIR\_COEF\_H12\_C\_1 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1334. DSS0\_WB\_FIR\_COEF\_H12\_C\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase n
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.64 DSS0\_WB\_FIR\_COEF\_H12\_C\_2 Register (Offset = FCh) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H12\_C\_2 is shown in [Figure 12-992](#) and described in [Table 12-1336](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter. If the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1335. DSS0\_WB\_FIR\_COEF\_H12\_C\_2  
Instances**

Instance	Physical Address
DSS0_WB	04AF 00FCh

**Figure 12-992. DSS0\_WB\_FIR\_COEF\_H12\_C\_2 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1336. DSS0\_WB\_FIR\_COEF\_H12\_C\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase n
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved



### 12.6.4.11.13.5.65 DSS0\_WB\_FIR\_COEF\_H12\_C\_3 Register (Offset = 100h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H12\_C\_3 is shown in [Figure 12-993](#) and described in [Table 12-1338](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1337. DSS0\_WB\_FIR\_COEF\_H12\_C\_3  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0100h

**Figure 12-993. DSS0\_WB\_FIR\_COEF\_H12\_C\_3 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1338. DSS0\_WB\_FIR\_COEF\_H12\_C\_3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase n
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.66 DSS0\_WB\_FIR\_COEF\_H12\_C\_4 Register (Offset = 104h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H12\_C\_4 is shown in [Figure 12-994](#) and described in [Table 12-1340](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1339. DSS0\_WB\_FIR\_COEF\_H12\_C\_4  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0104h

**Figure 12-994. DSS0\_WB\_FIR\_COEF\_H12\_C\_4 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1340. DSS0\_WB\_FIR\_COEF\_H12\_C\_4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase n
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.67 DSS0\_WB\_FIR\_COEF\_H12\_C\_5 Register (Offset = 108h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H12\_C\_5 is shown in [Figure 12-995](#) and described in [Table 12-1342](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1341. DSS0\_WB\_FIR\_COEF\_H12\_C\_5  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0108h

**Figure 12-995. DSS0\_WB\_FIR\_COEF\_H12\_C\_5 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1342. DSS0\_WB\_FIR\_COEF\_H12\_C\_5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase n
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.68 DSS0\_WB\_FIR\_COEF\_H12\_C\_6 Register (Offset = 10Ch) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H12\_C\_6 is shown in [Figure 12-996](#) and described in [Table 12-1344](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1343. DSS0\_WB\_FIR\_COEF\_H12\_C\_6  
Instances**

Instance	Physical Address
DSS0_WB	04AF 010Ch

**Figure 12-996. DSS0\_WB\_FIR\_COEF\_H12\_C\_6 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1344. DSS0\_WB\_FIR\_COEF\_H12\_C\_6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase n
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.69 DSS0\_WB\_FIR\_COEF\_H12\_C\_7 Register (Offset = 110h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H12\_C\_7 is shown in [Figure 12-997](#) and described in [Table 12-1346](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1345. DSS0\_WB\_FIR\_COEF\_H12\_C\_7 Instances**

Instance	Physical Address
DSS0_WB	04AF 0110h

**Figure 12-997. DSS0\_WB\_FIR\_COEF\_H12\_C\_7 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1346. DSS0\_WB\_FIR\_COEF\_H12\_C\_7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase n
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.70 DSS0\_WB\_FIR\_COEF\_H12\_C\_8 Register (Offset = 114h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H12\_C\_8 is shown in [Figure 12-998](#) and described in [Table 12-1348](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter. If the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1347. DSS0\_WB\_FIR\_COEF\_H12\_C\_8  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0114h

**Figure 12-998. DSS0\_WB\_FIR\_COEF\_H12\_C\_8 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1348. DSS0\_WB\_FIR\_COEF\_H12\_C\_8 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase n
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.71 DSS0\_WB\_FIR\_COEF\_H12\_C\_9 Register (Offset = 118h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H12\_C\_9 is shown in [Figure 12-999](#) and described in [Table 12-1350](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1349. DSS0\_WB\_FIR\_COEF\_H12\_C\_9  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0118h

**Figure 12-999. DSS0\_WB\_FIR\_COEF\_H12\_C\_9 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1350. DSS0\_WB\_FIR\_COEF\_H12\_C\_9 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase n
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.72 DSS0\_WB\_FIR\_COEF\_H12\_C\_10 Register (Offset = 11Ch) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H12\_C\_10 is shown in [Figure 12-1000](#) and described in [Table 12-1352](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1351. DSS0\_WB\_FIR\_COEF\_H12\_C\_10  
Instances**

Instance	Physical Address
DSS0_WB	04AF 011Ch

**Figure 12-1000. DSS0\_WB\_FIR\_COEF\_H12\_C\_10 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1352. DSS0\_WB\_FIR\_COEF\_H12\_C\_10 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase n
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved



### 12.6.4.11.13.5.73 DSS0\_WB\_FIR\_COEF\_H12\_C\_11 Register (Offset = 120h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H12\_C\_11 is shown in [Figure 12-1001](#) and described in [Table 12-1354](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter. If the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1353. DSS0\_WB\_FIR\_COEF\_H12\_C\_11  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0120h

**Figure 12-1001. DSS0\_WB\_FIR\_COEF\_H12\_C\_11 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1354. DSS0\_WB\_FIR\_COEF\_H12\_C\_11 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase n
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

#### 12.6.4.11.13.5.74 DSS0\_WB\_FIR\_COEF\_H12\_C\_12 Register (Offset = 124h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H12\_C\_12 is shown in [Figure 12-1002](#) and described in [Table 12-1356](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter. If the pixel format at the input of the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1355. DSS0\_WB\_FIR\_COEF\_H12\_C\_12  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0124h

**Figure 12-1002. DSS0\_WB\_FIR\_COEF\_H12\_C\_12 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1356. DSS0\_WB\_FIR\_COEF\_H12\_C\_12 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase n
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.75 DSS0\_WB\_FIR\_COEF\_H12\_C\_13 Register (Offset = 128h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H12\_C\_13 is shown in [Figure 12-1003](#) and described in [Table 12-1358](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1357. DSS0\_WB\_FIR\_COEF\_H12\_C\_13  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0128h

**Figure 12-1003. DSS0\_WB\_FIR\_COEF\_H12\_C\_13 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1358. DSS0\_WB\_FIR\_COEF\_H12\_C\_13 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase n
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.76 DSS0\_WB\_FIR\_COEF\_H12\_C\_14 Register (Offset = 12Ch) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H12\_C\_14 is shown in [Figure 12-1004](#) and described in [Table 12-1360](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1359. DSS0\_WB\_FIR\_COEF\_H12\_C\_14  
Instances**

Instance	Physical Address
DSS0_WB	04AF 012Ch

**Figure 12-1004. DSS0\_WB\_FIR\_COEF\_H12\_C\_14 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1360. DSS0\_WB\_FIR\_COEF\_H12\_C\_14 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase n
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.77 DSS0\_WB\_FIR\_COEF\_H12\_C\_15 Register (Offset = 130h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_H12\_C\_15 is shown in [Figure 12-1005](#) and described in [Table 12-1362](#).

Return to [Summary Table](#).

The bank of registers configures the up/down-scaling coefficients for the horizontal resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1361. DSS0\_WB\_FIR\_COEF\_H12\_C\_15  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0130h

**Figure 12-1005. DSS0\_WB\_FIR\_COEF\_H12\_C\_15 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRHC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRHC2				FIRHC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRHC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1362. DSS0\_WB\_FIR\_COEF\_H12\_C\_15 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRHC2	R/W	0h	Signed coefficient C2 for the horizontal up/down-scaling with the phase n
19-10	FIRHC1	R/W	0h	Signed coefficient C1 for the horizontal up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.78 DSS0\_WB\_FIR\_COEF\_V0\_0 Register (Offset = 134h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V0\_0 is shown in [Figure 12-1006](#) and described in [Table 12-1364](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1363. DSS0\_WB\_FIR\_COEF\_V0\_0  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0134h

**Figure 12-1006. DSS0\_WB\_FIR\_COEF\_V0\_0 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1364. DSS0\_WB\_FIR\_COEF\_V0\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase n

### 12.6.4.11.13.5.79 DSS0\_WB\_FIR\_COEF\_V0\_1 Register (Offset = 138h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V0\_1 is shown in [Figure 12-1007](#) and described in [Table 12-1366](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1365. DSS0\_WB\_FIR\_COEF\_V0\_1  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0138h

**Figure 12-1007. DSS0\_WB\_FIR\_COEF\_V0\_1 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1366. DSS0\_WB\_FIR\_COEF\_V0\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase n

### 12.6.4.11.13.5.80 DSS0\_WB\_FIR\_COEF\_V0\_2 Register (Offset = 13Ch) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V0\_2 is shown in [Figure 12-1008](#) and described in [Table 12-1368](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1367. DSS0\_WB\_FIR\_COEF\_V0\_2  
Instances**

Instance	Physical Address
DSS0_WB	04AF 013Ch

**Figure 12-1008. DSS0\_WB\_FIR\_COEF\_V0\_2 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1368. DSS0\_WB\_FIR\_COEF\_V0\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase n



#### 12.6.4.11.13.5.81 DSS0\_WB\_FIR\_COEF\_V0\_3 Register (Offset = 140h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V0\_3 is shown in [Figure 12-1009](#) and described in [Table 12-1370](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1369. DSS0\_WB\_FIR\_COEF\_V0\_3  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0140h

**Figure 12-1009. DSS0\_WB\_FIR\_COEF\_V0\_3 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1370. DSS0\_WB\_FIR\_COEF\_V0\_3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase n

### 12.6.4.11.13.5.82 DSS0\_WB\_FIR\_COEF\_V0\_4 Register (Offset = 144h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V0\_4 is shown in [Figure 12-1010](#) and described in [Table 12-1372](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1371. DSS0\_WB\_FIR\_COEF\_V0\_4  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0144h

**Figure 12-1010. DSS0\_WB\_FIR\_COEF\_V0\_4 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1372. DSS0\_WB\_FIR\_COEF\_V0\_4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase n

#### 12.6.4.11.13.5.83 DSS0\_WB\_FIR\_COEF\_V0\_5 Register (Offset = 148h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V0\_5 is shown in [Figure 12-1011](#) and described in [Table 12-1374](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1373. DSS0\_WB\_FIR\_COEF\_V0\_5  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0148h

**Figure 12-1011. DSS0\_WB\_FIR\_COEF\_V0\_5 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1374. DSS0\_WB\_FIR\_COEF\_V0\_5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase n

#### 12.6.4.11.13.5.84 DSS0\_WB\_FIR\_COEF\_V0\_6 Register (Offset = 14Ch) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V0\_6 is shown in [Figure 12-1012](#) and described in [Table 12-1376](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1375. DSS0\_WB\_FIR\_COEF\_V0\_6  
Instances**

Instance	Physical Address
DSS0_WB	04AF 014Ch

**Figure 12-1012. DSS0\_WB\_FIR\_COEF\_V0\_6 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1376. DSS0\_WB\_FIR\_COEF\_V0\_6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase n

#### 12.6.4.11.13.5.85 DSS0\_WB\_FIR\_COEF\_V0\_7 Register (Offset = 150h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V0\_7 is shown in [Figure 12-1013](#) and described in [Table 12-1378](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1377. DSS0\_WB\_FIR\_COEF\_V0\_7  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0150h

**Figure 12-1013. DSS0\_WB\_FIR\_COEF\_V0\_7 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1378. DSS0\_WB\_FIR\_COEF\_V0\_7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase n

### 12.6.4.11.13.5.86 DSS0\_WB\_FIR\_COEF\_V0\_8 Register (Offset = 154h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V0\_8 is shown in [Figure 12-1014](#) and described in [Table 12-1380](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1379. DSS0\_WB\_FIR\_COEF\_V0\_8  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0154h

**Figure 12-1014. DSS0\_WB\_FIR\_COEF\_V0\_8 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1380. DSS0\_WB\_FIR\_COEF\_V0\_8 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase n

#### 12.6.4.11.13.5.87 DSS0\_WB\_FIR\_COEF\_V0\_C\_0 Register (Offset = 158h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V0\_C\_0 is shown in [Figure 12-1015](#) and described in [Table 12-1382](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter. If the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1381. DSS0\_WB\_FIR\_COEF\_V0\_C\_0  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0158h

**Figure 12-1015. DSS0\_WB\_FIR\_COEF\_V0\_C\_0 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1382. DSS0\_WB\_FIR\_COEF\_V0\_C\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase n

### 12.6.4.11.13.5.88 DSS0\_WB\_FIR\_COEF\_V0\_C\_1 Register (Offset = 15Ch) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V0\_C\_1 is shown in [Figure 12-1016](#) and described in [Table 12-1384](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter. If the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1383. DSS0\_WB\_FIR\_COEF\_V0\_C\_1  
Instances**

Instance	Physical Address
DSS0_WB	04AF 015Ch

**Figure 12-1016. DSS0\_WB\_FIR\_COEF\_V0\_C\_1 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1384. DSS0\_WB\_FIR\_COEF\_V0\_C\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase n



#### 12.6.4.11.13.5.89 DSS0\_WB\_FIR\_COEF\_V0\_C\_2 Register (Offset = 160h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V0\_C\_2 is shown in [Figure 12-1017](#) and described in [Table 12-1386](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter. If the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1385. DSS0\_WB\_FIR\_COEF\_V0\_C\_2  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0160h

**Figure 12-1017. DSS0\_WB\_FIR\_COEF\_V0\_C\_2 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1386. DSS0\_WB\_FIR\_COEF\_V0\_C\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase n

### 12.6.4.11.13.5.90 DSS0\_WB\_FIR\_COEF\_V0\_C\_3 Register (Offset = 164h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V0\_C\_3 is shown in [Figure 12-1018](#) and described in [Table 12-1388](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter. If the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1387. DSS0\_WB\_FIR\_COEF\_V0\_C\_3  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0164h

**Figure 12-1018. DSS0\_WB\_FIR\_COEF\_V0\_C\_3 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1388. DSS0\_WB\_FIR\_COEF\_V0\_C\_3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase n

#### 12.6.4.11.13.5.91 DSS0\_WB\_FIR\_COEF\_V0\_C\_4 Register (Offset = 168h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V0\_C\_4 is shown in [Figure 12-1019](#) and described in [Table 12-1390](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter. If the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1389. DSS0\_WB\_FIR\_COEF\_V0\_C\_4  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0168h

**Figure 12-1019. DSS0\_WB\_FIR\_COEF\_V0\_C\_4 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1390. DSS0\_WB\_FIR\_COEF\_V0\_C\_4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase n

### 12.6.4.11.13.5.92 DSS0\_WB\_FIR\_COEF\_V0\_C\_5 Register (Offset = 16Ch) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V0\_C\_5 is shown in [Figure 12-1020](#) and described in [Table 12-1392](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter. If the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1391. DSS0\_WB\_FIR\_COEF\_V0\_C\_5  
Instances**

Instance	Physical Address
DSS0_WB	04AF 016Ch

**Figure 12-1020. DSS0\_WB\_FIR\_COEF\_V0\_C\_5 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1392. DSS0\_WB\_FIR\_COEF\_V0\_C\_5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase n

#### 12.6.4.11.13.5.93 DSS0\_WB\_FIR\_COEF\_V0\_C\_6 Register (Offset = 170h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V0\_C\_6 is shown in [Figure 12-1021](#) and described in [Table 12-1394](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1393. DSS0\_WB\_FIR\_COEF\_V0\_C\_6  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0170h

**Figure 12-1021. DSS0\_WB\_FIR\_COEF\_V0\_C\_6 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1394. DSS0\_WB\_FIR\_COEF\_V0\_C\_6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase n

### 12.6.4.11.13.5.94 DSS0\_WB\_FIR\_COEF\_V0\_C\_7 Register (Offset = 174h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V0\_C\_7 is shown in [Figure 12-1022](#) and described in [Table 12-1396](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter. If the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1395. DSS0\_WB\_FIR\_COEF\_V0\_C\_7  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0174h

**Figure 12-1022. DSS0\_WB\_FIR\_COEF\_V0\_C\_7 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1396. DSS0\_WB\_FIR\_COEF\_V0\_C\_7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase n

#### 12.6.4.11.13.5.95 DSS0\_WB\_FIR\_COEF\_V0\_C\_8 Register (Offset = 178h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V0\_C\_8 is shown in [Figure 12-1023](#) and described in [Table 12-1398](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1397. DSS0\_WB\_FIR\_COEF\_V0\_C\_8  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0178h

**Figure 12-1023. DSS0\_WB\_FIR\_COEF\_V0\_C\_8 Register**

31	30	29	28	27	26	25	24
RESERVED				RESERVED			
R-0h				R-0h			
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED						FIRVC0	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
FIRVC0							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1398. DSS0\_WB\_FIR\_COEF\_V0\_C\_8 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-10	RESERVED	R	0h	Reserved
9-0	FIRVC0	R/W	0h	Unsigned coefficient C0 for the vertical up/down-scaling with the phase n

### 12.6.4.11.13.5.96 DSS0\_WB\_FIR\_COEF\_V12\_0 Register (Offset = 17Ch) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V12\_0 is shown in [Figure 12-1024](#) and described in [Table 12-1400](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1399. DSS0\_WB\_FIR\_COEF\_V12\_0  
Instances**

Instance	Physical Address
DSS0_WB	04AF 017Ch

**Figure 12-1024. DSS0\_WB\_FIR\_COEF\_V12\_0 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1400. DSS0\_WB\_FIR\_COEF\_V12\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase n
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved



### 12.6.4.11.13.5.97 DSS0\_WB\_FIR\_COEF\_V12\_1 Register (Offset = 180h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V12\_1 is shown in [Figure 12-1025](#) and described in [Table 12-1402](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1401. DSS0\_WB\_FIR\_COEF\_V12\_1  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0180h

**Figure 12-1025. DSS0\_WB\_FIR\_COEF\_V12\_1 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1402. DSS0\_WB\_FIR\_COEF\_V12\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase n
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.98 DSS0\_WB\_FIR\_COEF\_V12\_2 Register (Offset = 184h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V12\_2 is shown in [Figure 12-1026](#) and described in [Table 12-1404](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1403. DSS0\_WB\_FIR\_COEF\_V12\_2  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0184h

**Figure 12-1026. DSS0\_WB\_FIR\_COEF\_V12\_2 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1404. DSS0\_WB\_FIR\_COEF\_V12\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase n
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

#### 12.6.4.11.13.5.99 DSS0\_WB\_FIR\_COEF\_V12\_3 Register (Offset = 188h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V12\_3 is shown in [Figure 12-1027](#) and described in [Table 12-1406](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1405. DSS0\_WB\_FIR\_COEF\_V12\_3  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0188h

**Figure 12-1027. DSS0\_WB\_FIR\_COEF\_V12\_3 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1406. DSS0\_WB\_FIR\_COEF\_V12\_3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase n
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.100 DSS0\_WB\_FIR\_COEF\_V12\_4 Register (Offset = 18Ch) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V12\_4 is shown in [Figure 12-1028](#) and described in [Table 12-1408](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1407. DSS0\_WB\_FIR\_COEF\_V12\_4  
Instances**

Instance	Physical Address
DSS0_WB	04AF 018Ch

**Figure 12-1028. DSS0\_WB\_FIR\_COEF\_V12\_4 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1408. DSS0\_WB\_FIR\_COEF\_V12\_4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase n
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.101 DSS0\_WB\_FIR\_COEF\_V12\_5 Register (Offset = 190h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V12\_5 is shown in [Figure 12-1029](#) and described in [Table 12-1410](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1409. DSS0\_WB\_FIR\_COEF\_V12\_5  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0190h

**Figure 12-1029. DSS0\_WB\_FIR\_COEF\_V12\_5 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1410. DSS0\_WB\_FIR\_COEF\_V12\_5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase n
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.102 DSS0\_WB\_FIR\_COEF\_V12\_6 Register (Offset = 194h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V12\_6 is shown in [Figure 12-1030](#) and described in [Table 12-1412](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1411. DSS0\_WB\_FIR\_COEF\_V12\_6  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0194h

**Figure 12-1030. DSS0\_WB\_FIR\_COEF\_V12\_6 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1412. DSS0\_WB\_FIR\_COEF\_V12\_6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase n
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.103 DSS0\_WB\_FIR\_COEF\_V12\_7 Register (Offset = 198h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V12\_7 is shown in [Figure 12-1031](#) and described in [Table 12-1414](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1413. DSS0\_WB\_FIR\_COEF\_V12\_7  
Instances**

Instance	Physical Address
DSS0_WB	04AF 0198h

**Figure 12-1031. DSS0\_WB\_FIR\_COEF\_V12\_7 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1414. DSS0\_WB\_FIR\_COEF\_V12\_7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase n
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.104 DSS0\_WB\_FIR\_COEF\_V12\_8 Register (Offset = 19Ch) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V12\_8 is shown in [Figure 12-1032](#) and described in [Table 12-1416](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1415. DSS0\_WB\_FIR\_COEF\_V12\_8  
Instances**

Instance	Physical Address
DSS0_WB	04AF 019Ch

**Figure 12-1032. DSS0\_WB\_FIR\_COEF\_V12\_8 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1416. DSS0\_WB\_FIR\_COEF\_V12\_8 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase n
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved



### 12.6.4.11.13.5.105 DSS0\_WB\_FIR\_COEF\_V12\_9 Register (Offset = 1A0h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V12\_9 is shown in [Figure 12-1033](#) and described in [Table 12-1418](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1417. DSS0\_WB\_FIR\_COEF\_V12\_9  
Instances**

Instance	Physical Address
DSS0_WB	04AF 01A0h

**Figure 12-1033. DSS0\_WB\_FIR\_COEF\_V12\_9 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1418. DSS0\_WB\_FIR\_COEF\_V12\_9 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase n
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.106 DSS0\_WB\_FIR\_COEF\_V12\_10 Register (Offset = 1A4h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V12\_10 is shown in [Figure 12-1034](#) and described in [Table 12-1420](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1419. DSS0\_WB\_FIR\_COEF\_V12\_10  
Instances**

Instance	Physical Address
DSS0_WB	04AF 01A4h

**Figure 12-1034. DSS0\_WB\_FIR\_COEF\_V12\_10 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1420. DSS0\_WB\_FIR\_COEF\_V12\_10 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase n
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

#### 12.6.4.11.13.5.107 DSS0\_WB\_FIR\_COEF\_V12\_11 Register (Offset = 1A8h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V12\_11 is shown in [Figure 12-1035](#) and described in [Table 12-1422](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1421. DSS0\_WB\_FIR\_COEF\_V12\_11  
Instances**

Instance	Physical Address
DSS0_WB	04AF 01A8h

**Figure 12-1035. DSS0\_WB\_FIR\_COEF\_V12\_11 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1422. DSS0\_WB\_FIR\_COEF\_V12\_11 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase n
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.108 DSS0\_WB\_FIR\_COEF\_V12\_12 Register (Offset = 1ACh) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V12\_12 is shown in [Figure 12-1036](#) and described in [Table 12-1424](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1423. DSS0\_WB\_FIR\_COEF\_V12\_12  
Instances**

Instance	Physical Address
DSS0_WB	04AF 01ACh

**Figure 12-1036. DSS0\_WB\_FIR\_COEF\_V12\_12 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1424. DSS0\_WB\_FIR\_COEF\_V12\_12 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase n
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

#### 12.6.4.11.13.5.109 DSS0\_WB\_FIR\_COEF\_V12\_13 Register (Offset = 1B0h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V12\_13 is shown in [Figure 12-1037](#) and described in [Table 12-1426](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1425. DSS0\_WB\_FIR\_COEF\_V12\_13  
Instances**

Instance	Physical Address
DSS0_WB	04AF 01B0h

**Figure 12-1037. DSS0\_WB\_FIR\_COEF\_V12\_13 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1426. DSS0\_WB\_FIR\_COEF\_V12\_13 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase n
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.110 DSS0\_WB\_FIR\_COEF\_V12\_14 Register (Offset = 1B4h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V12\_14 is shown in [Figure 12-1038](#) and described in [Table 12-1428](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1427. DSS0\_WB\_FIR\_COEF\_V12\_14  
Instances**

Instance	Physical Address
DSS0_WB	04AF 01B4h

**Figure 12-1038. DSS0\_WB\_FIR\_COEF\_V12\_14 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1428. DSS0\_WB\_FIR\_COEF\_V12\_14 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase n
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.111 DSS0\_WB\_FIR\_COEF\_V12\_15 Register (Offset = 1B8h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V12\_15 is shown in [Figure 12-1039](#) and described in [Table 12-1430](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for ARGB and Y setting. Shadow register

**Table 12-1429. DSS0\_WB\_FIR\_COEF\_V12\_15  
Instances**

Instance	Physical Address
DSS0_WB	04AF 01B8h

**Figure 12-1039. DSS0\_WB\_FIR\_COEF\_V12\_15 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1430. DSS0\_WB\_FIR\_COEF\_V12\_15 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase n
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.112 DSS0\_WB\_FIR\_COEF\_V12\_C\_0 Register (Offset = 1BCh) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V12\_C\_0 is shown in [Figure 12-1040](#) and described in [Table 12-1432](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1431. DSS0\_WB\_FIR\_COEF\_V12\_C\_0  
Instances**

Instance	Physical Address
DSS0_WB	04AF 01BCh

**Figure 12-1040. DSS0\_WB\_FIR\_COEF\_V12\_C\_0 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1432. DSS0\_WB\_FIR\_COEF\_V12\_C\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase n
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved



### 12.6.4.11.13.5.113 DSS0\_WB\_FIR\_COEF\_V12\_C\_1 Register (Offset = 1C0h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V12\_C\_1 is shown in [Figure 12-1041](#) and described in [Table 12-1434](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1433. DSS0\_WB\_FIR\_COEF\_V12\_C\_1  
Instances**

Instance	Physical Address
DSS0_WB	04AF 01C0h

**Figure 12-1041. DSS0\_WB\_FIR\_COEF\_V12\_C\_1 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1434. DSS0\_WB\_FIR\_COEF\_V12\_C\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase n
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.114 DSS0\_WB\_FIR\_COEF\_V12\_C\_2 Register (Offset = 1C4h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V12\_C\_2 is shown in [Figure 12-1042](#) and described in [Table 12-1436](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1435. DSS0\_WB\_FIR\_COEF\_V12\_C\_2  
Instances**

Instance	Physical Address
DSS0_WB	04AF 01C4h

**Figure 12-1042. DSS0\_WB\_FIR\_COEF\_V12\_C\_2 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1436. DSS0\_WB\_FIR\_COEF\_V12\_C\_2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase n
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.115 DSS0\_WB\_FIR\_COEF\_V12\_C\_3 Register (Offset = 1C8h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V12\_C\_3 is shown in [Figure 12-1043](#) and described in [Table 12-1438](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1437. DSS0\_WB\_FIR\_COEF\_V12\_C\_3  
Instances**

Instance	Physical Address
DSS0_WB	04AF 01C8h

**Figure 12-1043. DSS0\_WB\_FIR\_COEF\_V12\_C\_3 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1438. DSS0\_WB\_FIR\_COEF\_V12\_C\_3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase n
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.116 DSS0\_WB\_FIR\_COEF\_V12\_C\_4 Register (Offset = 1CCh) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V12\_C\_4 is shown in [Figure 12-1044](#) and described in [Table 12-1440](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1439. DSS0\_WB\_FIR\_COEF\_V12\_C\_4  
Instances**

Instance	Physical Address
DSS0_WB	04AF 01CCh

**Figure 12-1044. DSS0\_WB\_FIR\_COEF\_V12\_C\_4 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1440. DSS0\_WB\_FIR\_COEF\_V12\_C\_4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase n
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.117 DSS0\_WB\_FIR\_COEF\_V12\_C\_5 Register (Offset = 1D0h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V12\_C\_5 is shown in [Figure 12-1045](#) and described in [Table 12-1442](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1441. DSS0\_WB\_FIR\_COEF\_V12\_C\_5  
Instances**

Instance	Physical Address
DSS0_WB	04AF 01D0h

**Figure 12-1045. DSS0\_WB\_FIR\_COEF\_V12\_C\_5 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1442. DSS0\_WB\_FIR\_COEF\_V12\_C\_5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase n
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.118 DSS0\_WB\_FIR\_COEF\_V12\_C\_6 Register (Offset = 1D4h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V12\_C\_6 is shown in [Figure 12-1046](#) and described in [Table 12-1444](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter. If the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1443. DSS0\_WB\_FIR\_COEF\_V12\_C\_6  
Instances**

Instance	Physical Address
DSS0_WB	04AF 01D4h

**Figure 12-1046. DSS0\_WB\_FIR\_COEF\_V12\_C\_6 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1444. DSS0\_WB\_FIR\_COEF\_V12\_C\_6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase n
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.119 DSS0\_WB\_FIR\_COEF\_V12\_C\_7 Register (Offset = 1D8h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V12\_C\_7 is shown in [Figure 12-1047](#) and described in [Table 12-1446](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1445. DSS0\_WB\_FIR\_COEF\_V12\_C\_7  
Instances**

Instance	Physical Address
DSS0_WB	04AF 01D8h

**Figure 12-1047. DSS0\_WB\_FIR\_COEF\_V12\_C\_7 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1446. DSS0\_WB\_FIR\_COEF\_V12\_C\_7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase n
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.120 DSS0\_WB\_FIR\_COEF\_V12\_C\_8 Register (Offset = 1DCh) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V12\_C\_8 is shown in [Figure 12-1048](#) and described in [Table 12-1448](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter. If the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1447. DSS0\_WB\_FIR\_COEF\_V12\_C\_8  
Instances**

Instance	Physical Address
DSS0_WB	04AF 01DCh

**Figure 12-1048. DSS0\_WB\_FIR\_COEF\_V12\_C\_8 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1448. DSS0\_WB\_FIR\_COEF\_V12\_C\_8 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase n
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved



### 12.6.4.11.13.5.121 DSS0\_WB\_FIR\_COEF\_V12\_C\_9 Register (Offset = 1E0h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V12\_C\_9 is shown in [Figure 12-1049](#) and described in [Table 12-1450](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1449. DSS0\_WB\_FIR\_COEF\_V12\_C\_9 Instances**

Instance	Physical Address
DSS0_WB	04AF 01E0h

**Figure 12-1049. DSS0\_WB\_FIR\_COEF\_V12\_C\_9 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1450. DSS0\_WB\_FIR\_COEF\_V12\_C\_9 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase n
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.122 DSS0\_WB\_FIR\_COEF\_V12\_C\_10 Register (Offset = 1E4h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V12\_C\_10 is shown in [Figure 12-1050](#) and described in [Table 12-1452](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1451. DSS0\_WB\_FIR\_COEF\_V12\_C\_10 Instances**

Instance	Physical Address
DSS0_WB	04AF 01E4h

**Figure 12-1050. DSS0\_WB\_FIR\_COEF\_V12\_C\_10 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1452. DSS0\_WB\_FIR\_COEF\_V12\_C\_10 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase n
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.123 DSS0\_WB\_FIR\_COEF\_V12\_C\_11 Register (Offset = 1E8h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V12\_C\_11 is shown in [Figure 12-1051](#) and described in [Table 12-1454](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1453. DSS0\_WB\_FIR\_COEF\_V12\_C\_11  
Instances**

Instance	Physical Address
DSS0_WB	04AF 01E8h

**Figure 12-1051. DSS0\_WB\_FIR\_COEF\_V12\_C\_11 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1454. DSS0\_WB\_FIR\_COEF\_V12\_C\_11 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase n
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.124 DSS0\_WB\_FIR\_COEF\_V12\_C\_12 Register (Offset = 1ECh) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V12\_C\_12 is shown in [Figure 12-1052](#) and described in [Table 12-1456](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter. If the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1455. DSS0\_WB\_FIR\_COEF\_V12\_C\_12  
Instances**

Instance	Physical Address
DSS0_WB	04AF 01ECh

**Figure 12-1052. DSS0\_WB\_FIR\_COEF\_V12\_C\_12 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1456. DSS0\_WB\_FIR\_COEF\_V12\_C\_12 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase n
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.125 DSS0\_WB\_FIR\_COEF\_V12\_C\_13 Register (Offset = 1F0h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V12\_C\_13 is shown in [Figure 12-1053](#) and described in [Table 12-1458](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1457. DSS0\_WB\_FIR\_COEF\_V12\_C\_13  
Instances**

Instance	Physical Address
DSS0_WB	04AF 01F0h

**Figure 12-1053. DSS0\_WB\_FIR\_COEF\_V12\_C\_13 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1458. DSS0\_WB\_FIR\_COEF\_V12\_C\_13 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase n
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.126 DSS0\_WB\_FIR\_COEF\_V12\_C\_14 Register (Offset = 1F4h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V12\_C\_14 is shown in [Figure 12-1054](#) and described in [Table 12-1460](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1459. DSS0\_WB\_FIR\_COEF\_V12\_C\_14  
Instances**

Instance	Physical Address
DSS0_WB	04AF 01F4h

**Figure 12-1054. DSS0\_WB\_FIR\_COEF\_V12\_C\_14 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1460. DSS0\_WB\_FIR\_COEF\_V12\_C\_14 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase n
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.127 DSS0\_WB\_FIR\_COEF\_V12\_C\_15 Register (Offset = 1F8h) [reset = 0h]

DSS0\_WB\_FIR\_COEF\_V12\_C\_15 is shown in [Figure 12-1055](#) and described in [Table 12-1462](#).

Return to [Summary Table](#).

The bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with the write back pipeline for the phases from 0 to 15. It is used for Crb and Cr setting. It is used only when the pixel format at the input of the filter is one of the YUV formats. If the pixel format at the input of the filter is ARGB all ARGB, RGB, RGBA are converted to ARGB32-8888 by the color space conversion before going to the filter is the color space conversion is done before the filter. When the register is not used by the HW, any value can be used for the bit-fields. Shadow register

**Table 12-1461. DSS0\_WB\_FIR\_COEF\_V12\_C\_15  
Instances**

Instance	Physical Address
DSS0_WB	04AF 01F8h

**Figure 12-1055. DSS0\_WB\_FIR\_COEF\_V12\_C\_15 Register**

31	30	29	28	27	26	25	24
RESERVED				FIRVC2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
FIRVC2				FIRVC1			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
FIRVC1						RESERVED	
R/W-0h						R-0h	
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1462. DSS0\_WB\_FIR\_COEF\_V12\_C\_15 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-20	FIRVC2	R/W	0h	Signed coefficient C2 for the vertical up/down-scaling with the phase n
19-10	FIRVC1	R/W	0h	Signed coefficient C1 for the vertical up/down-scaling with the phase n
9-0	RESERVED	R	0h	Reserved

### 12.6.4.11.13.5.128 DSS0\_WB\_MFLAG\_THRESHOLD Register (Offset = 204h) [reset = 0h]

DSS0\_WB\_MFLAG\_THRESHOLD is shown in [Figure 12-1056](#) and described in [Table 12-1464](#).

Return to [Summary Table](#).

**Table 12-1463. DSS0\_WB\_MFLAG\_THRESHOLD Instances**

Instance	Physical Address
DSS0_WB	04AF 0204h

**Figure 12-1056. DSS0\_WB\_MFLAG\_THRESHOLD Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HT_MFLAG																LT_MFLAG															
R/W-0h																R/W-0h															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1464. DSS0\_WB\_MFLAG\_THRESHOLD Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	HT_MFLAG	R/W	0h	MFlag High Threshold
15-0	LT_MFLAG	R/W	0h	MFlag Low Threshold



#### 12.6.4.11.13.5.129 DSS0\_WB\_PICTURE\_SIZE Register (Offset = 208h) [reset = X]

DSS0\_WB\_PICTURE\_SIZE is shown in [Figure 12-1057](#) and described in [Table 12-1466](#).

Return to [Summary Table](#).

The register configures the DSS0\_WB\_SIZE of the write-back picture associated with the write back pipeline after up/down-scaling DSS0\_WB\_SIZE of the image stored in DDR memory, generated by WB pipe. Shadow register

**Table 12-1465. DSS0\_WB\_PICTURE\_SIZE Instances**

Instance	Physical Address
DSS0_WB	04AF 0208h

**Figure 12-1057. DSS0\_WB\_PICTURE\_SIZE Register**

31	30	29	28	27	26	25	24
RESERVED				MEMSIZEY			
R/W-X				R/W-0h			
23	22	21	20	19	18	17	16
MEMSIZEY							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED				MEMSIZEX			
R/W-X				R/W-0h			
7	6	5	4	3	2	1	0
MEMSIZEX							
R/W-0h							

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1466. DSS0\_WB\_PICTURE\_SIZE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	X	
29-16	MEMSIZEY	R/W	0h	Number of lines of the wb picture in memory Encoded value [from 1 to 16384] to specify the number of lines of the picture store in memory [program to value minus one]
15-14	RESERVED	R/W	X	
13-0	MEMSIZEX	R/W	0h	Number of pixels of the wb picture in memory Encoded value [from 1 to 16384] to specify the number of pixels of the picture stored in memory [program to value minus one]

### 12.6.4.11.13.5.130 DSS0\_WB\_SIZE Register (Offset = 210h) [reset = X]

DSS0\_WB\_SIZE is shown in [Figure 12-1058](#) and described in [Table 12-1468](#).

Return to [Summary Table](#).

The register configures the DSS0\_WB\_SIZE of the output of overlay connected to the write-back pipeline when the overlay output is only used by the write-back pipeline. When the overlay is output on the primary LCD or secondary LCD or TV outputs, the DSS0\_WB\_SIZE of the frame is defined in the DISPC\_SIZE\_LCD1, DISPC\_SIZE\_LCD2, and DISPC\_SIZE\_TV respectively. Shadow register.

**Table 12-1467. DSS0\_WB\_SIZE Instances**

Instance	Physical Address
DSS0_WB	04AF 0210h

**Figure 12-1058. DSS0\_WB\_SIZE Register**

31	30	29	28	27	26	25	24
RESERVED				SIZEY			
R/W-X				R/W-0h			
23	22	21	20	19	18	17	16
SIZEY							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED				SIZEX			
R/W-X				R/W-0h			
7	6	5	4	3	2	1	0
SIZEX							
R/W-0h							

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1468. DSS0\_WB\_SIZE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	X	
29-16	SIZEY	R/W	0h	Number of lines of the Write-back picture Encoded value [from 1 to 16384] to specify the number of lines of the write-back picture from overlay or pipeline
15-14	RESERVED	R/W	X	
13-0	SIZEX	R/W	0h	Number of pixels of the Write-back picture Encoded value [from 1 to 16384] to specify the number of pixels of the write-back picture from overlay or pipeline

#### 12.6.4.11.13.5.131 DSS0\_WB\_POSITION Register (Offset = 214h) [reset = 0h]

DSS0\_WB\_POSITION is shown in [Figure 12-1059](#) and described in [Table 12-1470](#).

Return to [Summary Table](#).

The register configures the start DSS0\_WB\_POSITION of the window on overlay which wb will capture. Shadow register. Only applicable when WB is operating in capture\_mode

**Table 12-1469. DSS0\_WB\_POSITION Instances**

Instance	Physical Address
DSS0_WB	04AF 0214h

**Figure 12-1059. DSS0\_WB\_POSITION Register**

31	30	29	28	27	26	25	24
RESERVED				POSY			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
POSY							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED				POSX			
R-0h				R/W-0h			
7	6	5	4	3	2	1	0
POSX							
R/W-0h							

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1470. DSS0\_WB\_POSITION Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	0h	Reserved
29-16	POSY	R/W	0h	Y DSS0_WB_POSITION of the video window Encoded value [from 0 to 16384] to specify the Y DSS0_WB_POSITION of the video window 1 The line at the top has the Y-DSS0_WB_POSITION 0
15-14	RESERVED	R	0h	Reserved
13-0	POSX	R/W	0h	X DSS0_WB_POSITION of the video window Encoded value [from 0 to 16384] to specify the X DSS0_WB_POSITION of the video window 1 The first pixel on the left of the display screen has the X-DSS0_WB_POSITION 0

### 12.6.4.11.13.5.132 DSS0\_WB\_CSC\_COEF7 Register (Offset = 21Ch) [reset = 0h]

DSS0\_WB\_CSC\_COEF7 is shown in [Figure 12-1060](#) and described in [Table 12-1472](#).

Return to [Summary Table](#).

The register configures the color space conversion matrix coefficients. Shadow register

**Table 12-1471. DSS0\_WB\_CSC\_COEF7 Instances**

Instance	Physical Address
DSS0_WB	04AF 021Ch

**Figure 12-1060. DSS0\_WB\_CSC\_COEF7 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
POSTOFFSET3												RESERVED			
R/W-0h												R-0h			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POSTOFFSET2												RESERVED			
R/W-0h												R-0h			

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1472. DSS0\_WB\_CSC\_COEF7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-19	POSTOFFSET3	R/W	0h	Row-3 post-offset. Encoded signed value [from -4096 to 4095]
18-16	RESERVED	R	0h	Reserved
15-3	POSTOFFSET2	R/W	0h	Row-2 post-offset. Encoded signed value [from -4096 to 4095]
2-0	RESERVED	R	0h	Reserved

#### 12.6.4.11.13.5.133 DSS0\_WB\_ROW\_INC Register (Offset = 224h) [reset = 1h]

DSS0\_WB\_ROW\_INC is shown in [Figure 12-1061](#) and described in [Table 12-1474](#).

Return to [Summary Table](#).

The register configures the number of bytes to increment at the end of the row for the buffer associated with the WB window. For YUV420 formats this corresponds to the Y Buffer Shadow register.

**Table 12-1473. DSS0\_WB\_ROW\_INC Instances**

Instance	Physical Address
DSS0_WB	04AF 0224h

**Figure 12-1061. DSS0\_WB\_ROW\_INC Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ROWINC																															
R/W-1h																															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1474. DSS0\_WB\_ROW\_INC Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	ROWINC	R/W	1h	Number of bytes to increment at the end of the row Encoded signed value [from $-2^{31}-1$ to $2^{31}$ ] to specify the number of bytes to increment at the end of the row in the video buffer The value 0 is invalid The value 1 means next pixel The value $1+n*bbp$ means increment of n pixels The value $1-[n+1]*bbp$ means decrement of n pixels

### 12.6.4.11.13.5.134 DSS0\_WB\_ROW\_INC\_UV Register (Offset = 228h) [reset = 1h]

DSS0\_WB\_ROW\_INC\_UV is shown in [Figure 12-1062](#) and described in [Table 12-1476](#).

Return to [Summary Table](#).

The register configures the number of bytes to increment at the end of the row for the UV buffer associated with the WB window for YUV420 formats. For non-YUV420 formats this register is unused. Shadow register

**Table 12-1475. DSS0\_WB\_ROW\_INC\_UV Instances**

Instance	Physical Address
DSS0_WB	04AF 0228h

**Figure 12-1062. DSS0\_WB\_ROW\_INC\_UV Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ROWINC																															
R/W-1h																															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1476. DSS0\_WB\_ROW\_INC\_UV Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	ROWINC	R/W	1h	Number of bytes to increment at the end of the row Encoded signed value [from $-2^{31}-1$ to $2^{31}$ ] to specify the number of bytes to increment at the end of the row in the video buffer The value 0 is invalid The value 1 means next pixel The value $1+n*bbp$ means increment of n pixels The value $1-[n+1]*bbp$ means decrement of n pixels

#### 12.6.4.11.13.5.135 DSS0\_WB\_BA\_EXT\_0 Register (Offset = 22Ch) [reset = 0h]

DSS0\_WB\_BA\_EXT\_0 is shown in [Figure 12-1063](#) and described in [Table 12-1478](#).

Return to [Summary Table](#).

The register configures the 16-bit base address extension. It is the base-address of the single video buffer for single plane ARGB or YUV. For the Y buffer for two plane YUV. For the Alpha buffer for two plane RGB565-A8. 0 & 1 : For ping-pong mechanism with external trigger, based on the field polarity. Shadow register

**Table 12-1477. DSS0\_WB\_BA\_EXT\_0 Instances**

Instance	Physical Address
DSS0_WB	04AF 022Ch

**Figure 12-1063. DSS0\_WB\_BA\_EXT\_0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																BA_EXT															
R-0h																R/W-0h															

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1478. DSS0\_WB\_BA\_EXT\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	BA_EXT	R/W	0h	Video base address extension [16 bits]. Addr extension to make the address space 48b wide

### 12.6.4.11.13.5.136 DSS0\_WB\_BA\_EXT\_1 Register (Offset = 230h) [reset = 0h]

DSS0\_WB\_BA\_EXT\_1 is shown in [Figure 12-1064](#) and described in [Table 12-1480](#).

Return to [Summary Table](#).

The register configures the 16-bit base address extension. It is the base-address of the single video buffer for single plane ARGB or YUV. For the Y buffer for two plane YUV. For the Alpha buffer for two plane RGB565-A8. 0 & 1 : For ping-pong mechanism with external trigger, based on the field polarity. Shadow register

**Table 12-1479. DSS0\_WB\_BA\_EXT\_1 Instances**

Instance	Physical Address
DSS0_WB	04AF 0230h

**Figure 12-1064. DSS0\_WB\_BA\_EXT\_1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																BA_EXT															
R-0h																R/W-0h															

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1480. DSS0\_WB\_BA\_EXT\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	BA_EXT	R/W	0h	Video base address extension [16 bits]. Addr extension to make the address space 48b wide



#### 12.6.4.11.13.5.137 DSS0\_WB\_BA\_UV\_EXT\_0 Register (Offset = 234h) [reset = 0h]

DSS0\_WB\_BA\_UV\_EXT\_0 is shown in [Figure 12-1065](#) and described in [Table 12-1482](#).

Return to [Summary Table](#).

The register configures the 16-bit base address extension of the UV buffer for two plane YUV or the RGB buffer for two plane RGB565-A8. 0 & 1 : For ping-pong mechanism with external trigger, based on the field polarity. Shadow register

**Table 12-1481. DSS0\_WB\_BA\_UV\_EXT\_0 Instances**

Instance	Physical Address
DSS0_WB	04AF 0234h

**Figure 12-1065. DSS0\_WB\_BA\_UV\_EXT\_0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																BA_UV_EXT															
R-0h																R/W-0h															

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1482. DSS0\_WB\_BA\_UV\_EXT\_0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	BA_UV_EXT	R/W	0h	Video base address extension [16 bits]. Addr extension to make the address space 48b wide

### 12.6.4.11.13.5.138 DSS0\_WB\_BA\_UV\_EXT\_1 Register (Offset = 238h) [reset = 0h]

DSS0\_WB\_BA\_UV\_EXT\_1 is shown in [Figure 12-1066](#) and described in [Table 12-1484](#).

Return to [Summary Table](#).

The register configures the 16-bit base address extension of the UV buffer for two plane YUV or the RGB buffer for two plane RGB565-A8. 0 & 1 : For ping-pong mechanism with external trigger, based on the field polarity. Shadow register

**Table 12-1483. DSS0\_WB\_BA\_UV\_EXT\_1 Instances**

Instance	Physical Address
DSS0_WB	04AF 0238h

**Figure 12-1066. DSS0\_WB\_BA\_UV\_EXT\_1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																BA_UV_EXT															
R-0h																R/W-0h															

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1484. DSS0\_WB\_BA\_UV\_EXT\_1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15-0	BA_UV_EXT	R/W	0h	Video base address extension [16 bits]. Addr extension to make the address space 48b wide

### 12.6.4.11.13.5.139 DSS0\_WB\_SECURE Register (Offset = 248h) [reset = 0h]

DSS0\_WB\_SECURE is shown in [Figure 12-1067](#) and described in [Table 12-1486](#).

Return to [Summary Table](#).

Security bit settings for the sub-module

**Table 12-1485. DSS0\_WB\_SECURE Instances**

Instance	Physical Address
DSS0_WB	04AF 0248h

**Figure 12-1067. DSS0\_WB\_SECURE Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							SECURE
R-0h							R/W-0h

LEGEND: R = Read Only; R/W = Read/Write; -n = value after reset

**Table 12-1486. DSS0\_WB\_SECURE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	SECURE	R/W	0h	DSS0_WB_SECURE bit 0h = DSS0_WB_SECURE bit is reset 1h = DSS0_WB_SECURE bit is set

## 12.6.5 MIPI Display Serial Interface (DSI) Controller

### 12.6.5.1 DSI Block Diagram

The MIPI DSI v1.3.1 Transmitter Controller (DSITX) provides an interface that receives data and control from the host processor display system using either the DPI or SDI input bus interfaces. The DSITX will translate the incoming pixel information and control signals into an internal packed byte format, in the case of DPI, or pass in the prepacked SDI byte format, before the internal byte format data is packetized and sent to the MIPI DSI Compatible display via the MIPI D-PHY physical interface. It supports video and command mode displays and can work in multi-display mode using virtual channel identification on the packets.

The DSITX controller provides two interfaces for connection to a display panel. Normal operation for a panel supporting DCS commands can be done using either SDI interface, or using the APB (Advanced Peripheral Bus) access to DIRECT\_CMD registers. Video streaming applications can only use the SDI or DPI interface. The DSI controller supports flow control using the SDI video interface.

The DSITX implements the stream arbitration and low-level protocol layer functionalities required by MIPI DSI specification v1.3. It supports up to 4 x 2.5 Gbps D-PHY data lanes in a single-link configuration and handles the byte lane mapping per use case (1, 2, 3, or 4-lanes).

The DSITX controller block diagram is illustrated in [Figure 12-1068](#).

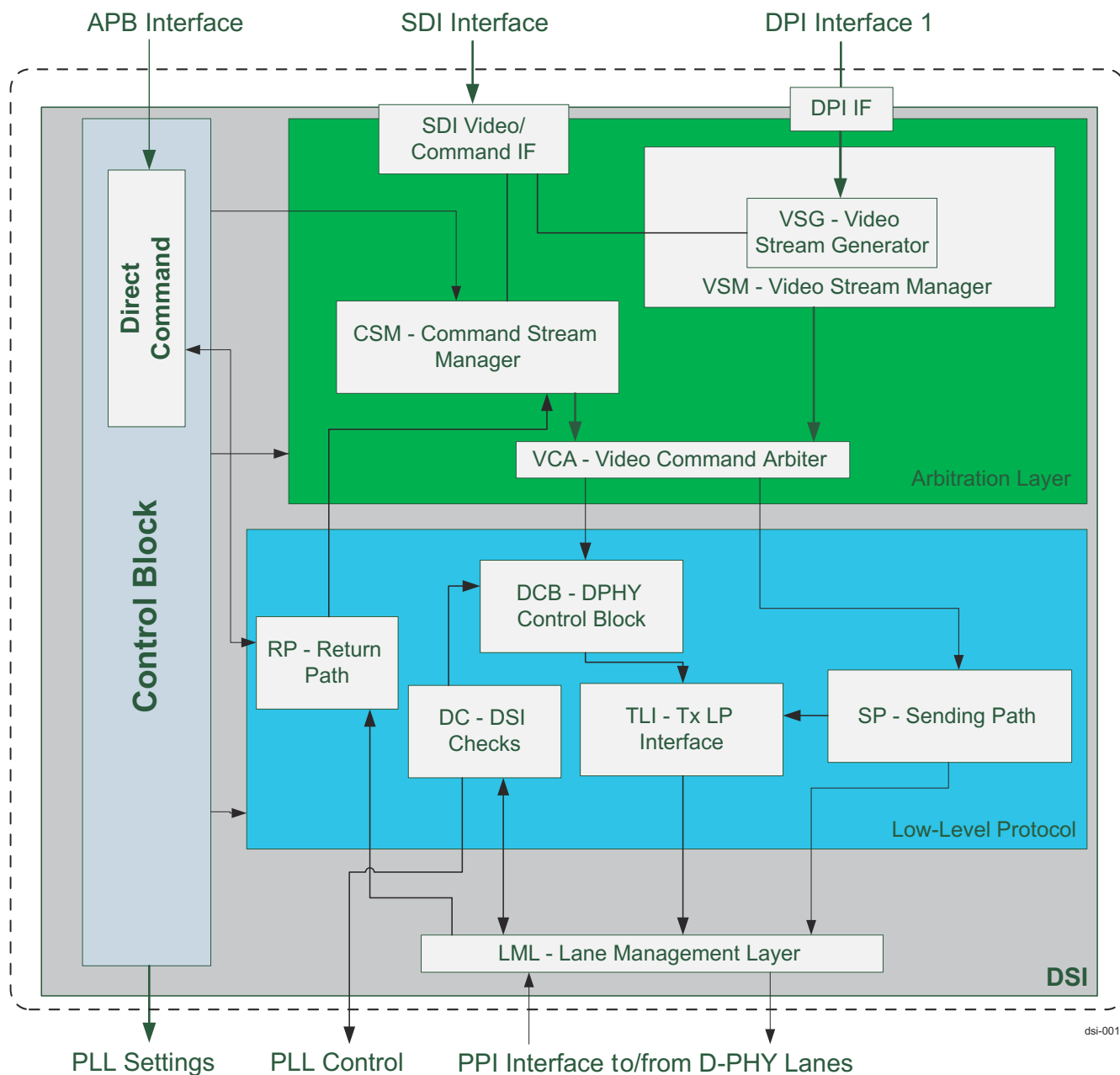


Figure 12-1068. DSITX Controller Block Architecture

### 12.6.5.2 DSI Clocking

The DSI receives all the clocks from DISPC or DPHY\_TX modules. There are no other dedicated PLLs. For more details on DSI clocks mapping at DSS and SoC level, see *DSI Integration*.

The DSI requires multiple clocks running asynchronously, that are shown in [Table 12-1487](#).

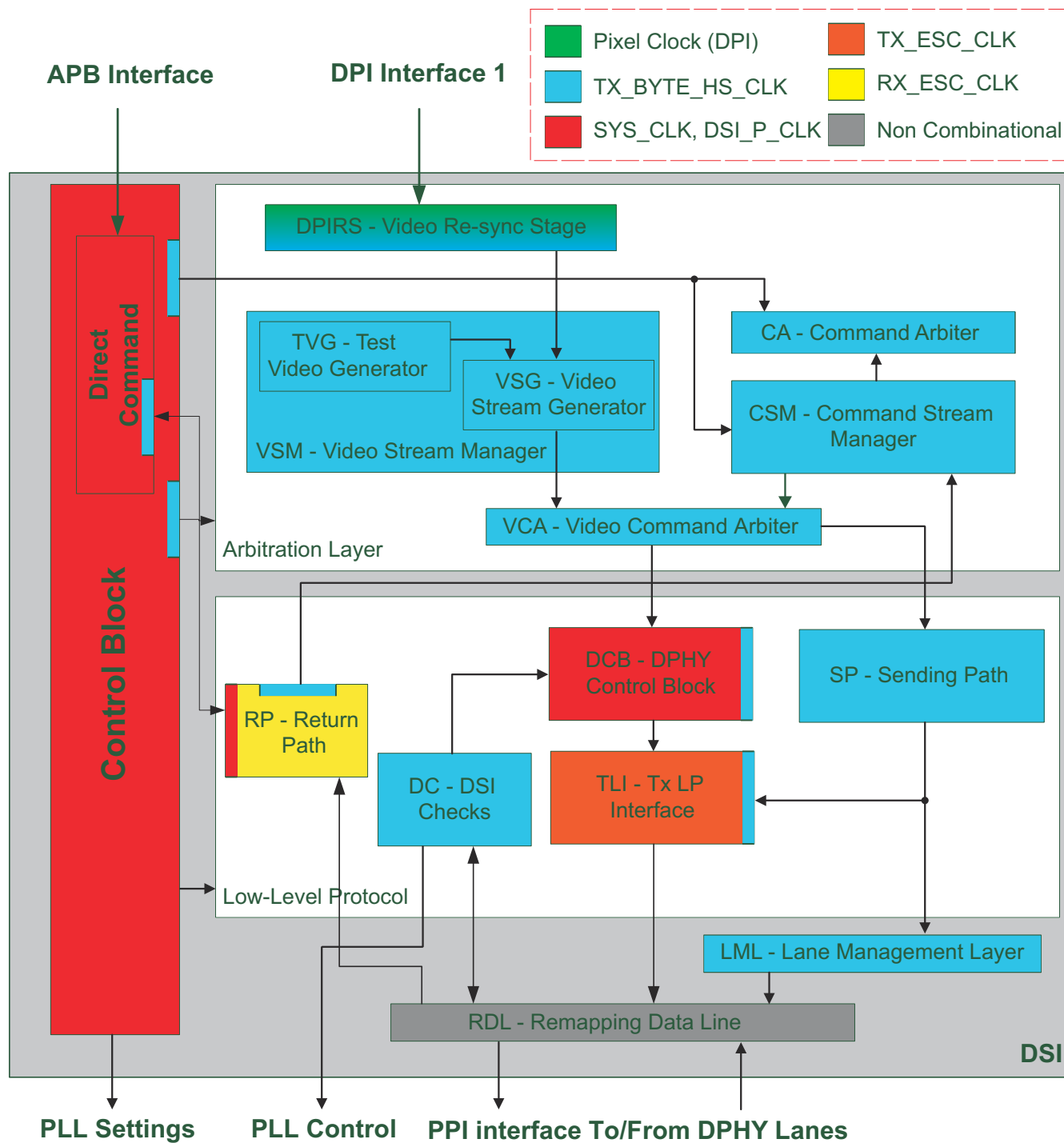
Table 12-1487. DSI Clocking

Clock	Direction	Min	Max	Constraints	Description
dpi_0_clk (Async to all other clocks)	Input	7 MHz	425 MHz	$F_{\text{pixel\_clk}} = F_{\text{tx\_byte\_hs\_clk}} \times \text{active\_lanes} \times 8 / (\text{bits\_per\_pixel})$	Used for DPI interface only. Frequency range depends on expected data rate with respect of number of data lanes, data lane frequency and frame rate

**Table 12-1487. DSI Clocking (continued)**

Clock	Direction	Min	Max	Constraints	Description
sys_clk (Async to all other clocks)	Input	7 MHz	250 MHz	1. Cannot be slower than $\text{tx\_byte\_hs\_clk} \times \text{datapath\_size} / \text{if\_datasize}$ (risk of underrun) in SDI mode. 2. Must be greater than: $\text{rx\_esc\_clk} \times 8$ . 3. Speed should be sufficient to provide enough data in SDI operation. 4. Must be faster than the $\text{tx\_byte\_hs\_clk}$ for Direct command operation.	Main functional clock Also used for the VBUS/APB interface
dphy_0_tx_byte_hs_clk (Async to all other clocks)	Input	10 MHz	312.5 MHz	The max value in SDI interface operation depends on sys_clk: can't exceed sys_clk freq x if_datasize datapath_size Otherwise 312.5 MHz to match the 2.5Gbps limit on DPHY v1.3 specification	DPHY PPI Byte Clock Frequency set by the DPHY and its High Speed input clock ( $\text{tx\_byte\_hs\_clk} = \text{dphy bit rate} / 8$ ) Maximum limit due to risk of underrun
dphy_0_tx_esc_clk (Async to all other clocks)	Input	1 MHz	20 MHz	No minimum limit is given by the DSI itself - while regular function mode max limit is 20 MHz	TX Escape Clock
dphy_0_rx_esc_clk (Async to all other clocks)	Input	1 MHz	10 MHz	No minimum limit is given by the DSI itself - while regular function mode max limit is 10 MHz	RX Escape Clock

Figure 12-1069 describes the clock scheme and domains of the DSI\_TX.



dsi\_spruij7-057

Figure 12-1069. DSITX Clock Scheme and Domains

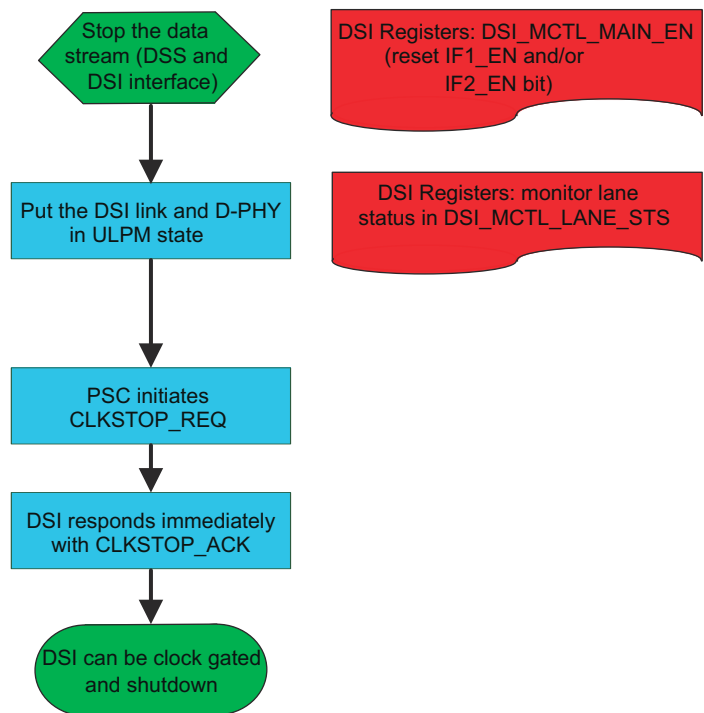
### 12.6.5.3 DSI Reset

The DSI has one active low reset input. The entire internal logic is reset by this reset. Internal resets (synchronous and asynchronous) for different clock domains are generated internally (synchronized to the respective clock domain).

### 12.6.5.4 DSI Power Management

The DSI implements a power management protocol to interface to a PSC (Power and Sleep Controller) module SoC level.

Figure 12-1070 shows the expected sequence from SW while performing a *clkstop\_req* to the DSI.



ds\_i\_spruij7-058

**Figure 12-1070. DSI Clock Gate / Power Off Procedure**

### 12.6.5.5 DSI Interrupts

Table 12-1488 shows the interrupts that are generated by the DSI module.

**Table 12-1488. DSI Interrupts**

Interrupt (n = 0)	Description	Source Module	Source Signal Name
dsi_#n_func_intr	DSI-#n Func Interrupt	DSI Controller	dsi_irq_n
dsi_#n_safety_error_fatal_intr	DSI #n Internal Diagnostic Fatal interrupt	DSI Controller	asf_int_fatal
dsi_#n_safety_error_nonfatal_intr	DSI #n Internal Diagnostic Non-fatal interrupt	DSI Controller	asf_int_nonfatal
ecc_intr_uncorr_level_sys_intr	ECC aggr uncorrectable error	-	-

### 12.6.5.6 DSI Internal Interfaces

#### 12.6.5.6.1 Video Input Interfaces

DSITX supports following video streaming interfaces:

- DPI (Uncompressed video stream)
- SDI (Serial Display Interface)

The DSI does not require DSI sub-link support. Therefore, only one video mode source is supported.





### 12.6.5.6.3 SDI (Serial Data Interface)

DSITX uses a SDI interface (32-bit) to receive active video data from an internal frame memory using a DMA or other data pull mechanism. The interface uses req/ready handshake signaling to control the data flow.

The DSITX SDI Interface supports one of the following modes:

- Command-only

SDI-DSI Interface requires following signals:

- dsi\_if\_valid
- dsi\_if\_stall
- dsi\_if\_start (Line Start)
- dsi\_if\_frame\_sync (frame\_sync)
- dsi\_if\_data

The SDI interface of DSI\_TX receives data directly from DSS VP output and is functional when the user configures the DSS to operate in COMMAND\_STALLMODE (DSS0\_VP\_CONTROL[11] STALLMODE = '1' and DSS0\_VP\_CONTROL[12] STALLMODETYPE = '0').

#### 12.6.5.6.3.1 Secure Display Support

Only secure peripherals can be connected to a secure DPI (VP) output of DSS. In order to support this requirement, the DSS exports a secure qualifier to the DSI\_TX. Inside the DSI a SECURE register is implemented (DSI\_WRAP\_DPI\_SECURE), which contains SECURE bits, [0] DPI\_0\_SECURE and [1] DPI\_0\_SECURE\_VIOLATION, corresponding to the DPI port. These SECURE bits can only be set or reset by a secure host (vbusp transactions with secure qualifier set).

The behavior of DSI module for different settings of SECURE register bit and SECURE qualifier from DSS is as shown below:

**Table 12-1490. Secure Display Support**

SECURE Register Bit (DSI)	SECURE mqualifier (DSS)	Security Violation Status	Comments
0	0	0	Data is passed (non-secure DSS data through non-secure DSI)
1	0	0	Data is passed (non-secure DSS data through secure DSI)
1	1	0	Data is passed (secure DSS data through secure DSI)
0	1	1	<b>Security Violation.</b> Data is blocked (secure DSS data through non-secure DSI)

Once a security violation is detected, it is captured in the DSI\_WRAP\_DPI\_SECURE[1] DPI\_0\_SECURE\_VIOLATION register status bit.

### 12.6.5.7 DSI Programming Guide

This section describes the programming guidelines for the DSI Controller and D-PHY.

The goal of this section is to present more in detail how the DSI link should be used at application level plus some scenarios of use of the DSI link (start-up, display switch on/off, dual-display).

#### 12.6.5.7.1 Application Guidelines

The purpose of this section is to provide guidelines on how to drive the DSI host controller, provide general sequence of operations and some typical application notes.

### 12.6.5.7.1.1 Overview of a Display Subsystem

Figure 12-1072 shows an overview of a display subsystem. It covers the point to point interfaces between the DPI/SDI and DSI blocks. Other blocks and interfaces are included to give a better understanding of the environment for this interface.

A DSI interface is made up from one DSI controller and one D-PHY block. D-PHY is the physical layer of the DSI interface. The DSI block handles protocol. The DSI interfaces support displays that comply with the MIPI DSI standard.

The displays can be of either command mode type or video mode type. Command mode displays have a frame memory that can hold the entire image. The display is refreshed from this memory and only changes of the display content are sent over the display interface. Video mode displays do not have this memory and thus all data for every refresh must be transferred in real time.

The SDI block can support one display in command mode. By using command mode displays and rerouting of connections, between display updates, more than two displays can be connected and active.

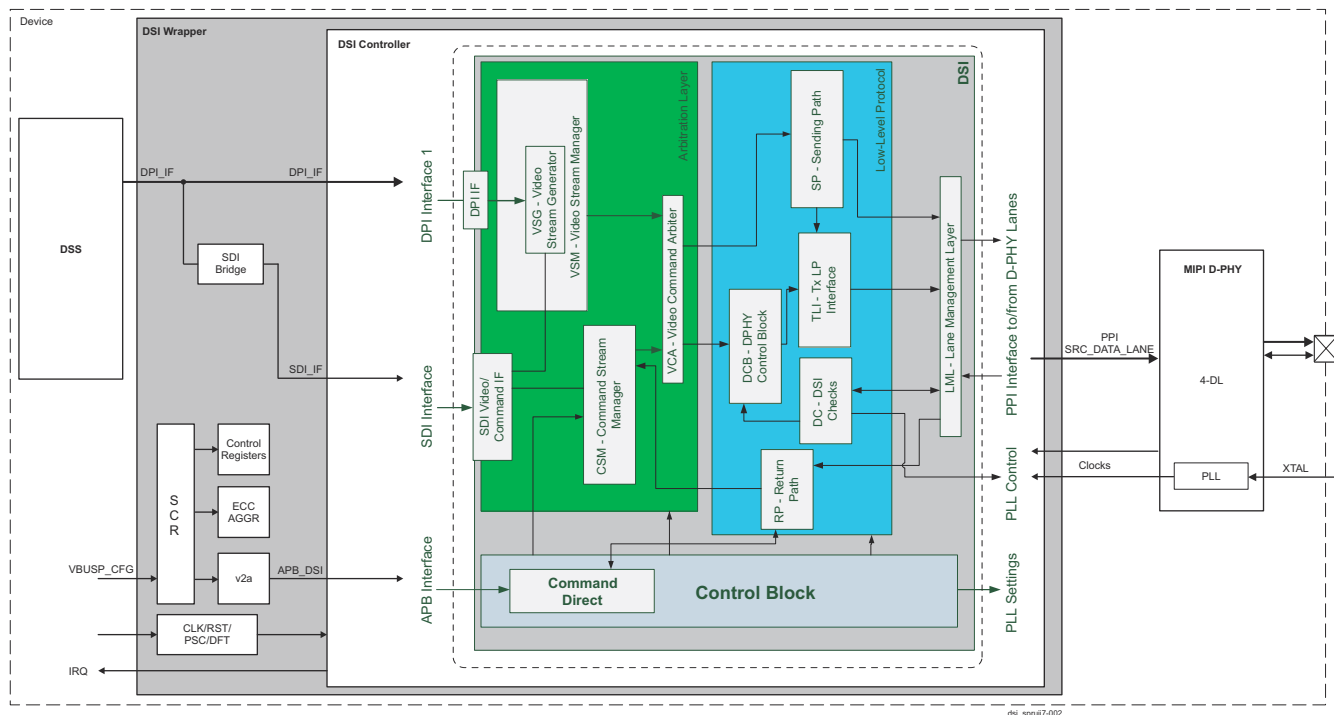


Figure 12-1072. Example Display Subsystem Showing Controller and PHY Blocks

#### 12.6.5.7.1.2 D-PHY And DSI Configuration

The DPHY and DSI controller must be configured before any panel video display can be used. All accesses to the DPHY to configure it for use with the DSI Controller should be performed through the DPHY own APB interface .

#### 12.6.5.7.1.3 DSI Controller Initialization

The DSI controller will always require the DPHY and data path configuration to be set after power up and any hard reset. The minimum sequence should define the data path lane configuration; mappings, available lanes, DPHY power, resets and lane swap. All these register values should be set before the DSI controller is enabled for functional operation.

The DSI registers are accessed through the APB interface. The programming sequence for command, video and controller configuration must follow a sequence of accesses to ensure the clock gate control updates the internal nodes before the link is enabled.

All DSI\_VID\_xxx registers must be programmed, if required, before the write to the DSI\_VID\_MAIN\_CTL register.

All DSI\_MCTL\_DPHY\_xxx registers must be programmed, if required, before the write to the DSI\_MCTL\_MAIN\_DATA\_CTL register to set the [0] LINK\_EN bit.

#### 12.6.5.7.1.4 Panel Configuration Using Command Mode

The DSI controller can provide DCS command access using either the SDI interface or through the APB interface control of the DIRECT COMMAND registers (see [Table 12-1491](#)). The panel registers will be programmed using the LP mode before the high-speed video is enabled. Commands can either be sent using the command interface or through the Command FIFO using the APB registers.

The command FIFO access is done through the APB Command mode registers by forming packets of DCS commands. The DCS commands can be found in the MIPI DCS specification or the display panel data sheet.

DCS commands are essentially register write or read transfers to panel registers. The initialization of a panel will require APB register writes to fill the FIFO with the DCS commands. The commands can be grouped into a long packet by sending the packet split into 4 bytes to fill the FIFO using a write to the DSI\_DIRECT\_CMD\_WRDAT register.

**Table 12-1491. DSI Direct Command Mode Registers**

DSI Register	Description
DSI_DIRECT_CMD_SEND	Direct Command - trigger the direct command sending - write only
DSI_DIRECT_CMD_MAIN_SETTINGS	Direct Command - main settings
DSI_DIRECT_CMD_STS	Direct Command - status - read only
DSI_DIRECT_CMD_RD_INIT	Direct Command - stop read operation
DSI_DIRECT_CMD_WRDAT	Direct Command - data to write byte 0 to 3
DSI_DIRECT_CMD_FIFO_RST	Direct Command- reset the write FIFO pointer

Each DCS command packet to be sent can be built by first selecting the packet configuration using the DSI\_DIRECT\_CMD\_MAIN\_SETTINGS. The DCS Command data types are defined in the DSI specification based on the number of parameters that are expected to be sent. Configuration of a panel display will normally use DCS write data types.

[Table 12-1492](#) shows the DSI\_DIRECT\_CMD\_MAIN\_SETTINGS register bit description.

**Table 12-1492. DSI Main Settings Register Description**

DSI_DIRECT_CMD_MAIN_SETTINGS Register Bit	Description
[24] CMD_LP_EN	Enables LP sending for the command request.
[23:16] CMD_SIZE	Size of the command in case of write command - when written value is bigger than 0x10, the value is rounded to 0x10 for write - when size is bigger to 0x2 for read it is rounded to 0x2.
[15:14] CMD_ID	In case of read/write command, Virtual Channel of the command.
[13:8] CMD_HEAD	In case of read/write command, data type of the command:
	0x05 DCS Write 0 parameters Short Packet
	0x15 DCS Write 1 parameter Short packet
	0x39 DCS Long Write N parameters Long Packet
[3] CMD_LONGNOTSHORT	This bit must be tied to one.
[2:0] CMD_NAT	Nature of the direct command: 000: write command.

Once the packet is loaded, issue a DIRECT\_CMD\_SEND (via the DSI\_DIRECT\_CMD\_SEND register) and the data will be sent in LP mode.

#### 12.6.5.7.1.5 VIDEO Interface Configuration

The DSI video operation requires the configuration of the VSG and TVG registers to match the panel configuration. The two video interface options will also require careful selection of the clock and registers to achieve error free video streaming.

The DSITX controller supports the mechanism for initial skew calibration for D-PHY data rates greater than 1.5 Gbps.

#### DPI Video Interface Operation

The time taken to output a frame on the PPI needs to match what is coming in on the DPI. This is best achieved by using the recommended clock ratio between the pixel and byte clocks. If this is the case, then the blanking and active data periods must be matched up. The DSI controller will slave its frame timing to the incoming DPI **VSYNC**, if it is programmed to generate a frame of slightly less than the same size when the VFP blanking is considered.

The controller works by transitioning to LP during the last programmed line of VFP. It will then remain in LP until the start of the next frame. So, programming the controller to match the DPI, but with at least one fewer line of VFP should result in a reliable configuration.

Program the DSI vertical size registers as follows:

- VSA = DPI\_VSI (lines, minimum of 2)
- VBP = DPI\_VBP (lines, minimum of 0)
- VACT = DPI\_VACT (lines)
- VFP < DPI\_VFP (minimum of 1)

The timing should also be matched per line, therefore the blanking and active periods should match. DPI horizontal timing is measured in pixels, whereas the DSI controller uses bytes. If any of the DPI related interrupts are triggered, then this highlights that the FIFO depth and/or the vsync\_delay settings require to be tuned to the current configuration. Simulating the core operation with the expected clocks is the best way to ensure the FIFO depth and vsync\_delay is suitable.

The relationship therefore depends upon the pixel format, which could be 24, 18 or 16 bpp. Additionally, the PPI short packets and packet headers that are inserted by the controller must be accounted for. HSA should be reduced by 14 bytes to account for the HSS short packet (4 bytes), the long blanking packet header and footer (4 + 2 bytes) and the HSE short packet (4 bytes). HBP should be reduced by 12 to account for the header and footer on the blanking packet (6 bytes) plus the header on the active data packet (6 bytes). HFP should be reduced by 6 bytes to account for the long packet header and footer. Finally, for lines with no active data, the BLKLINE\_PULSE\_PCK is the total size minus 20 bytes (14 for HSA, 6 for the remaining blanking which is all combined into a single packet).

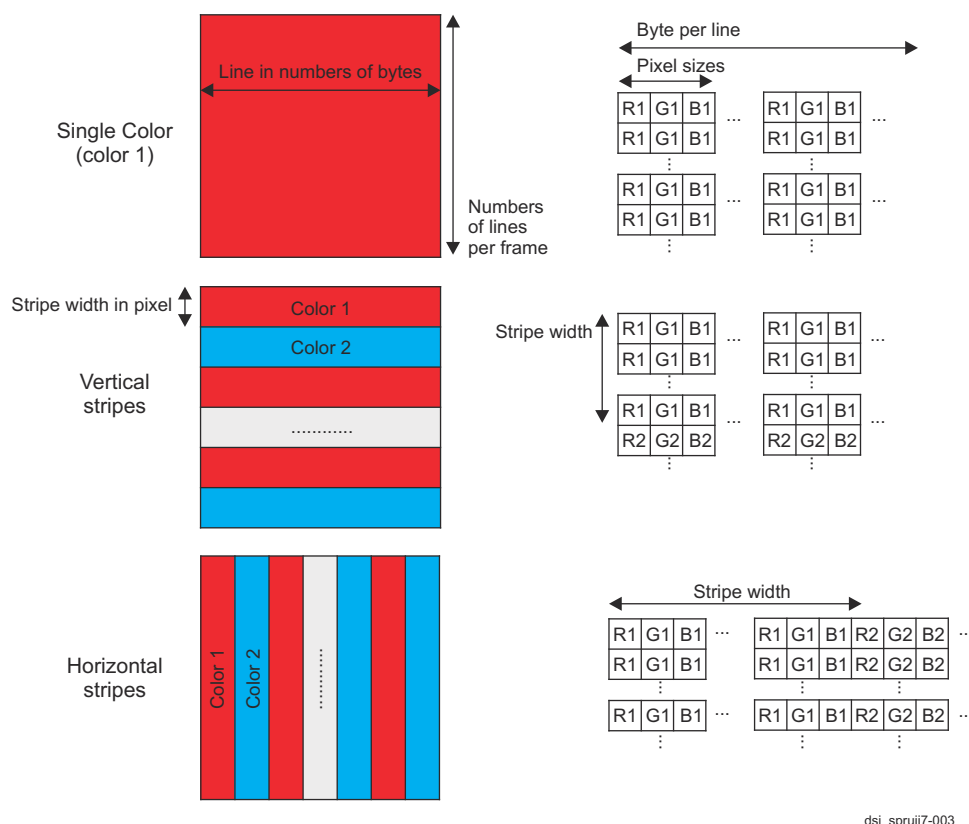
Program the DSI horizontal size registers as follows:

- HSA = (DPI\_HSA × bpp/8) - 14
- HBP = (DPI\_HBP × bpp/8) - 12
- HACT = (DPI\_HACT × bpp/8) NOTE: (DPI\_HACT × bpp/32) must be an integer.
- HFP = (DPI\_HFP × bpp/8) - 6
- BLKLINE\_PULSE\_PCK = ((DPI\_HSA + DPI\_HBP + DPI\_HACT + DPI\_HFP) × bpp/8) - 20;

#### TVG Generator

The TVG oversees generating dummy data (to display something even if there is no video stream running). It is also able to ease verification or validation of the DSI without having a complete environment (application or verification test bench). The controller has a Test Video Generator that can be programmed to generate a set of test colour patterns based on the display panel that will be used. The panel parameters for horizontal and vertical resolution along with the frame rate and pixel colour depth need to be set based on the datasheet information for the panel.

Figure 12-1073 presents TVG MODE patterns.



dsi\_sprui7-003

**Figure 12-1073. TVG MODE Patterns**

The TVG generates a data stream in the same format than the one specified for the normal functional video stream (as set in VSG register). The content of that flow is specified by registers. Those registers specify the following information: Image size: Number of bytes per line (see TVG\_LINE\_SIZE[14:0] register) and number of lines per frame (TVG\_NBLINE[12:0]).

Image kind: Single color, vertical stripes or horizontal stripes (see TVG\_MODE[1:0] register). Stripe width: Significant when a stripe mode is enabled. Possible values are 1, 2, 4, 8, 16, 32, 64 and 128 (see TVG\_STRIPE\_SIZE[2:0] register).

Pixel kind: 16 bits RGB, 18 bits RGB, 18 bits loosely RGB, 24 bits RGB, 30 bits RGB or 36 bits RGB (see VID\_PIXEL\_MODE[3:0] register).

Color one: Color used in single color mode or first color when in stripe mode. 12 bits per component R/G/B. For formats using fewer bits than 12, the least significant ones are the ones considered (see COL1\_GREEN[11:0], COL1\_RED[11:0], COL1\_BLUE[11:0] registers).

Color two: Color two when in stripe mode. 12 bits per component R/G/B. For formats using fewer bits than 12, the least significant are the one considered. (see COL2\_GREEN[11:0], COL2\_RED[11:0], COL2\_BLUE[11:0] registers). Start pulse and stop handshake (+ stop mode) (see TVG\_CTL register).

An example of the sequence is as follows:

- Select TVG for video stream generation instead of SDI 1 interface:
  - IF1\_EN bit to 0 (stall signal at 1) in MCTL\_MAIN\_EN.
- Select video mode for interface 1 in MCTL\_MAIN\_DATA\_CTL:
  - TVG\_SEL bit to 1 in MCTL\_MAIN\_DATA\_CTL register to select stream from TVG.
- Select generated frame format in TVG\_CTL register:
  - Image format (single color, H stripes, V stripes) in TVG\_MODE field.

- Image color selection in TVG\_COLOR1, TVG\_COLOR1\_BIS, TVG\_COLOR2, VG\_COLOR2\_BIS.
- Stripes size in TVG\_STRIPE\_SIZE field.
- Image size in TVG\_IMG\_SIZE register.

**Note:** The number of lines per frame and number of bytes per line. It is required that TVG settings on active area match VSG settings on active area. Any mismatch will create an error that is detected in the VSG and forces recovery mode in TVG, stopping test frame generation.

- Select TVG stop mode with TVG\_STOPMODE field of TVG\_CTL:
  - TVG video stream generation start/stop controlled by the TVG\_RUN bit in TVG\_CTL register. Polling TVG run status from the TVG\_RUNNING bit in TVG\_STS register is useful when setting TVG\_RUN to 0. TVG video stream generation does not stop instantaneously (depends on TVG stop mode), so this should be checked to confirm the TVG has stopped.

#### 12.6.5.7.2 Application Considerations

The following sections outline the normal operation of the DSITX controller and the programming and sequences require to operate the DSITX controller for normal video and command operation.

##### 12.6.5.7.2.1 D-PHY Timings Control

Several DPHY timing constraints must be respected by the system, which are described in this section. When a switch in ULP is requested, the application should be aware that the D-PHY needs 1 ms to leave this state (this timing is guaranteed by the DSI itself). It makes no sense to start an ULP request if the expected time in ULP is short.

When PLL power down is asserted, the system should continue to assert for a specific minimum period of time (PLL internal requirement – please refer to DPHY documentation). Please note that shutting down the PLL implies that no clock is present on tx\_byte\_hs\_clk input and thus the DSI link is stopped; even if the system-side interface clocks remain active, no more HS transfers can be done.

Force\_stop, ppi\_c\_force\_tx\_stop (clock lane) and ppi\_d\_force\_tx\_stop (1 per data lane), can be useful to resolve certain deadlock situations, for example: DSI slave does not give back control to master, or critical DPHY error. When a 'force stop mode' is issued by the application, the system should maintain this state on the bus to ensure the request is correctly considered by the D-PHY cells and the **stop\_state** is asserted and direction signal is deasserted.

reg\_wait\_burst\_time (for the LML) must be programmed to ensure that two HS bursts are separated by at least 100 ns. The value will be based on the period of the tx\_byte\_clk cycle.

VCA: When trying to interleave commands to video stream, it is assumed that only write/read commands and BTA request are sent (no TE). The reason is that such transfers are slow and difficult to predict in term of duration. They could take longer than the slots available for video. The read commands are unpredictable and may cause some break in the video stream, however, they can still be used, if an appropriate response time can be guaranteed. It is the responsibility of the system integrator and the application to plan and implement recovery procedures when the video stream is stalled due to a read that takes too long.

##### 12.6.5.7.2.2 Control Block

Most of the register contents must be resynchronized against a DSI internal clock (tx\_hs\_byte\_clk). This means there is a certain time to pass data from one clock domain to the other. If two writes in the same register are too close, the register itself is updated but the synchronization may fail.

The formula to calculate the number of system clock periods between two writes is:

$$\text{nb\_cycle}(\text{dsi\_p\_clk}) = 6 * (\text{f}_{\text{dsi\_p\_clk}} / \text{f}_{\text{tx\_byte\_hs\_clk}} + 1)$$

The application must respect a period of TX ns between two write accesses to the same register (there is no issue writing different registers back-to-back).

[Table 12-1493](#) shows an example of required time between two write accesses.



**Table 12-1493. Example of Required Time between Two Write Accesses**

$f_{\text{dsi\_p\_clk}}$	$f_{\text{tx\_byte\_hs\_clk}}$	TX minimum time between 2 writes accesses
200 MHz	106 MHz	~ 90 ns
160 MHz	10 MHz	~ 650 ns
200 MHz	10 MHz	~ 650 ns

Another place where asynchronous behaviour may interfere with programming in the CB is the direct command status register `DIRECT_CMD_STS`. Some of the bits of this register are set when a pulse (that lasts one clock period) coming from `tx_hs_byte_clk` domain is observed (after resynchronization on `dsi_p_clk`) and is reset when the clear bit is written. In use case where `tx_byte_hs_clk` is slow (10 MHz range), the bit can be read and a clear attempted before the end of the source pulse. This can result in the re-assertion of the bit immediately after it is cleared. For that reason, it is recommended to wait for a while between reading the bits of the `DIRECT_CMD_STS` register and clearing them.

### Application Issues

Some register fields cannot take all the possible values but are restricted to a certain number of combinations (mode control). The DSI controller does not check that all the fields match a valid setup so it is the responsibility of the system integrator and the application to check that the written values are amongst the permitted combinations. Amongst the register fields that fall in to this category are most of the mode settings (stop mode, recovery mode, direct command type, image sizes, etc.)

### Programming Coherency

The application level must ensure that the register configurations are valid for the system to operate correctly. There are several possible combinations are possible in the CB registers however should be avoided. A few examples:

- No HS transfer should be enabled when PLL is shut-off.
- TVG should not be run when the video interface is running.
- No access (from interface or from register) should be attempted while TBG is active.
- Prior to shut-down of the DPHY PLL, application should verify that the DSI link is in proper state. If the bit that enables BTA is not set and that any operation implying a BTA is sent, the system is stalled. Then the bit that enables BTA must be aligned with application needs. Moreover, read enable or TE enable cannot be set if `bta_en` is not set (these operations cannot be enabled when BTA is not enabled).
- TE feature must not be enabled in SDI interface.

#### 12.6.5.7.2.3 Video Coherency

In video mode, there are a lot of sizes defined for the video stream generation. This data corresponds to the payload of the various generated packets and not to their duration. However, they are closely linked. It is the responsibility of the system integrator and the application to use correct sizes to ensure the correct video timing and behaviour, because the system may use 1 to 4 active data lanes with a fixed number of bytes for the header/checksum overhead. In the same way, when programming D-PHY time (to switch from LP to HS), the application must consider D-PHY time plus overhead due to LLP and LML crossing time (see [Section 12.6.5.7.7.2, Video stream settings \(VSG\)](#)).

### VSG Control

Start and stop sequences can take a long time to be performed. If the requests are not maintained up to the time the status bit are indicating effective start or stop, they can be ignored and the VSG may not have started or stopped as expected by application. It is then up to the application to carefully manage the request and to verify that requests are being processed before changing the state of the VSG.

### Test Generator:

The DSITX provides test generator to provide a video data stream. When using TVG, all sources can be used (provided VCA permits it) however the SDI interface must not be programmed to send video data and must be restricted to command mode.



The pixel modes supported are: RGB 16-, 18-, 24-, 30- and 36-bit; YCbCr422 16-bit; and YCbCr420 12-bit (note that additional YCbCr422 20-bit and YCbCr420 24-bit may be supported using SDI interface, but are restricted to command mode only).

The supported display sizes are listed below:

- QQVGA (160 x 120) - 15/20/30/60 fps - RGB 16-18-24 (-30 and 36) bits per pixels
- QCIF (176x144), QCIF + (176x208 and 176x220) - 15/20/30/60 fps - RGB 16-18 and 24 bits per pixels
- QVGA (320x240) - 15/20/30/60 fps - RGB R16-18 and 24 bits per pixels
- CIF (352x288), CIF + (352x416 or 352x440) - 15/20/30/60 fps - RGB 16-18 and 24 bits per pixels
- 1/2 VGA (320x480) and 2/3 VGA (640x320) - 15/20/30/60 fps - RGB 16-18 and 24 bits per pixels
- VGA (640x480) - 15/20/30/60 fps - RGB 16-18-24(-30 and 36) bits per pixels
- WVGA (800x480 - 848x480 - 854x480 - 852x480) - RGB 15/20/30/60 fps - 16-18 and 24 bits per pixels
- SVGA (800x600) - 15/20/30/60 fps - RGB 16-18 and 24 bits per pixels
- QHD (960x540) - 15/20/30/60 fps - RGB 16-18 and 24 bits per pixels
- XVGA (1024x768) - 15/20/30/60 fps - 16-18 and 24 bits per pixels
- Full HD (1920x1080) - 15/20/30/60 fps - 16-18 and 24 bits per pixels
- WUXGA(1920x1200) - 15/20/30/60/(120 fps – 16 bpp only) 16, 18 and 24 bits per pixels
- 4Kx4K(4096x4096) - 15/20/30 fps - 16-18 and 24 bits per pixels

### 12.6.5.7.3 Start-up Procedure

The start-up sequence described in [Figure 12-1074](#) assumes that the rest of the chip is ready (all clocks are present except tx\_byte\_hs\_clk, application it is ready...) and that the DSI is ready (reset de-asserted).

The following is the most common start up sequence:

Firstly, application layer programs and enables the PLL, then waits for a period for the PLL to lock (minimum time will be defined by the PHY documentation). At the end of the minimum wait period, the application layer should poll the PLL lock status register periodically to confirm lock status, repeating until confirmed. At the end of this step, it is assumed that a clock is present on the tx\_byte\_hs\_clk input (this step may be replaced by an interrupt-based alternative if available from the selected DPHY or provided in the system integration logic).

During the time taken by the PLL to lock, the application can program the configuration registers and prepare the DSI link. This can also be done after the PLL is locked. It should at least configure enough to be able to use LP mode to send direct commands.

Enable clock and data lane(s) per the needs. This action is only concerned with programming registers that control the D-PHY lane enable signal. The active lane configuration with the DSI\_MCTL\_MAIN\_PHY\_CTL and DSI\_MCTL\_MAIN\_EN registers must be programmed to match, i.e. for two data lanes (0 and 1).

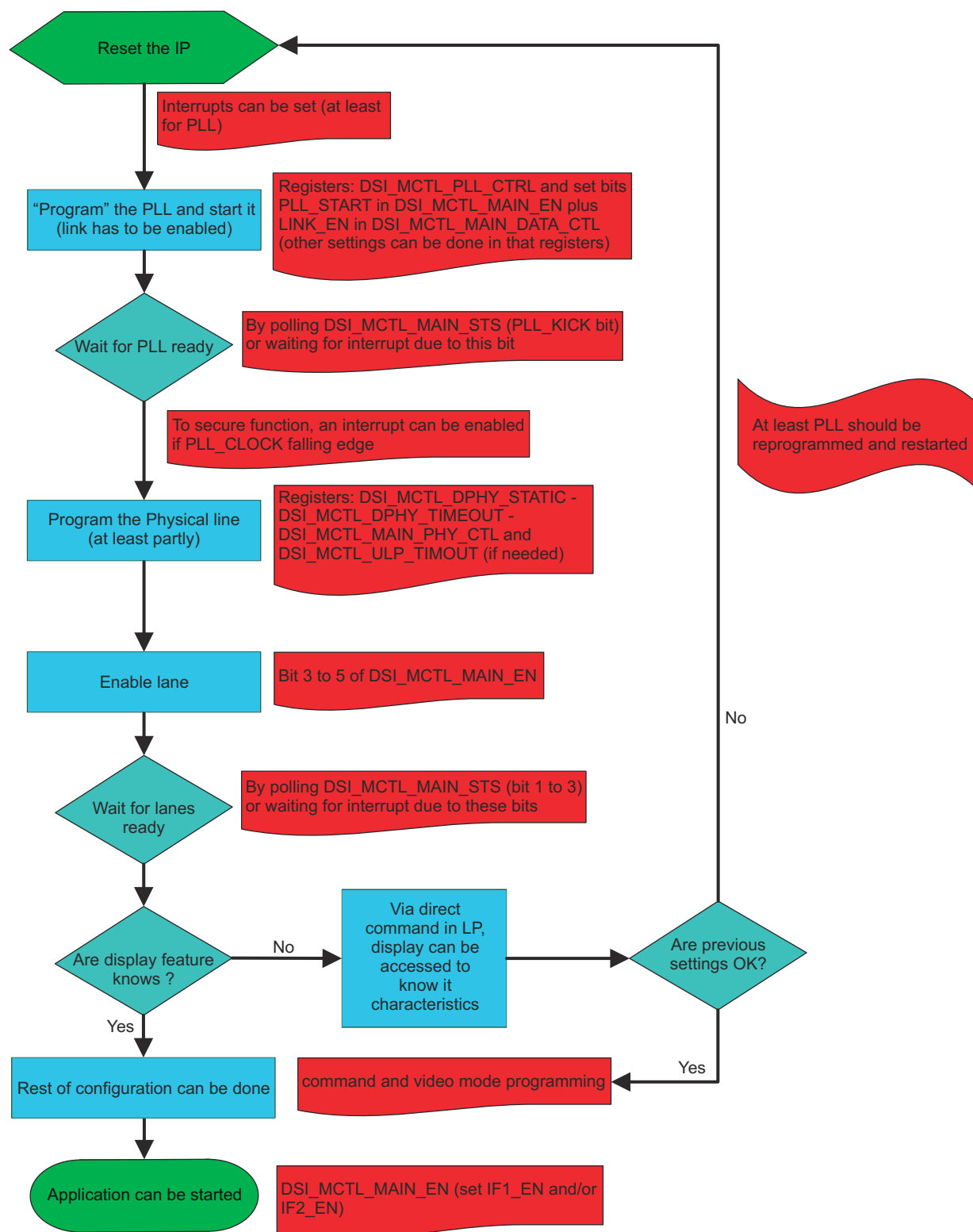
- DSI\_MCTL\_MAIN\_PHY\_CTL (0x08) register – LANE3/4\_EN - bits[2:1] set to zero to disable lanes 2/3.
- DSI\_MCTL\_MAIN\_EN (0x24) register – DAT3/4\_EN fields - bits[6:5] set to zero to disable lanes 2/3.

Start the lane in the DCB. The DCB state machine puts the lanes in the correct state to be ready.

At this step, the DSI link should be able to access to display - at least in LP mode. If the system (DSI link and display) is fully programmed, code to display image can be started. If display characteristics must be read from the display and the system is programmed in LP mode, this is time to read the displays parameters to allow correct programming of the DSI. A new round of programming can then take place to set-up the system, based on the parameters which have been read fro the display.

The DSI internal setting is now completed and it is ready to accept data. The application can now enable the interface so that the stall signal is removed and that DSI link can accept data.

When the procedure described above is terminated, it does not imply that display itself is ready. At this stage, the D-PHY and DSI link are ready to send/receive data to/from display and thus application may still need to initialize the display itself.



dsi\_spruij7-005

**Figure 12-1074. Start-up Procedure Summary**

#### 12.6.5.7.4 Interrupt Management

There are numerous error and status conditions that can be monitored via the interrupt mechanism. The edge on which these conditions are generated is programmable.

Each interrupt source has four associated registers:

1. The status register itself "<sts\_reg>"
2. a control register "<sts\_reg>\_ctl" to control corresponding interrupt behaviour
3. a flag register "<sts\_reg>\_flag" to observe when interrupt/event has been triggered
4. A clear register "<sts\_reg>\_clr" to clear the event flag register.

The decision of which status bits (<sts\_bit>) can generate interrupt is taken when the corresponding enable bit "<sts\_bit>\_en" is set in the status control register. In the same register the status bit edge "<sts\_bit>\_edge" states on which edge of the status bit the interrupt is triggered. When the selected edge is observed, the flag bit "<sts\_bit>\_flag" is set. The interrupt signal is an OR of all flag bits that are enabled. When the register "<sts\_reg>\_clr" is written with the bit "<sts\_bit>\_clr", the "<sts\_bit>\_flag" is cleared.

##### 12.6.5.7.4.1 Error and Status Registers

The error register bank handles storing all error flags that are identified. It also stores some status bits that must be observable and detectable by the application. These status and error registers can generate an interrupt on the rising or falling edge if this generation is enabled.

This mechanism is different depending on the nature of the status or error bits. For those that are generated in the DSI block (basically all status and error bits except direct command), the status bits cannot be easily controlled and toggle according to the internal status meaning only the current value is observed in the status/error register itself.

If the interrupt generation is enabled with the associated enable bit, a rising edge (associated edge bit set to 0) or a falling edge (associated edge bit set to 1) of that status bit toggles the corresponding flag bit. At the end, all the flag bits are put together with an OR to generate the interrupt signal. The flag register can be reset by writing in the clear register.

The described behavior can be summarized by the following pseudo-HDL. The signal named signal\_sts\_bit is the bit that is observable by reading status bit.

- reg\_sts\_q is the status register (accessible via APB)
- reg\_flag\_sts\_q is the flag register
- clear\_the\_flag is set when the register clear is written
- edge\_sts\_ctl is the edge bit
- enable\_sts\_ctl is the enable bit

```
reg_sts_d <= signal_sts_bit;
```

```
reg_flag_sts_d <= '0' WHEN clear_the_flag = '1' ELSE
```

- '1' WHEN (reg\_sts\_q = '0' AND reg\_sts\_d = '1' AND edge\_sts\_ctl = '0') ELSE – detect rising
- '1' WHEN (reg\_sts\_q = '1' AND reg\_sts\_d = '0' AND edge\_sts\_ctl = '1') ELSE – detect falling
- reg\_flag\_sts\_q;

```
irq_n <= NOT (OR_OF_ALL(reg_flag_sts_q AND enable_sts_ctl));
```

The code is slightly different in cases where the observed bit is a generated condition (in the control block), as is the case for the direct command status and error flags. The status/error information is automatically generated but is cleared only when writing in the corresponding clear bit. The interrupt generation behaves similarly to the error condition cases, however, as the status falling edge is observed only when the bit is cleared, it is not possible to use the falling edge detection on flag (however the code is kept as-is to provide a standard implementation method). This leads to the following code:

- reg\_sts\_d <= '0' WHEN clear\_the\_flag = '1' ELSE
- '1' WHEN the\_sts\_bit (Note 1) = '1' AND reg\_sts\_q = '0' ELSE -- (see Note)

- reg\_sts\_q;
- reg\_flag\_sts\_d <= '0' WHEN clear\_the\_flag = '1' ELSE
  - '1' WHEN (reg\_sts\_q = '0' AND reg\_sts\_d = '1' AND edge\_sts\_ctl = '0') ELSE – detect rising
  - '1' WHEN (reg\_sts\_q = '1' AND reg\_sts\_d = '0' AND edge\_sts\_ctl = '1') ELSE – detect falling
- reg\_flag\_sts\_q;
- int <= NOT(OR\_OF\_ALL(reg\_flag\_sts\_q AND enable\_sts\_ctl));

**Note:** In some case (pulse generated), the rising edge detection is not done and the code becomes simply '1' WHEN signal\_sts\_bit = '1' ELSE...

#### 12.6.5.7.4.2 Interrupt Management for Direct Command Registers

The following direct command status bits/registers are only set when error is generated and only cleared when the clear bit is written:

- all the status bits of the register DSI\_DIRECT\_CMD\_RD\_STS
- all the status bits of the register DSI\_DIRECT\_CMD\_STS
- all the status bits except bit 0 (cmd\_transmission) of the register DSI\_DIRECT\_CMD\_STS

As these registers are reset only when the clear register is written, it is not possible to detect the falling edge on them to generate interrupts. Only the rising edges can generate the interrupt.

**Note:** There can be issues with detection when using the bits for all the signals that are a pulse generated in the tx\_byte\_hs\_clk domain. When the speed of that clock is slower in relation to dsi\_p\_clk, it is possible to have the interrupt set, the bit read and the clear attempted before the end of the tx\_byte\_hs\_clk pulse, after the clear, the bit is set again.

#### 12.6.5.7.5 Direct Command Usage

The direct command mechanism is a way to send commands (with a maximum size of direct\_cmd\_fifodepth bytes) by writing and reading registers. It allows commands to be sent that are not allowed through the SDI-DSI interface (DSI commands, triggers), to perform read operations or to send DCS / generic write / generic read operations when the DSI link is not fully programmed and cannot use the SDI interface for commands.

Figure 12-1075 shows direct command management sequence.

Basically, the procedure to do so is as follows:

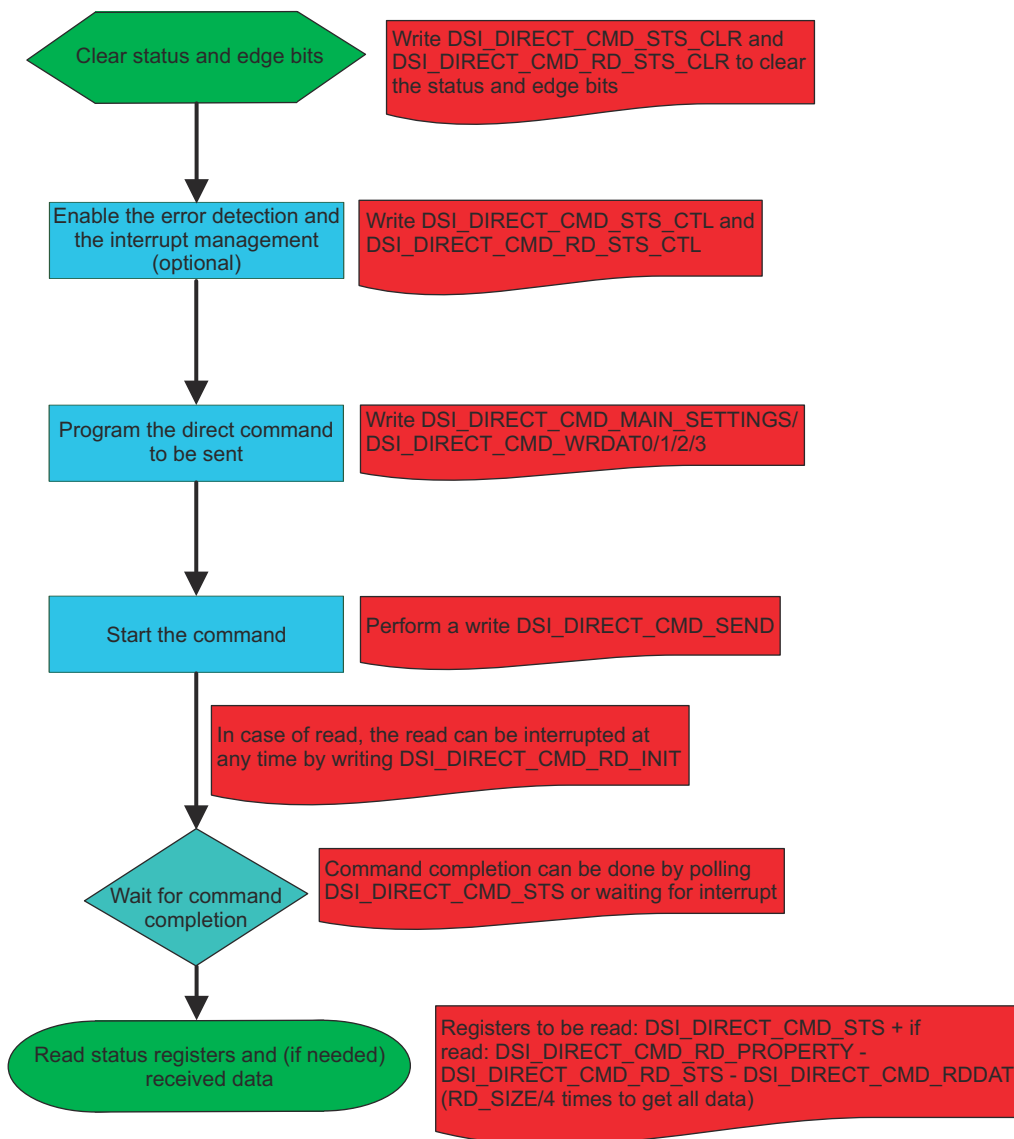
- Clear the status bits (when reading the status bits, only the DSI reads the current access status).
- Program the various registers that define the command to send.
- Write the direct\_cmd\_send register. Writing in that register (any value) triggers the command sending mechanism.

The command should be started only if the command request is amongst the defined ones for DCS or Generic Write or Read with 0, 1, or more parameters:

- DCS Write codes 0x05, 0x15, 0x39
- DCS Read codes 0x06
- Generic Write codes 0x03, 0x13, 0x23, 0x29
- Generic Read codes 0x04, 0x14, 0x24

**Note:** The direct\_cmd\_main\_settings register can use other data type values for cmd\_head to support future types and no check is performed for invalid types.

- Wait for the assertion of the command\_completed bit in the direct\_cmd\_sts register (or any of the relevant bits that indicate a specific command termination). Instead of polling the registers, it is possible to enable the corresponding interrupts and wait for its detection.
- Read the received data and (if any) the different status bits.



dsi\_spruij7-006

**Figure 12-1075. Direct Command Management**

There are several checks performed when sending a direct command to verify that requested command sizes are supported and to check that the requested command exists (otherwise the request to send the command is ignored).

- For Short packets commands the DCS and Generic Read and Write the data size is rounded to maximum of 2.
- For Long packet DCS and Generic Read and Write the data size is rounded to the maximum size of the FIFO.
- No checks are made for the write FIFO having the expected number of bytes programmed into the CMD\_SIZE in the dir\_cmd\_main register. In this case the data written out will loop back through the value in the FIFO.

At completion of a read command, registers can be read (they contain read data and/or errors detected during the communication) and trigger events are sent to the application to announce the request completion (using the status and error register bank).

When a read request is performed, the read data is put in the RP FIFO. If more bytes are sent, they are ignored and errors are set accordingly. The received read can then be accessed by reading several times the register that gives an access to the FIFO.

For every new command, several status bits are present: one signals that a command is being transmitted (cmd\_transmission). When the command is complete, several bits can be toggled to signal it (write\_completed, trigger\_completed, te\_received, read\_completed and read\_completed\_with\_err).

Some complex commands provide intermediate information. For BTA requests, there is a bit that signals when the BTA request has been sent (BTA\_completed) and another one that signals when the DSI is master of the D-PHY interface again (BTA\_finished). For commands that imply a BTA, some other bits specify whether specific data has been received (such as trigger, acknowledge with or without error).

#### 12.6.5.7.5.1 Trigger Mapping Information

Triggers are one of the groups of direct commands that can be sent and received using the DPHY in Low Power escape mode. The trigger\_val[3:0] bits (in the registers DIRECT\_CMD\_MAIN\_SETTINGS and DIRECT\_CMD\_STS) are used to define which of the four possible trigger entry codes has to be sent or has been received. The register fields described above are copied in signal ppi\_d1\_tx\_trigger\_esc[3:0] (for trigger\_val[3:0] of DIRECT\_CMD\_MAIN\_SETTINGS) or are a copy of ppi\_d1\_rx\_trigger\_esc[3:0] (for trigger\_val[3:0] of DIRECT\_CMD\_STS). Only one bit out of the four should be set else the D-PHY behavior is not very precisely defined. [Table 12-1494](#) gives the trigger mapping in the system.

**Table 12-1494. Trigger Mapping**

Trigger name	Trigger entry code	trigger_val	meaning in TX direction	meaning in RX direction
trigger 0 - reset	01100010	1b0001	Reset	Not affected by DSI spec
trigger 1 - unknown 3	01011101	1b0010	Not affected by DSI spec	TE response <sup>(1)</sup>
trigger 2- unknown 4	00100001	1b0100	Not affected by DSI spec	Acknowledge with no error <sup>(1)</sup>
trigger 3 - unknown 5	10100000	1b1000	Not affected by DSI spec	Not affected by DSI spec

(1) These triggers are not observed in the trigger\_val field of the direct\_cmd\_sts register because they are used by the DSI link for a specific purpose.

**Trigger Reset** – This will request from the display and expects an immediate reset of the DSI command/video mode with all pending requests in TX sending path FIFO for transfer to be discarded.

**Acknowledge** – This is a response to DSITX controller. The host processor may request a command acknowledge and error information related to any transmission by asserting Bus Turnaround with the transmission. The peripheral shall respond with ACK Trigger Message if there are no errors and with Acknowledge and Error Report packet if any errors were detected in previous transmissions. Appropriate flags shall be set to indicate what errors were detected on the preceding transmissions.

If the transmission was a Read request, the peripheral shall return READ data without issuing additional ACK Trigger Message or an Acknowledge and Error Report packet if no errors were detected. If there was an error in the Read request, the peripheral shall return the appropriate Acknowledge and Error Report unless the error was a single-bit correctable error. In that case, the peripheral shall return the requested READ data packet followed by Acknowledge and Error Report packet with appropriate error bits set.

**Tearing Effect** - The Tearing effect on the display is avoided by having synchronization information from the display. It is used only in command mode. Users are responsible for selecting the command mode for the VC using the TE feature and selecting the polling or automatic mechanism.

The user is responsible for ensuring that there is a delay after a trigger is issued by the host to the panel, before any other action, message or trigger is performed. When the host issues a reset trigger, the system will expect the controller to stop and re initialise the clock and data links. All other triggers issued by the host will be application specific and are outside the scope of this document.



#### 12.6.5.7.5.2 Command Mode Settings

The command mode is enabled on the SDI interface. This is controlled using the register `mctl_main_en`. Other settings can be set using the registers: `mctl_main_data_ctrl` (to decide about TE usage, read enable...); and `cmd_mode_ctl` (Virtual Channel of the command packets, arbitration between requests of the two interfaces, possibility to use LPDT, TE timer programming and padding value in case of an error).

When programming Direct READ commands, a BTA request is sent automatically following the read transmission, for the peripheral to respond – it is therefore not necessary to explicitly send a BTA request at that time. The system must allow the BTA request to be completed and the bus returned to the host before issuing any new read or write command. The system must either: poll the `read_completed` (and `read_completed_with_err`) status bits; or, wait for an interrupt caused by these bits before issuing any new command.

When programming Direct WRITE commands, it is advisable to ensure that each transaction can complete successfully before moving on to a subsequent command. There are two recommended methods to achieve this, either: explicitly request a BTA between write transactions while checking for the associated interrupt or polling the appropriate status flag (this method also provides the added security of an ACK response from the peripheral); or allowing a gap between commands sufficient to allow completion of the first before commencing with the second. The recommended approximate gap time required to ensure that all commands complete successfully is equivalent to 100 TX ESC byte periods. If required (for instance, to minimize the start-up time), it is also possible to calculate the minimum delay time required for each transaction, based on the TX escape clock frequency used in the application and the length of each command (no active video transmission should be enabled at the time – a typical use case would be peripheral parameter configuration). As with read operations, any write operation followed by an explicit BTA request must allow the BTA to be completed and the bus returned to the host before issuing any new read or write command.

For single parameter commands, the upper bytes of the 32-bit value written to the Direct Command write data register (`direct_cmd_wrdat`) should be masked to zero to comply with the DSI specification requirement that unused parameter locations are set to zero. For zero parameter writes, assuming there have been previous write commands sent, it is necessary to clear the sending path FIFO by writing to the `direct_cmd_fifo_rst` register to clear the data path, then write a zero value to the `direct_cmd_wrdat` register to send the zero parameter write command.

**Note:** Failure to follow these procedures may result in non-zero data in the unused parameter locations; this may or may not affect the peripheral, however it is recommended to use the process described to ensure compliance with the DSI specification in this regard.

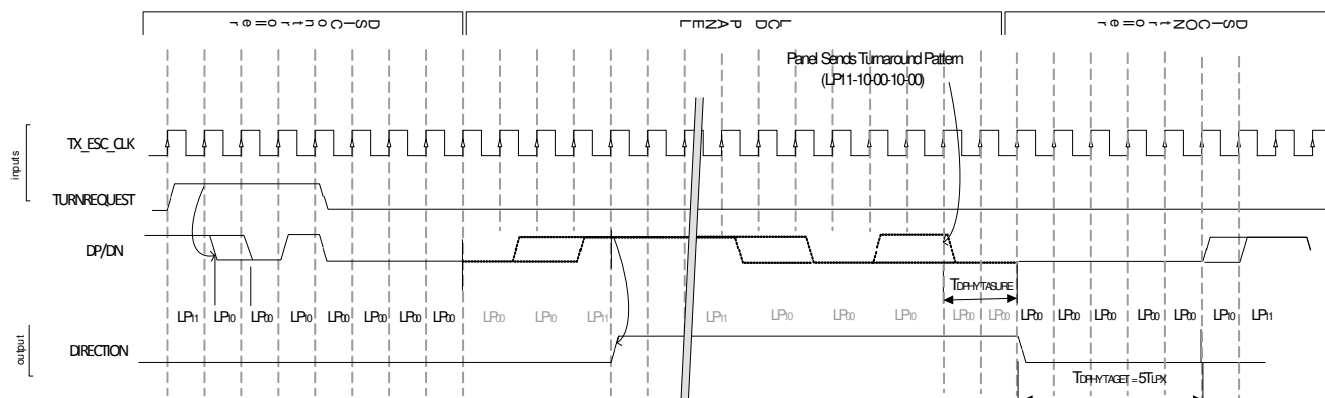
If STOP mode is forced during a command mode read or write transaction, all outstanding transactions should be considered lost and all status and error flags should be disregarded and cleared. Similarly, if an LPDT read data operation is stopped by writing to the `STOP_READ_OPERATION` field of the `DIRECT_CMD_RD_INIT` register, the status for the aborted read operation should be disregarded and cleared before resuming normal operation.

#### 12.6.5.7.5.3 Bus Turnaround Sequence

The command operation can request a read or an acknowledge response from the panel after a long sequence of packets is sent. The Bus turnaround is also used to control the Tearing Effect control.

When a Bus Turnaround is requested from the command stream manager, the DPHY will be given a turn request. The request will make the DPHY interface perform a sequence of LP states (LP11-10-00-10-00) to indicate that it is releasing control of the DP/DN signals. The DPHY Direction signal will change once it detects the Panel side driving the LP00-10-11 states. The panel will then be able to send any response packets (see section 0) and release the bus by issuing a bus turnaround sequence to the controller DPHY. The DSITX controller DPHY will detect the sequence and change the direction signal once the pattern is sent and LP00 state is valid for 2-3 TX\_ESC\_CLK cycles.

Figure 12-1076 shows Bus turnaround timing sequence from controller to panel and panel to controller.



**Figure 12-1076. Bus Turnaround Timing Sequence from Controller to Panel and Panel to Controller**

#### 12.6.5.7.5.4 Tearing Effect Control

The DSI command control for a tearing effect request can be performed using the polling method or automatically. The enable bits for BTA and TE must be enabled for the mechanism to be performed with either the SDI or DIRCMD interface. For automatic operation, the DSI will send a first BTA, and then waits for BTA reception and if no TE has been received during the first BTA <sup>(1)</sup>, it sends another BTA and then waits for TE. There are three possible error conditions that can happen (but only the two first are detected by the DSITX controller and passed to registers):

- If the display answers to the second BTA with another BTA and thus does not sent the TE - this means that the display does not support TE generation OR that TE generation mechanism has not been enabled (reg\_err\_no\_te).
- If the TE has not arrived within a given programmable period – this means that the TE request arrived too late and that the Display Application Processor is not synchronized with the display. The error reported is reg\_err\_te\_miss but the system continues to wait until the TE is received. <sup>(2)</sup>
- If a time-out has been detected by the DSI showing that a problem happened during the BTA and that the system is forced back to idle without BTA – in this case, a BTA without TE is seen and the CSM generates a reg\_err\_no\_te (as for first case).

For the polling method to request a tearing effect (TE polling mode is enabled with te\_hw\_polling\_en = 1) the DSI sends a first BTA, waits for peripheral TE + BTA, if only BTA has been received after the first BTA, it keeps sending BTAs until peripheral responds to the BTA by a TE + BTA instead of BTA only; in this specific mode reg\_err\_no\_te will never be asserted. To stop sending BTA and waiting for TE a force stop <sup>(3)</sup> ) is needed.

**Note 1:** TE received is checked on the first BTA in case a BTA was the last command issued by the DSI link (BTA or read command sent via direct command interface).

**Note 2:** Again to support the cases where a BTA was emitted just before the TE request, the TE windows is counted on the first BTA (in case the display understands this BTA as the TE request and thus may emits too late the TE), and is restarted after the second BTA.

**Note 3:** The time to wait before forcing a stop should be around the duration of a frame. The reason is that if you miss the TE time at the end of the current frame, you've to wait till the end of the next frame before new TE can be sent by display.

The following table describes the TE timeout counter operation, Programming of the IP. The counter value should be calculated based on the tx\_byte\_clk period.

**Table 12-1495. TE Timeout Programming**

te_timeout(11)	te_timeout(10)	timeout value
0	0	$256 \times \text{te\_timeout}<9:0>$
0	1	$512 \times \text{te\_timeout}<9:0>$
1	0	$1024 \times \text{te\_timeout}<9:0>$



**Table 12-1495. TE Timeout Programming (continued)**

te_timeout(11)	te_timeout(10)	timeout value
1	1	$2048 \times te\_timeout<9:0>$

#### 12.6.5.7.5.5 Tearing Effect Control on Panels with Frame Buffer

Display panels will use command mode sequencing when an on-board frame buffer is used to store the video image rather than using the video streaming and synchronisation packets to control the update of the image information. This means that the host display driver must know when the panel is processing the image and avoid changing pixel information before it is updated. The mechanism is known as the tearing effect. The DSITX controller can support two schemes as outlined in the MIPI DSI specification.

#### TE control using the Polling Method

For polling to the display module, the host processor shall detect the current scan line information with a DCS command such as **get\_scan\_line** to avoid Tearing Effects. DSITX controller will issue a READ using this command until it gets to the expected value for the last line.

#### DCS Command - get\_scanline (45h)

##### Command

Direction H->D

Hex Code 0 1 0 0 0 1 0 1 45h

##### Parameter 1

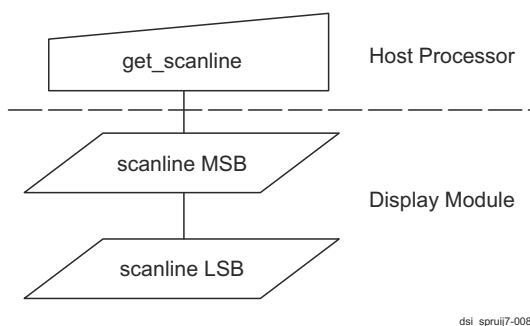
Direction D->H

Hex Code N15 N14 N13 N12 N11 N10 N9 N8 XXh

##### Parameter 2

Direction D->H

Hex Code N7 N6 N5 N4 N3 N2 N1 N0 XXh



**Figure 12-1077. get\_scan\_line Command**

#### Description

The display module returns the current scanline, N, used to update the display device. The total number of scanlines on a display device is defined as VSYNC + VBP + VACT + VFP. The first scanline is defined as the first line of V Sync and is denoted as Line 0.

In Sleep Mode, the value returned by **get\_scanline** is undefined. See [MIPI-DPI] for definitions of VSYNC, VBP, VACT, and VFP.

- In 2D mode, the scanline value of the display memory and the display panel is the same.
- In 3D Mode, the scanline value of the display memory and the display panel can be different; **get\_scanline** shall return the current scanline of the display panel.

## Restrictions

None

## TE control using the Automatic Method

For TE-reporting from the display module, the TE-reporting function is enabled and disabled by three DCS commands to the display module controller: `set_tear_on`, `set_tear_scanline`, and `set_tear_off`. See [MIPI-DCS] for details.

`set_tear_on` and `set_tear_scanline` are sent to the display module as DSI Data Type 0x15 (DCS Short Write, one parameter) and DSI Data Type 0x39 (DCS Long Write/write\_LUT), respectively. The host processor ends the transmission with Bus Turn-Around asserted, giving bus possession to the display module.

Since the display module DSI Protocol layer does not interpret DCS commands, but only passes them through to the display controller, it responds with a normal Acknowledge and returns bus possession to the host processor. In this state, the display module cannot report TE events to the host processor since it does not have bus possession.

To enable TE-reporting, the host processor shall give bus possession to the display module without an accompanying DSI command transmission after TE reporting has been enabled. This is accomplished by the host processor protocol logic asserting (internal) Bus Turn-Around signal to its D-PHY functional block. The PHY layer will then initiate a Bus Turn-Around sequence in LP mode, which gives bus possession to the display module.

Since the timing of a TE event is unknown to the host processor, the host processor shall give bus possession to the display module and then wait for up to one video frame period for the TE response. During this time, the host processor cannot send new commands, or requests to the display module, because it does not have bus possession.

## DCS Command - `set_tear_on` (35h)

### Command

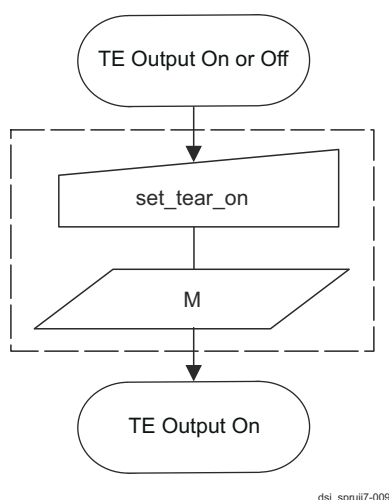
Direction H->D

Hex Code 35h

### Parameter

Direction H->D

Hex Code X X X X X X M XXh



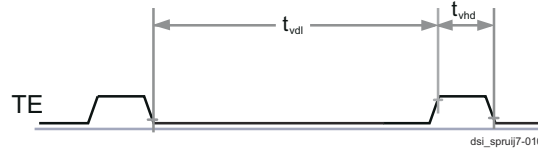
**Figure 12-1078. `set_tear_on` Command - 1**

## Description

This command turns on the display module Tearing Effect output signal on the TE signal line. The TE signal is not affected by changing set\_address\_mode bit B4. set\_tear\_on has one parameter that describes the Tearing Effect Output Line mode.

If M = 0 (Mode 0):

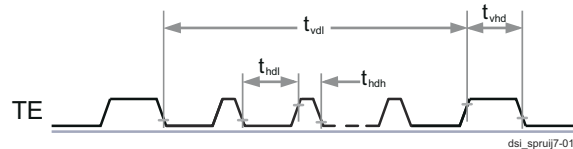
The Tearing Effect Output line consists of V-Blanking information only.



**Figure 12-1079. set\_tear\_on Command - 2**

If M = 1 (Mode 1):

The Tearing Effect Output Line consists of both V-Blanking and H-Blanking information.



**Figure 12-1080. set\_tear\_on Command - 3**

The Tearing Effect Output line shall be active low when the display module is in Sleep mode. See [MIPI- DPI] for definitions of tvdl, tvdh, thdl and thdh.

## Restrictions

This command takes effect on the frame following the current frame. Therefore, if the Tearing Effect (TE) output is already ON, the TE output shall continue to operate as programmed by the previous set\_tear\_on, or set\_tear\_scanline, command until the end of the frame.

### DCS Command - set\_tear\_scanline (44h)

#### Command

Direction H->D

Hex Code 0 1 0 0 0 1 0 0 44h

#### Parameter 1

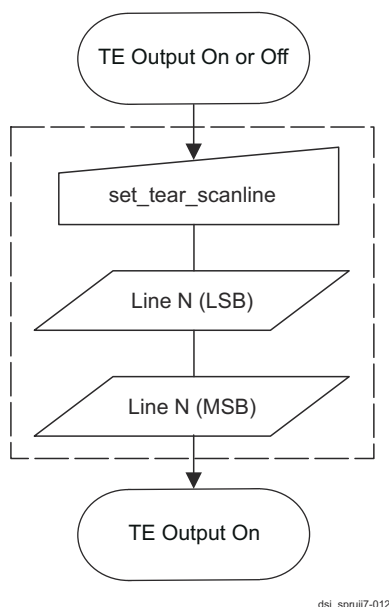
Direction H->D

Hex Code N15 N14 N13 N12 N11 N10 N9 N8 XXh

#### Parameter 2

Direction H->D

Hex Code N7 N6 N5 N4 N3 N2 N1 N0 XXh



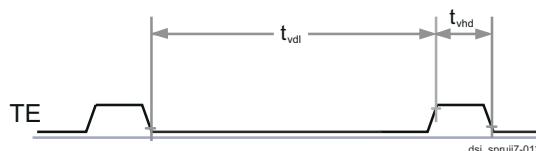
**Figure 12-1081. set\_tear\_scanline Command - 1**

### Description

This command turns on the display module Tearing Effect output signal on the TE signal line when the display module reaches line N. The TE signal is not affected by changing set\_address\_mode bit B4.

The Tearing Effect Line On has one parameter that describes the Tearing Effect Output Line mode.

After issuing a set\_tear\_scanline command to the display module, the Tearing Effect output signal, e.g. as in DBI- 2 systems, shall be a delayed version of V-Blanking information as illustrated in [Figure 12-1082](#).



**Figure 12-1082. set\_tear\_scanline Command - 2**

Note that set\_tear\_scanline with N = 0 is equivalent to set\_tear\_on with M = 0. The Tearing Effect Output line shall be active low when the display module is in Sleep mode. See [MIPI-DBI] for definitions of tvdl and tvdh and [MIPI-DSI] for definition of display module line numbers. In 2D mode, the scanline value of the display memory and the display panel is the same.

### Restrictions

This command takes effect on the frame following the current frame. Therefore, if the Tear Effect (TE) output is already ON, the TE output shall continue to operate as programmed by the previous set\_tear\_on, or set\_tear\_scanline, command until the end of the frame.

### DCS Command - set\_tear\_off (34h)

#### Command

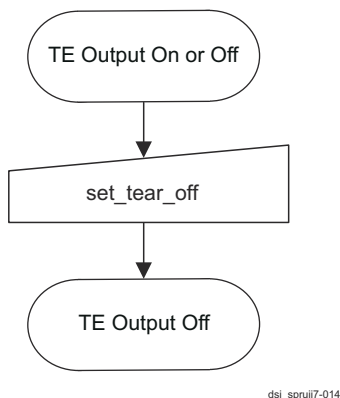
Direction H->D

Hex Code 0 0 1 1 0 1 0 0 34h

Parameters None

## Description

This command turns off the display module Tearing Effect output signal on the TE signal line.



**Figure 12-1083. set\_tear\_off Command**

## Restrictions

This command has no effect when the Tearing Effect output is already off.

### 12.6.5.7.5.6 Return Path Operation

The DSI Return Path behaves like a slave, by collecting data from the DPHY receive interface when the DPHY is configured to accept a transmission from the peripheral. The RP receives 6 signals from the DPHY (see [Table 12-1496](#)).

The data (and trigger) reception is inactive if the DPHY PPI direction signal is at low level. When it is active, data is received using the RX ESC\_CLK clock (the clock is transmitted with the data itself - see D-PHY protocol description from MIPI) and may stop if there is a pause in the data transmission but data is generated on clock falling edge. New data is available on rx\_data\_esc on a rising edge of the clock, if both rx\_lpd\_t\_esc and rx\_valid\_esc are asserted.

**Table 12-1496. Interface Between Return Path and DPHY**

Signal Name	I/O	From/To	Description
rx_lpd_t_esc	I	DPHY	Escape Low Power data receive mode This active high signal is asserted to indicate that the lane module is in low-power data receive mode. While in this mode, received data bytes are driven onto the rx_data_esc output when rx_valid_esc is active. The lane module remains in this mode with rx_lpd_t_esc asserted until a stop state is detected on the lane interconnect.
rx_trigger_esc(3:0)	I	DPHY	Escape trigger received These signals indicate that an escape trigger command has been received. Only one of these signals will be asserted at any time and the signal will remain asserted until the lane module returns to stop state.
rx_data_esc(7:0)	I	DPHY	Escape receive data For Low Power data receptions, this eight-bit value is driven by the D-PHY and is valid on rising edges of rx_clk_esc with rx_valid_esc asserted. The bit connected to rx_data_esc[0] was received first.
rx_valid_esc	I	DPHY	Escape receive data valid This signal indicates that the lane module is driving data on rx_data_esc and expects the protocol layers to take the data at the current rising edge of rx_clk_esc. There is no "ready" signal to throttle the receive data.

**Table 12-1496. Interface Between Return Path and DPHY (continued)**

Signal Name	I/O	From/To	Description
stop_state_dl1	I	DPHY	Lane is in stop state This signal indicates that the lane module is in STOP state. Note that this signal is asynchronous to any clock in the protocol interface.
direction	I	DPHY	Lane direction. When this signal is high, the lane module is in receive mode. When direction is low, the lane module is in transmit mode. (Mnemonic: 1 = Input, 0 = Output.)

The received data messages are identified as one of two possible types:

- Trigger
- Read packet

**Trigger** messages are passed almost directly to the register (reg\_req = reg\_trigger = 1 and reg\_rd\_data<3:0> equals to trigger value during one clock cycle). However, triggers are decoded to see if they are a TE trigger or an acknowledge with no error.

- The TE data is passed separately to the CSM using csm\_te\_received pin. It needs to be resynchronized with tx\_byte\_clk byte clock because of the CSM working clock domain.
- The acknowledge is only passed to register using reg\_req and reg\_ack pins. All other trigger values are passed undecoded to the registers without any processing. It is the responsibility of the application to decode and decide what to do with them.

**Read Packet:** When data is received, the system waits for the 4 bytes of the **HEADER**, and performs an ECC correction (if it is enabled) and then decodes the packets.

The following cases are considered when the system performs the ECC check:

- If the received command is not in the list of legal display opcodes, (i.e. errors in the packet header), it discards that packet and all further incoming bytes, even if they are legal packets (until the next BTA - i.e. direction change). It sends errors to the register bank (err\_undecodable and uncorrectable\_err). This may also result in an EOT error being reported as no further bytes have been decoded. But even if ECC correction shows an uncorrected error, decoding is attempted (if the errors are in the ECC byte or in the payload bytes, the opcode can be understood and data correctly recovered). In this case only err\_uncorrectable error bit is sent to register and it falls in the situation of regular commands (acknowledge with error or read).
- If "an acknowledge with error", the two status bytes are sent to the registers (reg\_req = 1 and reg\_ack\_err = 1 and reg\_rd\_data<15:0> contains the correct value).
- If the message type is a short read (either DCS or generic), data is passed to the registers (2 bytes without any specific decoding). Interface signals must be set accordingly and the two bytes are sent to registers (reg\_req = reg\_start = reg\_end = reg\_read = 1 with data and dscnotgen set accordingly). Size information is decoded in the first byte of the received packet depending on the data type information and then sent to register using reg\_size port.
- If a long read, the size is sent to the register using reg\_size. The system goes to **LONG state**. All data received are sent to register using reg\_data and are used for checksum detection. If during receive, return packet fifo (in control block) becomes full (rd\_data\_fifo\_full), the error flag err\_oversize must be set. All remaining data is used for checksum calculation but are not transferred to registers. When the number of received bytes is equal to the packet size, the system will check the **CHECKSUM**. The two checksum bytes are never transferred to register. They are used to make checksum detection and indicate err\_checksum status.
- If the transfer is shorter than the size specified in the header, the error err\_wrong\_length is issued along with any subsequent errors such as err\_missing\_eot (if needed). The case where it is longer drives to a tentative checksum decoding that generates an error and by a tentative to decode a new header (that is in fact a part of the previous packet) that drives to an error err\_undecodable and the throwing of the rest of the received bytes (with corresponding errors such as err\_missing\_eot).

- If it is an EoT packet, it is not passed to the control block. The following data is ignored (if any). If the system is waiting for **EoT**, this is the only action. If system is not waiting for EoT packet, the err\_eot\_with\_err is sent.
- If after an "acknowledge with error", the system waits for EoT packet and counts 4 bytes but is not able to decode them, it emits an err\_eot\_with\_err and err\_undecodable.
- If direction is removed at a "wrong" time (i.e before the fourth byte of a small packet, the system stops to works but emit errors to the register: err\_receive with eventually err\_missing\_eot.

The DSI is designed to support multi-peripheral integration, the virtual channel of the received packet is also stored in register using reg\_vc port.

The application FW layer is responsible of all read actions, it is assumed that once a read is requested, application reads return packets before requesting other read actions. Based on this assumption, the receive FIFO implemented in the register block is cleared each time a read action is started. All data not read before the new read command are lost.

A re-initialization of the return path is also performed using register access on reg\_init\_rp.

All errors related to short or long received data packets should be captured in the register when the direction changes (to capture all errors at the same time and not generate several errors and possibly several interrupts on the same reverse transaction). As soon as an error is detected on a packet, the error must be recorded and maintained up to the end of the packet transfer.

The maximum packet size supported by the DSI return path is 16 bytes. If a longer packet is received, only the first 16 bytes are available (by register way) for application.

#### **12.6.5.7.5.7 EoT Packet Management**

The DSI protocol states that, in the sending path, only the HS transmission requires use of EoT packet. For transfer from display to host, it is not recommended to use EoT but it is possible. The host needs to be able to detect the EoT packet and then behaves accordingly.

In case the display makes use of the EoT packet, the bit disp\_eot\_gen must be set to one. In that case, if after the ECC correction, the packet header is 0x08, we've detected an EoT packet and all bytes that may arrive between this packet and the direction change must be ignored.

If additional data is received between the EOT packet and the next BTA (end of read process), or if an unwanted EOT packet is detected, the unwanted bytes must be rejected and err\_receive is asserted.

If the EoT packet is not detected although it should be present, the error err\_missing\_eot is set. If after an "acknowledge with error", 4 bytes are received before BTA but are not identified as EoT (because of errors), it is assumed these 4 bytes are the EoT. In that case, the error err\_eot\_with\_err has to be set with err\_undecodable.

#### **12.6.5.7.5.8 ECC Correction**

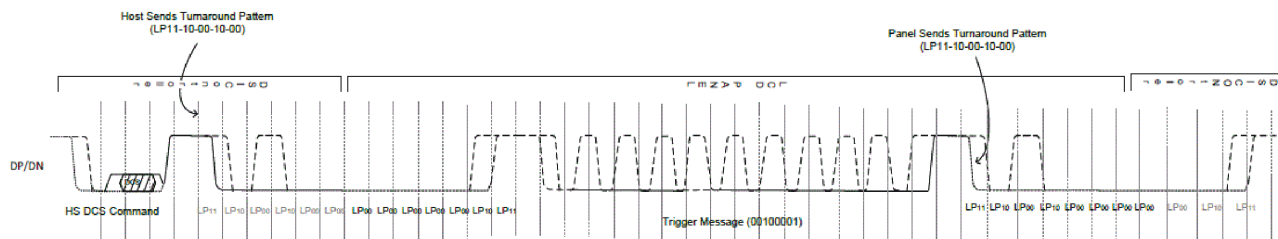
The DSITX controller uses the packet header ECC information to detect and correct one error, and can detect two or more errors without correction.

#### **12.6.5.7.5.9 LP Transmission and BTA**

The DSI host may send a command sequence and then request an acknowledge message from the panel. The host will send the command and release the bus by performing a BTA sequence. The panel will then respond with an acknowledge message (or error response) followed by a BTA to return the bus to the host. This sequence is illustrated in Overview of a Display Subsystem.

Overview of a Display Subsystem.

[Figure 12-1084](#) shows LP transmission timing diagram.

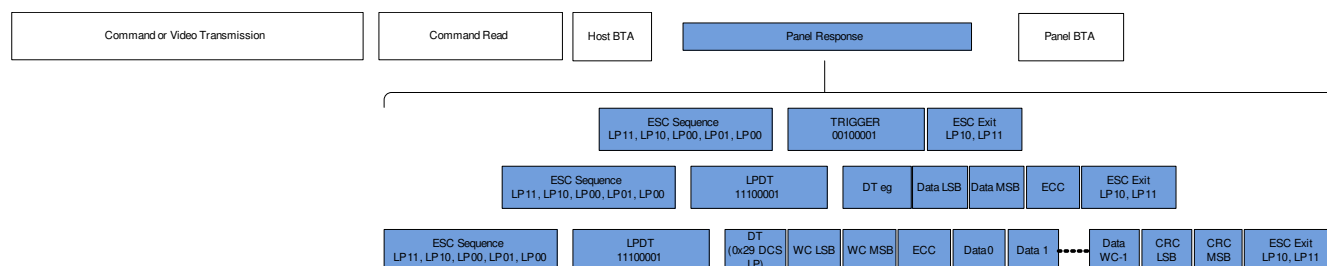


dsi\_spruij7-015

**Figure 12-1084. LP Transmission Timing Diagram for Host Read and Trigger ACK**

The Host will not be able to transmit any other packets once the BTA is sent to the panel and will expect either an ACK trigger or response LPDT message with a payload. The system must ensure there is sufficient time to send any command and issue the BTA, receive the response (Trigger, Short or long packet) and accept the BTA from the panel, if this request is made in the line blanking intervals.

Figure 12-1085 presents panel read response sequences.



**Figure 12-1085. Panel Read Response Sequences**

The panel response time will be based on the tx\_esc\_clk used inside the panel and the expected number of bytes for the payload. The expected responses are illustrated in the blocks in the MIPI DSI standard.

#### 12.6.5.7.6 Low-power Management

There are several low-power scenarios which consist of switching off various parts of the design:

1. Switch D-PHY cells in ULP mode when there are no data to be sent (bits clk\_lane\_ulpm\_en, dat1\_ulpm\_en, dat2\_ulpm\_en, dat3\_ulpm\_en and dat4\_ulpm\_en in register mctl\_main\_en set to one with ULP mode enabled in register mctl\_main\_phy\_ctl). Please, note that the time to leave ULP state is 1 ms and needs to be set in clock cycles in the register mctl\_ulpout\_time.
2. Switch the DPHY byte/bit clock PLL off (refer to DPHY documentation for control sequence). In this case, the D-PHY cell can only transmit data in LPDT thus maximum bandwidth is 10 Mb/s.
3. More drastic methods are to shut-down some parts of the design. The D-PHY and the DSI themselves can be shut down.

When D-PHY and DSI are both shut-down and the display is powered down, the restart is a normal start procedure as described in [Section 12.6.5.7.3, Start-up procedure](#).

When only the D-PHY is stopped, it is placed in either the stop or ULP state. Restarting the D-PHY is similar to the regular start-up procedure, however, after the lanes are enabled by the DSI Controller (clk\_lane\_en, dat1\_en, dat2\_en, dat3\_en and dat4\_en in register mctl\_main\_en are set), the D-PHY generates a rising edge on the stopstate signal to indicate to the DSI Controller that the enable request has been effective within the D-PHY. After that the start-up procedure may continue up to the point that the DSI link is ready to begin transmission.

Note that when setting the D-PHY into the ULP state, the application needs to wait enough time between the time it sets the ULP request in register and the time the D-PHY is disabled – this ensures that the ULP request is detected on the D-PHY interface.

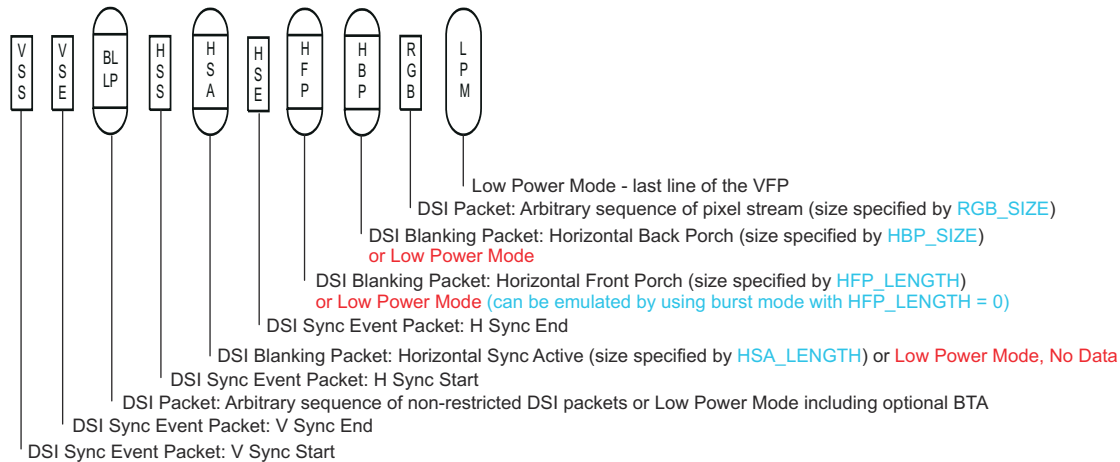


### 12.6.5.7.7 Video Mode Settings

All values contained in the Control Block (CB) registers must be set with meaningful values and are expressed with a range of different units: clock cycles, bytes, number of lines.

#### 12.6.5.7.7.1 Video Stream Presentation

In [Figure 12-1086](#), it shows the list of packets that can be part of a video stream and the associated register fields. In the four following figures ([Figure 12-1087](#), [Figure 12-1088](#), [Figure 12-1089](#), [Figure 12-1090](#)), different video streams that can be generated are presented.



ds1-017

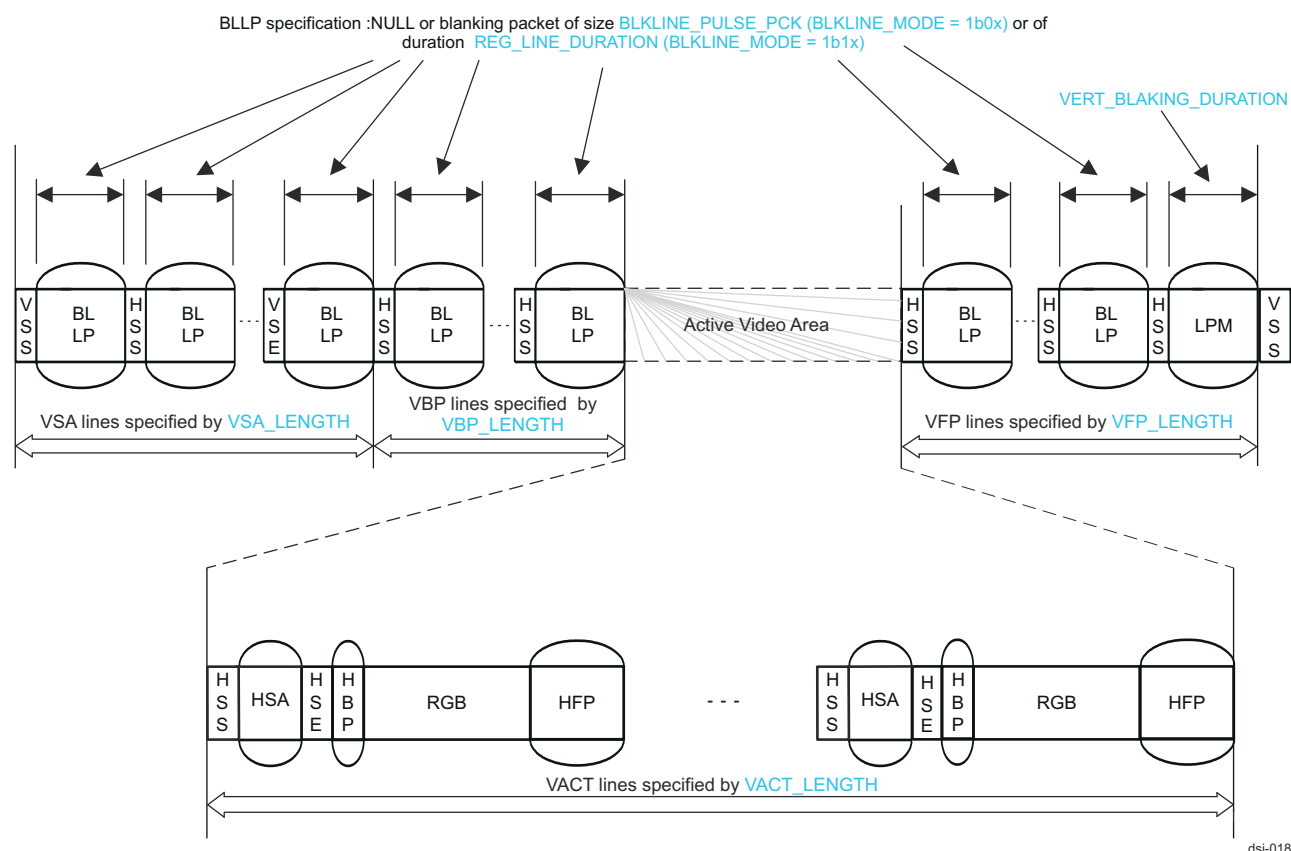
**Figure 12-1086. DSI Packet Terminology**

**Red text** - behaviour not possible.

**Blue text** - register fields used to specify the packet size.

The BLLP (Blanking – Low Power) periods allow the DPHY link to perform the following sequences:

- Host Transmit blanking packets in HS.
- Host Transmit command packets in HS or LP (Escape mode).
- Host Transmit packets using interleaving to a different virtual Channel in HS or LP (Escape mode).
- Host performs BTA and waits for response packets from the panel using Escape Mode followed by a panel BTA.

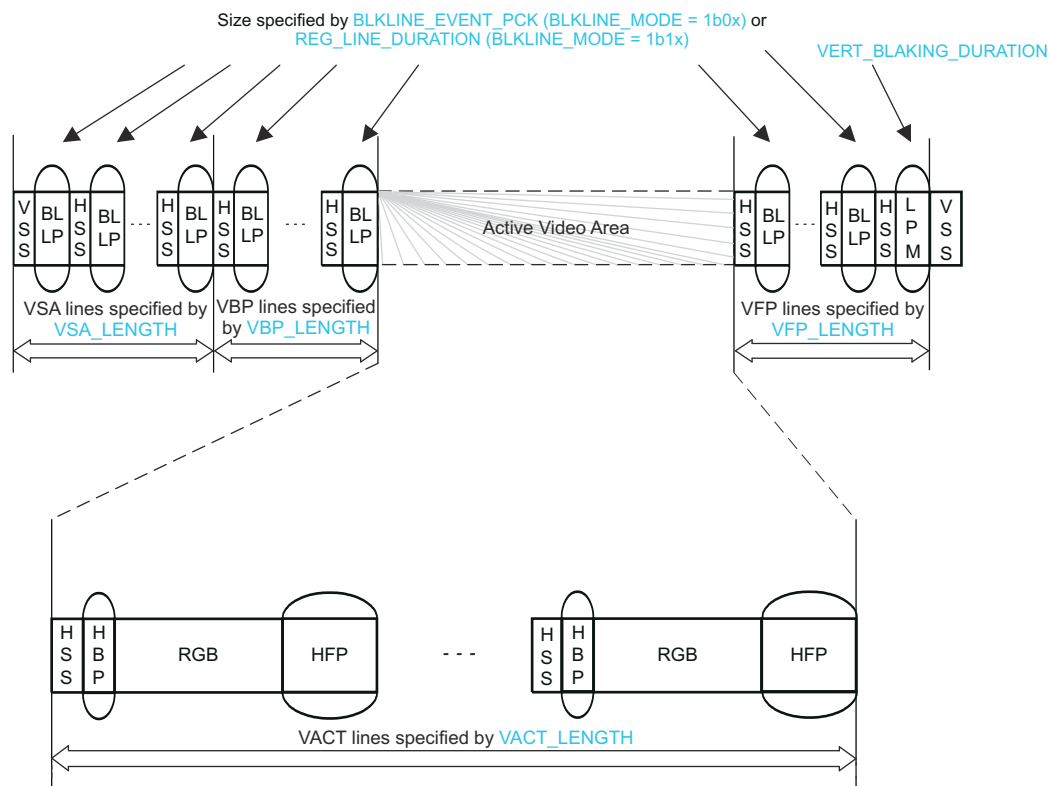


**Figure 12-1087. Video Pulse Mode (Non-Burst) Timing Diagram**

**Note**

This sequence is no longer specified in the MIPI spec Version 1.02.00 28-Jun-2010 but supported by the DSI-controller.

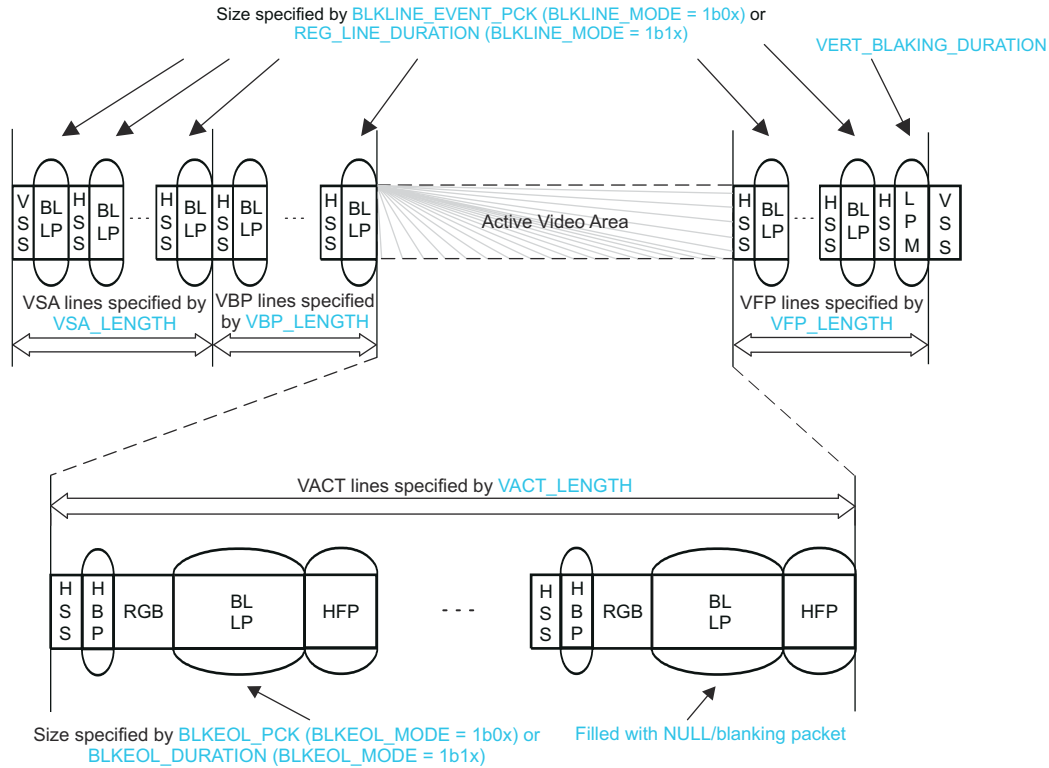




dsi-020

**Figure 12-1089. Video Event Mode (Non-Burst)**

burst\_mode = sync\_pulse\_active = sync\_pulse\_horizontal = 0 - all spec versions.



dsi-021

**Figure 12-1090. Video Sync Event in Burst Mode**

burst\_mode = 1 and sync\_pulse\_active = sync\_pulse\_horizontal = 0 - all spec versions.

Burst Mode operation requires the tx\_byte\_clk to be set to double the bit rate compared to the non-burst case. The calculations for the video parameters must then be increased to match the higher rate. The DPI FIFO must also be large enough to store at least half of the active line bytes before the output transmission begins otherwise it will underrun.

#### 12.6.5.7.7.2 Video Stream Settings (VSG)

The following is all the registers that need to be set before using the video in SDI or DPI interface operation.

- vid\_vsize1, 2
- vid\_hsize1, 2
- vid\_blksize1, 2
- vid\_pck\_time
- vid\_dphy\_time
- vid\_error\_color1, 2
- vid\_vca\_setting1, 2
- vid\_main\_ctl – programmed last in sequence of vid registers to apply to DSI registers

This section addresses the dependencies between all video registers, and details how the TVG and VCA registers must be set per VSG registers.

#### vid\_main\_ctl

- header information must correspond with the vid\_pixel\_mode used to control the generation of the recovery streams. The header information is used to be enable management of future cases where the DSI link could be used to pass streams with formats that differ feom those currently supported directly.
- sync\_pulse\_horizontal can only be asserted if sync\_pulse\_active = 1.

When `sync_pulse_active` is set to one, the HSYNC pulses are emulated only during the active part of the frame. When both `sync_pulse_horizontal` and `sync_pulse_active` are set, the HSYNC pulses are present on all lines of the frame.

**vid\_vsize** - the variables are expressed in line numbers in this register.

- `vsa_length` should be at least one (in order VSG can insert at least the vertical synchronization start VSS line) and greater or equal to 2 when pulse mode (`sync_pulse_active` = 1).
- `vbp_length` 0 to 63.
- `vfp_length` must be greater than 0.
- `vact_length` must be greater than 0.
- `vid_hsize1`: - the numbers are expressed in number of bytes (the fields are specifying the payload).

**vid\_hsize1** - the numbers are expressed in number of bytes (the fields are specifying the payload of the corresponding packet).

- `hsa_length` need to be at least set to one if pulse mode is enabled. The expected normal operating range is 1 to 255, large values for the HAS length will impact on the size of the DPI FIFO required.
- `hbp_length` can have any value. However, when set to 0, a HBP packet is generated with 0 payload.

**vid\_hsize2** - again the fields are specifying the payload of the corresponding packet (in byte).

- `hfp_length` can have any value. However, when set to 0, no HFP blanking packet is generated.
- `rgb_size` must match the number of pixel per line of the display multiplied by the number of byte per pixel.

**vid\_blksize1** - still specifying size of the payload in byte.

- `blkline_event_pck` defines the payload length of the blanking or NULL packet to be inserted in blanking line (with sync event). This value is used when `reg_blkline_mode` = 0b01 or 0b00 and when `sync_pulse_active` = 0. It is also needed by the VSG when `reg_blkline_mode` = 0b1x. Its value depends on the `hbp`, `rgb`, `blkeol`, and `hfp` packet length.
- `blkline_event_pck` defines the payload length of the blanking or NULL packet to be inserted in blanking line (with sync event). This value is used when `reg_blkline_mode` = 0b01 or 0b00 and when `sync_pulse_active` = 0. It is also needed by the VSG when `reg_blkline_mode` = 0b1x. Its value depends on the `hbp`, `rgb`, `blkeol`, and `hfp` packet length.

-- 32767 = 0x7FFF = max value on 15 bits and -100 to take

margin `blkeol_pck` < 32767 - `hfp_length` - `hbp_length` - `hsa_length` - `rgb_size` - 100;

The `blkeol_pck` (and `blkeol_duration`) must be adapted to the `reg_wakeup_time` in case VCA is authorized to go back in LP.

**vid\_blksize2** - still specifying size of the payload in byte.

- `blkline_pulse_pck` has almost the same definition than `blk_event_pck` but it concerns blanking lines with sync pulse.

**vid\_pck\_time** - specifies duration in clock cycles.

- `blkeol_duration`: `blkeol_duration` = `div_round_up` <sup>(1)</sup> ((`blkeol_pck` + 6), `lane_nb`);
- `vert_blanking_duration` is no longer used and can always be set to 0. It remains present in the design for legacy reason.

**Note 1:** The function `div_round_up` is a function that performs a division then round the result to the first entire number superior or equal to the division result.

**vid\_dphy\_time**: specifies duration in clock cycles. These are values that have external dependencies and thus need to be calculated first (and rest of VSG programming is calculated from these values).

**reg\_line\_duration**: depends on display type and size and of its programming (in some display, the blanking sizes can be adapted to provide a given number of frame per second with one of the D-PHY possible clock frequency (as PLL can only generate a given number of discrete frequencies).

**if (pulse mode) {**

```

line_length = (blkline_pulse_pck + 6);
}
else (event mode) {
line_length = (blkline_event_pck + 6);
if (burst_mode and burst_lp and lane_nb == 2 and
hsa_length[0:0] == 1 and hfp_length[0:0] == 1)
line_length = line_length - 1;
}
result = div_round_up(line_length, lane_nb);

```

**Note 1:** The function `div_round_up` is a function that performs a division then round the result to the first entire number superior or equal to the division result.

Additional Notes about the line duration:

- If line length (in bytes) is a multiple of the number of lanes enabled, DSI will never face any timing jitter except due to the resynchronization in SP/DCB.
- If line length (in bytes) is NOT a multiple of the number of lanes enabled For Non LP operation, there should not have any problem since the line is once longer, once shorter than the "normal" value. For example: if line length is 491 bytes, it will be 245 then 246 in clock cycles with 2 lanes for an average value of 245.5.

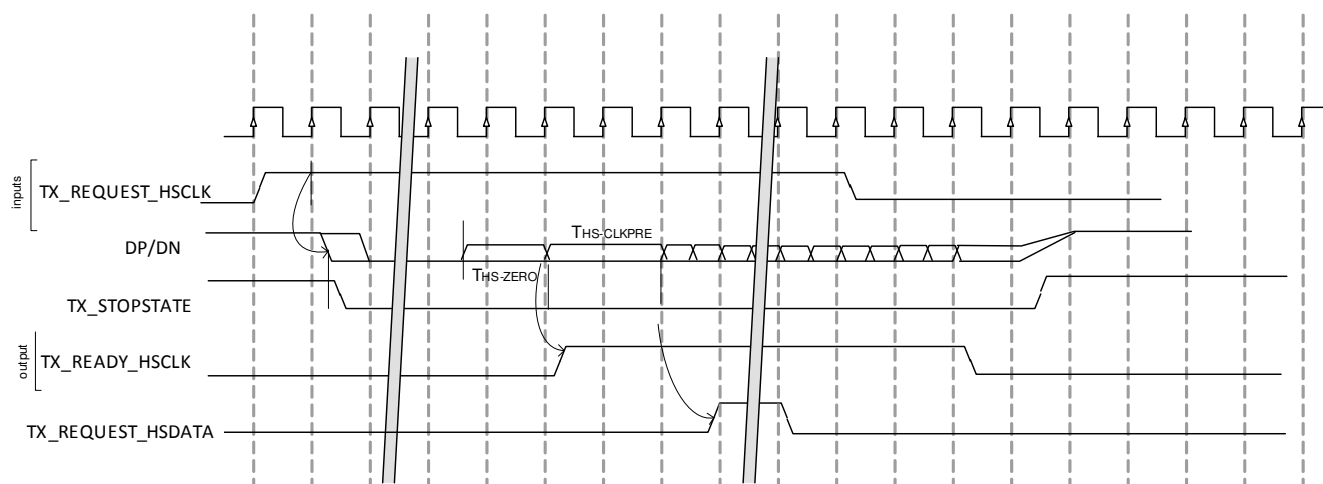
For LP enabled operation, the controller cannot allow underflow of the line and frame timing. With the previous example, the LP line length in clock cycles will be 246 since we cannot start the HS transmission in the middle of a clock cycle. And there is no mechanism to have one line in 245 and the next one in 246 cycles. For LP see section 0.

**reg\_wakeup\_time** must be shorter than line duration and depends on the D-PHY cell plus some pipelines delays inserted by the DSI link. This value strongly depends on the DPHY PLL programming and configuration, and is a mix of both internal and external analog and digital timing factors, therefore it is very difficult to provide an exact formula. The recommended approach to selecting a suitable value for this parameter is to characterize behaviour in at the system level environment using simulation, emulation (e.g. Palladium) or validation (e.g. FPGA).

The DPHY needs 100 ns + time for THXS-EXIT to go to Low Power mode.

The timing for the clock lane TXRequestHS going active to the TXReadyHS will be determined by the DPHY DDR bit rate clock frequency. The timing will be the total number of DDR bit clk cycles for the TXRequestHS to be detected and the clock lane to transition to High Speed, so  $TLPX + TCLK\_PREPARE + TCLK\_ZERO$ . Additional time is required between the clock lane driving HS clocks and the Data Lanes being activate,  $TCLK\_PRE$ . All of these parameters can be calculated from the DPHY datasheet.

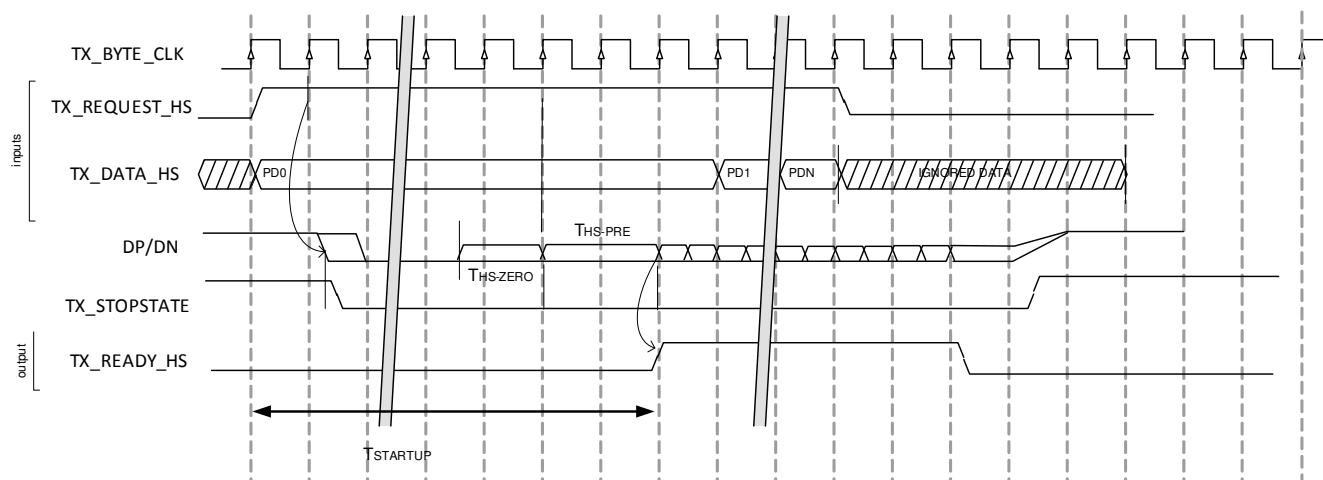
[Figure 12-1091](#) shows timing diagram for clock lane activation/deactivation.



**Figure 12-1091. Timing Diagram for Clock Lane Activation/Deactivation**

The data lane request follows a similar timing sequence from the TX\_Request\_HS to TX\_Ready\_HS, TLPX + THS-PREPARE + THS\_ZERO. The data lane requests will be activated once the clock lane is ready, i.e TX\_Ready\_HSCLK is high.

Figure 12-1092 shows timing diagram for data lane activation/deactivation.



**Figure 12-1092. Timing Diagram for Data Lane Activation/Deactivation**

**Table 12-1497. Timing Parameters**

Parameter	Description	Min	Max	Example 650 Mbps (UI = 1.538 ns)
TLPX	Transmitted length of any Low- Power state period.	50 ns		50 ns
TCLK-PREPARE	Time that the transmitter drives the Clock Lane LP-00 Line state immediately before the HS-0 Line state starting the HS transmission.	38 ns	95 ns	38-95 ns
TCLK-PREPARE + TCLK-ZERO	TCLK-PREPARE + time that the transmitter drives the HS-0 state prior to starting the Clock.	300 ns		300 ns
Wakeup Time CL				388-445 ns
TCLK-PRE	Time that the HS clock shall be driven by the transmitter prior to any associated Data Lane beginning the transition from LP to HS mode.	8 UI		12.3 ns

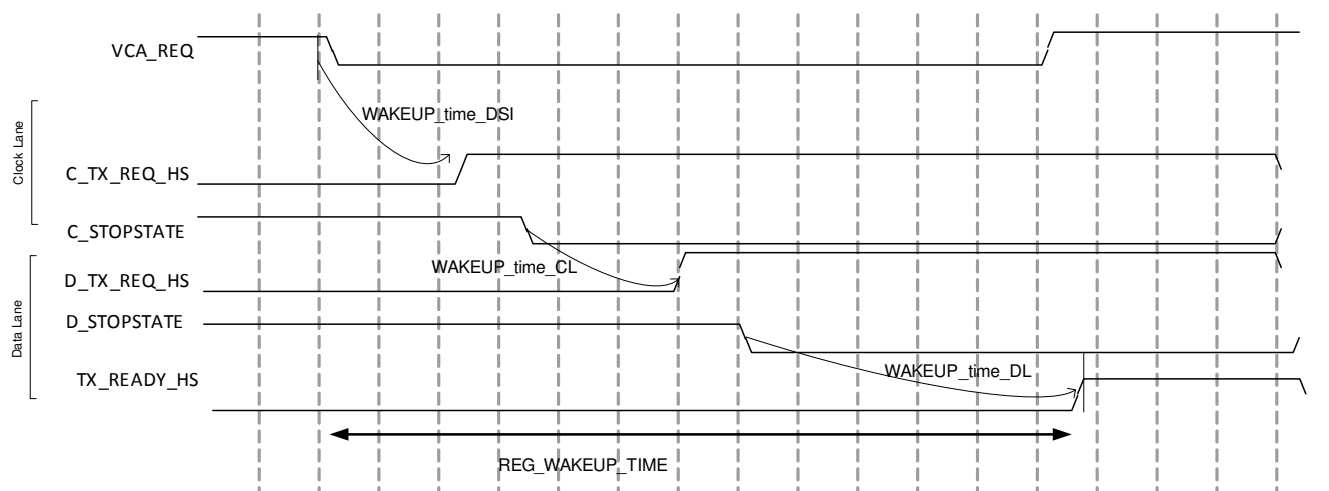


**Table 12-1497. Timing Parameters (continued)**

Parameter	Description	Min	Max	Example 650 Mbps (UI = 1.538 ns)
THS-PREPARE	Time that the transmitter drives the Data Lane LP-00 Line state immediately before the HS-0 Line state starting the HS transmission.	40 ns + 4 × UI	85 ns + 6 × UI	~ 46-95 ns
THS-PREPARE + THS-ZERO	THS-PREPARE + time that the transmitter drives the HS-0 state prior to transmitting the Sync sequence.	145 ns + 10 × UI		160 ns
Wakeup Time DL				~ 270 ns

So the wakeup time values expressed as tx\_byte\_clk (12.3ns) cycles, CL = 36 and CLKPRE + DL = 22.

Figure 12-1093 shows wakeup time timing diagram.


**Figure 12-1093. Wakeup Time Timing Diagram**

Here is the example of the calculation done for the system running at **650 MHz**.

```

if (clk_continuous == TRUE) {
wakeup_time_cl = 0x1;
} else {
wakeup_time_cl = 0x24;
};

wakeup_time_dsi = 0xA; -- from request on VSG to request HS on DL1 (+ 1 for VSG internal cycle)
wakeup_time_dl = 0x14; -- from stop state falling edge to tx_ready rising edge
reg_wakeup_time = wakeup_time_dsi + wakeup_time_cl + wakeup_time_dl + (hs_host_eot × 4 / lane_nb)

```

**Note:** It is not necessary to program all the register values every time because some of them are used only in some modes. For instance, blkline\_event\_pck and blkline\_pulse\_pck are used depending on the way SYNC is generated, and then in pulse mode, there is no need to program blkline\_event\_pck and vice-versa.

#### 12.6.5.7.7.3 VCA Configuration

The VCA setting done in two registers: vid\_vca\_setting\_1 and vid\_vca\_setting\_2. Their programming must respect the following rules:

- vid\_vca\_setting\_1:

- max\_burst\_limit specifies the size of the bigger packet that generates the following sequence RGB + packet + NULL packet + BLK packet; it must be linked with blkeol\_pck (same size minus the smaller NULL packet). It is equal to blkeol\_pck - 6.
  - burst\_lp can be set to 1'b1 if blkeol\_pck > 2 × reg\_wakeup\_time × lane\_number (reg\_wakeup\_time is only specifying the time to go from LP to HS thus need to consider the time to go from HS to LP too).
- vid\_vca\_setting\_2:
  - exact\_burst\_limit: as it specifies (in byte) the payload of the packet that generates a sequence RGB + packet + HFP, it must have the same value as of blkeol\_pck when the DSI link is in burst mode.
  - max\_line\_limit specifies the maximum size of a packet that generates HSS + (HSA + HSE) + packet + NULL packet. It then depends on line\_duration, hsa\_length regardless if the system is using pulse or event synchronizing. Its size is linked to vert\_blanking\_duration.

```

if (sync_pulse_active == 0) {
-- size of the payload of the inserted packet followed by a NULL pkt
max_line_limit = (blkline_event_pck-6);
};
if (sync_pulse_active == 1) {
-- size of the payload of the inserted packet followed by a NULL pkt
max_line_limit = gen_blkline_pulse_pck-6;

```

Another constraint to respect when using VCA is to ensure that only supported operations are sent from command side when video is active. For instance, no TE should be attempted, because it could break the video stream (TE completion time can be long). Command messages can be inserted into the LP states provided there is sufficient space for the message bytes and any associate bytes for the header and CRC, if a long packet type.

#### 12.6.5.7.7.4 TVG Configuration

The TVG has a single register: tvg\_img\_size that should complies with the following rules against VSG settings:

tvg\_line\_size: must be equal to rgb\_size set for VSG

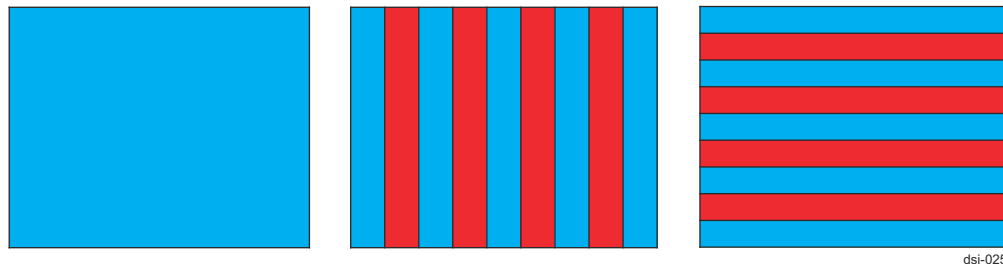
tvg\_nblne: must be equal to vact\_length set for VSG

The TVG generates a data stream in the same format than the one specified at the output of the VRS. The content of that flow is specified by the following registers:

- Image size - TVG\_LINE\_SIZE[14:0] and TVG\_NBLINE[12:0] registers: Number of bytes per line and number of lines per frame.
- Image kind - TVG\_MODE[1:0] register: Single color, vertical stripes or horizontal stripes.
- Stripe width - TVG\_STRIPE\_SIZE[2:0] register: Significant when a stripe mode is enabled. Possible values are 1, 2, 4, 8, 16, 32, 64 and 128.
- Pixel kind VID\_PIXEL\_MODE[3:0] register: 16 bits RGB, 18 bits RGB, 18 bits loosely RGB, 24 bits RGB, 30 bits RGB or 36 bits RGB.
- Color one - COL1\_GREEN[11:0], COL1\_RED[11:0], COL1\_BLUE[11:0] registers: Color used in single color mode or first color when in stripe mode. 12 bits per component R/G/B. For formats using fewer bits than 12, the least significant ones are the ones considered.
- Color two - COL2\_GREEN[11:0], COL2\_RED[11:0], COL2\_BLUE[11:0] registers: Color two when in stripe mode. 12 bits per component R/G/B. For formats using fewer bits than 12, the least significant are the one considered.
- Start pulse and stop handshake (+ stop mode) (see TVG\_CTL register).

The Test Video Generator can be programmed to generate a set of test colour patterns based on the display panel that will be used. The panel parameters for horizontal and vertical resolution along with the frame rate and pixel colour depth need to be set based on the datasheet information for the panel.

Figure 12-1094 presents TVG MODE patterns.



**Figure 12-1094. TVG MODE Patterns**

An example of the sequence is as follows:

- Select TVG for video stream generation instead of DPI/SDI 1 interface.
  - IF1\_EN bit to 0 (SDI stall signal at 1) in MCTL\_MAIN\_EN.
- Select video mode for interface 1 in MCTL\_MAIN\_DATA\_CTL.
  - TVG\_SEL bit to 1 in MCTL\_MAIN\_DATA\_CTL register to select stream from TVG.
- Select generated frame format in TVG\_CTL register.
  - Image format (single color, H stripes, V stripes) in TVG\_MODE field.
  - Image color selection in TVG\_COLOR1, TVG\_COLOR1\_BIS, TVG\_COLOR2, VG\_COLOR2\_BIS.
  - Stripes size in TVG\_STRIPE\_SIZE field.
  - Image size in TVG\_IMG\_SIZE register.

**Note:** The TVG settings on active area for the number of lines per frame and number of bytes per line must match VSG settings on active area. Any mismatch will create an error that is detected in the VSG and forces recovery mode in TVG, stopping test frame generation.

- Select TVG stop mode with TVG\_STOPMODE field of TVG\_CTL.
  - TVG video stream generation start/stop controlled by the TVG\_RUN bit in TVG\_CTL register.
  - Polling TVG run status from the TVG\_RUNNING bit in TVG\_STS register is useful when setting TVG\_RUN to 0. TVG video stream generation does not stop the instantaneously (depends on TVG stop mode), so this should be checked to confirm the TVG has stopped.

#### 12.6.5.7.8 DPI To DSI Programming

The DPI interface will normally be driven from a graphics driver that will be configured to match a frame geometry and refresh rate for a panel display. The DPI driver will use the expected refresh rate and geometry for the active and blanking parts of a frame to determine the pixel rate clock. The DSITX controller will then translate this to match the incoming byte information to send across the DSI DPHY link.

A system will always expect the bandwidth of the incoming pixels × BPP to match the transmission of bytes from the DSI DPHY based on the number of active lanes. The relationship of DPI clock to TX byte clock must ideally hold with the following formula:

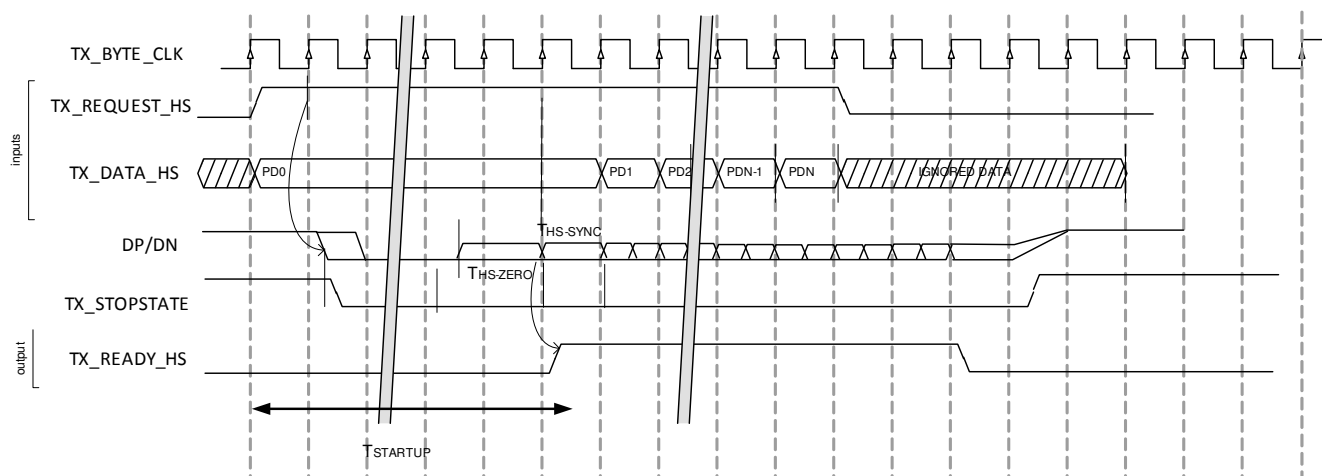
$$F_{\text{pixel\_clk}} = F_{\text{tx\_byte\_clk}} \times \text{active\_lanes} \times 8 \div \text{bits\_per\_pixel};$$

The DSI will supply packets to the DPHY interface using the internal sequencing and counters clocked from the tx\_byte clock. The normal system operation will mean that the DSI must always have the correct number of tx\_byte clock cycles to match the number of pixel clock cycles. This means that all the events used for the packet generation will be directly impacted by the delay that exists in the DPHY High Speed request to ready.

The DSITX controller relies upon the DPHY PLL being programmed and locked before starting the DPI video. The DSITX controller can then be configured and enabled (clock lane and data lanes) before starting the video transmission.

The DSITX controller will begin data transmission with a High Speed request after it receives the falling edge of the VSYNC on the DPI interface. The controller will then begin to build the bytes for the VSYNC packets (VSS, HSA, Blanking packets) during the delay period before the ready is returned from the DPHY. This delay will add an interval effect on all the packets that follow.

The DPI FIFO size must be sufficient to buffer all the incoming pixel data during the active line stage while the DSITX controller continues to send the previous lines packet information due to this interval effect.



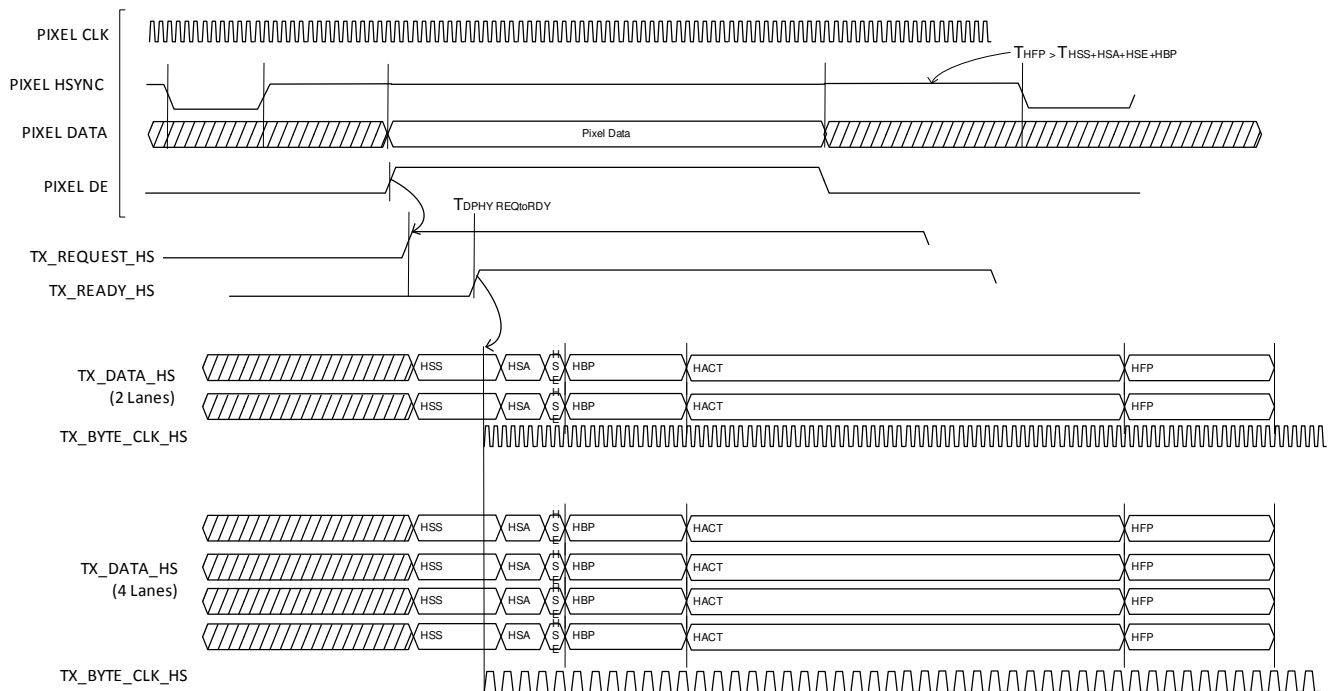
**Figure 12-1095. Timing Diagram**

The DPI and DSI system may not be able to guarantee the exact frequency on the pixel clock input or tx\_byte clock from the DPHY. The DSI configuration can be adjusted to support a small deviation in frequencies using the size of the HFP packet so that it changes the alignment of packets on the active lanes to absorb the different pixel clock cycles during the line. More details are given in section 0.

#### 12.6.5.7.8.1 DSI and DPHY Operation

The DPI FIFO is used to buffer all the active pixel data from the DPI interface during the active line. The depth of the FIFO must be selected to allow the pixel to be stored after the DPHY request to ready delay is introduced at the start of every frame. The FIFO will fill to a depth based on the tx\_byte clock cycles used from the point where VSYNC is identified to the response from the DPHY link after it has exited Low Power and entered the zeroes state on the active data lanes.

Figure 12-1096 presents DPI interface to DSI DPHY timing diagram.



**Figure 12-1096. DPI Interface to DSI DPHY Timing Diagram**

The timing diagram above illustrates how the packets are generated for a DPHY with two and with four lanes activated. The diagram shows that the tx\_byte clock changes based on the number of lanes, and that the configuration register values for the number of bytes used for HSA, HBP and HFP will remain the same.

The register values calculated for each of the line synchronization stages needs to take the number of bytes used to form the packet into account so that the timing for each stage aligns to the timing of the DPI input and most importantly that the total number of bytes exactly matches the number of byte clock cycles. The values used for the Horizontal Front porch must be larger than the combined horizontal sync and back porch to allow the DPI FIFO to empty.

The basic rule is that the values used for the DSI will be based on the  $\text{DPI} \times \text{BPP}/8$ .

The DPI graphics driver must be configured with horizontal values that can be directly translated to DSI packet payloads.

- The minimum values for the DPI HSA must convert to be greater than:
  - Non-Burst Sync Pulses – 15 bytes
  - Non-Burst Sync Events – 11 bytes
- The minimum values for the DPI HBP must convert to be greater than:
  - Non-Burst Sync Pulses – 7 bytes
  - Non-Burst Sync Events – 7 bytes

These values will then form the number of tx\_byte\_clk cycles used to send the bytes over the number of active lanes. So, for an RGB888, DSI configuration the minimum values will be:

- $\text{HSA} = \text{roundup}(15/3) = 5$
- $\text{HBP} = \text{roundup}(11/3) = 4$

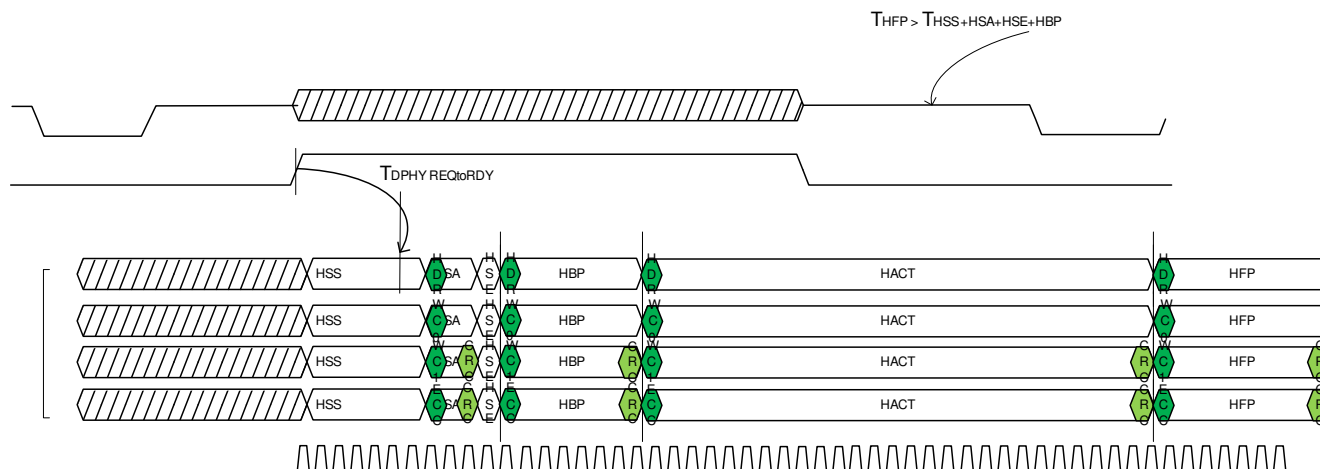
The diagram below, D-PHY Timings Control (see [Figure 12-1097](#)), illustrates the byte mapping for a four lane DPHY link. In this example, the calculation for the horizontal sync stage aligns exactly across the four byte lanes.

- HSS – Four Bytes
- HSA – Blank Packet with Header (4) + HSA\_Count + CRC(2)
- HSE – Four Bytes

The next stage is the horizontal back porch:

- HBP – Blank Packet with Header (4) + HBP\_Count + CRC(2)

The value used in the HBP count can be used to account for the Header bytes used in the active data packet in next stage, although it is best to add this to the HFP stage calculation and allow time to empty the FIFO.



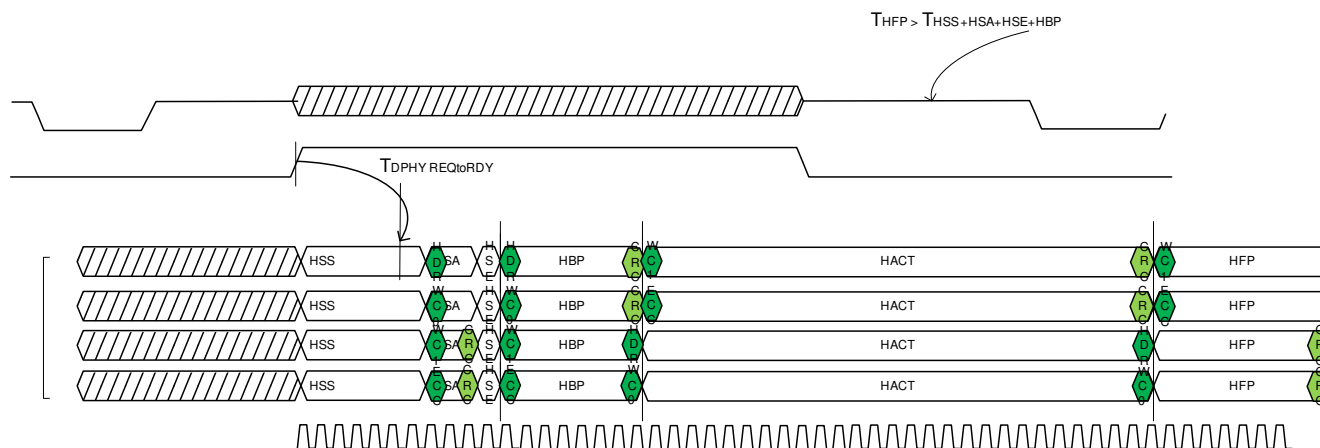
**Figure 12-1097. Horizontal Line Packet Timing Diagram**

The Active part of the line will be a fixed size based on the line size and BPP, and include the six bytes for the packet header and CRC. The final stage is the horizontal front porch.

- HFP – Blank Packet with Header (4) + HFP\_Count + CRC(2)

The HFP value should be adjusted to ensure that the number of tx\_byte clock cycles for the horizontal line exactly matches the pixel clock cycles for the line times the BPP.

The timing diagram below, Control Block, illustrates the byte alignment changing when the HBP calculated is two bytes shorter than the previous case. This leads to the active and front porch packet headers beginning two bytes earlier, however the HFP count is adjusted to keep the end of the line on exactly the same tx\_byte\_clk cycle count.



**Figure 12-1098. Horizontal Line Packet - Shorter HBP - Longer HFP Timing Diagram**

The recommendation is to ensure that the total bytes for each horizontal line results in an exact integer tx\_byte clock cycles (Total\_bytes MOD num\_of\_lanes = zero), except for the case where the clocks are not exactly matched.

#### 12.6.5.7.8.2 Pixel Clock to TX\_BYTE\_CLK Variation

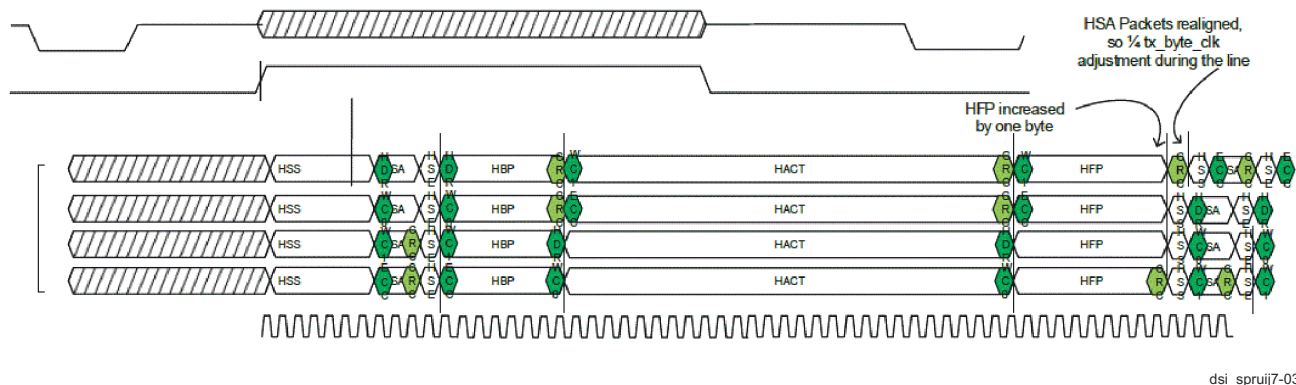
The selection of the Pixel clock rate and DPHY Bit clock rate should ideally match based on the ratio of bpp on the pixel side compared to the tx\_byte\_clk. The timing for the horizontal sync, back and front porch will also be configured to allow the pixel data to be transferred without any over or underflow of the DPI FIFO.

The DPI FIFO fill level register (DPI\_CFG) can be used to track that the parameters are sufficient and that any clock variation can still be handled by the FIFO depth. This register will show the pixel data held in the buffer, so during the HSA and HBP packet generation stages DPI pixel data will be stored and this value will increment. Once the data packet is being generated, the DPI\_CFG level will remain constant, as the buffer is emptied at the same bandwidth as it is filled. Finally, the value will reduce to zero during the HFP stage.

The packet length of the HFP can be adjusted to help absorb small differences in the pixel\_clk to tx\_byte\_clk relationship. The DSITX controller will concatenate all of the packets during High Speed transmission, so for example adding to the byte value of the HFP will make the bytes at the end of the packet move to align on another lane.

The packets that follow will then be concatenated and aligned to the next available lane, so if the tx\_byte\_clk is faster than the expected pixel clock times bpp the extra byte will delay the start of the new line tx\_byte\_clk × extra\_byte/lanes.

Figure 12-1099, Video coherency, illustrates the impact of increasing the HFP by one byte so that the next line begins ¼ tx\_byte clock later. This allows averaging out across the line length of small differences in the pixel to tx\_byte clocks when the tx\_byte clock is not ideally matched to the expected pixel clock multiplied by bpp.



dsi\_spruij7-030

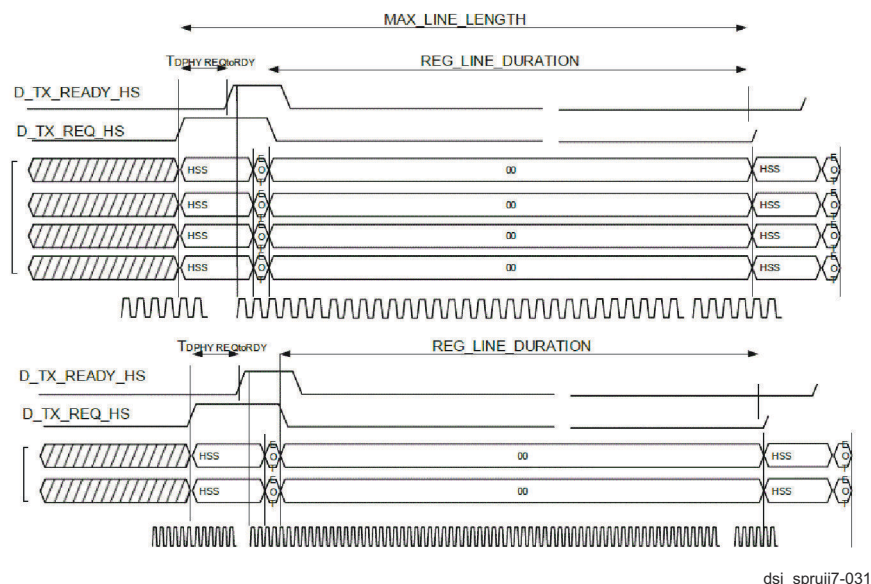
**Figure 12-1099. Packet Alignment Control Using HFP Byte Increase - tx\_byte\_clk Faster than Ideal Rate**

The DPI\_CFG register will show any clock misalignment as an increasing/decreasing value for the FIFO fill level on each line of pixels (must be read mid-line for an accurate reading of the level). When the byte count adjustment is made, the DPI\_CFG register will return to the original fill level value every four lines.

#### 12.6.5.7.8.3 LP Operation

The DSITX controller can be configured to perform a transition between LP state and HS state during the horizontal lines which have long periods of blanking. The registers controlling the timing for the DPHY wakeup time and the reg\_line\_duration must be programmed to exactly match the DPI cycles of the horizontal line to the tx\_byte\_clk cycles required for the number of active lanes selected.



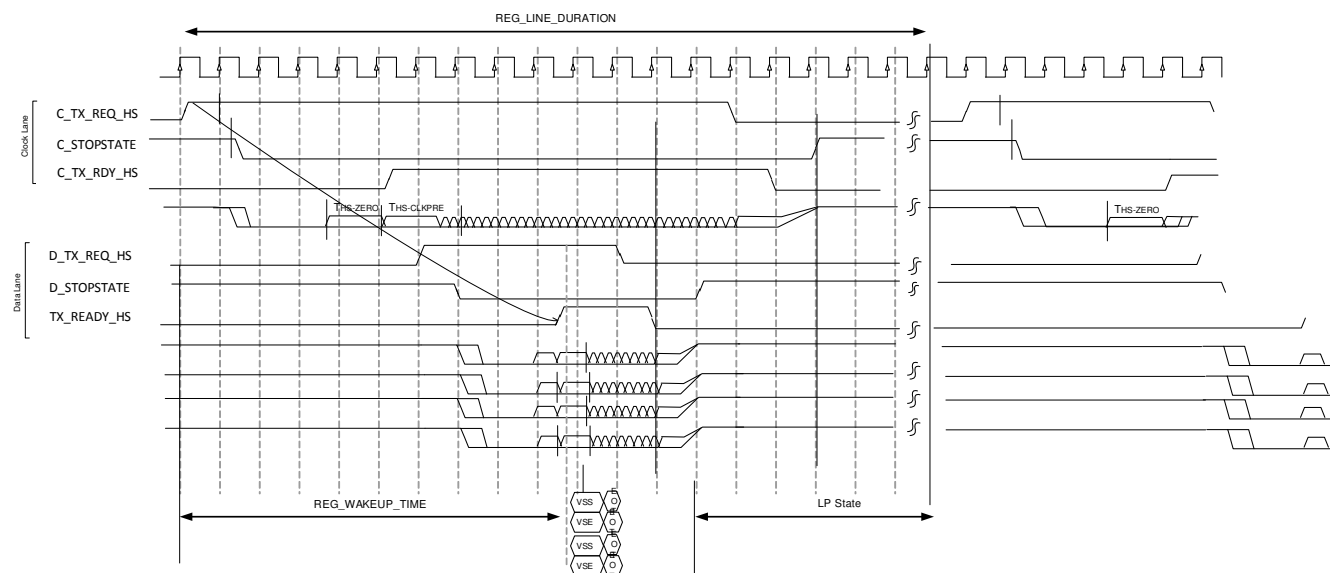


**Figure 12-1100. Low Power Operation with Four and Two Active Lanes**

The LP operation for each vertical blanking line will require the reg\_line\_duration to be configured to match the following:

- Max\_line\_length (in tx\_byte\_clk cycles see 2.2.4) minus Eot (if enabled), also minus further 10 if CLOCK lane is non-continuous.

Note REG\_WAKEUP\_TIME is used internally to adjust the cycles for each vertical LP line.



**Figure 12-1101. REG\_LINE\_DURATION Timing Example for LP Operation**

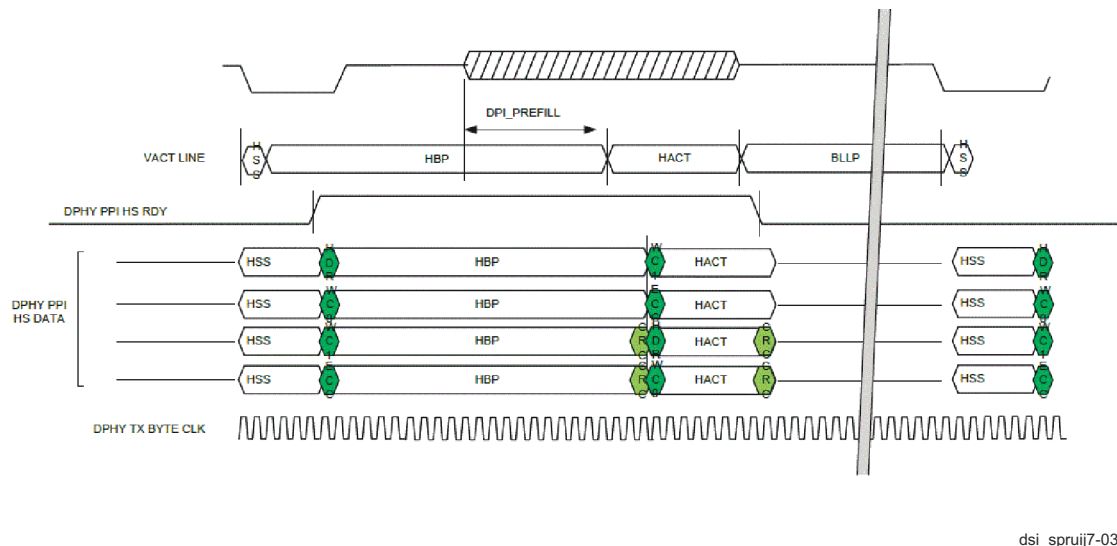
#### 12.6.5.7.8.4 DPI Interface Burst Operation

The DSITX controller can support burst operation on the DPI interface provided the DPI FIFO size is large enough to store the line pixel data with enough data to not underflow. The burst operation will use a tx\_byte\_clk that is faster than normal event based operation, so the DPI FIFO size will be impacted by the ratio of the pixel\_clk and BPP input to the tx\_byte\_clk and number of lanes on the output.



The recommended operation is to set the DSI HSA and DSI HFP registers to zero, and adjust the DSI HBP size to control the number of bytes prefilled in the DPI FIFO before the burst is sent.

Figure 12-1102 shows DPI operation with event burst mode.



**Figure 12-1102. DPI Operation with Event Burst Mode**

#### 12.6.5.7.9 Programming the DSITX Controller to Match the Incoming DPI Stream

The time taken to output a frame on the PPI needs to match what is coming in on the DPI. This is best achieved by using the recommended clock ratio between the pixel and byte clocks. Assuming that is the case, then the blanking and active data periods must be matched up.

For each frame of video the time will be:

$$(V_{\text{total}} * H_{\text{total}})_{\text{@pixel\_clk}} = (V_{\text{total}} * H_{\text{total}})_{\text{@tx\_byte\_clk}} \times \text{bits\_per\_pixel}/8;$$

So for example, the time for a full HD frame 1920x1080 with reduced blanking in RGB565 will be:

$$2200 \times 1200 \text{ pixels} = 2200 \times 1200 \text{ tx\_bytes} \times 16/8;$$

The DSITX controller will slave its frame timing to the incoming DPI vsync, as long as it is programmed to generate a frame of slightly less than the same size when the VFP blanking is considered. The controller works by transitioning to LP during the last programmed line of VFP. It will then remain in LP until the start of the next frame. So programming the controller to match the DPI, but with at least one fewer line of VFP should result in a reliable configuration.

##### 12.6.5.7.9.1 Vertical Timing

The DSITX controller will generate vertical packets for each part of the frame beginning with the VSS and VSE combined with blanking packets to fill the VSA. The blanking packets for the pulse mode program the DSI vertical size registers as follows:

- VSA = DPI\_VSI (lines, minimum of 2 → VSS and VSE)
- VBP = DPI\_VBP (lines, minimum of 0)
- VACT = DPI\_VACT (lines)
- VFP < DPI\_VFP (minimum of 1)

DPI horizontal timing is measured in pixel clocks, whereas the DSITX controller uses byte clocks. The horizontal timing should also be matched per line; therefore the blanking and active periods of each line should match. The relationship therefore depends upon the pixel format, which could be 24, 18 or 16 bpp. The timing for a horizontal line for the DPI driver side will be the number of pixel cycles for the **HLINE** = (HSA + HBP + HACT +

HFP) and this will form **HLINE** × bpp/8 bytes for the controller. These bytes will then be sent using 1, 2, 3 or 4 data lanes, so the DSITX controller will required 1, 1/2, 1/3, 1/4 this number of tx\_byte\_clk cycles respectively.

e.g a DPI with HSA = 12, HBP = 12, HACT = 1920, HFP = 24 and 16bpp will be 1968 pixel clocks for each horizontal line.

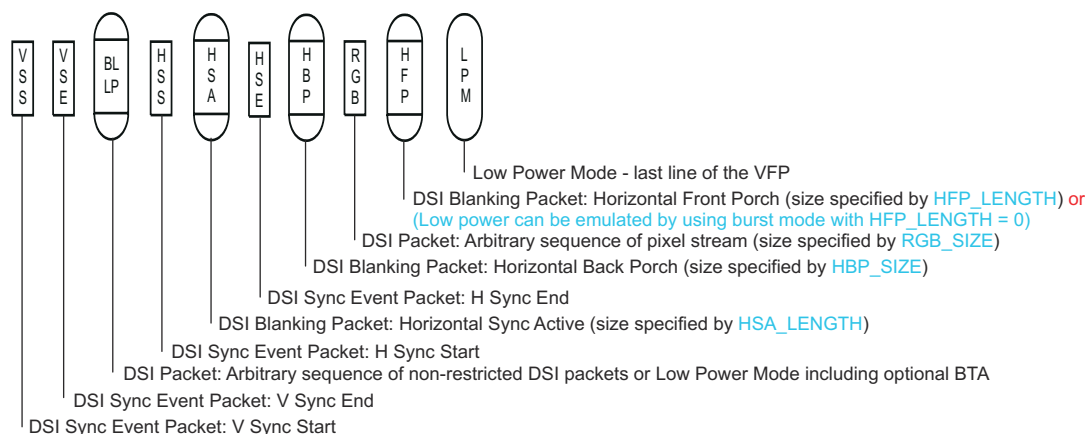
This gives the DSI horizontal lines 3936 bytes to transmit and would use 984 tx\_byte\_clks on a four lane system, or 1312 on a three lane system etc..

The horizontal configuration uses different registers depending on the Mode (Pulse or Event) and the control of the BLKLINE\_MODE register bit.

- Non-Burst Mode with Sync Pulses – enables the peripheral to accurately reconstruct original video timing, including sync pulse widths.

Note that for accurate reconstruction of timing, packet overhead including Data ID, ECC, and Checksum bytes should be taken into consideration.

- Non-Burst Mode with Sync Events – similar to above, but accurate reconstruction of sync pulse widths is not required, so a single Sync Event is substituted.
- Burst mode – RGB pixel packets (active video) portion are time-compressed, leaving more time during a scan line for LP mode (saving power) or for multiplexing other transmissions onto the DSI link.

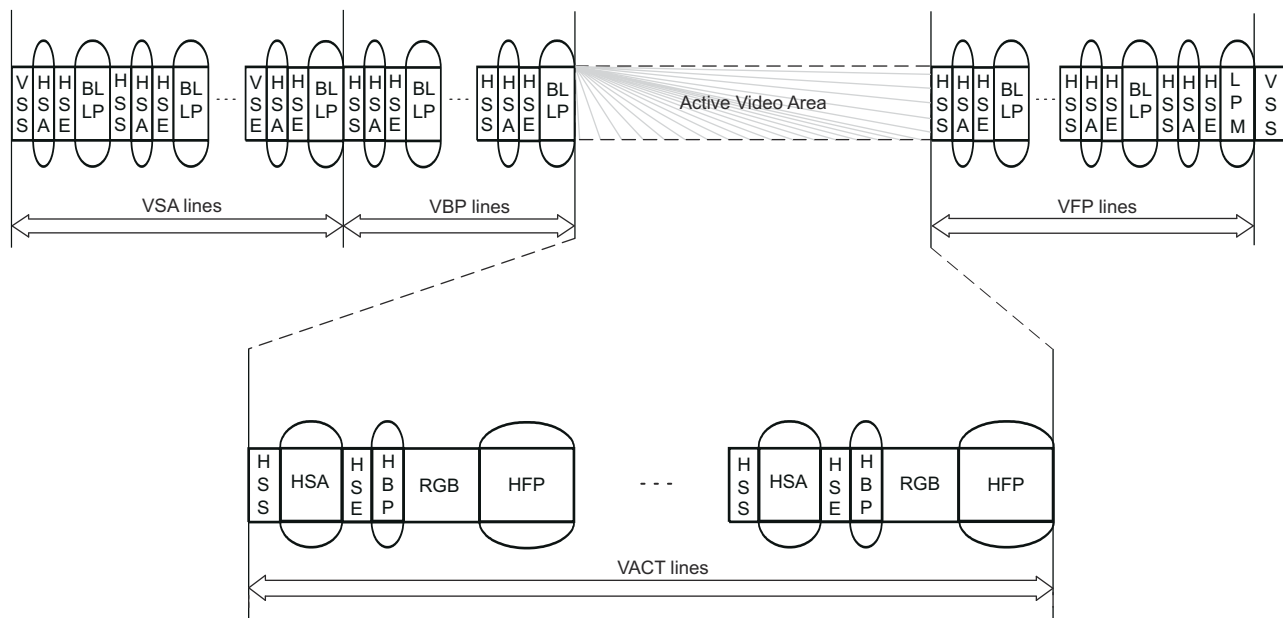


dsi-034

**Figure 12-1103. Vertical Timing**

#### 12.6.5.7.9.2 Horizontal Timing for Non-Burst Mode with Sync Pulses

Sync Pulse Mode uses the Short packet and Long packet structures to accurately track the DPI interface timing. Figure 12-1104 illustrates the packet sequence for a single frame.

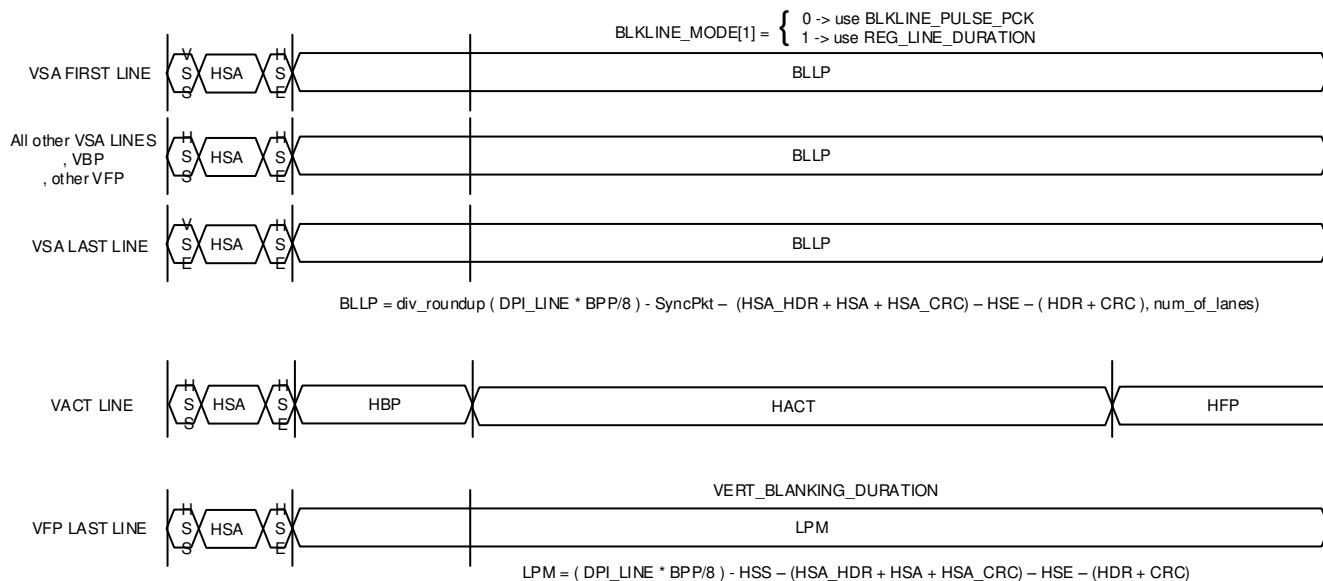


dsi-035

**Figure 12-1104. Vertical Timing**

The DSITX controller registers for this mode are used to generate the exact number of bytes required for each horizontal line based on the DPI line configuration and the BPP for the colour pixel data.

The packet structure for each type of line during the frame is shown below, Trigger Mapping Information. The calculation of the total number of bytes in any horizontal line must always be  $\text{HLINE} \times \text{bpp}/8$  bytes.



**Figure 12-1105. Non-Burst Mode With Sync Pulse Line Structures**

The DSI short packets and packet headers that are inserted by the controller must be accounted for:

- HSA should be reduced by 14 bytes to account for the HSS short packet (4 bytes), the long blanking packet header and CRC footer (4 + 2 bytes) and the HSE short packet (4 bytes).
- HBP should be reduced by 12 to account for the header and footer on the blanking packet (6 bytes) plus the header/footer on the active data packet (6 bytes).
- HFP should be reduced by 6 bytes to account for the long packet header and CRC footer.

Finally for lines with no active data, the controller will use either:

- **BLKLINE\_PULSE\_PCK**: Total line size ( $H_{LINE} \times \text{bpp}/8$ ) in bytes minus the HSA 20 bytes (14 for HSA header and CRC, 6 for the remaining blanking which is all combined into a single packet).
- **REG\_LINE\_DURATION**: Total line size ( $H_{LINE} \times \text{bpp}/8$ ) in tx\_byte\_clk cycles minus the calculated HSA in tx\_byte\_clk cycles ( $HSA \times \text{bpp}/8$  minus 14 bytes (14 for HSA header and CRC) minus the EOT packet (4 bytes) if required, divided by number of lanes). This should be aligned to the number of active lane so rounding the value so that  $REG\_LINE\_DURATION \bmod \text{Lanes}$  equals zero.
- **VERT\_BLANKING\_DURATION**: Total line size ( $H_{LINE} \times \text{bpp}/8$ ) in tx\_byte\_clk cycles minus the calculated HSA in tx\_byte\_clk cycles ( $HSA \times \text{bpp}/8$  minus 14 bytes (14 for HSA header and CRC) divided by number of lanes).

Program the DSI horizontal size registers as follows:

burst\_mode = 0 and sync\_pulse\_active = sync\_pulse\_horizontal = 1;

$HSA = (DPI\_HSA \times \text{bpp}/8) - 14$  – DPI HSA min value 5 for RGB888

$HBP = (DPI\_HBP \times \text{bpp}/8) - 12$  – DPI HBP min value 5 for RGB888

$HACT = (DPI\_HACT \times \text{bpp}/8)$

$HFP = (DPI\_HFP \times \text{bpp}/8) - 6$  – DPI HFP min value 10 for RGB888

**Note:**  $(DPI\_HACT \times \text{bpp}/32)$  must be an integer. Total Line Length =  $\text{div\_roundup}((H_{LINE} \times \text{bpp}/8), \text{num of lanes})$ ;

$(BLKLINE\_PULSE\_PCK) \text{ bytes} = (H_{LINE} \times \text{bpp}/8) - 20 - HSA$ ;

$(REG\_LINE\_DURATION)_{tx\_byte\_clks} = \text{Total Line Length} - \text{div\_roundup}((HSA \times \text{bpp}/8) - 14, \text{number of lanes})$ ;

$(VERT\_BLANKING\_DURATION)_{tx\_byte\_clks} = \text{Total Line Length} - \text{div\_roundup}((HSA \times \text{bpp}/8) - 14, \text{number of lanes})$ ;

For example; a DPI with  $HSA = 12$ ,  $HBP = 12$ ,  $HACT = 1920$ ,  $HFP = 24$  and 16bpp will be 1968 pixel clocks for each horizontal line. This give each DSI  $H_{LINE}$  of 3936 bytes. So for 4 lanes, 984 tx\_byte\_clk cycles;

$HSA = (12 \times 16/8) - 14 = 10$

$HBP = (12 \times 16/8) - 12 = 12$

$HACT = (1920 \times 16/8) = 3840$

$HFP = (24 \times 16/8) - 6 = 42$

Total Line =  $\text{div\_roundup}((12 + 12 + 1920 + 24) \times 16/8, 4) = 984 \text{ tx\_byte\_clk cycles} = 3936 \text{ bytes}$ ;

$BLKLINE\_PULSE\_PCK = (3936) - 20 - 10 = 3906$

$REG\_LINE\_DURATION = 984 - \text{div\_roundup}(12 \times 2, 4) = 978$

$VERT\_BLANKING\_DURATION = 984 - \text{div\_roundup}(12 \times 2, 4) = 978$

Confirming the calculation for each line with  $BLKLINE\_MODE = 0$ ;

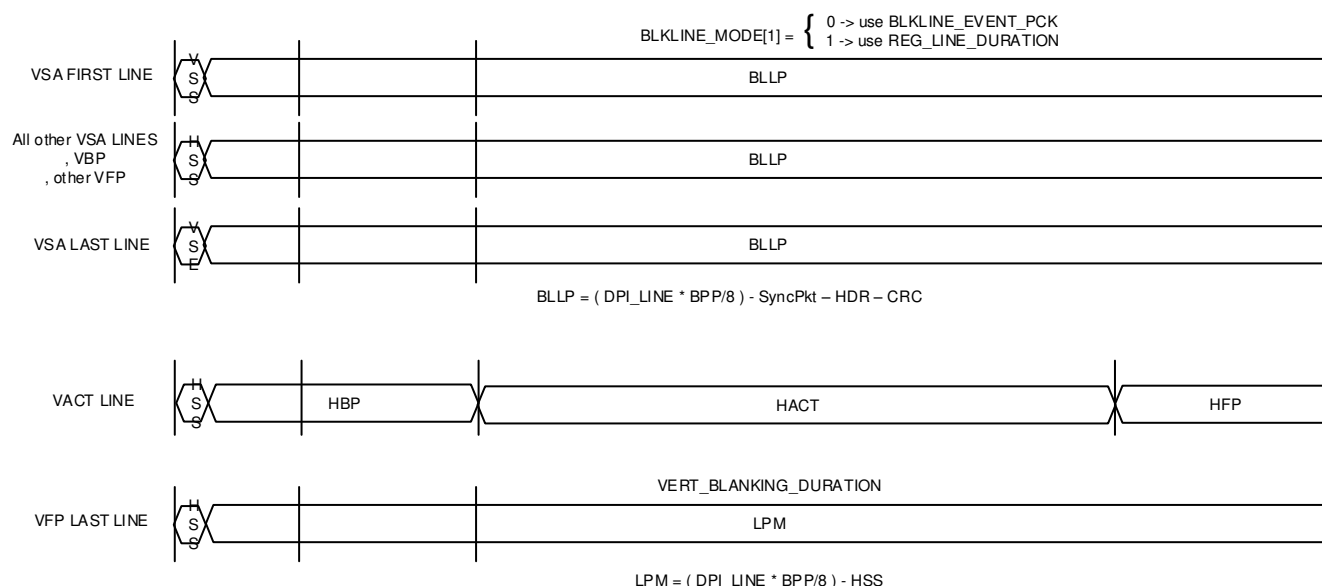
$VSS = VSSPkt + HSAPkt + HSEPkt + BLK\_LINE\_PULSE\_PCKPkt = 4 + (4 + 10 + 2) + 4 + (4 + 3906 + 2) = 3936$ ;

$VSA_{Line} = HSSPkt + HSAPkt + HSEPkt + BLK\_LINE\_PULSE\_PCKPkt = 4 + (4 + 10 + 2) + 4 + (4 + 3906 + 2) = 3936$ ;

$VSE = VSEPkt + HSAPkt + HSEPkt + BLK\_LINE\_PULSE\_PCKPkt = 4 + (4 + 10 + 2) + 4 + (4 + 3906 + 2) = 3936$ ;

$VACT = HSSPkt + HSAPkt + HSEPkt + HBPPkt + HACTPkt + HFPPkt = 4 + (4 + 10 + 2) + 4 + (4 + 12 + 2) + (4 + 3840 + 2) + (4 + 42 + 2) = 3936$ ;




**Figure 12-1107. Horizontal Timing - 1**

The DSI short packets and packet headers that are inserted by the controller must be accounted for:

- VSS/VSE/HSS short packet (4 bytes).
- HBP should account for the header and footer on the blanking packet (6 bytes) plus the header on the active data packet (2 bytes). This will match the DPI\_HSA + DPI\_HBP minus the Sync Short packet.
- HFP should be reduced by 6 bytes to account for the long packet header and CRC footer and footer of the active data.

Finally, for lines with no active data, the controller will use either:

- BLKLINE\_EVENT\_PCK: Total line size (HLINE \* bpp/8) in bytes minus 10 bytes (4 for VSS/VSE/HSS, 6 for the remaining blanking header/footer which is all combined into a single packet).
- REG\_LINE\_DURATION: Total line size div\_roundup(HLINE \* bpp/8, num\_of\_lanes) in tx\_byte\_clk cycles minus tx\_byte cycles for the 8 bytes (4 for VSS/VSE/HSS, and 4 for EOT). This should be aligned to the number of active lane so round the value so that REG\_LINE\_DURATION MOD Lanes equals zero.
- VERT\_BLANKING\_DURATION: Total line size div\_roundup(HLINE \* bpp/8, num\_of\_lanes) in tx\_byte\_clk cycles minus tx\_byte cycles for the 4 bytes (4 for HSS).

Program the DSI horizontal size registers as follows:

burst\_mode = 0 and sync\_pulse\_active = sync\_pulse\_horizontal = 0;

- HSA = 0
- HBP = ((DPI\_HSA + DPI\_HBP) \* bpp/8) - 12 - DPI HSA + HBP min value 5 for RGB888
- HACT = (DPI\_HACT \* bpp/8)
- HFP = (DPI\_HFP \* bpp/8) - 6 - DPI HFP min value 10 for RGB888

**Note:** (DPI\_HACT \* bpp/32) must be an integer. Total Line Length = div\_roundup((HLINE \* bpp/8), num of lanes).

(BLKLINE\_EVENT\_PCK) bytes} = (HLINE \* bpp/8) - 10

(REG\_LINE\_DURATION) tx\_byteclks} = Total Line Length - div\_roundup(8, number of lanes)

(VERT\_BLANKING\_DURATION) tx\_byteclks} = Total Line Length - div\_roundup(4, number of lanes)

For example: a DPI with HSA = 12, HBP = 12, HACT = 1920, HFP = 24 and 16bpp will be 1968 pixel clocks for each horizontal line. This give each DSI HLINE of 3936 bytes.

HSA = 0

$$HBP = ((12 + 12) \times 16/8) - 12 = 24$$

$$HACT = (1920 \times 16/8) = 3840$$

$$HFP = (24 \times 16/8) - 6 = 42$$

$$\text{Total Line} = \text{div\_roundup}((12 + 12 + 1920 + 24) \times 16/8, 4) = 984$$

$$\text{BLKLINE\_EVENT\_PCK} = (\text{HLIN} \times \text{bpp}/8) - 10 = 981$$

$$\text{REG\_LINE\_DURATION} = 984 - \text{div\_roundup}(8, 4) = 982$$

$$\text{VERT\_BLANKING\_DURATION} = 984 - \text{div\_roundup}(4, 4) = 983$$

Confirming the calculation for each line:

$$VSS = VSSPkt + \text{BLK\_LINE\_EVENT\_PCK}Pkt = 4 + (4 + 3926 + 2) = 3936$$

$$VSAline = HSSPkt + \text{BLK\_LINE\_EVENT\_PCK}Pkt = 4 + (4 + 3926 + 2) = 3936$$

$$VSE = VSEPkt + \text{BLK\_LINE\_EVENT\_PCK}Pkt = 4 + (4 + 3926 + 2) = 3936$$

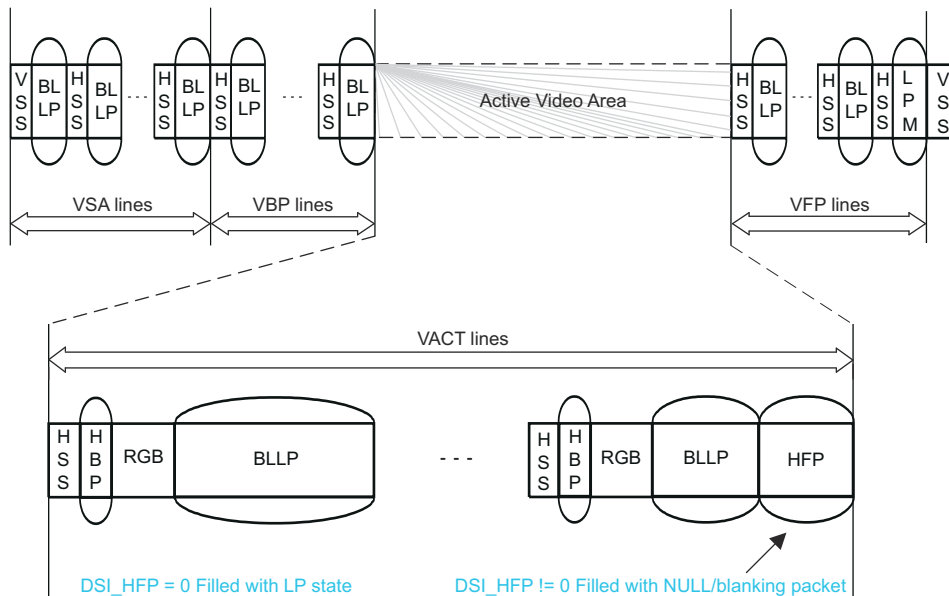
$$VACT = HSSPkt + HBPPkt + HACTPkt + HFPPkt = 4 + (4 + 24 + 2) + (4 + 3840 + 2) + (4 + 42 + 2) = 3936$$

$$VFP = HSSPkt + \text{BLK\_LINE\_EVENT\_PCK}Pkt = 4 + (4 + 3926 + 2) = 3936$$

$$VFPLast = HSSPkt + \text{VERT\_BLANKING\_DURATION} = 4 + (983 \times 4) = 3936$$

#### 12.6.5.7.9.4 Burst Event Mode Horizontal Timing

Burst Event Mode uses the Short packet and Long packet structures to transmit the video information in short bursts and allow the system to use the Low Power states to save active power. This can only be supported with DSITX controllers able to store a horizontal line in the DPI FIFO. The burst mode will expect the tx\_byte\_clk to be twice the rate of the non burst event mode, and the register calculation will expect the total bytes for each line to double. [Figure 12-1108](#) illustrates the packet sequence for a single frame.

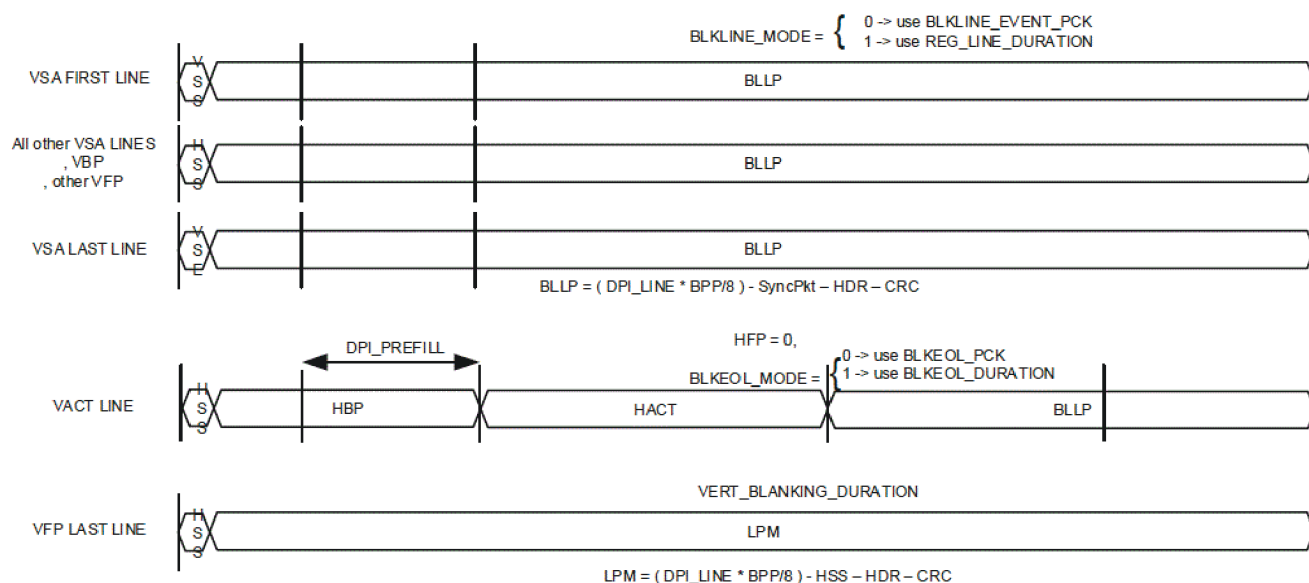


**Figure 12-1108. Burst Event Mode Horizontal Timing - 1**

The DSITX controller registers for this mode are used to generate the exact number of bytes required for each horizontal line based on the DPI line configuration and the BPP for the colour pixel data.



The packet structure for each type of line during the frame is shown below, with the DSI HFP = 0 to select BLLP operation.



ds\_i\_spruij7-040

**Figure 12-1109. Burst Event Mode Horizontal Timing - 2**

The DSI short packets and packet headers that are inserted by the controller for active lines must be accounted for:

- VSS/VSE/HSS short packet (4 bytes).
- HBP should account for the header and footer on the blanking packet (6 bytes) plus the header/footer on the active data packet (6 bytes). This will match the DPI\_HSA + DPI\_HBP minus the Sync Short packet. In this case the DPI must also reduce the HSA and HFP timing.
- BLLP uses BLKEOL\_PCK or BLKEOL\_DURATION and will change depending on the HFP use case:
  - HFP can be zero to extend the BLKEOL\_x values (recommended).
  - or a non-zero value to transmit a Blanking Packet instead of LP state. The Blanking Packet HFP should be reduced by 6 bytes to account for the long packet header and CRC footer.
  - The host\_eot state will cause the BLLP to either add and EOT cycle or not. The BLKEOL\_PCK or BLKEOL\_DURATION value will need to be adjusted to include this extra cycle if host\_eot is disabled.

Finally for lines with no active data, the controller will use either:

- BLKLINE\_EVENT\_PCK in bytes, excluding the header and crc.
- REG\_LINE\_DURATION or VERT\_BLANKING\_DURATION.
- Both are the total expected tx\_byte\_clk cycle for the line minus the number of cycles required to send the header packet bytes (4 for VSS/VSE/HSS) over the active lanes.

Program the DSI horizontal size registers as follows:

burst\_mode = 1 and sync\_pulse\_active = sync\_pulse\_horizontal = 0;

- HSA = 0
- HBP = (2 \* (DPI\_HSA + DPI\_HBP) \* bpp/8) - 12 + DPI FIFO Prefill
- HACT = (DPI\_HACT \* bpp/8) NOTE: (DPI\_HACT \* bpp/32) must be an integer.
- HFP = Zero

Total Line Length = div\_roundup((HLINE \* bpp/8), num of lanes);

{BLKLINE\_EVENT\_PCK} = (HLINE \* bpp/8) \* 2 - 4



$(\text{REG\_LINE\_DURATION})\} = \text{Total Line Length} \times 2 - \text{div\_roundup}(4, \text{number of lanes})$

$(\text{VERT\_BLANKING\_DURATION})\} = \text{Total Line Length} \times 2 - \text{div\_roundup}(4, \text{number of lanes})$

$(\text{BLKEOL\_DURATION})\} = \text{Total Line Length} \times 2 - \text{TX\_BYTE\_CYCLES}$

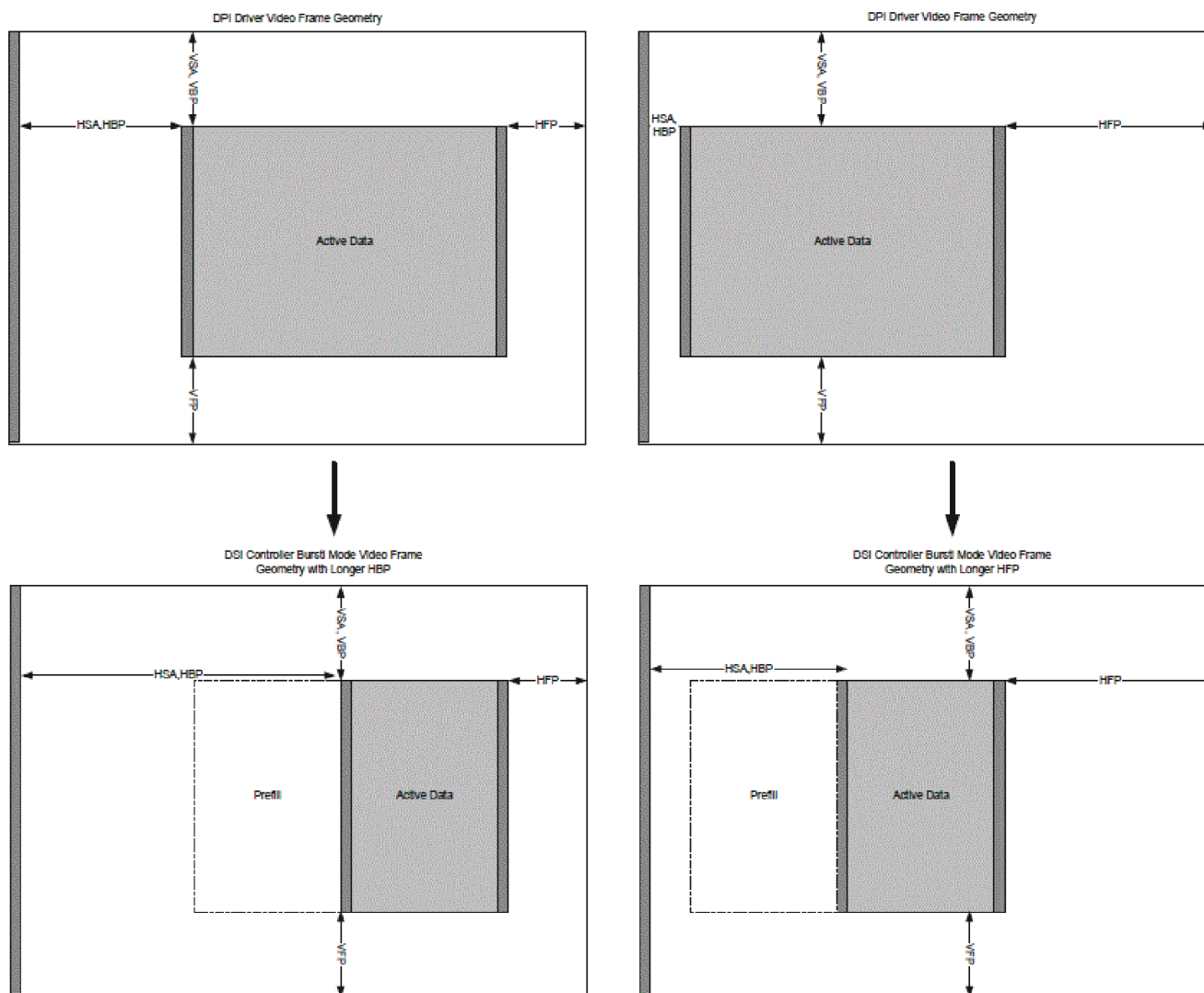
$(\text{BLKEOL\_PCK}) \text{ bytes} = \text{Total Line Length} \times 2 - (\text{HACT} + \text{HBP} + \text{HSS})$

**Note:**  $\text{TX\_BYTE\_CYCLES} = \text{TX Byte clock cycles to send Active Line bytes on the number of active lanes} = \text{div\_roundup}(\text{HBP} + \text{HACT}, \text{number of lanes})$ .

### Burst Operation Frame Configuration

Burst mode operation can be used to save power provided the system configuration can support the higher tx\_byte clock rate. The user should consider if the power saving during LP states is more than reducing the number of active lanes and running in non-burst mode.

The burst mode can only be used if the controller configuration can support it; either using SDI video interface, of DPI interface with a buffer size large enough to prefill the data. The video driver timing from the DPI side will also need to be changed.



ds\_i\_spruij7-041

**Figure 12-1110. DSITX Controller Video Frames for Burst Mode Compared with DPI Driver Side**

For example:

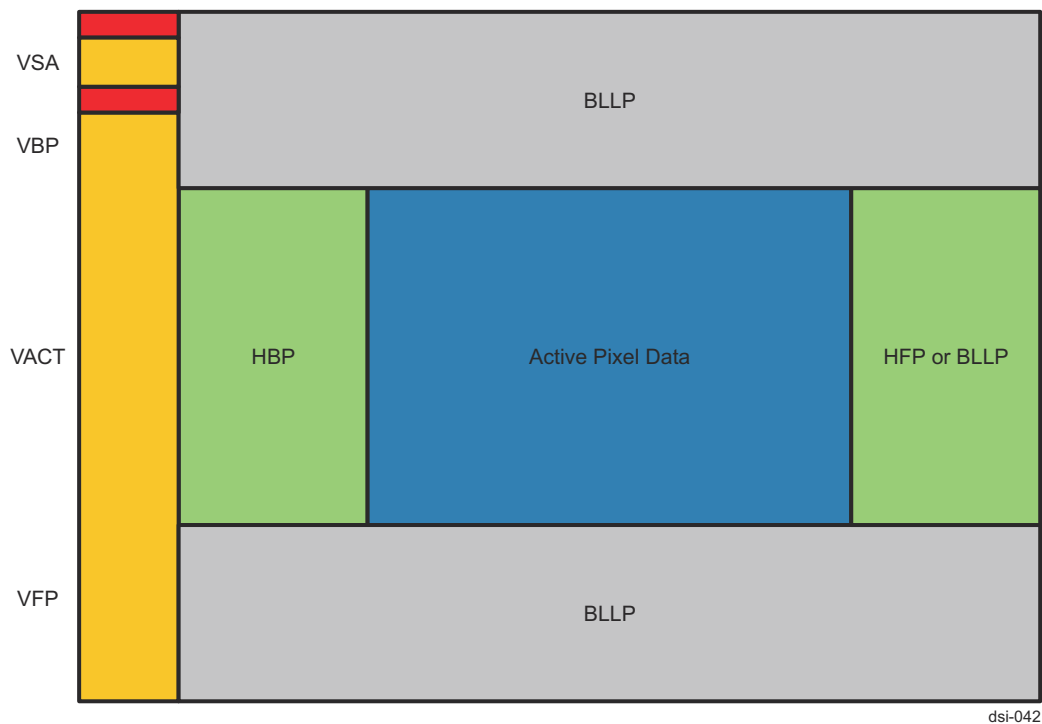
The VESA configuration for a 1920x1200 frame at 60-frames per second with normal blanking would have a pixel clock at 193.25 MHz, HSA 200 pixels, HBP 336 pixels, HFP 136 pixels and total line 2592 pixels. This would need to be reconfigured to move the active area earlier in the horizontal line for the DSITX controller, so could be changed to; HSA 20 pixels, HBP 30 pixels, HFP 622 pixels. This would allow the HFP LP state to be almost  $\frac{1}{4}$  of the line ( $622/2592$ ).

#### 12.6.5.7.9.5 Burst Mode Operation

The DSI IP offers different options for behavior in the BLLP areas:

- Insertion of blanking or null packets.
- Switch to LP (power saving).
- Insertion of commands (if any) + blanking or null packet.

Figure 12-1111 shows burst mode operation.



**Figure 12-1111. Burst Mode Operation**

The user should balance the increase in the clock frequencies for transmission of the active display, against the power saving by using BLLP(Null) or LP state.

For Burst with the LP state the register fields must be programmed for `burst_lp = 1` and `reg_blkeol_mode = 2'b2`.

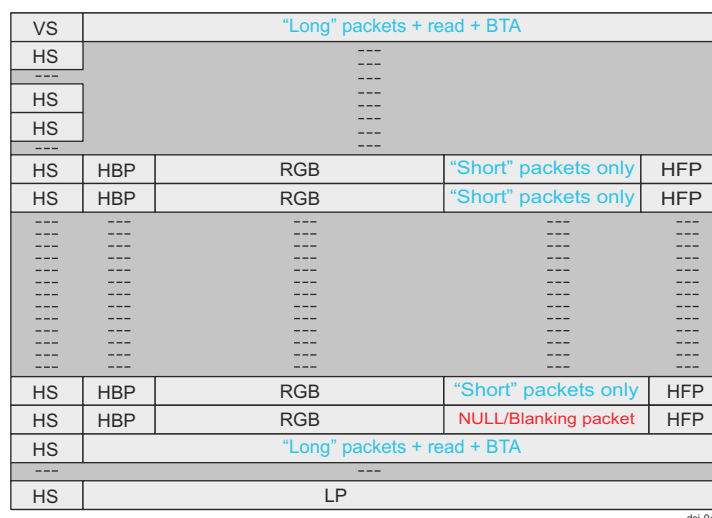
### Command Insertion Operation

The DSITX can use the burst operation to allow the blanking/LP stages to be used for DCS/GEN command insertion by programming the size of the space available after the sync of each line.

Only one command packet can be passed within each video line. The decision to insert a command in a video line is done at the beginning of the slot just after the previous video packet. If a command arrives in the middle of the slot, it is not processed in that line, the decision to accept the command will be taken in the next line.

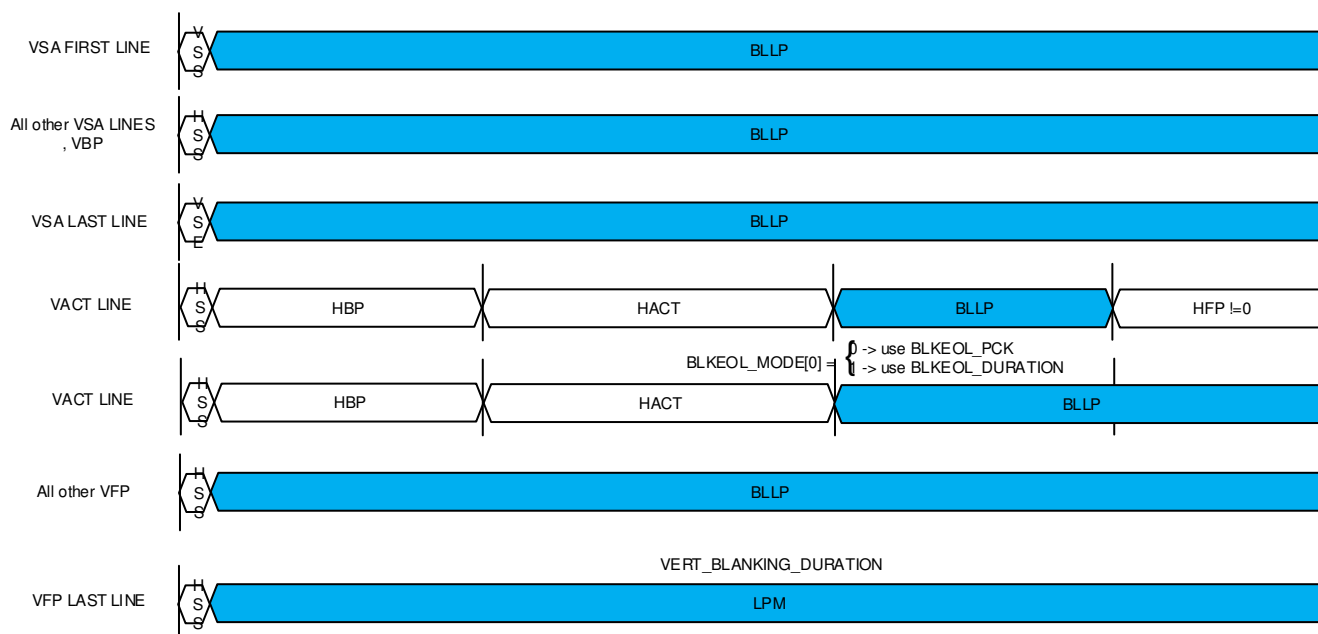
If the packet does not fit in a short slot (size bigger than `reg_max_burst_limit` and not equal to `reg_exact_burst_limit`), it is delayed up to next long slot. The signal `reg_err_burstwrite` is flagged (it is only an information that can help application software control). If its length is longer than `reg_max_line_limit`, it is discarded and the error `reg_err_linewrite` is generated.

Read commands and BTA requests are only passed during long slots, as there is no way to calculate their duration exactly. In the case where their duration takes longer time than the vertical blanking period, the error `reg_err_longread` is flagged.


**Figure 12-1112. Burst Mode Command Locations During Video Frame**

### Command Insertion Registers in Burst Operation

The burst operation requires the configuraton of the registers to control the size of the packets that will fit in the space available on each type of horizontal line.



In VSA, VBP or VFP: Insertion of short and long command packets, read or BTA are allowed.



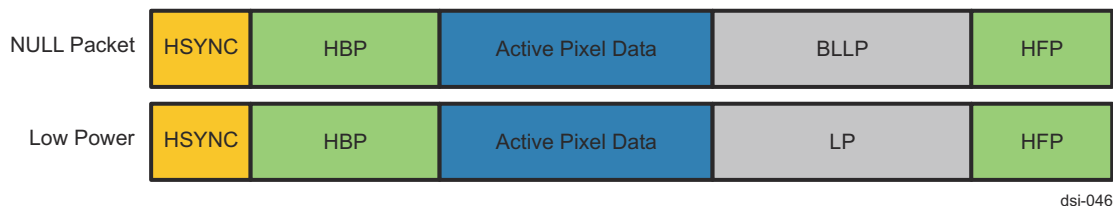
ds1-045

- In VACT: Only short packets can be inserted.
- Command, if inserted, is inserted right after the HSA or HSE or the video packet.
- Only one command can be inserted by line.

- When a command is inserted on a video line, no switch to LP but Null packet insertion.
- Trigger and TE insertion not supported.

VID\_VCA\_SETTING1 register:

[16] BURST\_LP: After an active line, in burst mode, the system can switch in LP (1) or should complete the line with NULL packet (0).



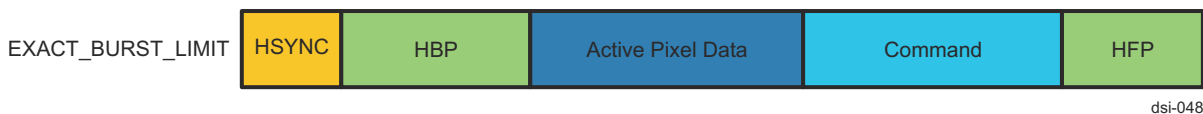
VID\_VCA\_SETTING1 register:

[15:0] MAX\_BURST\_LIMIT: Maximum length of a packet (number of bytes) that can be passed during the blanking period at the end of an active line (in burst mode) followed by a NULL packet with no data (6 bytes).



VID\_VCA\_SETTING2 register:

[15:0] EXACT\_BURST\_LIMIT: Exact maximum size of the burst packet (packet that fits after pixel data in burst mode), that is with no Null packet insertion afterwards.



VID\_VCA\_SETTING2 register:

[31:16] MAX\_LINE\_LIMIT: The "maximum" packet (number of bytes) that can be passed during the vertical blanking period.



If the packet does not fit in a short slot (size bigger than MAX\_BURST\_LIMIT and not equal to EXACT\_BURST\_LIMIT), it is delayed up to next long slot.

- If the packet length is longer than MAX\_LINE\_LIMIT, it is thrown away (an error is asserted).

[Table 12-1498](#) outlines the expected behaviour of each packet type during the video frame.

**Table 12-1498. Behavior of the DSITX Transmission when Both Video and Command are Running**

Packet type	Packet Size	Behavior	Error Flagged
VACT Active Line - Read and BTA do not pass			
No packet	-	LP mode null packet sent	
Short packet		Packet passed followed by a null packet	
Long packet	size < reg_max_burst_limit	Packet passed followed by a null packet	

**Table 12-1498. Behavior of the DSITX Transmission when Both Video and Command are Running  
(continued)**

Packet type	Packet Size	Behavior	Error Flagged
Long packet	size = reg_exact_burst_limit	Packet passed	
Long packet	reg_max_burst_limit < size < reg_exact_burst_limit	Packet delayed, LP mode packet delayed, null packet sent	Burstwrite Burstwrite
Long packet	reg_exact_burst_limit < size <= reg_max_line_limit	packet is delayed to non-active part of frame: Packet delayed, LP mode packet delayed, null packet sent	Burstwrite Burstwrite
Packet type	Packet Size	Behavior	Error Flagged
Long packet	size > reg_max_line_limit	Packet thrown, LP mode packet thrown, null packet sent	Linewrite Linewrite
Read packet	-	Packet delayed, LP mode Packet delayed, null packet sent	-
BTA request	-	Packet delayed, LP mode Packet delayed, null packet sent	-
VSA, VBP, VFP - Long packet, read and BTA can be passed			
No packet	-	LP mode	-
Short packet	-	Packet passed followed by a null packet	-
Long packet	size < reg_max_burst_limit	Packet passed followed by a null packet	
Long packet	size > reg_exact_burst_limit	packet thrown, LP mode	Linewrite
Read packet	read completed (direction) before end of slot	Packet passed	
Read packet	read not completed (direction) before end of slot	Packet passed	Longread If operation lasts too long
BTA request	-	Packet passed	Longread If operation lasts too long

Note that for burst operation with Low Power, the command will be sent as a high-speed packet even if the request is made to send as Low Power. Also, the next active line will return to using LP during the HFP phase if no other commands are requested.

#### 12.6.5.7.9.6 Example Configurations

1. Here is an example for a VGA frame with 24bpp @60fps using non burst pulse sync mode: With blanking, total size 500 lines, each line is 800 pixels.

400000 DPI cycles/frame @ 24 MHz pixel clock <=> 300000 DSI cycles/frame @ 18 MHz byte clock;

800 DPI cycles/line @ 24 MHz <=> 600 DSI cycles/line @ 18 MHz;

**Table 12-1499. DPI and DSI Parameters - 1**

Parameter	DPI	DSI
VSA	5	5
VBP	5	5
VACT	480	480

**Table 12-1499. DPI and DSI Parameters - 1 (continued)**

Parameter	DPI	DSI
VFP	10	5
HSA	40	106 (120-14)
HBP	40	108 (120-12)
HACT	640	1920
HFP	80	234 (240-6)
HTOTAL	800 pixel cycles	2400 bytes

burst\_mode = 0

sync\_pulse\_active = 1

sync\_pulse\_horizontal = 1

BLKLINE\_PULSE\_PCK = 2380 (2400-20)

REG\_LINE\_DURATION = 2380

VERT\_BLANKING\_DURATION = 2384

**Note:** The controller will send 4 VFP lines then transition to LP for the duration equivalent to 6 lines.

2. Here are some example values for a 24fps UHD frame size with 24bpp using non-burst event mode: With blanking, total size 2250 lines, each line is 4000 pixels.

9000000 DPI cycles/frame @ 216 MHz pixel clock <-> 6750000 DSI cycles/frame @ 162 MHz byte clock

4000 DPI cycles/line @ 216 MHz <-> 3000 DSI cycles/line @ 162 MHz

**Table 12-1500. DPI and DSI Parameters - 2**

Parameter	DPI	DSI
VSA	2	2
VBP	0	0
VACT	2160	2160
VFP	40	1
HSA	40	0
HBP	40	228 (120 + 120 - 12)
HACT	3840	11520
HFP	80	234 (240 - 6)
HTOTAL	4000	12000

burst\_mode = 0

sync\_pulse\_active = 0

sync\_pulse\_horizontal = 0

BLKLINE\_EVENT\_PCK = 11990 (12000-10)

REG\_LINE\_DURATION = 11990

VERT\_BLANKING\_DURATION = 11996

**Note:** After the active data the controller will transition immediately to LP for the duration equivalent to 88 lines, giving the maximum power saving between lines.

If any of the DPI related interrupts are triggered, then this highlights that the FIFO depth and/or the vsync\_delay settings require to be tuned to the current configuration. Simulating the core operation with the expected clocks is the best way to ensure.

#### 12.6.5.7.9.7 Stereoscopic Video Support

The DSI controller can support stereoscopic display format (SDF) using both command and video modes.

- Command Mode
  - APB Direct Command 3D Commands – Use get\_3D\_control with Read
  - SDI Command Mode – DCS Writes
- Video Mode – Single Link Operation
  - VSYNC Parameter1 Definition (L/R, Mode, Format)
  - SDI Video Mode
  - DPI Video Mode – Video stream will expect to follow expected format; frame, line, pixel
    - 3DVSNC = 0 not currently supported.

The DSI must be configured to enable the 3D operation and identify the format that is being used using the register control to enable the 3D features and the configuration that matches the video stream from the system.

The SDF format requires the system to provide frame and synchronisation information using the two parameter fields within the vertical sync short packet (VSS).

Configured the DSI mctl\_3dvideo\_ctl register to match format – L/R Pixel, L/R Line, L/R Frame Drive video stream based on orientation and format.

**Note:** DSI Controller will not perform any on-the-fly pixel manipulation for L-R ordering.

#### Example Video Operation

For a 3D video stream where the DSI video is given the combined images as one continuous large frame, the VSYNC packet would identify that the left image is sent first, it is a line based format and in landscape mode (see [Table 12-1501](#)).

**Table 12-1501. Example Video Operation**

Position	D7	D6	D5	D4	D3	D2	D1	D0
Value	0	0	0	0	0	0	1	0
Description	Rsvd	Rsvd	Left First	No Sync	Line-based		Landscape	

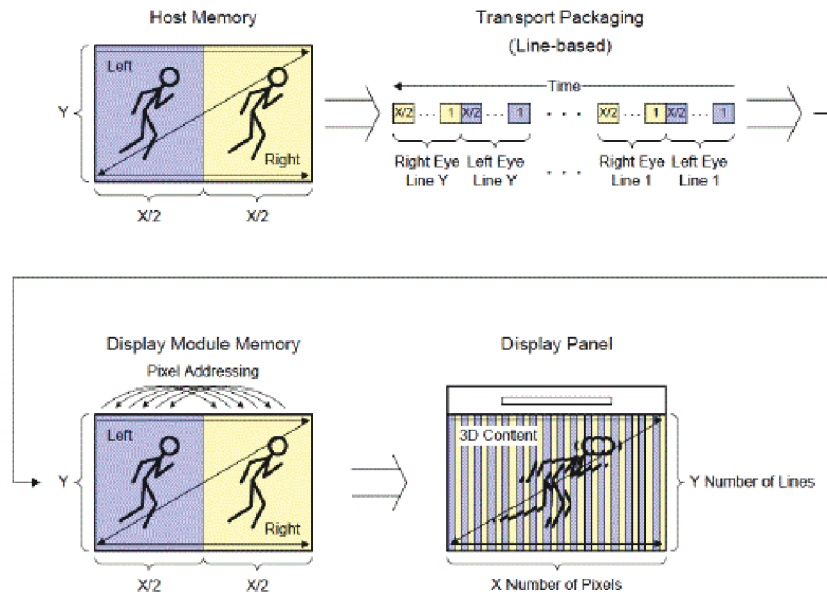
mctl\_3dvideo\_ctl (0x20)

**Table 12-1502. VSYNC Parameter 1**

[7]	vid_vsync_3d_en	1
[5]	vid_vsync_3d_lr	0
[4]	vid_vsync_3d_second_en	1
[3:2]	vid_vsync_3dformat	00
[1:0]	vid_vsync_3dmode	10

The video stream would then be configured in the DSI to be twice the line length of the normal image.

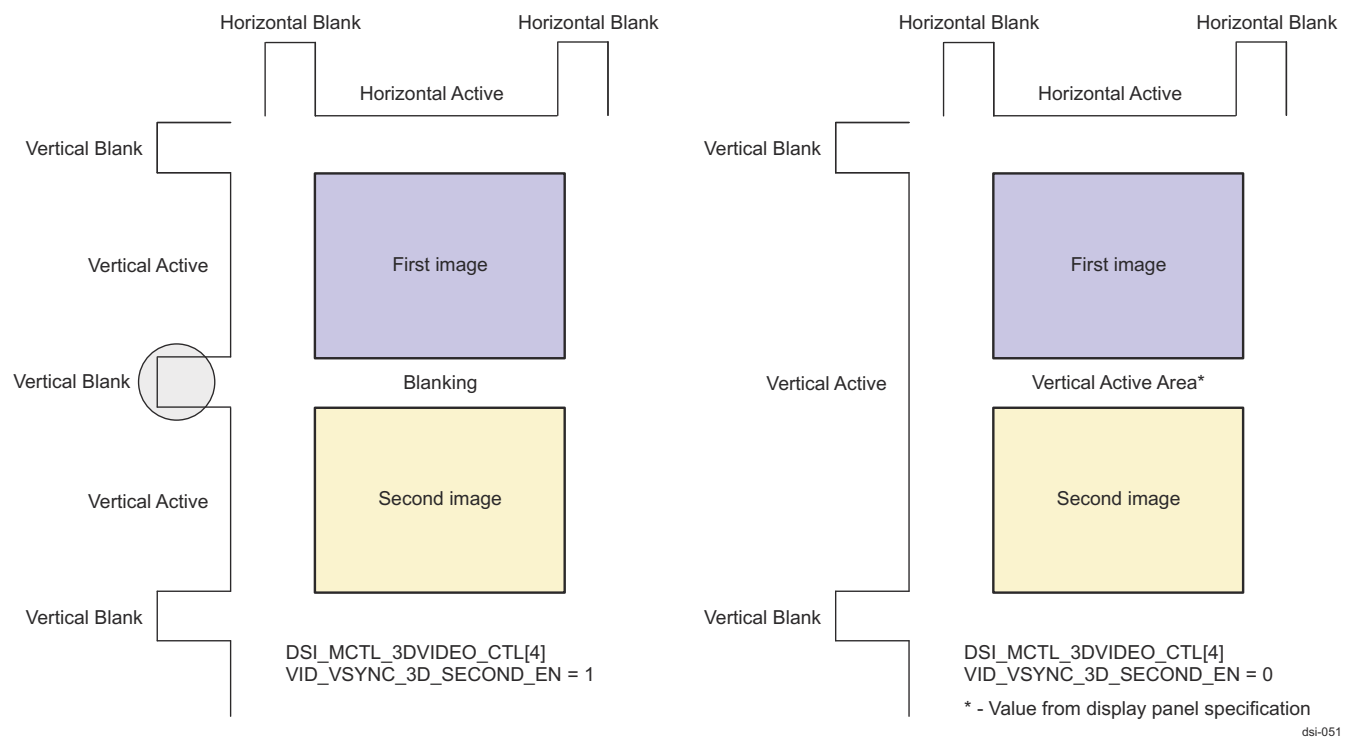




dsi\_spruij7-050

**Figure 12-1113. Example DPI Image Format for SDF Combining Left And Right Pixels**

**Note:** The vertical operation DSI controller will expect the system to provide the left and right as two individual frames with SYNC to allow the blanking to be generated. The DSI does not insert the vertical blanking area when VSYNC = 0.



dsi-051

**Figure 12-1114. 3D Image format for 3DVSYNC splitting first and second images**

#### 12.6.5.7.10 DSITX Video Stream Variable Refresh

The DSITX Controller can be programmed to allow the video stream sequence from a video source to halt between frames and enter Low Power state. The DPI video stream can halt before the new VSYNC is sent and remain in this state until the host video system wants to begin transmitting the new frame.

The DSITX will not perform the normal frame recovery mechanism when this is enabled, however it will continue to recover from line errors until the end of the frame.

**Table 12-1503. DSITX Video Stream Variable Refresh**

Variable refresh rate operation enable control	Configures how the video generation timing is controlled.0, 1 DPI mode operation supports = 1 (enabled). For SDI operation, this parameter can optionally be set to FALSE (= 0) when the system clock and tx_byte_clk are generated from the same reference clock source, or TRUE to allow variable refresh rate control.
--	--

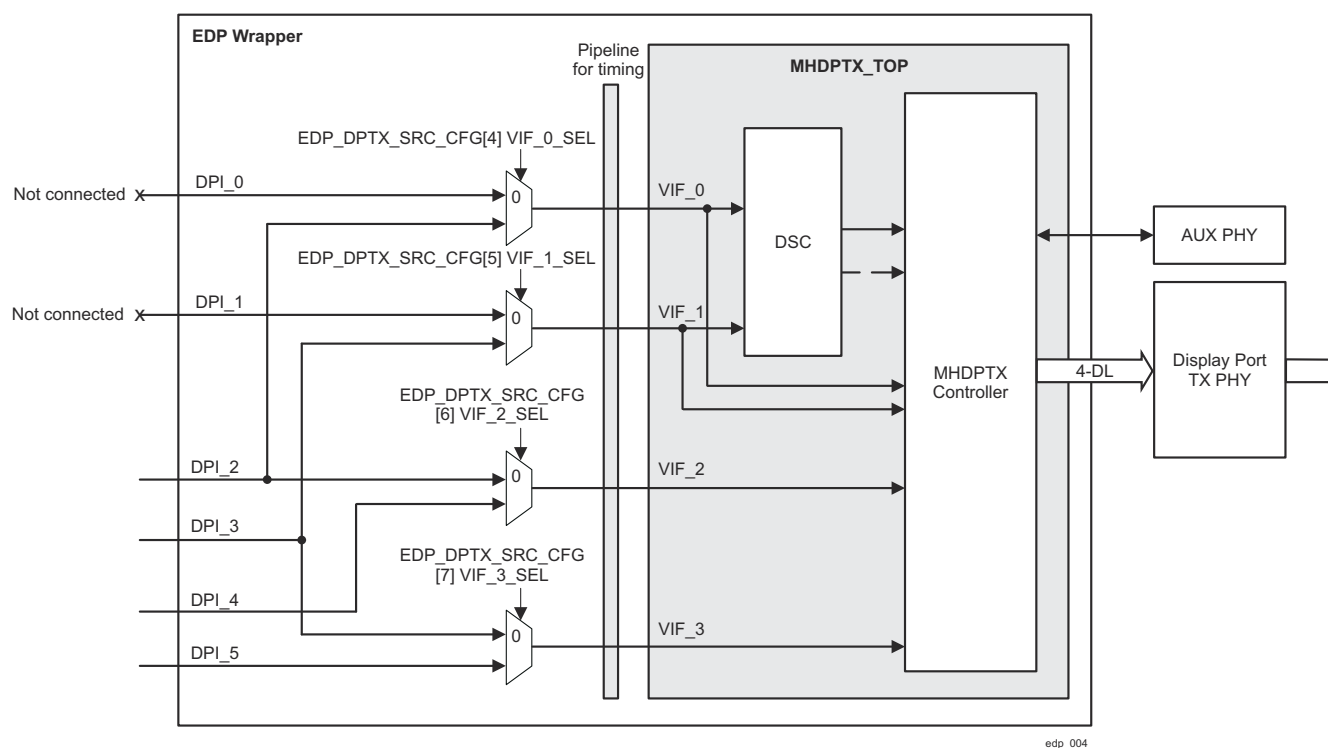


**Table 12-1504. EDP Wrapper DPI Interface Signals**

Signal Name <sup>(1)</sup>	Direction	Default Value	Description
dpi_n_data[47:0]	Input	48'd0	Pixel Data
dpi_n_m_mdata[47:0]	Input	48'd0	Master Meta Data Only bit [10] secure mode bit is used
dpi_n_de	Input	1'd0	Data Enable (active high)
dpi_n_vs	Input	1'd0	Start of frame or Vsync (active low)
dpi_n_hs	Input	1'd0	Start of line or Hsync (active low)

(1) n = 0 to 5

Figure 12-1116 shows the DPI data mapping to VIF ports via muxing in the EDP wrapper.


**Figure 12-1116. DP Wrapper DPI Data Muxing and VIF Mapping**

The following data stream transport modes are supported by the DPI data mapping to VIF ports:

**SST Mode (Single Stream Transport) – VIF0 used**

- DPI\_2 - Single Panel Input (optional DSC compression) mapped to VIF0
- DPI\_2/DPI\_3 - Split Panel Dual Inputs (for DSC compression only) mapped to VIF0 (via DSC) (see Note 1)

**MST Mode (Multi-Stream Transport) – All VIFs used**

- DPI\_2 - Single Panel Input (optional DSC compression) mapped to VIF0
- DPI\_3 - Single Panel Input (optional DSC compression in dual panel mode) mapped to VIF1 (see Note 2 and Note 3)
- DPI\_2/DPI\_3 - Split Panel Dual Inputs (for DSC compression only - two L/R streams compressed separately but outputs combined as one stream) with compressed output mapped to VIF0 (via DSC)
- DPI\_2 or DPI\_4 - Single Panel Input (no DSC compression) mapped to VIF2
- DPI\_3 or DPI\_5 - Single Panel Input (no DSC compression) mapped to VIF3

Notes:

1. DSC can be enabled to perform compression on a single input stream as Left and Right panel data for high resolution video (pixel clock > ~400 MHz) that cannot be compressed with a single DSC encoder. In this mode, two input streams share the same video timing (vs/hs/de) and identical input pixel clocks. This mode is referred to as **"split panel mode"**.
2. DSC can also be enabled to perform compression of two unrelated input streams of the same resolution. These streams must share the same pixel input clock since the DSC encoders require two encoder clocks to be synchronous. But, in this case, two streams can be asynchronous in terms of video timing (vs/hs/de). This mode is also referred to as **"dual panel mode"**.
3. When DSC is using only one encoder, then it is required to be on Enc0 input (meaning the stream to be compressed must be on DPI\_2). In MST mode (where this constraint is applicable), the single stream that is to be compressed must be placed on DPI\_2 and therefore the DP sink device (that receives the compressed stream) will be assigned to receive the STREAM\_0.

When DSC is in split or dual panel mode, the DSS uses a common clock to send two streams. In the EDP, two DSC input clocks are sourced from the same DPI\_2 source clock to keep these clocks to be the same (per DSC requirement) in order to minimize the skew between two DSC encoder clocks.

For detailed description of the clock diagrams, including the clock muxes per data selection, refer to [Section 12.6.6.5.1, Clock Diagram](#).

#### 12.6.6.2.2 Secure Video Content Protection

Each DPI\_n input port includes a 48-bit DPI\_n\_M\_MDATA bus from DSS (DISPC) which carries attribute sideband signals associated with the DPI interface. The EDP sets the security status (DPISECURE) of the incoming DPI interface using the DPI\_n\_M\_MDATA[10] bit as shown in [Table 12-1505](#).

**Table 12-1505. EDP Secure Video Content Protection**

Sideband bit(s)	Name	Description
DPI_n_M_MDATA[10]	DPISECURE	DPI-np secure bit 1: DPI-np is secure 0: DPI-np is not secure

The EDP performs a simple security check on the selected DPI connection by comparing the value of the selected DPISECURE bit (per EDP\_DPTX\_SRC\_CFG[7-4] VIF\_n\_SEL bits (where n = 0 to 3)) to the EDP\_DPTX\_VIF\_SECURE\_MODE\_CFG[3-0] VIF\_n bits.

If the selected DPISECURE bit is set to 1, the video content is considered to be secure and the connection (transmission via DPTX) is only made if the EDP\_DPTX\_VIF\_SECURE\_MODE\_CFG[3-0] VIF\_n bits are also set to 1 (indicating the display is also secure). Otherwise, the EDP forces the DSS\_DPI\_DATA to be 0h while leaving the sync signals alone – in effect, causing the video to go black. If the DPISECURE bit is set to 0, then the connection is also enabled.

#### 12.6.6.2.3 DPI\_DATA Input Pixel Format Supported

[Figure 12-1117](#) shows the supported by the EDP pixel packing formats in the DSS\_DPI\_DATA.

Default: **RGB121212** - 36-bit pixel data (12 bpc) LSB aligned in 48-bit data bus



Optional: **RGB101010** - 30-bit pixel data (10 bpc) LSB aligned in 48-bit data bus



EDP\_005

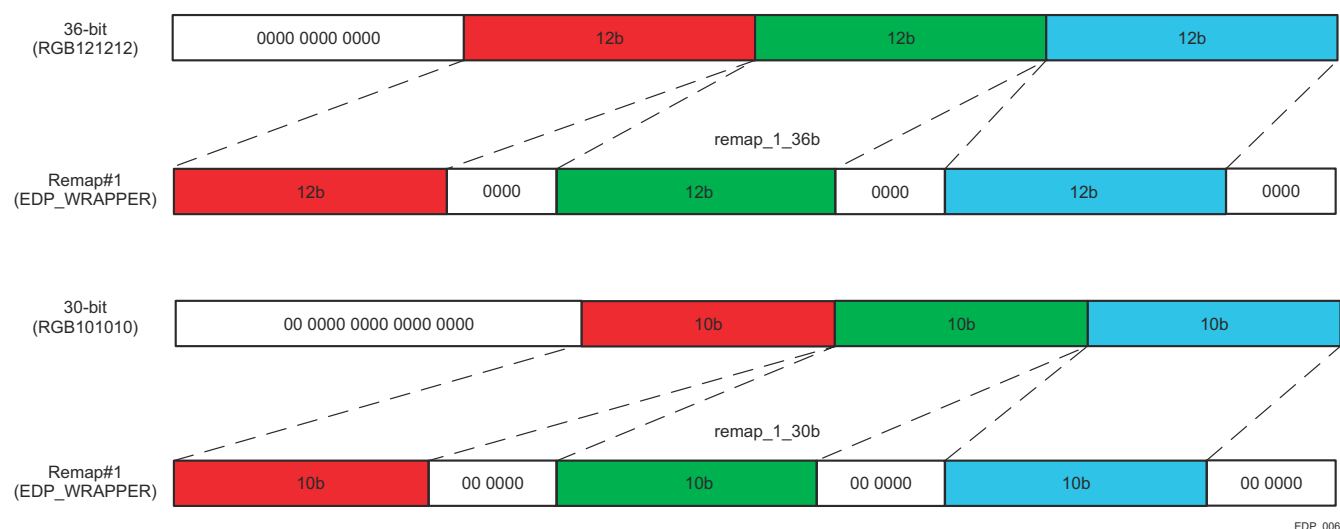
**Figure 12-1117. EDP Wrapper Supported Input DSS\_DPI\_DATA Pixel Formats**

The selection between these two formats is made per VIF\_n source by configuration EDP\_DPTX\_SRC\_CFG[11-8] VIF\_n\_IN30B parameters. The default is 36-bit format (RGB121212). Optionally,

the 30-bit format (which is the pixel format when the DPI\_data is generated from the DSS's merge/split module) can be selected to match the DSS configuration.

Internally, the EDP wrapper performs following pixel data/component realignments to match the input pixel data alignment required by the MHDPTX core (MSB alignment) and DSC (native LSB alignment).

Figure 12-1118 shows the internal mapping from the LSB packed pixel bus to MSB-aligned container packed bus: the 30 or 36-bit pixel data remapped to 48-bit (16 bpc (bits per component) MSB aligned within color component). The MHDPTX core extracts MSB 8/10/12-bits from each 16-bit color component to include in the display port video transport packet (see Table 12-1506, *EDP Video Interface Pixel Mapping (MSB Mapping)*).



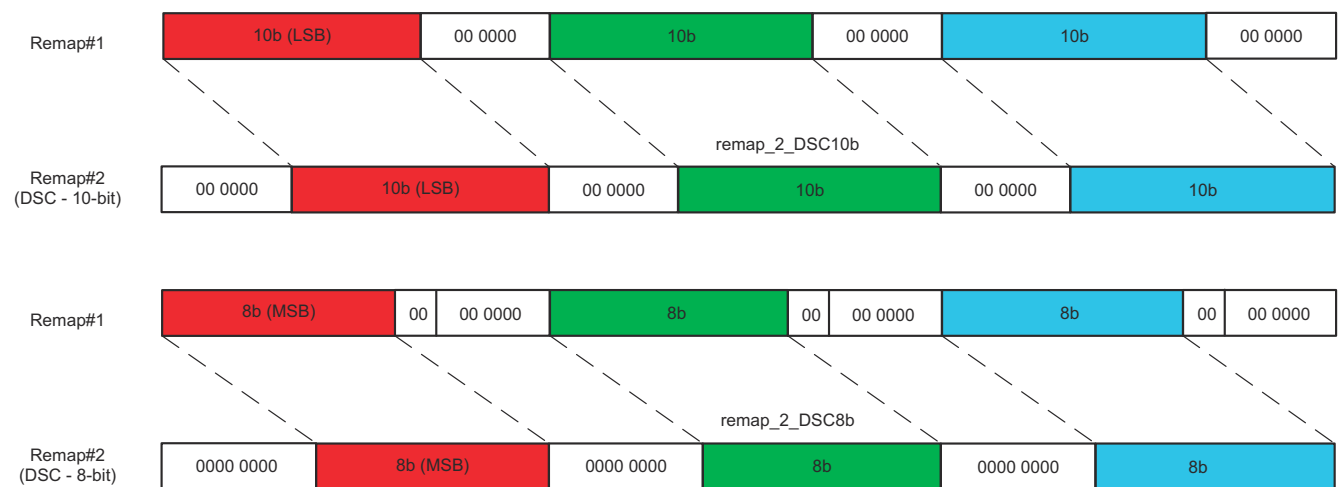
EDP\_006

**Figure 12-1118. EDP Wrapper Internal Pixel Bus Remapping (First Stage)**

DSC Input Data Re-alignment (Optional for DSC enabled VIF\_0 and VIF\_1) – From the 48-bit aligned data bus, the EDP wrapper performs additional re-alignment (LSB aligned component data per DSC requirement) and pixel data alignment per DSC input processing bit size configuration.

For this LSB alignment, EDP\_DPTX\_DSC\_CFG[6-5] DSC\_1/0\_10BPC bits must be configured to extract significant 8 or 10-bit component data from the input pixel bus, as shown on Figure 12-1119.

The parameters in these bits must match the parameters in the EDP\_CORE\_ENC0\_MAIN\_CONF\_P/EDP\_CORE\_ENC1\_MAIN\_CONF\_P[1-0] IPUT\_BPC bitfields.



EDP\_007

**Figure 12-1119. EDP Wrapper Internal Pixel Remapping (Additional for DSC Bound Pixel Data Bus)**

Pixel data alignment within the MHDPTX\_TOP should be set to be MSB always to enable the pixel mapping, as shown in [Table 12-1506](#). DSC bound pixel data bus is to be LSB aligned and requires no other MHDPTX\_TOP configuration.

**Table 12-1506. EDP Wrapper Video Interface Pixel Mapping (MSB Mapping)**

Video Bus	Source RGB/YCbCr444 (C of YCbCr422 <sup>(2)</sup> )				
	Bit Width	8-bit	10-bit	12-bit	16-bit <sup>(1)</sup>
Channel 2	[47:40]	R/Cr[15:8] / C[15:8]	R/Cr[15:6] / C[15:6]	R/Cr[15:4] / C[15:4]	R/Cr[15:0] / C[15:0]
	[39:38]				
	[37:36]				
	[35:32]				
Channel 1	[31:24]	G/Y[15:8]	G/Y[15:6]	G/Y[15:4]	G/Y[15:0]
	[23:22]				
	[21:20]				
	[19:16]				
Channel 0	[15:8]	B/Cb[15:8]	B/Cb[15:6]	B/Cb[15:4]	B/Cb[15:0]
	[7:6]				
	[5:4]				
	[3:0]				

(1) 16-bit source format is not supported. When selected, lower 4 bits are set to 0x0.

(2) YCbCr422 format is locally generated from YCbCr444 input from DSS (by a simple chroma decimation).

#### 12.6.6.2.4 Audio Input Interface

The audio input interface provides the following features:

- One I2S interface configured to support multiple physical audio channels (up to 8-channels)
- Frame Format – TDM mode
  - Multiple Physical Channels N = 1, 2, or 4
  - Variable TDM time slots M = 2 or 8
  - Time slot size: 32, 24, or 16
  - Variable word length (28, 24, 20, 16) configured as left or right justified
- I2S\_DATA - 4-bits wide
- Supports data rate up to 8-channel L-PCM @192 kHz (Buffer dependent)
- SPDIF mode is not supported

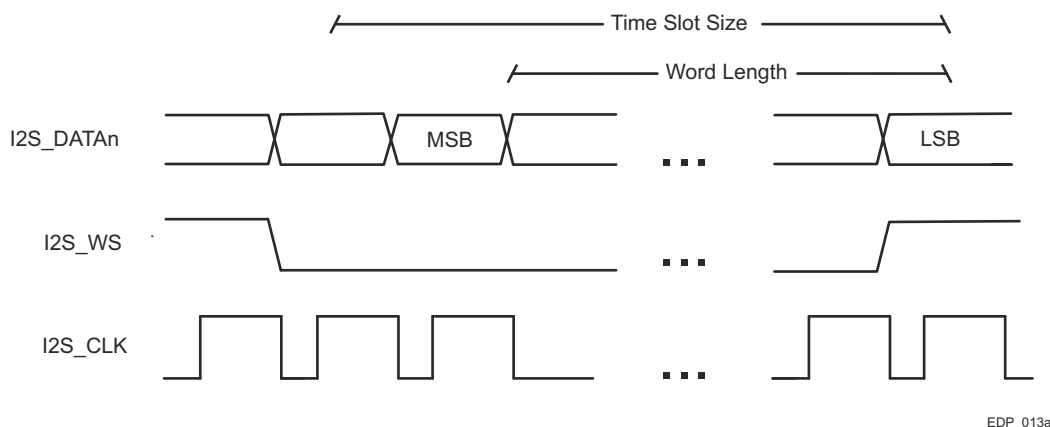
##### 12.6.6.2.4.1 Audio I2S Signals/Timing

[Table 12-1507](#) shows the I2S signals and timing.

**Table 12-1507. EDP Audio I2S Signals/Timing**

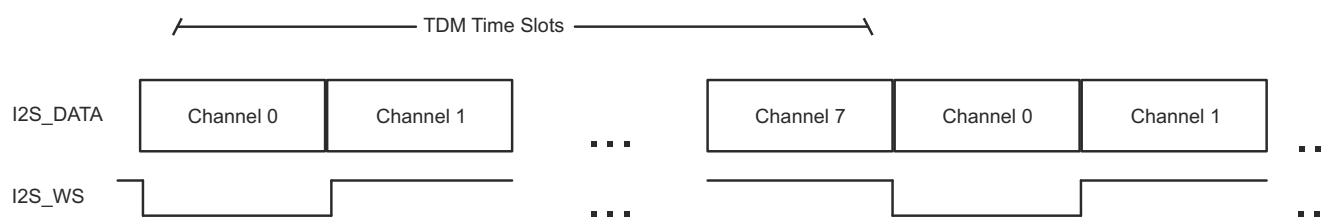
Signal Name	Direction	Default Value	Description
AIF_I2S_CLK	Input	1'd0	I2S clock
AIF_I2S_DATA[3:0]	Input	4'd0	I2S data for up to 4*M logical channels. Number of physical channels: 1, 2, or 4 M – number of time slots: 2 or 8
AIF_I2S_WS	Input	1'd0	I2S word-select indication. In TDM mode, WS = 0 indicates channel 0.

[Figure 12-1120](#) and [Figure 12-1121](#) show the I2S timing.



EDP\_013a

**Figure 12-1120. EDP Audio I2S Timing - Bit Allocation (Right Justification)**



EDP\_013b

**Figure 12-1121. EDP Audio I2S Timing - TDM Time Slot Allocation (M = 8)**

#### 12.6.6.2.4.2 Audio I2S Clock Frequency

The bit clock frequency is defined as below:

- Bit clk freq = Sampling\_rate \* Number\_of\_bits\_per\_channel \* Number\_of\_channels

For 16-bit/44.1 kHz stereo channels:

- I2S\_CLK (freq) = 44.1 kHz x 16 x 2 = 1.4112 MHz

For 24-bit/192 kHz stereo channels:

- I2S\_CLK (freq) = 192 kHz x 24 x 2 = 9.216 MHz

#### 12.6.6.3 EDP Transmitter Controller Subsystem (MHDPTX\_TOP)

The EDP Transmitter Controller Subsystem (MHDPTX\_TOP) integrates the following major components:

- Display Stream Compression (DSC) v1.1 Encoder – VESA Compression Engine
- Display Port TX Controller (MHDPTX Controller) - EDP/DP Transmitter Controller

##### 12.6.6.3.1 Display Stream Compression Encoder (DSC)

This module performs the VESA Digital Stream Compression encoding for the Display Port TX interface. The DSC contains two parallel Hard Slice Encoders, which support compression for VESA Display Port transmitter.

##### 12.6.6.3.1.1 DSC Encoder Features

The DSC Encoder supports the following main features:

- Maximum image width and height per encoder slice can be configured with up to 8K
  - Programmable slice width and height
  - All slices must be the same height, including the final slice of the picture
- 8 bits/component (24 bits/pixel)
- 10 bits/component (30 bits/pixel)



- An encoder built for 10 bits/component will be able to run in either 8 or 10 bits/component mode based on a register value
- RGB/YCbCr with 4:4:4 sampling support only on the input streams
- Single or Dual Pixel input(s)
  - Single pixel input (one encoder slice processing only)
  - Dual pixel inputs
    - Split Panel (Left/Right) streams – requires low skew between streams
- Support for all DSC 1.1 encoding mechanisms
  - MMAP, BP, MPP predictions and ICH
  - Flatness detection and signaling
- Supports CBR (Constant Bit Rate) mode only - Configurable target bpp (bits per pixel), non-integer values supported. The following modes are available:
  - 8 bpp and 12 bpp compressed bit rates for 8 bits/component - Corresponds to 3:1 and 2:1 compression
  - 10 bpp and 15 bpp compressed bit rates for 10 bits/component - Corresponds to 3:1 and 2:1 compression
  - 12 bpp and 18 bpp compressed bit rates for 12 bits/component
- Each Hard Slice encoder instance can be configured (with the parameter NB\_SS\_ENC) to support the following number of slices per line
  - Supports 1 or 2 slices (soft slices) per line
    - When configured with support for 2 slices per line, the encoder support dynamically the configuration of both 1 and 2 slices per line.
  - Thus a total of up to 4 slices per line can be supported with two Hard Slice Encoders
- Encoder data flow supported:
  - Synchronous Streaming mode – no back pressure support
- Implements the following active internal diagnostic mechanisms:
  - Self-Checking during VBLANK
  - Control output diagnostics
  - SRAM Protection
  - Configuration and Status Register (CSR) Protection
  - Fault avoidance mechanism

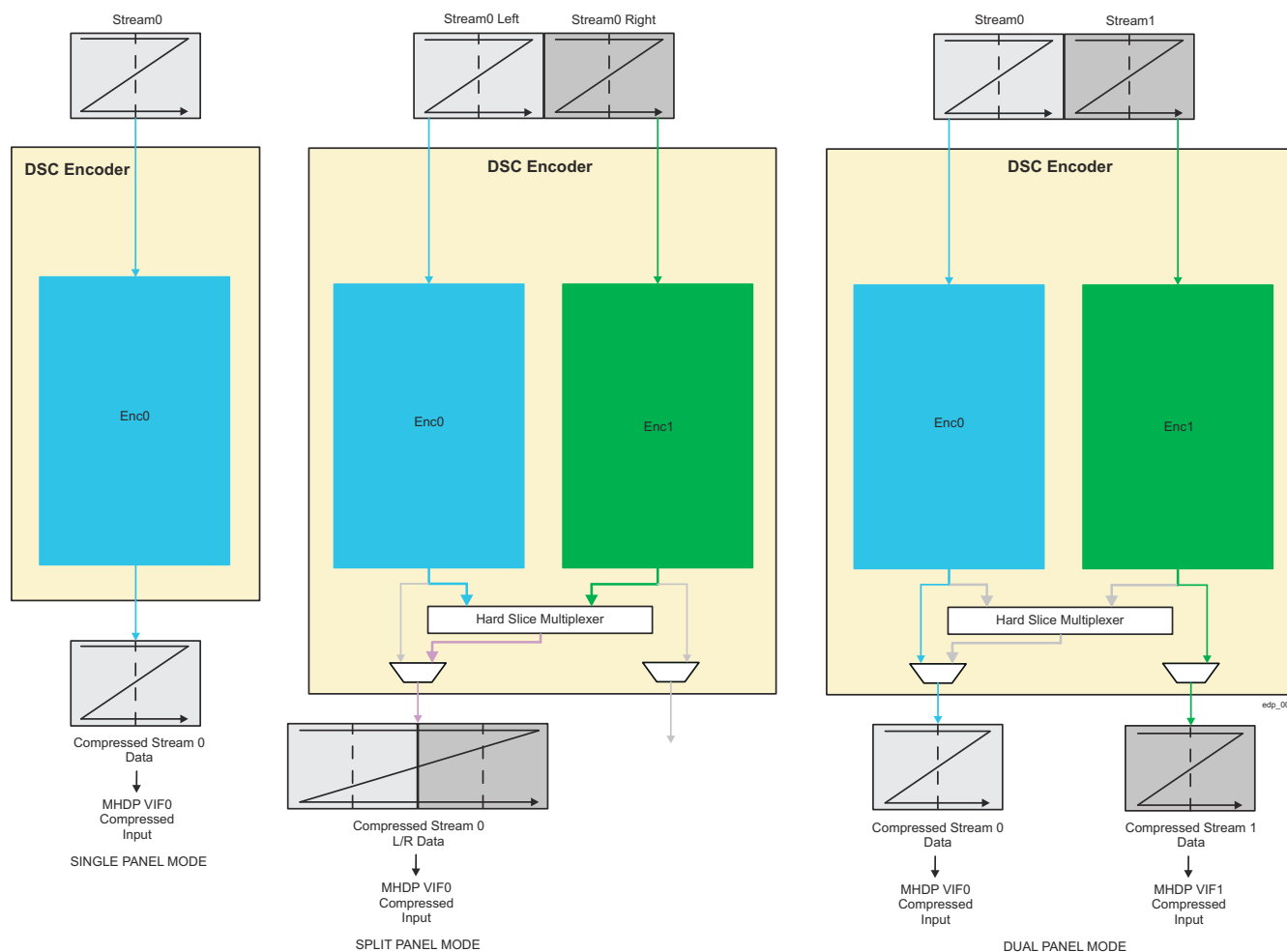
#### 12.6.6.3.1.2 Usage Models for EDP

[Table 12-1508](#) summarizes the usage models supported by DSC instantiated in the MHDPTX\_TOP subsystem.

**Table 12-1508. EDP DSC Usage Models**

Usage Models			
	Single Panel Mode (SST/MST)	Split Panel Mode (SST/MST)	Dual Panel Mode (Only in MST)
ENC0	Stream 0 Full Frame	Stream 0 Left Frame	Stream 0
ENC1	Not Used	Stream 0 Right Frame	Stream 1

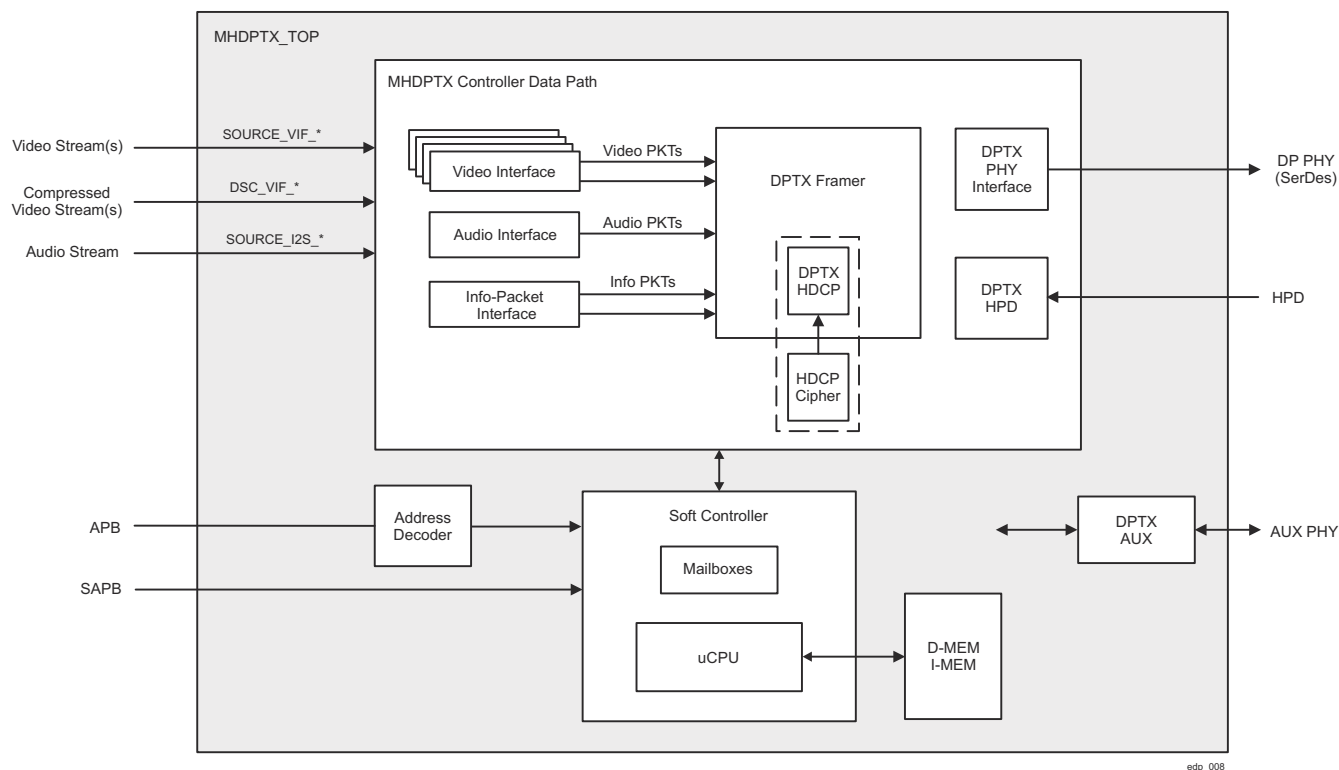
The shaded boxes in [Table 12-1508](#) indicate the encoder slice enabled for EDP usage. The active paths for these two modes are shown on [Figure 12-1122](#).



**Figure 12-1122. EDP DSC Single Input/1-Transport Link Mode and Split Panel/Dual Panel Mode**

#### 12.6.6.3.2 Display Port Transmitter Controller (MHDPTX Controller)

Figure 12-1123 shows a simplified block diagram of the EDP Display Port Transmitter Controller (MHDPTX) that is included in the EDP subsystem.



**Figure 12-1123. EDP Display Port Transmitter Controller Functional Block Diagram**

n = 0 to 3

#### 12.6.6.3.2.1 EDP Transmitter Controller Mode Configurations

This section describes the configuration mode input signals of the MHDPTX\_TOP module.

##### SOURCE\_SECURE\_MODE

This mode configuration signal sets the "source\_secure\_mode" of the MHDPTX's APB interface to provide additional security control on the host interface. The main APB interface in "debug" mode can be used to access almost all internal registers as well as IIRAM/DRAM of the internal uCPU. This signal must be set to 0 to load FW during boot time. By setting SOURCE\_SECURE\_MODE to 1, this functionality can be disabled to only allow mailbox and basic control register accesses.

In EDP, this signal is mapped to a wrapper level configuration register within the EDP\_CFG region: EDP\_DPTX\_IPCFG[0] APB\_SECURE\_REG\_BLOCK\_EN. If tempering of this register should be blocked, the EDP\_CFG region should be firewalled to grant access only to a secure host only.

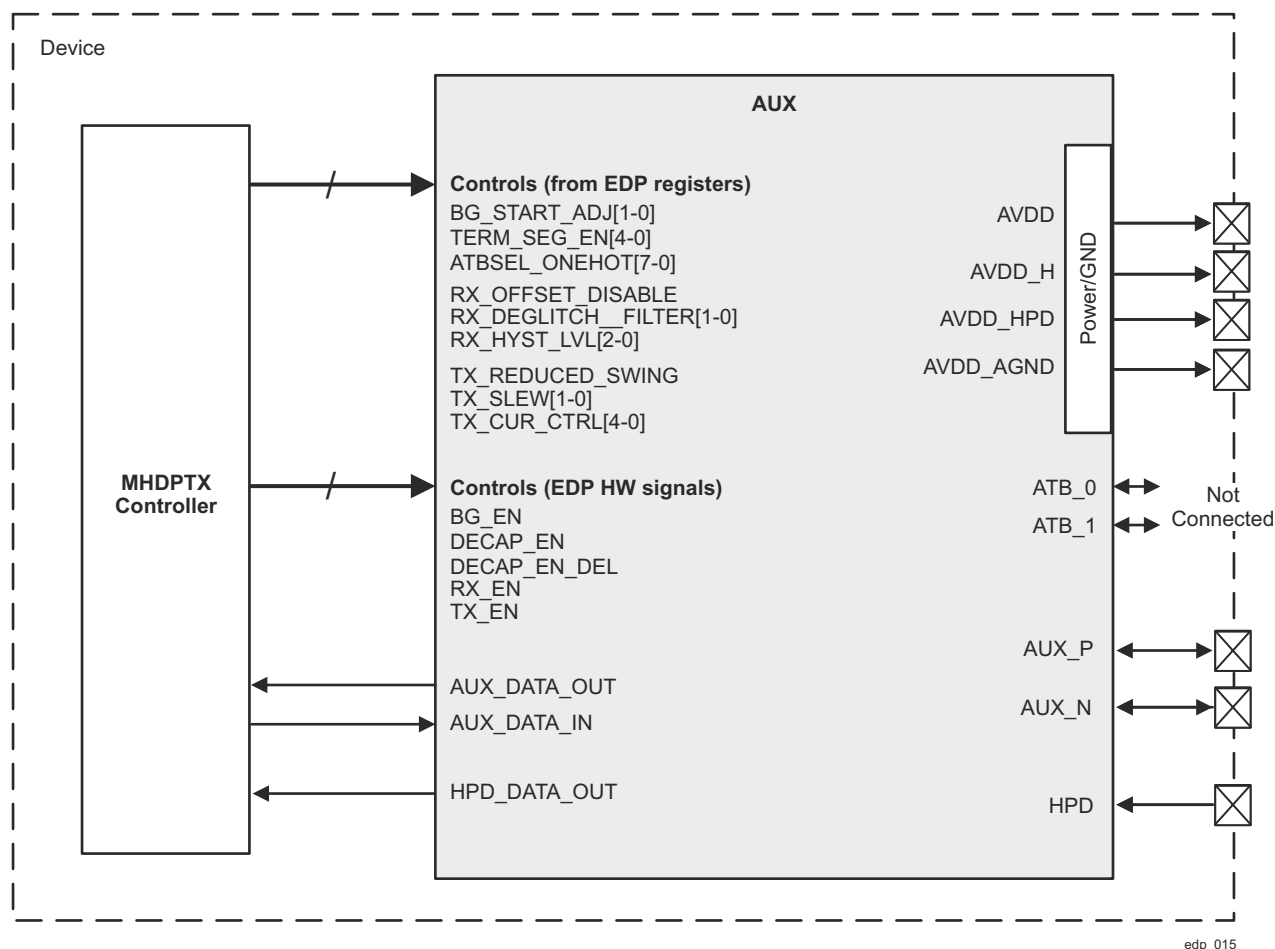
If the DP firmware regions are needed to be protected (to prevent either corruption or altering of the firmware), a secure host can change this setting to 1 after initializing the firmware memories. Then, a non-secure host can come and take the CPU out of reset and enable the DP function without an ability to alter the firmware.

##### SOURCE\_CRYPTO\_DIS

This mode configuration signal is used to completely disable HDCP functionality by disabling Crypto module (module responsible for keys encryption during authentication) . Refer to *EDP eFuse Tie-Off*, for eFuse tie-off information.

#### 12.6.6.4 EDP AUX\_PHY Interface

The AUX PHY module is required for each Display Port interface. The AUX\_PHY module is integrated with the EDP controller as shown on [Figure 12-1124](#). The AUX PHY has no configuration interface. All configuration and control signals are supplied from EDP\_CORE\_APB memory-mapped registers.



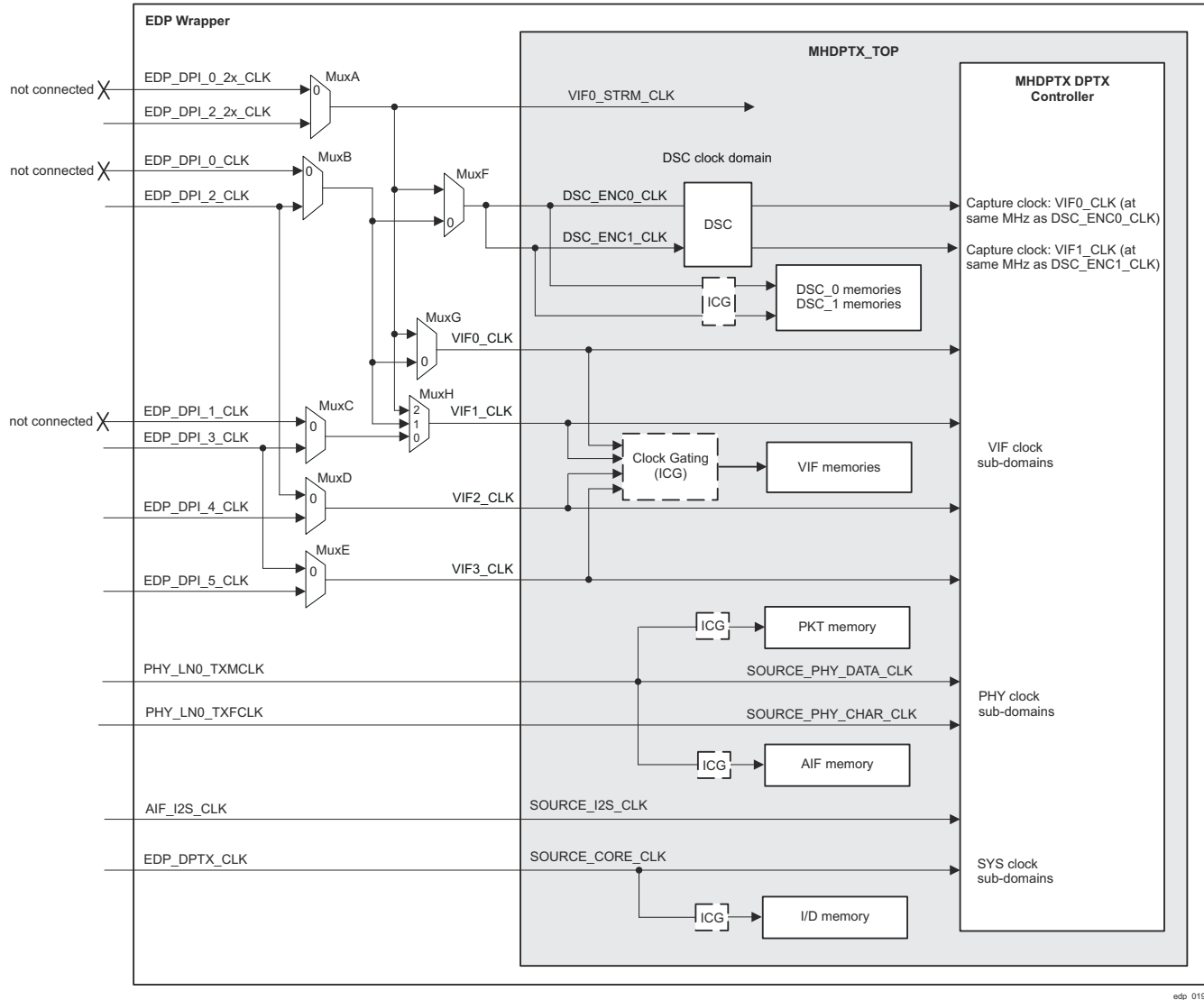
**Figure 12-1124. EDP Interface to AUX PHY**

The configurations and controls in [Figure 12-1124](#) are specified through EDP\_CORE\_AUX\_CONFIG\_P and EDP\_CORE\_AUX\_CTRL\_P registers.

#### 12.6.6.5 EDP Clocks

##### 12.6.6.5.1 Clock Diagram

[Figure 12-1125](#) shows the input clocks of eDP and muxing on the DPI input clocks to match the pixel data selection.



**Figure 12-1125. EDP Clock Diagram**

#### 12.6.6.5.1.1 DPI Interface Clock Sourcing

In default (non-DSC enabled) modes, the EDP performs the matching pixel data clock selection (MuxA - MuxE) based only on the VIF input source selection (EDP\_DTPX\_SRC\_CFG[7-4] VIF\_n\_SEL register fields) settings.

For VIF0\_CLK, the source selection in non-DSC enabled mode comes from the output of MuxA (EDP\_DPI\_0\_2x\_CLK or EDP\_DPI\_2\_2x\_CLK). These clocks are full pixel data frequency clocks.

When DSC is enabled, the DSC\_ENC0\_CLK and DSC\_ENC1\_CLK clocks are sourced (MuxF) from:

- Full pixel data frequency clock (MuxA selected clock) - when DSC is configured in a non-split panel mode (EDP\_DPTX\_DSC\_CFG[1-0] MODE\_SEL != 0h AND EDP\_DPTX\_DSC\_CFG[2] SPLIT\_PANEL\_EN = 0h)
- Half pixel data frequency clock (MuxB selected clock) – when DSC is configured in the split panel mode (EDP\_DPTX\_DSC\_CFG[1-0] MODE\_SEL = 2h AND eDP\_DPTX\_DSC\_CFG[2] SPLIT\_PANEL\_EN = 1h)

VIF0\_CLK and VIF1\_CLK selection goes through additional muxing (MuxG/MuxH) to match the DSC encoder clocks.

#### 12.6.6.5.1.2 Memory Clock Gating

Memory clock gating is handled at the wrapper based on the scheme described below.

Functional mode clock enable (functional clock gating – for memory):

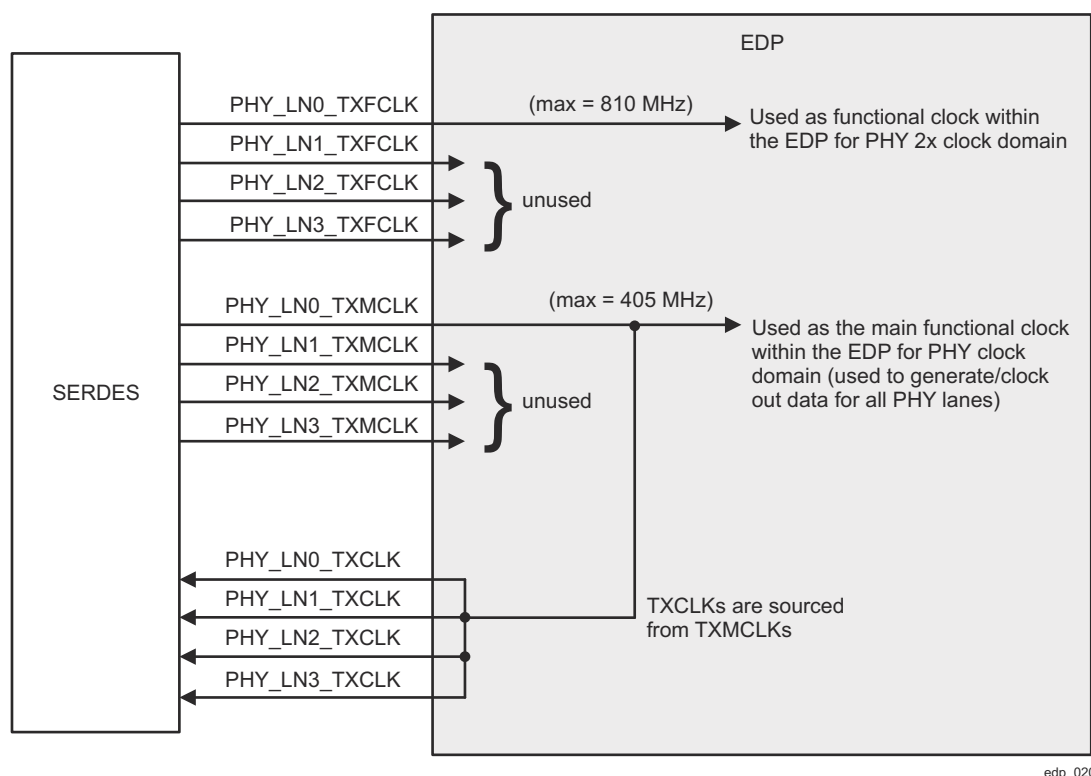
- DPTX firmware memory (I/D mem) clocks are enabled when `EDP_DPTX_IPCFG[1] FW_MEM_CLK_EN = 1`.
- Memory clocks for all DPTX data memories in the normal functional mode are only enabled when the corresponding source (video/audio/pif) is enabled per `DPTX_SRC_CFG`.
  - Clocks for the video stream memories (vif\_mem) are enabled whenever the stream is enabled (`EDP_DPTX_SRC_CFG[3-0] VIF_n_EN`).
  - Clocks for the packet memories (pkt\_mem) are enabled whenever the packet stream interface is enabled (`EDP_DPTX_SRC_CFG[3-0] VIF_n_EN`).
  - Similarly clock for the audio buffer (aif\_mem) is enabled whenever the audio stream is enabled (`EDP_DPTX_SRC_CFG[16] AIF_EN`).
- Clocks for the DSC memories (DSC\_ENC0 and DSC\_ENC1 mem) are enabled based on the ENC0/ENC1 usage (`EDP_DPTX_DSC_CFG[1-0] MODE_SEL`). When `EDP_DPTX_DSC_CFG[1-0] MODE_SEL` is not 0 (that is, at least one DSC encoder is enabled), the wrapper enables `VIF_n_MEM` clock based on the DSC usage

Functional mode clock enable override:

- Clocks for all ECC memories can be forced to turn on (that is, functional clock gating bypassed) during ECC diagnostic access mode by setting `EDP_ECC_MEM_CFG[0] CLK_EN = 1`.

#### 12.6.6.5.1.3 PHY Clock Connections

There are multiple clocks from/to the physical interface. [Figure 12-1126](#) shows clocks used for the PHY clock domain logic (TXMCLK/TXFCLK) and for the return data clock (TXCLK). Not shown in the diagram are RX mode input clocks unused by EDP (TX only mode) – `PHY_INn_RXCLK/RXFCLK/REFCLK`.



**Figure 12-1126. EDP PHY Clock Connections**

#### 12.6.6.5.2 Clock Groups

Following are the major groups of clocks in the EDP subsystem:

- System clock – This is the VBUSP configuration clock. It is used to source the clock for all internal configuration interfaces (VBUSP and PAB) and the core logic of the MHDPTX controller.
- DPI clocks - These are video stream source clocks. These can be sourced either directly from PLL or DSS (that is, DPI\_CLK outputs, DSS sourced mode).
- PHY I/O clocks - These are clocks used to transmit display port raw data to the SERDES or the clock used to receive audio data from MCASP.

Table 12-1509 describes clocks at the EDP boundary.

**Table 12-1509. EDP Clocks**

Clock	Freq Min	Freq Max	Source	Description
EDP_DPTX_CLK	125 MHz	125 MHz	System PLL	Functional clock (all VBUSP/APB/DPTX_core logics are clocked by this clock) Valid Freq: 100 – 200 MHz (125 MHz chosen to be 1/4x of CBASS 500 MHz clock)
EDP_DPI_2_2x_CLK	25 MHz	600 MHz	Video PLL or DSS EDP_DPI_2x_CLK	DPI Streaming Clock – Typically 1x of the corresponding EDP_DPI_2_CLK. But, in DSC split-panel mode, 2x of the EDP_DPI_2_CLK
EDP_DPI_2/3/4/5_CLK	25 MHz	600 MHz	Video PLL or DSS EDP_DPI_CLK	DPI Clock – pixel clock of the EDP_DPI_DATA bus. In DSS-sourced mode, EDP_DPI_n_CLK is the pixel clock of the EDP_DPI_n_DATA bus. In PLL-sourced mode, the pixel clock needs to be internally muxed between EDP_DPI_2_2x_CLK and EDP_DPI_2_CLK depending on the EDP_DPI_DATA type (split panel data or not).
EDP_AIF_I2S_CLK	1 MHz	125 MHz	MCASP	Audio I2S Clock (192 kHz clock)
PHY_LN0_TXMCLK (SOURCE_PHY_DATA_CLK)	81 MHz	405 MHz	DP-SERDES (PHY) PLL	PHY Data clock (1/2 Char Clock)
PHY_LN0_TXFCLK (SOURCE_PHY_CHAR_CLK)	162 MHz	810 MHz	DP-SERDES (PHY) PLL	PHY Char Clock

#### 12.6.6.6 EDP Resets

The EDP subsystem has one synchronous active low reset input. The entire subsystem is reset by this reset. All resets for different clock domains (both synchronous and asynchronous) are generated internally using this reset.

All resets within the MHDPTX\_TOP are asynchronous. Each clock's sub-domain resets are synchronized with respective software reset.

#### 12.6.6.7 EDP Interrupt Requests

The interrupt signals generated by the EDP subsystem and propagated to SoC level are shown in Table 12-1510.

**Table 12-1510. EDP Subsystem Interrupts**

Interrupt	Description
EDP_INTR[3:0]	EDP Controller Interrupt Event (Functional)
EDP_INTR_ASF[6:0]	EDP Controller ASF Error Interrupt

#### 12.6.6.7.1 EDP\_INTR Interrupt Description

Table 12-1511 provides details on the EDP\_INTR[3:0] interrupt lines. Each of the interrupts listed in Table 12-1511 has a corresponding set of memory-mapped interrupt registers (status/mask/clear).

**Table 12-1511. EDP\_INTR Interrupts**

Interrupt Bit	Source	Description	Status Register	Mask Register	Clear Register
[3]	DSC	DSC Enc1 interrupt event detected	EDP_CORE_ENC1_IN_T_STAT_P	EDP_CORE_ENC1_IN_T_MASK_P	EDP_CORE_ENC1_IN_T_CLR_P
[2]		DSC Enc0 interrupt event detected	EDP_CORE_ENC0_IN_T_STAT_P	EDP_CORE_ENC0_IN_T_MASK_P	EDP_CORE_ENC0_IN_T_CLR_P
[1]	MHDPTX Controller	DPTX SIRQ - Secure APB domain interrupt event. Set when an interrupt in the status register is reported.	EDP_CORE_APB_STA_TUS_S	EDP_CORE_APB_INT_MASK_S	-
[0]		DPTX PIRQ - General DPTX interrupt event. Set when an interrupt in the status register is reported.	EDP_CORE_APB_INT_STATUS_P	EDP_CORE_APB_INT_MASK_P	-

### 12.6.6.7.2 EDP\_INTR\_ASF Interrupt Description

Table 12-1512 provides details on the EDP\_INTR\_ASF[6:0] interrupts lines. Each of the interrupts listed in Table 12-1512 has corresponding set of memory-mapped interrupt registers (status/mask/clear).

**Table 12-1512. EDP\_INTR\_ASF Interrupts**

Interrupt Bit	Source	Description	Status Register	Mask Register	Clear Register
[6]	ECC_AGGR_DSC	ECC Aggregator Uncorrected Error Interrupt	EDP_ECC_DSC_DED_STATUS_REG0	EDP_ECC_DSC_DED_ENABLE_SET_REG0	EDP_ECC_DSC_DED_ENABLE_CLR_REG0
[5]	ECC_AGGR_PHY	ECC Aggregator Uncorrected Error Interrupt	EDP_ECC_PHY_DED_STATUS_REG0	EDP_ECC_PHY_DED_ENABLE_SET_REG0	EDP_ECC_PHY_DED_ENABLE_CLR_REG0
[4]	ECC_AGGR_CORE	ECC Aggregator Uncorrected Error Interrupt	EDP_ECC_CORE_DED_STATUS_REG0	EDP_ECC_CORE_DED_ENABLE_SET_REG0	EDP_ECC_CORE_DED_ENABLE_CLR_REG0
[3]	DSC	ASF Corrected interrupt detected in DSC	EDP_CORE_ENC_ASF_INT_STAT_P	EDP_CORE_ENC_ASF_INT_MASK_P	EDP_CORE_ENC_ASF_INT_CLR_P
[2]		ASF Un-Corrected interrupt detected in DSC			
[1]	MHDPTX Controller	ASF Corrected (NonFatal) event detected in MHDPTX Controller. Set if non-fatal error occurs.	EDP_CORE_ASF_INT_STATUS <sup>(1)</sup>	EDP_CORE_ASF_INT_MASK <sup>(1)</sup>	-
[0]		ASF Un-corrected (Fatal) event detected in MHDPTX Controller. Set when fatal error occurs.			

(1) Each interrupt can be individually defined as fatal or non-fatal in the EDP\_CORE\_ASF\_FATAL\_NONFATAL\_SELECT register.

### 12.6.6.8 EDP Embedded Memories

#### 12.6.6.8.1 MHDPTX Controller Memories

**Table 12-1513. EDP MHDPTX Controller Memories**

Name	ECC	Description
IRAM	Yes	uCPU Instruction Memory. Contains uCPU FW, loaded by host during boot time.



**Table 12-1513. EDP MHDPTX Controller Memories (continued)**

Name	ECC	Description
DRAM	Yes	uCPU DATA Memory. Contains uCPU data-memory, mailbox for communication between uCPU and host processor and EDID segment. In DisplayPort mode, it contains DPCD registers and AUX mailbox. In HDMI mode, it contains SCDC standard registers. When HDCP is supported, it contains keys and protected information.
PKT_MEM_n (n = 0 to 3)	Yes	Source Packet Memory. Contains up to 16 info-frames. Accessed by host processor (over APB) for write and by HD Display TX Controller for read.
VIF_MEM_n (n = 0 to 3)	No (No ECC for the video buffer)	Video Memory. Elastic buffer, compensating the difference between the pixel clock and the line clock.
AIF_MEM	Yes	Audio Memory. Buffer audio samples during H active period.

#### 12.6.6.8.2 DSC Memories

**Table 12-1514. EDP DSC Memories**

Name	ECC	Description
ENC0/1_LB	Yes	Line Buffer RAM. Stores the reconstructed pixels that will be re-used for next line processing.
ENC0/1_OB0	Yes	Output Buffer RAM. Holds (as FIFO) results of encoder slices.
ENC0/1_SSM_S	Yes	Sub-stream Mux Size RAM. Holds context information required to generate the proper order of the mux words.
ENC0/1_SSM_D	Yes	Sub-stream mux balance FIFO RAM. Holds stream data for re-ordering.

#### 12.6.6.8.3 ECC Aggregation

The tasks of ECC detection/correction of 'ECC enabled' memories are handled within the MHDPTX Controller. The EDP wrapper provides 'ECC error injection' using a combination of ECC\_Wrapper (to access the memory) and ECC\_Aggregator (to allow system to access the ECC injection logic). An ECC\_Aggregator can access multiple ECC\_Wrappers of the same clock group. [Table 12-1515](#) shows how the ECC memories (and the EDC\_CTRL for Parity Inv) in the EDP are grouped to three ECC aggregators.

**Table 12-1515. EDP ECC\_Aggregator (Based on Clock Group)**

Memory / Logic	ECC Vector_ID	ECC_Aggregator
IRAM	0	ECC_AGGR_CORE
DRAM	1	
EDC_CTRL (Parity)	2	
PKT_MEM_0	0	ECC_AGGR_PHY
PKT_MEM_1	1	
PKT_MEM_2	2	
PKT_MEM_3	3	
AIF_MEM	4	
VIF_MEM_0	-	N/A (No ECC for the video buffer)
VIF_MEM_1	-	
VIF_MEM_2	-	
VIF_MEM_3	-	

**Table 12-1515. EDP ECC\_Aggregator (Based on Clock Group) (continued)**

Memory / Logic	ECC Vector_ID	ECC_Aggregator
ENC0_LB	0	ECC_AGGR_DSC
ENC0_SSM_S	1	
ENC0_SSM_D	2	
ENC0_OB0	3	
ENC1_LB	4	
ENC1_SSM_S	5	
ENC1_SSM_D	6	
ENC1_OB0	7	

Furthermore, each ECC\_Aggregator is connected to the VBUSP\_CFG via the main CBASS SCR (with asynchronous bridge). All ECC aggregator bound VBUSP\_CFG transactions are mapped to a separate RSEL (in EDP case, RSEL=3), see [Figure 12-1127](#).

Each ECC\_Aggregator and the asynchronous VBUSP interface of the aggregator are clocked using the read clocks used to read the memory, as follows:

- ECC\_AGGR\_CORE: EDP\_DPTX\_CLK
- ECC\_AGGR\_PHY: SOURCE\_PHY\_DATA\_CLK (PHY\_LN0\_TXMCLK)
- ECC\_AGGR\_DSC: EDP\_DPI\_0\_CLK or EDP\_DPI\_2\_CLK (selected by mux)

The clocks to some ECC memories may be disabled when the associated video or audio channel is not enabled. If none of the memories for an ECC aggregator is enabled (for example, DSC memory clocks are disabled, if the DSC is not enabled), then the clock for the aggregator is also disabled and the aggregator is disconnected from the CBASS (resulting in null return for any VBUSP access to the aggregator)

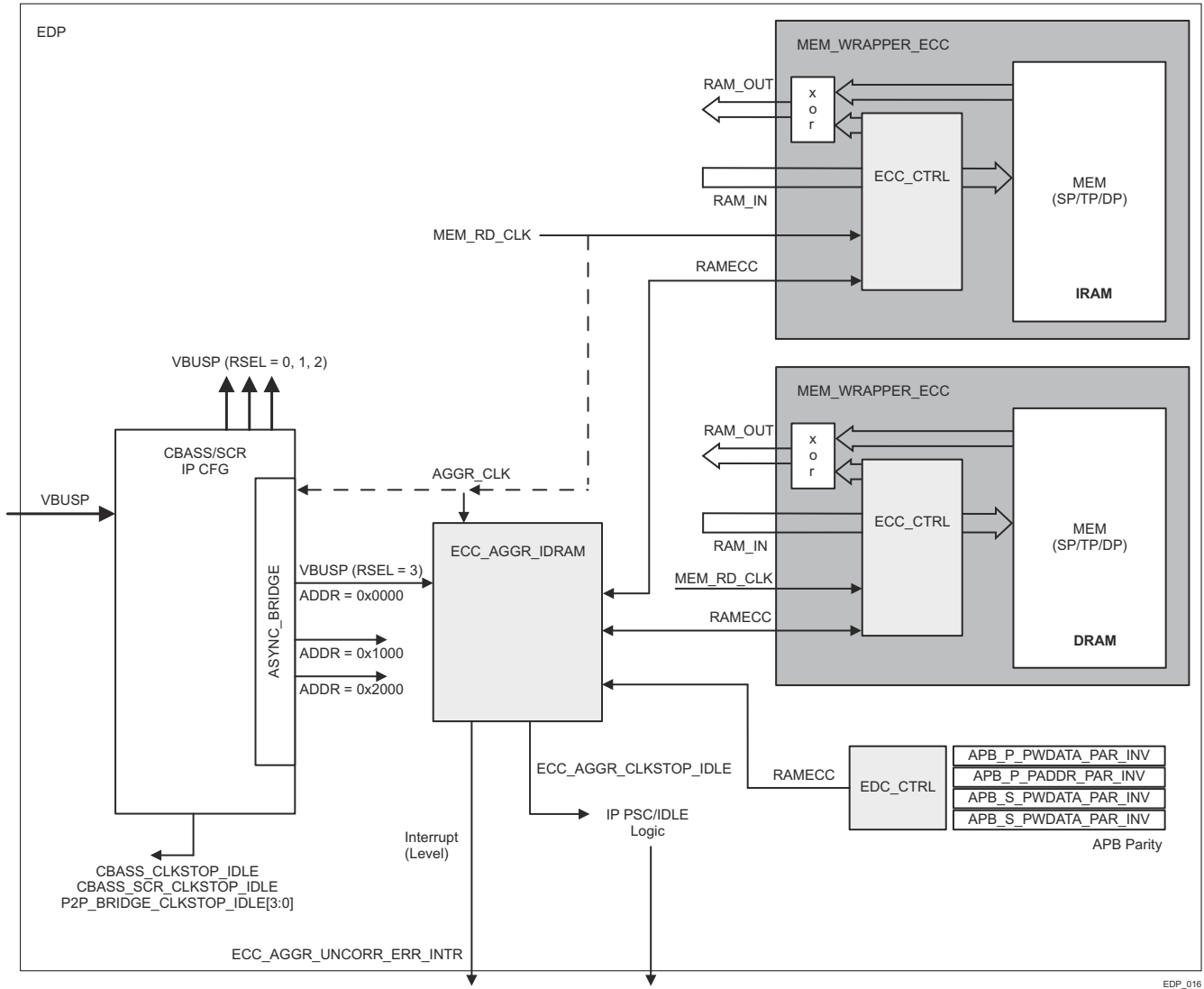
But, if any of the ECC memories connected to an ECC aggregator is enabled, the clock for the ECC aggregator must also be enabled. Any access to the ECC\_CTRL of the disabled memory will result in TIMEOUT error (interrupt generated) since the memory clock is not running.

If ECC diagnostics/access are needed for all ECC memories whether the memories are enabled or not, the ECC software can set the clock enable override configuration register - EDP\_ECC\_MEM\_CFG[0] CLK\_EN to bypass the clock gating.

Even with this bit set, the ECC\_CTRL connected to the ECC\_AGGR\_PHY aggregator may not get any clock if the PHY is not enabled and properly configured. The EDP\_PHY\_CLK\_STATUS[0] VALID status register tells whether the PHY data clock is running or not.

The DSC clock is sourced from the system video PLL and therefore it should be running. Therefore, the ECC software just need to ensure the clock gating is not active and/or set the ECC CLK\_EN override bit to get the clock to the memory.

The ECC\_CORE Aggregator clock should be running whenever the EDP is enabled. There is no clock gating to the IRAM/DRAM memories.



**Figure 12-1127. EDP ECC\_WRAP/ECC\_AGGR and PARITY\_INV Logic Connection**

#### 12.6.6.9 EDP Programmer's Guide

This chapter describes the programming guidelines for the MDPTX Controller and EDP PHY.

##### 12.6.6.9.1 EDP Controller Programming

###### 12.6.6.9.1.1 MHDPTX Register/Memory Regions

Table 12-1516 provides a summary of the bus interfaces and the MMR regions mapped to each interface in this module.

**Table 12-1516. EDP Memory Map Region - Firewall Secure Mode Settings**

Region Name	Firewall Secure Mode Settings <sup>(1) (2)</sup>
INTG_CFG_VP	Should be Secure
V2A_CORE_VP_REGS_APB	Secure (for Secure display) / Non-secure
V2A_S_CORE_VP_REGS_SAPB	Must be Secure always (HDCP configuration)
MHDPTX_WRAPPER_ECC_AGGR_CORE_CFG	Secure (for Secure display) / Non-secure
MHDPTX_WRAPPER_ECC_AGGR_PHY_CFG	Secure (for Secure display) / Non-secure

**Table 12-1516. EDP Memory Map Region - Firewall Secure Mode Settings (continued)**

Region Name	Firewall Secure Mode Settings <sup>(1) (2)</sup>
MHDPTX_WRAPPER_ECC_AGGR_DSC_CFG	Secure (for Secure display) / Non-secure

- (1) Firewall secure mode settings:
- The V2A\_S\_CORE\_VP\_REGS\_SAPB (secure APB) region must be restricted to secure host access only (for HDCP key protection)
  - The INTG\_CFG\_VP region should be set to secure host access only since this region controls the security of the DPI ports and controls the clock enables.
  - Other regions should be set to secure, if the display connected is to be secure.
- (2) Word-access only for RSEL (1 and 2) – accessing APB and SAPB regions of MHDPTX core. Any non-word access request will result in either rstatus=1 or sstatus=1.

Table 12-1517 shows the MHDPTX Controller register/memory regions that are accessible during various operational modes:

**Table 12-1517. EDP MHDPTX Controller APB/SAPB Memory Map and Access Modes**

Description	Address Base [19:0]	Register Bank	Direct access (x= APB (x= APB /SAPB)	Mailbox access (x= APB (x= APB /SAPB)	FW (uCPU) access	APB debug (Bootmode) access
Main configuration and Mailbox control registers. Each APB interface has its own set of these registers.	0x00000	APB_CFG	x			x
Source digital PHY control	0x00800	SOURCE_PHY		x	x	x
Clocks and Reset	0x00900	SOURCE_CAR		x	x	x
Clock Meters	0x00a00	CLOCK_METERS		x	x	x
Video Interface 0 control	0x00b00	SOURCE_VIF		x	x	x
Video Interface 1 control	0x00b20	SOURCE_VIF		x	x	x
Video Interface 2 control	0x00b40	SOURCE_VIF		x	x	x
Video Interface 3 control	0x00b60	SOURCE_VIF		x	x	x
DPTX Digital PHY	0x02000	DPTX_PHY		x	x	x
DPTX HPD	0x02100	DPTX_HPD		x	x	x
DPTX Framer	0x02200	DPTX_FRAMER		x	x	x
DPTX Stream	0x02200	DPTX_STREAM		x	x	x
DPTX Main control	0x02300	DPTX_GLBL		x	x	x
DPTX HDCP SM	0x02400	DPTX_HDCP		x	x	x
DPTX Auxiliary	0x02800	DP_AUX		x	x	x
DPTX Stream 0	0x03000	DPTX_STREAM		x	x	x
DPTX Stream 1	0x03080	DPTX_STREAM		x	x	x
DPTX Stream 2	0x03100	DPTX_STREAM		x	x	x
DPTX Stream 3	0x03080	DPTX_STREAM		x	x	x
HDCP Crypto	0x04000	Crypto		x	x	-
HDCP Cipher	0x05000	Cipher Accessible only through SAPB mailbox		x (SAPB)	x	-
memory	0x10000 ... 0x1ff00	IMEM			x	x
Data memory	0x20000 ... 0x2ff00	DMEM			x	x
Audio decoder	0x30000	SOURCE_AIF_DECODER	APB			x
SDP control stream 0	0x30800	SOURCE_PIF	APB			x
SDP control stream 1	0x30840	SOURCE_PIF	APB			x

**Table 12-1517. EDP MHDPTX Controller APB/SAPB Memory Map and Access Modes (continued)**

Description	Address Base [19:0]	Register Bank	Direct access (x= APB (x= APB /SAPB)	Mailbox access (x= APB (x= APB /SAPB)	FW (uCPU) access	APB debug (Bootmode) access
SDP control stream 2	0x30880	SOURCE_PIF	APB			x
SDP control stream 3	0x308c0	SOURCE_PIF	APB			x
Registers related with IPS configuration	0x30A00	IPS_REGS	APB			x
Fault reporting module	0x30B00	ASF	APB			x
DSC encoder	0x30C00 - 0x30F00	DSC	APB			x

**Access Modes:**

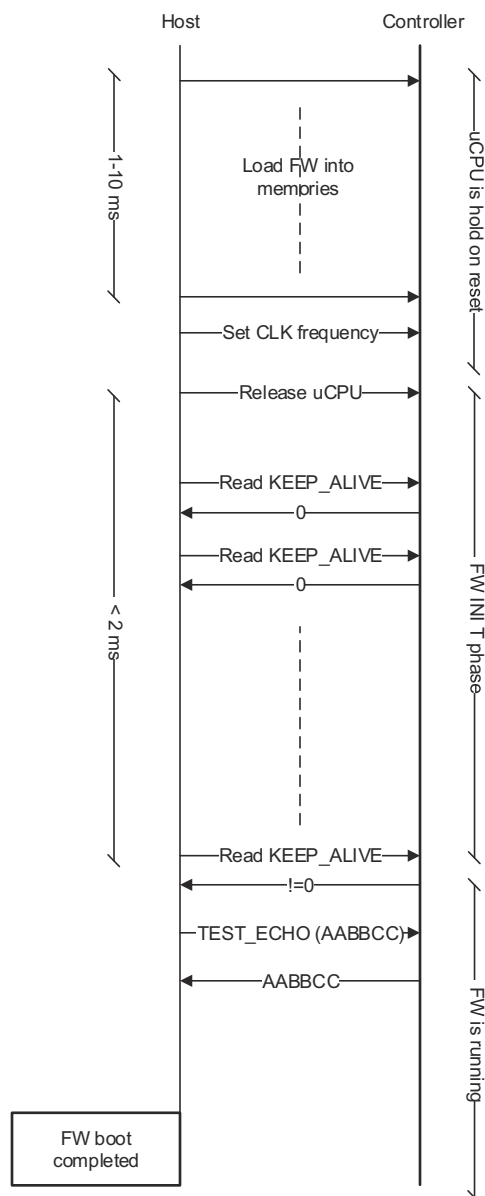
- APB Debug (Boot Mode) Access - After a hardware reset, MHDPTX comes up in this mode – allowing APB access to all regions including IMEM/DMEM (for FW download)
- Direct Access - Once the boot mode initialization is complete, APB\_CTRL register is cleared. This takes it out of the Boot mode and into the direct access mode in which only the mailbox control (APB\_CFG space) and regions marked as APB in the direct access column.
- Mailbox Access - While in Direct Access mode, all configurations of the DPTX for DP operation are done through the APB mailbox accesses. Secure transactions (secure refers to transactions to protect "secure" content) are done through the secure mailbox access via SAPB interface. The SAPB access is made through the secure EDP vbusp region (must be configured as such in the cfg Firewall)
- FW (uCPU) Access – This column indicates register/memory bank accessible by FW access.

**12.6.6.9.1.2 Boot Sequence**

After reset, the uCPU is disabled. The full MHDPTX Controller address space is accessible for the host processor for debugging purposes, and the I-MEM and D-MEM.

The host processor must perform the following:

- Load the FW into the memories.
- Enable the uCPU.
- Verify that FW is running as expected.



**Figure 12-1128. EDP MHDPTX Controller Boot Sequence**

#### 12.6.6.9.1.3 Setting Core Clock Frequency

The FW relies on a known core\_clock frequency for configuring time HW dividers and timeout counters.

Therefore, core\_clock frequency (SW\_CLK\_H, SW\_CLK\_I) must be configured by the host processor, prior to booting the FW.

#### 12.6.6.9.1.4 Loading Firmware

The FW binaries are provided in the hexadecimal format that is composed from two files IRAM0.DATA and DRAM0.DATA.

The IRAM0.DATA contains the instruction memory and the DRAM0.DATA contains the variables initialization.

The external host processor should load the FW into the I-MEM and D-MEM, and enable the uCPU. Accessing the I-MEM, D-MEM is enabled only after reset (after reset, the system is going into debug mode, after loading

FW and set EDP\_CORE\_APB\_CTRL\_P register to 0 system is going out from debug mode, and only reset can bring the system back to debug mode).

After loading the memories, the host processor should select the uCPU to be the master of the I-MEM and D-MEM memories, and enable the uCPU by writing 0x00 to the EDP\_CORE\_APB\_CTRL\_P register.

#### 12.6.6.9.1.5 FW Running indication

After loading the FW, the embedded FW takes approximately 2ms to complete the INIT phase and start running. The EDP\_CORE\_KEEP\_ALIVE\_P register specifies whether the FW is running.

**KEEP\_ALIVE** – The EDP\_CORE\_KEEP\_ALIVE\_P register is a counter initialized to 0 after reset and incremented by the FW on any scheduler loop.

When a valid FW is running, this register will be changed on every FW Scheduler loop (< 2ms). The host processor may use the mailbox channel immediately after reset, FW will response after the completion of INIT phase.

### Operational mode

After completion of boot sequence, the FW set the MHDPTX Controller in a standby mode. In the standby mode, it is ensured that no transaction is executed over the external interfaces (that is, AUX, PHY, Video).

Switching into active mode is done by executing the GENERAL\_MAIN\_CONTROL command.

#### 12.6.6.9.1.6 Software Events Handling

Software events are captured by the MHDPTX Controller FW, where the Event Value is stored in a designated area (in the D-MEM) and a bit is set by the FW in EDP\_CORE\_SW\_EVENTS0\_P register, see [Table 12-1518](#).

The host processor should poll the EDP\_CORE\_SW\_EVENTS0\_P register (cleared after read) and following a detection of event(s) the host processor should call the relevant command which returns the Event Value.

The Event Value holds the latest value associated with the event.

The EDP\_CORE\_SW\_EVENTS0\_P register is not cleared by the FW. Therefore, it is recommended to read this register (dummy read) to initialize the events to zero.

While any bit in EDP\_CORE\_SW\_EVENTS0\_P register is set, the MHDPTX Controller raises a hardware interrupt.

**Table 12-1518. EDP MHDPTX Controller Software Events**

Bit	Event	Description
31:8	RESERVED	-
7	HDCP_TX_IS_RECEIVER_ID_VALID	IP has an ID to check if it is valid, HDCP_TX_IS_RECEIVER_ID_VALID_REQ needs to be called.
6	HDCP2_TX_STORE_KM	IP has Km to store, HDCP2_TX_STORE_KM_REQ needs to be called.
5	HDCP2_TX_IS_KM_STORED	IP need to check if Km is stored, HDCP2_TX_IS_KM_STORED_REQ needs to be called.
4	HDCP_TX_STATUS	HDCP TX was changed, HDCP_TX_STATUS_REQ needs to be called.
3:1	RESERVED	-
0	DPTX_HPDP	HPD was changed, DPTX_READ_EVENT_REQUEST needs to be called.

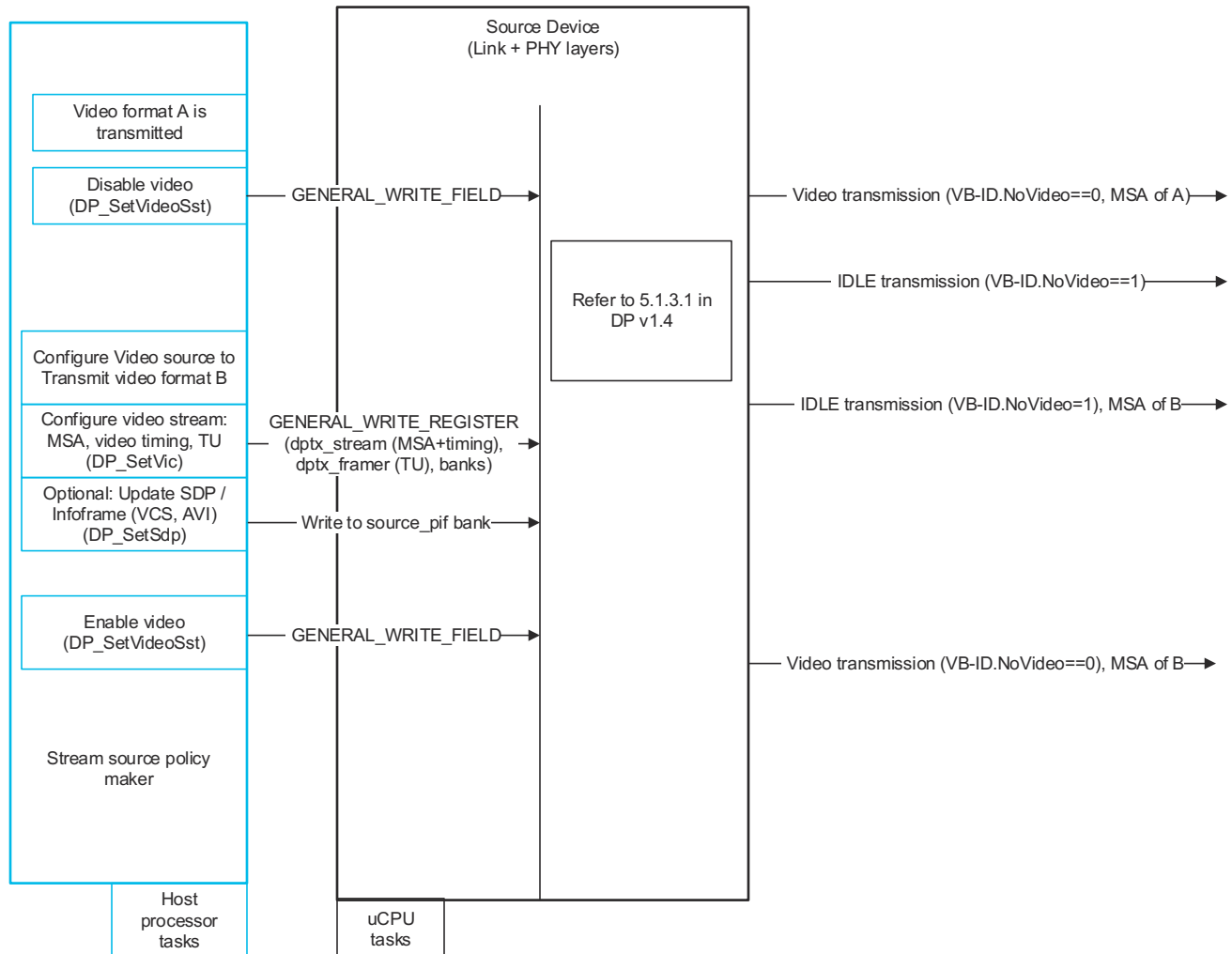
#### 12.6.6.9.1.7 DisplayPort Source (TX) Sequence

[Figure 12-1129](#) shows the expected host processor and uCPU tasks executed to start a video/audio transmission.

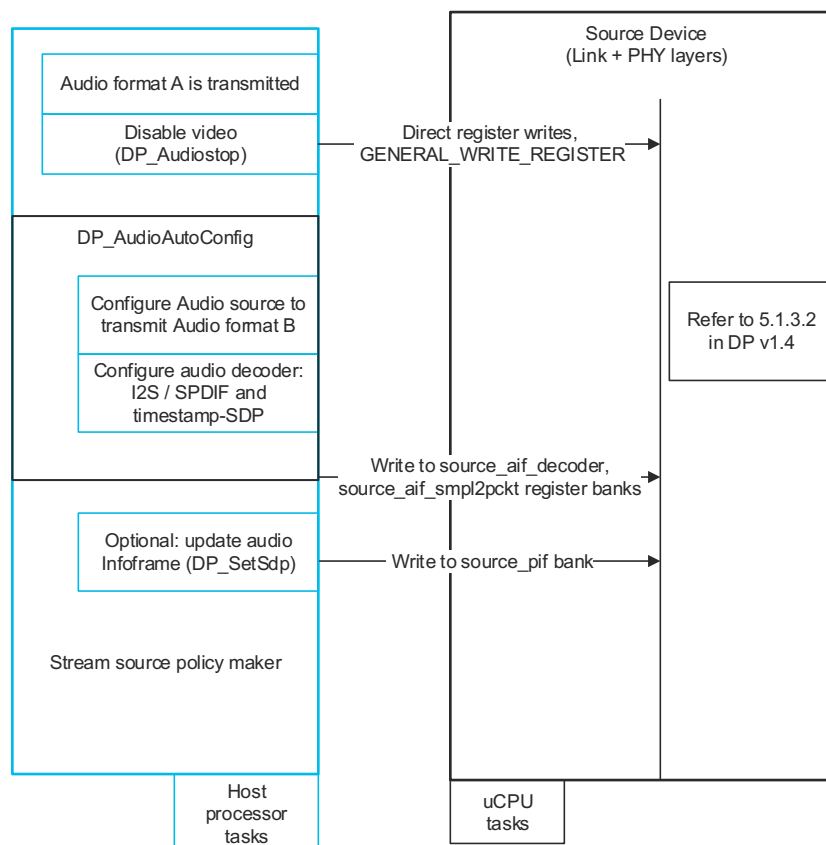


Copyright © 2024 Texas Instruments Incorporated





**Figure 12-1130. EDP Operation Sequence – DisplayPort Video Format Change**



**Figure 12-1131. EDP Operation Sequence – DisplayPort Audio Format Change**

#### 12.6.6.9.1.8 HDCP

The FW supports embedded HDCP crypto, with optional Km-key encryption.

##### 12.6.6.9.1.8.1 Embedded HDCP Crypto

In this mode, the HDCP functionality is fully supported by the MHDPTX Controller. The AKE phase and the corresponding cryptographic operations are handled by the FW.

Following the AKE phase, the FW initializes the stream cipher with the session key.

The host processor communicates with the MHDPTX Controller for:

- Configuring the capabilities
- Loading the HDCP keys (or certificate)
- Monitoring authentication status.

#### Note

Commands that carry secret information must be protected from tampering. Therefore, SAPB must be used to carry secured commands over mailbox.

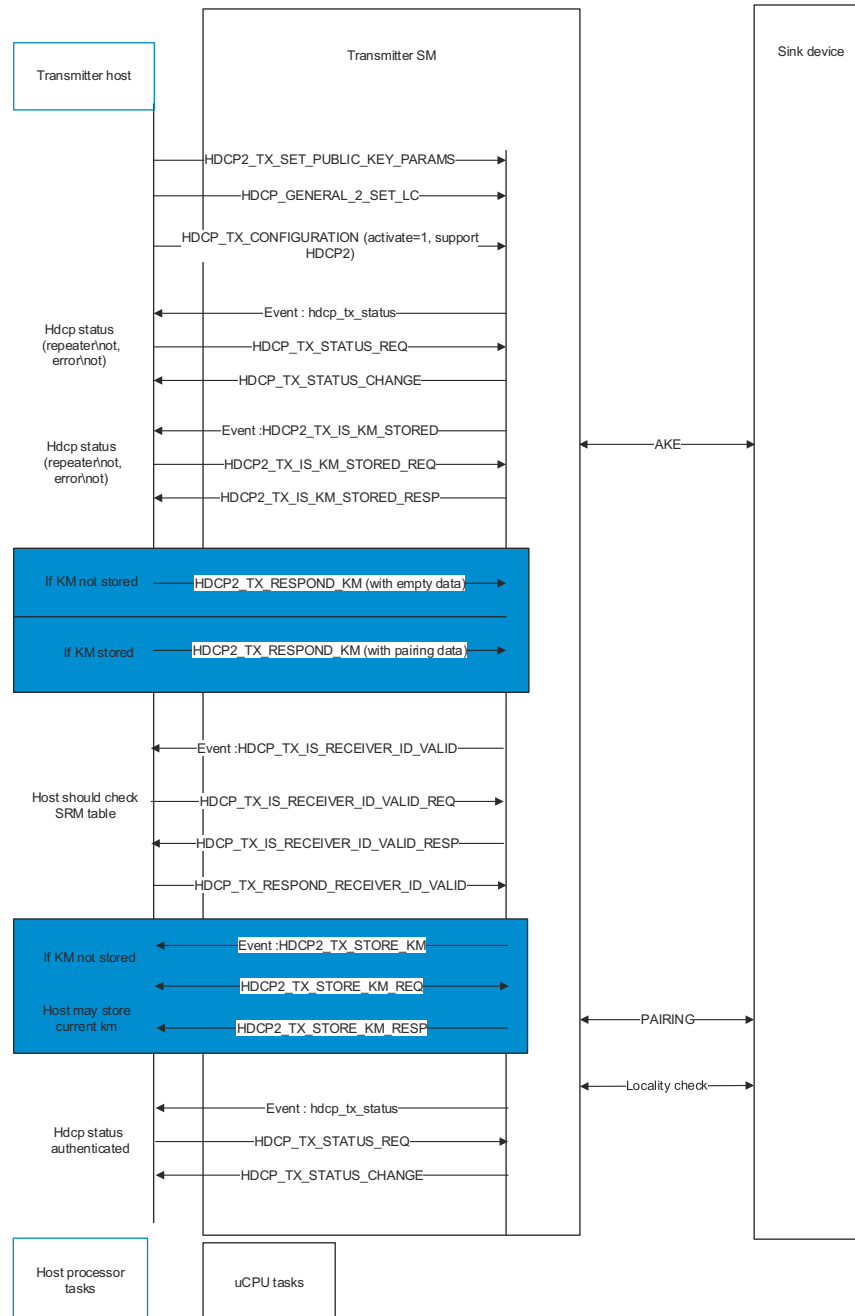
**Table 12-1519. EDP HDCP Commands With Secret Information**

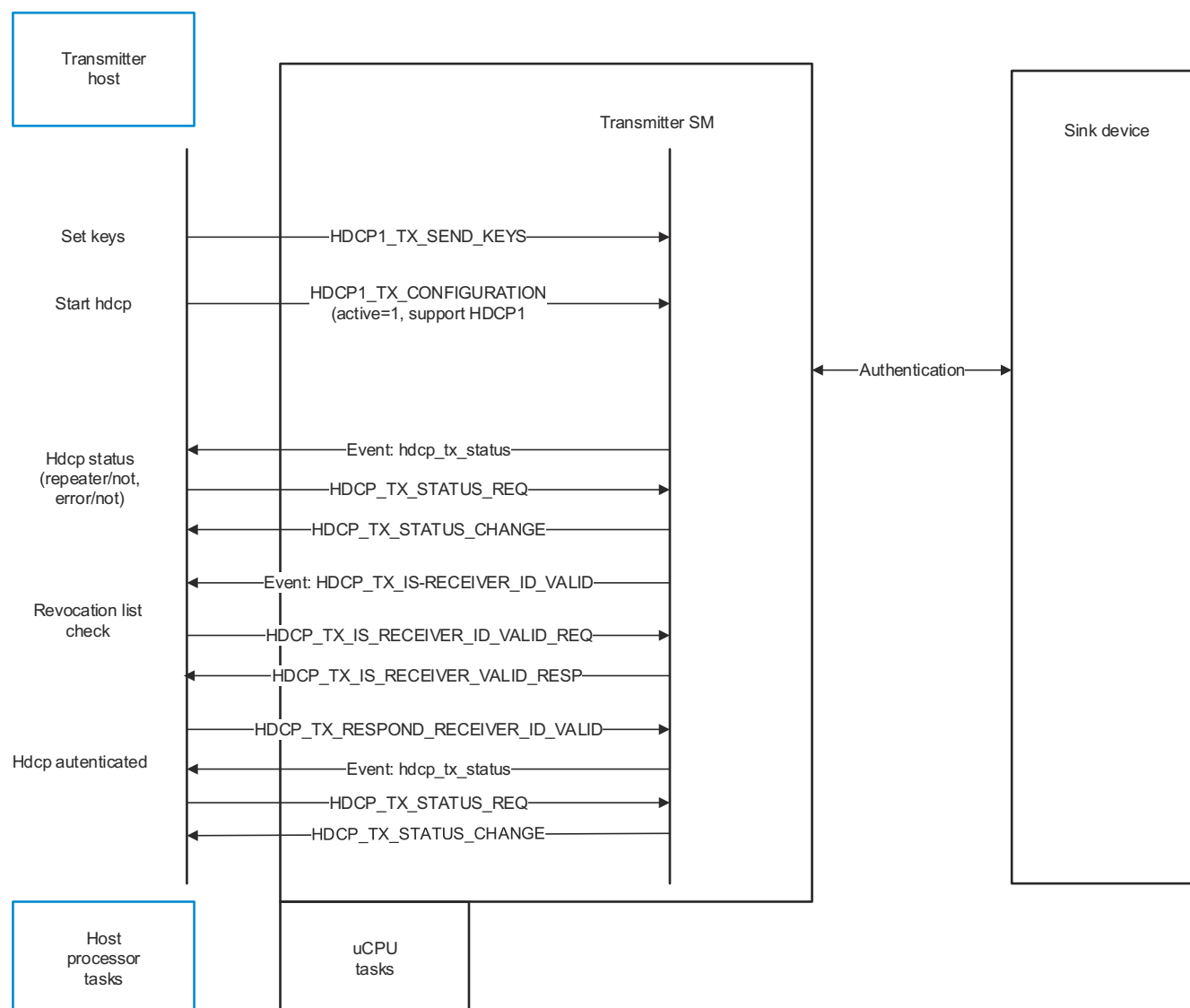
HDCP command	Description
HDCP_GENERAL_2_SET_LC	Refer to HDCP 2.2 Appendix A
HDCP_GENERAL_SET_SEED	External TRNG must be used for generating a true random seed number. Therefore, it is expected that the seed number will meet the high entropy level.
HDCP2_TX_RESPOND_KM	Refer to HDCP 2.2 Appendix A
HDCP1_TX_SEND_KEYS	Refer to HDCP 1.4 Appendix B

**Table 12-1519. EDP HDCP Commands With Secret Information (continued)**

HDCP command	Description
HDCP2_TX_STORE_KM	Refer to HDCP 2.2 Appendix A

Figure 12-1132 and Figure 12-1133 show the operation sequences for an embedded HDCP 2.2 and HDCP 1.4


**Figure 12-1132. EDP Operation Sequence – Embedded HDCP 2.2**



**Figure 12-1133. EDP Operation Sequence – Embedded HDCP 1.4**

### 12.6.6.9.1.8.2 Additional Security Features

#### 12.6.6.9.1.8.2.1 KM-Key Encryption

In order to provide higher security level an optional Km-key encryption is provided. If Km-key encryption is enabled while sending `HDCP_TX_CONFIGURATION` command, then HDCP controller will expect Master Key, 1.4 Device Keys and Global Constant. These keys will be decrypted using Km-key (loaded using `HDCP2_TX_SET_KM_KEY_PARAMS` command) by the controller's internal Firmware and used during Authentication and Key Exchange procedure. The Km-key encryption will also assure that when host reads Master Key it will receive it in encrypted form.

Enabling Km-key encryption affects the following commands or their parameters:

- `HDCP2_TX_RESPOND_KM/Km`
- `HDCP1_TX_SEND_KEYS/Transmitter Device Key`
- `HDCP2_TX_STORE_KM_RESP/Km`
- `HDCP_GENERAL_2_SET_LC`

#### **12.6.6.9.1.8.2.2 Cyphertext Stealing**

In order to use Km-key encryption for the 1.4 Device Keys and avoid changes in the API a Cyphertext Stealing technique is used. This technique allows to apply AES-ECB in Km-key encryption while 1.4 Device Key does not have size equal to multiple of 16.

#### **12.6.6.9.1.9 HD Display TX Controller**

##### **12.6.6.9.1.9.1 Info-Frame Handling**

Info-frames, which include meta-data (that is, auto type or video format), are initialized by the host processor.

The host processor directly initializes the PKT-MEM (which stores the info-frames). Initialization includes loading info-frame into PKT-MEM and updating the PKT LUT. Refer to DP\_SetSdp, DP\_RemoveSdp.

##### **12.6.6.9.1.9.1.1 EDID Handling**

The MHDPTX Controller issues command for reading EDID data from the remote side (sink). The EDID is sent to the host processor in a single block, that is, each block in one message. The MHDPTX Controller does not parse the EDID blocks. Refer to DP\_ReadEdid.

##### **12.6.6.9.1.9.1.2 Audio Control**

When changing an audio stream format, the host processor must follow a specific programming sequence by configuring the I2S decoder and audio info frame packet. Refer to DP\_AudioAutoConfig.

##### **12.6.6.9.1.9.1.3 Video Control**

The host processor need to configure the video timing before enabling the video stream over the VIF interface.

#### **12.6.6.9.1.10 DPTX TX Controller**

##### **12.6.6.9.1.10.1 Protocol over Auxiliary**

The firmware provides an interface for the AUX and I2C-over-AUX data transactions. The host processor may use it for direct DPCD (including the HDCP address range) or EDID access.

##### **12.6.6.9.1.10.2 PHY (Physical layer) Handling**

Separate PHY driver is used to initialize PHY.

DPTX PHY is handled by Core Driver, during link training (that is, Lane count, Link Rate, Pre-emphasis and Voltage Swing control).

#### **12.6.6.9.2 EDP PHY Wrapper Initialization**

EDP connects to the SERDES PHY wrapper (WIZ) using the RAW data mode. To enable this connection, the WIZ configuration registers in the SERDES PHY must be configured. For details on the SERDES PHY wrapper configuration, see Section 10-G SerDes Programming Guide.

The following is an example configuration for enabling 4-data lane RAW data mode for Display Port connection.

```
// SERDES_TOP_CTRL register bit-field settings:
```

```
// PMA_CMN_REF_CLK_MODE = 2'b10, PMA_CMN_REF_CLK_INT_MODE = 2'b10,  
PMA_CMN_REFCLK_DIV = 0, PMA_SUSPEND_OVERRIDE = 0
```

```
SERDES_TOP_CTRL register value = 32'h30000000
```

```
// SERDES_RST register bit-field settings:
```

```
// PHY_RESET_N = 0 (PHY in reset)
```

```
// PHY_EN_REFCLK = 0 (PHY reference clock output disabled)
```

```
// PLL1/0_REFCLK_SEL = 0 (for cmn_refclk_<p/m> select), 1 (for pma_cmn_refclk_int select)
```

```
// REFCLK_TERM_DIS = 1 (termination disabled)
```

```
// REFCLK_DIG_SEL = 1 (PMA common reference clock select: 0 for cmn_refclk_<p/m>, 1 –
pma_cmn_refclk_int)

// If the reference clock is sourced from internal digital input:

// PLL1_REFCLK_SEL = 1, PLL0_REFCLK_SEL = 1, REFCLK_TERM_DIS = 1, REFCLK_DIG_SEL = 1
SERDES_RST register value = 32'h39000000

// Alternatively, if the reference clock is sourced from <p/m> inputs:

// PLL1_REFCLK_SEL = 0, PLL0_REFCLK_SEL = 0, REFCLK_TERM_DIS = 0, REFCLK_DIG_SEL = 0
SERDES_RST alternative register value = 32'h00000000

// LANECTLx register bit-field settings:

// P0_ENABLE = 0, P0_FORCE_ENABLE = 1, P0_ALIGN = 1, P0_RAW_AUTO_START = 1,
P0_STANDARD_MODE = 0, P0_FULLRT_DIV = 0
LANECTL0 register value = 32'h70000000 // for lane 0

// P0_ENABLE = 1, P0_FORCE_ENABLE = 0, P0_ALIGN = 0, P0_RAW_AUTO_START = 0,
P0_STANDARD_MODE = 0, P0_FULLRT_DIV = 0
LANECTL1 register value = 32'h80000000 // for lane 1
LANECTL2 register value = 32'h80000000 // for lane 2
LANECTL3 register value = 32'h80000000 // for lane 3

// LANEDIVx register bit-field settings:

// P0_MAC_DIV_SEL0 = 7'b1, P0_MAC_DIV_SEL1 = 9'b1
LANEDIV0 register value = 32'h00010001 // for lane 0
LANEDIV1 register value = 32'h00010001 // for lane 1
LANEDIV2 register value = 32'h00010001 // for lane 2
LANEDIV3 register value = 32'h00010001 // for lane 3

// Set DIAG_REG register to 0
DIAG_TEST register value = 32'h0
```

### Note

The SERDES PHY reset is not driven by the EDP controller (that is, by writing to the EDP\_CORE\_PHY\_RESET\_P[8] PHY\_RESET register bit. Instead, the SERDES PHY reset is mapped to the SERDES\_RST[31] PHY\_RESET\_N register bit in the SERDES PHY wrapper. But, the EDP\_CORE\_PHY\_RESET[8] PHY\_RESET bit is still used in the EDP controller to disable the controller's output to the SERDES. Therefore, after reset, the EDP\_CORE\_PHY\_RESET[8] PHY\_RESET bit must be set to 1 (reset off). After this, the SERDES reset can be controlled strictly with the SERDES\_RST[31] PHY\_RESET\_N bit.

### 12.6.6.9.3 EDP PHY Programming

Table 12-1520 shows information on how to program the EDP PHY (SERDES).

**Table 12-1520. EDP PHY (SERDES) Programming Details**

General Information	
Standards / links / lanes supported	Display port and embedded display port / 2 links / 1, 2, or 4 lanes per link
Reference clock information and setup	

**Table 12-1520. EDP PHY (SERDES) Programming Details (continued)**

<b>General Information</b>	
External / internal reference clock selection	Set up external or internal reference clock.
Reference clock frequency selection	Set up the reference clock programming for the selected reference clock frequency
<b>PLL and high speed clocking information and setup</b>	
PLL 0, Mode 0	Display port link 0. Note, if only link 1 is used, PLL 0 must be programmed in the same as PLL 1.
PLL 0, Mode 1	Unused
PLL 1, Mode 0	Display port link 1. Note, if only link 0 is used, PLL 1 must be programmed in the same as PLL 0.
PMA common full rate and data rate clocks	Unused
PMA transceiver full rate and data rate clocks	Full rate and data rate clocks are used as specified in the PMA spec table Clock rates for supported standards.
<b>Top level pins</b>	
PMA: cmn_pll0_mode_sel (Driven by PHY internal logic)	1'b0
PMA: cmn_pll1_mode_sel (Driven by PHY internal logic)	1'b0
PHY: phy_ln{nn:00}_mode[1:0]	2'b11
PHY: phy_link_cfg_ln_{n:1}	1'b0 for slave lanes of each link 1'b1 for the master lane of each link
<b>PHY configuration specific registers</b>	
SERDES register PHY_AUTO_CFG_SPDUP, bit-fields [1] PHY_PLL_CFG_1 and [0] PHY_PLL_CFG_0	16'h0000 for single DP link configuration 16'h0002 for 2 DP link configuration
<b>Set PLL analog clock divider values</b>	
SERDES register CMN_PDIAG_PLL0_CLK_SEL_M0__CMN_PDIAG_PLL0_CTL_M0	Link 0: Program it as described
SERDES register CMN_PDIAG_PLL1_CLK_SEL_M0__CMN_PDIAG_PLL1_CTL_M0	Link 1: Program it as described
<b>Select analog high speed clock, and transceiver clock divider values</b>	
SERDES register XCVR_DIAG_HSCLK_DIV__XCVR_DIAG_HSCLK_SEL_j, bits [15:0] Link 0 Link 1	16'h0000 16'h0001
SERDES register XCVR_DIAG_HSCLK_DIV__XCVR_DIAG_HSCLK_SEL_j, bits [31:16]	Program it as described
<b>Select digital PLL clock and data rate divider values</b>	
SERDES register XCVR_DIAG_PLLDRC_CTRL__XCVR_DIAG_XDP_PWRI_S TAT_j Link 0 Link 1	16'h0001 16'h0009
<b>Protocol specific setup</b>	
Display port and embedded display	Program it as described

## 12.7 Camera Subsystem

This section describes the Camera Subsystem in the device.



## 12.7.1 Camera Streaming Interface Receiver (CSI\_RX\_IF)

The following sections describe the camera streaming receiver interface (CSI\_RX\_IF) modules in the device.

### 12.7.1.1 CSI\_RX\_IF Overview

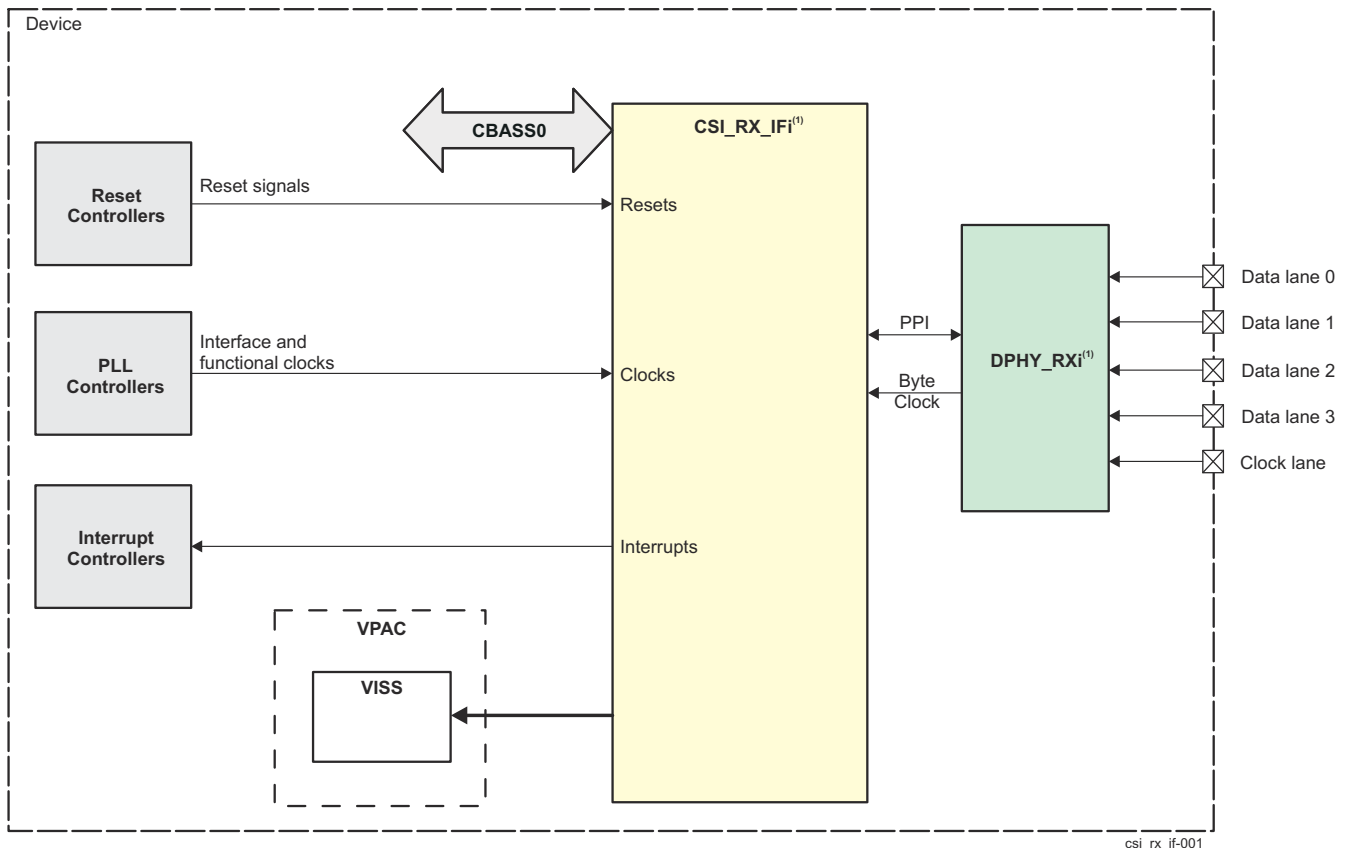
The integration of the CSI\_RX\_IF module allows the device to stream video inputs from multiple cameras to the image processing accelerator (VPAC) or to internal memory. The video input may also be retransmitted via the transmitter CSI (CSI\_TX\_IF) for debug and test purposes.

Table 12-1521 shows the CSI\_RX\_IF module allocation across device domains.

**Table 12-1521. CSI\_RX\_IF Modules Allocation Across Device Domains**

Module Instance	Domain		
	WKUP	MCU	MAIN
CSI_RX_IF0	-	-	✓
CSI_RX_IF1	-	-	✓

Figure 12-1134 shows the CSI\_RX\_IF module overview.



A. i = 0 to 1

**Figure 12-1134. CSI\_RX\_IF Module Overview**

#### 12.7.1.1.1 CSI\_RX\_IF Features

The CSI\_RX\_IF module supports the following features:

- Compliant to MIPI CSI v1.3
- Supports up to 16 virtual channels per input (partial MIPI CSI v2.0 feature)

- Data rate up to 2.5 Gbps per lane (wire rate)
- Supports 1, 2, 3, or 4 Data Lane connection to DPHY\_RX
- Programmable formats including YUV420, YUV422, RGB, Raw, and User Defined (over 25 different formats supported)
- Four independent (simultaneous) output streams:
  - Two VP 32-bit streams to VISS inputs of VPAC image processing accelerator:
    - 2x 16-bit pixels per clock cycle
    - One virtual channel and data type per port
    - Raw format only (8-16 bits)
    - 32bit, 2 pixels wide, elastic buffer mode
      - Data[15:0]: Pixel n (MSB zero padding)
      - Data[31:0]: Pixel n+1 (MSB zero padding)
      - Internal full flag (FF) based FIFO (2048x32)
      - VP clock asynchronous to CSI\_RX\_IF main clock. Crossing done internally.
  - One (up to 4 channels) PPI 16-bit pixel retransmission interface to CSI\_TX\_IF:
    - 2x 16-bit pixels per clock cycle
    - 32bit retransmission width
    - No external buffer
    - Raw format only (8-20 bits, partial MIPI CSI v2.0 feature)
    - CSI\_RX\_IF and CSI\_TX\_IF main clocks must be running at the same frequency and synchronous.
  - One (up to 32 Channels) DMA interface through a 128-bit PSI\_L connection to NAVSS for transfers to memory:
    - Byte packed (32x4) format, elastic buffer mode
    - Max rate 1 data cycle every 4 main clocks
    - ByteValid per byte in Last Data Phase (LDP)
    - 32 thread ID's supported (virtual channel & data type combinations); Flexible number of threads (32 Max)
    - Virtual channels and data types mapped via mmr to PSI\_L thread ID's
    - Internal FF based FIFO; RAM based buffer (2kx128)
- Functional and data path error interrupts
- ECC support

#### 12.7.1.1.2 CSI\_RX\_IF Not Supported Features

The CSI\_RX\_IF does not support the following features:

- MIPI CSI2 v2.0 scrambling
- MIPI CSI2 v2.0 optional no LP (low power state) between packets
- Cropping
- Pixel processing
- Virtualization
- YUV420-8 legacy not supported
- YUV420 interleave limitation. The entire frame must be sent.
- Full line buffer mode is not supported

#### 12.7.1.2 CSI\_RX\_IF Environment

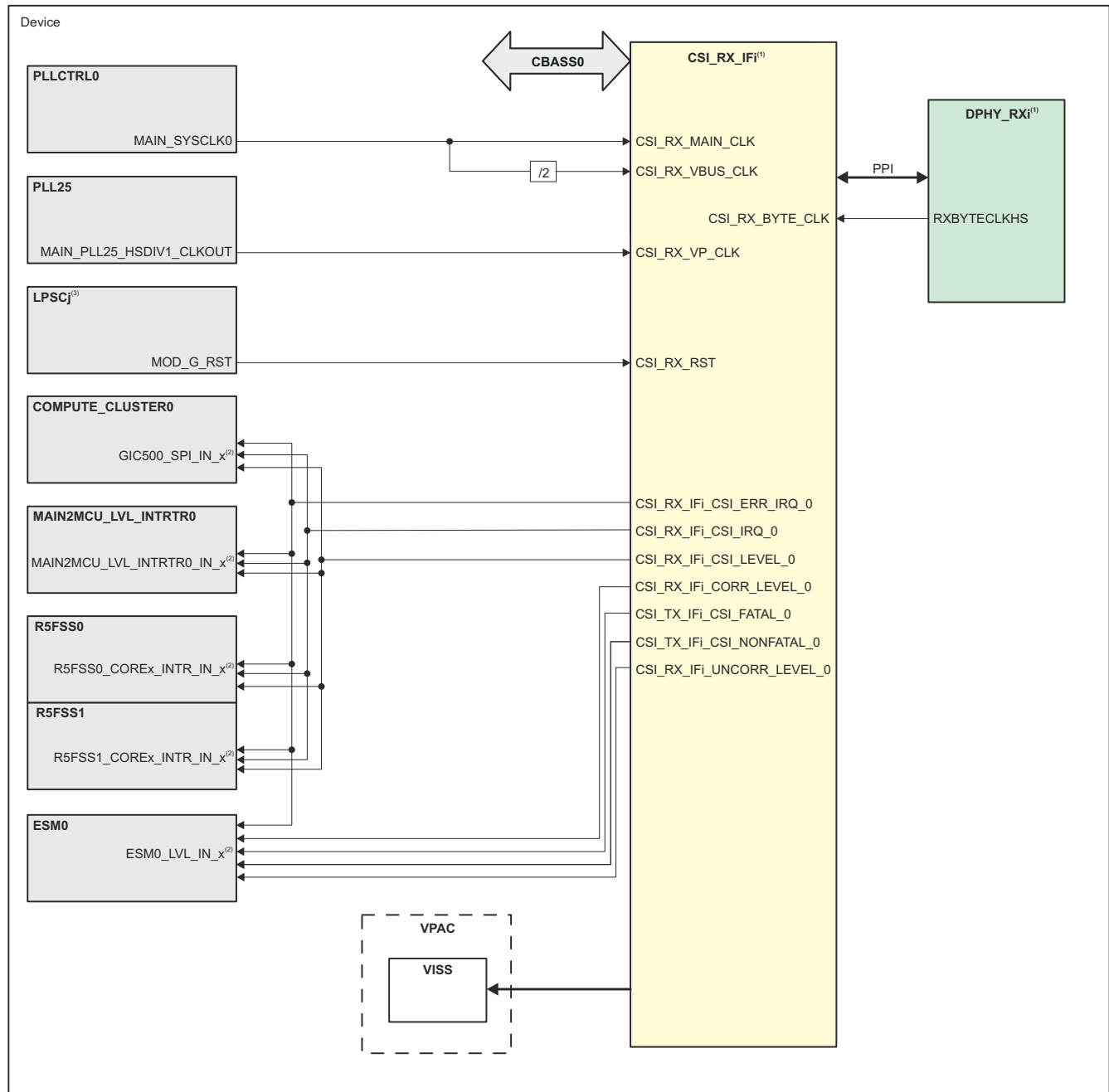
The CSI\_RX\_IF has no dedicated pins. At the device level, video inputs come from the DPHY\_RX, see [Section 12.7.2](#).

### 12.7.1.3 CSI\_RX\_IF Integration

This section describes the CSI\_RX\_IF integration in the device, including information about clocks, resets, and hardware requests.

#### 12.7.1.3.1 CSI\_RX\_IF Integration in MAIN Domain

There are several CSI\_RX\_IF modules integrated in the device MAIN domain. [Figure 12-1135](#) shows the integration of CSI\_RX\_IF modules.



csi\_rx\_if-002

- A.  $i = 0$  to  $1$
- B.  $x$  = Interrupt port index, see [Table 12-1524](#)

C. j = LPSC index, see [Table 12-1522](#)

**Figure 12-1135. CSI\_RX\_IF Integration**

[Table 12-1522](#) through [Table 12-1524](#) summarize the integration of CSI\_RX\_IF in the device MAIN domain.

**Table 12-1522. CSI\_RX\_IF Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
CSI_RX_IF0	PSC0	PD2	LPSC53	CBASS0
CSI_RX_IF1	PSC0	PD2	LPSC54	CBASS0

**Table 12-1523. CSI\_RX\_IF Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
CSI_RX_IF0	CSI_RX_MAIN_CLK	MAIN_SYSCLK0	PLLCTRL0	Main functional clock.
	CSI_RX_VBUS_CLK	MAIN_SYSCLK0 / 2	PLLCTRL0	The VBUS clock runs at always half the speed of the CSI_RX_MAIN_CLK.
	CSI_RX_VP_CLK	MAIN_PLL25_HSDIV1_CLKOUT	PLL25	Video port interface clock. It must run at the same speed or higher than the CSI_RX_MAIN_CLK. It can be async to the CSI_RX_MAIN_CLK clock. However, it must be sync to VPAC video clock.
	CSI_RX_BYTE_CLK	RXBYTECLKHS	DPHY_RX0	The byte clock is the clock supplied by the DPHY_RX.
CSI_RX_IF1	CSI_RX_MAIN_CLK	MAIN_SYSCLK0	PLLCTRL0	Main functional clock.
	CSI_RX_VBUS_CLK	MAIN_SYSCLK0 / 2	PLLCTRL0	The VBUS clock runs at always half the speed of the CSI_RX_MAIN_CLK.
	CSI_RX_VP_CLK	MAIN_PLL25_HSDIV1_CLKOUT	PLL25	Video port interface clock. It must run at the same speed or higher than the CSI_RX_MAIN_CLK. It can be async to the CSI_RX_MAIN_CLK clock. However, it must be sync to VPAC video clock.
	CSI_RX_BYTE_CLK	RXBYTECLKHS	DPHY_RX1	The byte clock is the clock supplied by the DPHY_RX.
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
CSI_RX_IF0	CSI_RX_RST	MOD_G_RST	LPSC53	Asynchronous module global reset, driving all collateral asynchronous resets of the 4 clock domains to the low state.
CSI_RX_IF1	CSI_RX_RST	MOD_G_RST	LPSC54	Asynchronous module global reset, driving all collateral asynchronous resets of the 4 clock domains to the low state

**Table 12-1524. CSI\_RX\_IF Hardware Requests**

Interrupt Requests				
--------------------	--	--	--	--

**Table 12-1524. CSI\_RX\_IF Hardware Requests (continued)**

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
CSI_RX_IF0	CSI_RX_IF0_CSI_ERR_IRQ_0	GIC500_SPI_IN_185	COMPUTE_CLUSTER0	Stream error detected. The CSI_RX_IF0 will detect the error condition and capture the associated VC, DT and WC information for the packet header before generating an error interrupt flag on CSI_RX_CSI_ERR_IRQ.	Level
		MAIN2MCU_LVL_INTRTR0_IN_255	MAIN2MCU_LVL_INTRTR0	Stream error detected. The CSI_RX_IF0 will detect the error condition and capture the associated VC, DT and WC information for the packet header before generating an error interrupt flag on CSI_RX_CSI_ERR_IRQ.	Level
		R5FSS0_CORE0_INTR_IN_78	R5FSS0	Stream error detected. The CSI_RX_IF0 will detect the error condition and capture the associated VC, DT and WC information for the packet header before generating an error interrupt flag on CSI_RX_CSI_ERR_IRQ.	Level
		R5FSS0_CORE1_INTR_IN_78	R5FSS0	Stream error detected. The CSI_RX_IF0 will detect the error condition and capture the associated VC, DT and WC information for the packet header before generating an error interrupt flag on CSI_RX_CSI_ERR_IRQ.	Level
		R5FSS1_CORE0_INTR_IN_78	R5FSS1	Stream error detected. The CSI_RX_IF0 will detect the error condition and capture the associated VC, DT and WC information for the packet header before generating an error interrupt flag on CSI_RX_CSI_ERR_IRQ.	Level
		R5FSS1_CORE1_INTR_IN_78	R5FSS1	Stream error detected. The CSI_RX_IF0 will detect the error condition and capture the associated VC, DT and WC information for the packet header before generating an error interrupt flag on CSI_RX_CSI_ERR_IRQ.	Level
		ESM0_LVL_IN_196	ESM0	Stream error detected. The CSI_RX_IF0 will detect the error condition and capture the associated VC, DT and WC information for the packet header before generating an error interrupt flag on CSI_RX_CSI_ERR_IRQ.	Level
CSI_RX_IF0_CSI_IRQ_0		GIC500_SPI_IN_184	COMPUTE_CLUSTER0	Global interrupt that various resynchronized sources converge into interrupt generation.	Level
		MAIN2MCU_LVL_INTRTR0_IN_254	MAIN2MCU_LVL_INTRTR0	Global interrupt that various resynchronized sources converge into interrupt generation.	Level
		R5FSS0_CORE0_INTR_IN_79	R5FSS0	Global interrupt that various resynchronized sources converge into interrupt generation.	Level

**Table 12-1524. CSI\_RX\_IF Hardware Requests (continued)**

	R5FSS0_CORE1_INTR_IN_79	R5FSS0	Global interrupt that various resynchronized sources converge into interrupt generation.	Level
	R5FSS1_CORE0_INTR_IN_79	R5FSS1	Global interrupt that various resynchronized sources converge into interrupt generation.	Level
	R5FSS1_CORE1_INTR_IN_79	R5FSS1	Global interrupt that various resynchronized sources converge into interrupt generation.	Level
CSI_RX_IF0_CSI_LEVEL_0	GIC500_SPI_IN_186	COMPUTE_CLUSTER0	PSI_L fifo overflow or VP0/VP1 frame/line mismatch	Level
	MAIN2MCU_LVL_INTRTR0_IN_256	MAIN2MCU_LVL_INTRTR0	PSI_L fifo overflow or VP0/VP1 frame/line mismatch	Level
	R5FSS0_CORE0_INTR_IN_80	R5FSS0	PSI_L fifo overflow or VP0/VP1 frame/line mismatch	Level
	R5FSS0_CORE1_INTR_IN_80	R5FSS0	PSI_L fifo overflow or VP0/VP1 frame/line mismatch	Level
	R5FSS1_CORE0_INTR_IN_80	R5FSS1	PSI_L fifo overflow or VP0/VP1 frame/line mismatch	Level
	R5FSS1_CORE1_INTR_IN_80	R5FSS1	PSI_L fifo overflow or VP0/VP1 frame/line mismatch	Level
CSI_RX_IF0_CORR_LEVEL_0	ESM0_LVL_IN_206	ESM0	This interrupt is for checking the interface signals of the CSI_RX_IF0 controller for parity.	Level
CSI_RX_IF0_CSI_FATAL_0	ESM0_LVL_IN_200	ESM0	ASF port fatal interrupt. Level sensitive.	Level
CSI_RX_IF0_CSI_NONFATAL_0	ESM0_LVL_IN_201	ESM0	ASF port non-fatal interrupt. Level sensitive.	Level
CSI_RX_IF0_UNCORR_LEVEL_0	ESM0_LVL_IN_207	ESM0	This interrupt is for checking the interface signals of the CSI_RX_IF0 controller for parity.	Level
CSI_RX_IF1_CSI_ERR_IRQ_0	GIC500_SPI_IN_189	COMPUTE_CLUSTER0	Stream error detected. The CSI_RX_IF1 will detect the error condition and capture the associated VC, DT and WC information for the packet header before generating an error interrupt flag on CSI_RX_CSI_ERR_IRQ.	Level
	MAIN2MCU_LVL_INTRTR0_IN_258	MAIN2MCU_LVL_INTRTR0	Stream error detected. The CSI_RX_IF1 will detect the error condition and capture the associated VC, DT and WC information for the packet header before generating an error interrupt flag on CSI_RX_CSI_ERR_IRQ.	Level
	R5FSS0_CORE0_INTR_IN_81	R5FSS0	Stream error detected. The CSI_RX_IF1 will detect the error condition and capture the associated VC, DT and WC information for the packet header before generating an error interrupt flag on CSI_RX_CSI_ERR_IRQ.	Level

**Table 12-1524. CSI\_RX\_IF Hardware Requests (continued)**

	R5FSS0_CORE1_INTR_IN_81	R5FSS0	Stream error detected. The CSI_RX_IF1 will detect the error condition and capture the associated VC, DT and WC information for the packet header before generating an error interrupt flag on CSI_RX_CSI_ERR_IRQ.	Level
	R5FSS1_CORE0_INTR_IN_81	R5FSS1	Stream error detected. The CSI_RX_IF1 will detect the error condition and capture the associated VC, DT and WC information for the packet header before generating an error interrupt flag on CSI_RX_CSI_ERR_IRQ.	Level
	R5FSS1_CORE1_INTR_IN_81	R5FSS1	Stream error detected. The CSI_RX_IF1 will detect the error condition and capture the associated VC, DT and WC information for the packet header before generating an error interrupt flag on CSI_RX_CSI_ERR_IRQ.	Level
	ESM0_LVL_IN_197	ESM0	Stream error detected. The CSI_RX_IF1 will detect the error condition and capture the associated VC, DT and WC information for the packet header before generating an error interrupt flag on CSI_RX_CSI_ERR_IRQ.	Level
CSI_RX_IF1_CSI_IRQ_0	GIC500_SPI_IN_188	COMPUTE_CLUSTER0	Global interrupt that various resynchronized sources converge into interrupt generation.	Level
	MAIN2MCU_LVL_INTRTR0_IN_257	MAIN2MCU_LVL_INTRTR0	Global interrupt that various resynchronized sources converge into interrupt generation.	Level
	R5FSS0_CORE0_INTR_IN_82	R5FSS0	Global interrupt that various resynchronized sources converge into interrupt generation.	Level
	R5FSS0_CORE1_INTR_IN_82	R5FSS0	Global interrupt that various resynchronized sources converge into interrupt generation.	Level
	R5FSS1_CORE0_INTR_IN_82	R5FSS1	Global interrupt that various resynchronized sources converge into interrupt generation.	Level
	R5FSS1_CORE1_INTR_IN_82	R5FSS1	Global interrupt that various resynchronized sources converge into interrupt generation.	Level
CSI_RX_IF1_CSI_LEVEL_0	GIC500_SPI_IN_190	COMPUTE_CLUSTER0	PSI_L fifo overflow or VP0/VP1 frame/line mismatch	Level
	MAIN2MCU_LVL_INTRTR0_IN_259	MAIN2MCU_LVL_INTRTR0	PSI_L fifo overflow or VP0/VP1 frame/line mismatch	Level
	R5FSS0_CORE0_INTR_IN_83	R5FSS0	PSI_L fifo overflow or VP0/VP1 frame/line mismatch	Level
	R5FSS0_CORE1_INTR_IN_83	R5FSS0	PSI_L fifo overflow or VP0/VP1 frame/line mismatch	Level
	R5FSS1_CORE0_INTR_IN_83	R5FSS1	PSI_L fifo overflow or VP0/VP1 frame/line mismatch	Level
	R5FSS1_CORE1_INTR_IN_83	R5FSS1	PSI_L fifo overflow or VP0/VP1 frame/line mismatch	Level

**Table 12-1524. CSI\_RX\_IF Hardware Requests (continued)**

CSI_RX_IF1_ CORR_LEVE L_0	ESM0_LVL_IN_208	ESM0	This interrupt is for checking the interface signals of the CSI_RX_IF1 controller for parity.	Level
CSI_RX_IF1_ CSI_FATAL_0	ESM0_LVL_IN_202	ESM0	ASF port fatal interrupt. Level sensitive.	Level
CSI_RX_IF1_ CSI_NONFAT AL_0	ESM0_LVL_IN_203	ESM0	ASF port non-fatal interrupt. Level sensitive.	Level
CSI_RX_IF1_ UNCORR_LE VEL_0	ESM0_LVL_IN_209	ESM0	This interrupt is for checking the interface signals of the CSI_RX_IF1 controller for parity.	Level



### 12.7.1.4 CSI\_RX\_IF Functional Description

#### 12.7.1.4.1 CSI\_RX\_IF Block Diagram

Figure 12-1136 depicts a simplified internal block diagram of the CSI\_RX\_IF and its surroundings.

**Figure 12-1136. CSI\_RX\_IF Block Diagram**

The CSI2 core streams 4 channels of data to the several export paths.

**Stream0** (high BW DMA up to 32 channel contexts) streams 32-bits worth of data, that is re-formatted and sent as pixel format into 128 bits as PSIL data. There are 32 sets of MMRs/Registers that map virtual channels and data type to PSIL threads. Data is sent out PSIL in FIFO order as it is received. There is no priority. Data re-formatting is done to end up with correct data organization in memory so that other ip can use the data. The data type must be configured in MMRs to ensure proper reformatting. See Section 12.7.1.4.5. The large buffer of PSIL data (2048x128) has ECC protection, see Section 12.7.1.4.7

**Stream1** (unlimited channel contexts, minimal buffering) streams an optional loopback output to CSI\_TX\_IF controller for diagnostic check.

#### 12.7.1.4.2 CSI\_RX\_IF Hardware and Software Reset

An active low asynchronous hardware reset is provided to CSI\_RX\_IF by device LPSC. It is internally resynchronized to the functional clock domain.

A software reset is triggered by configuring the CSI\_RX\_IF\_VBUS2APB\_SOFT\_RESET bit-fields for protocol reset and/or module reset.

#### 12.7.1.4.3 CSI\_RX\_IF Clock Configuration

There are four clock domain in the CSI\_RX\_IF.

1. The CSI\_RX\_MAIN\_CLK clock runs the CSI2 core, PSIL DMA, and retransmits to CSI\_TX\_IF. The minimum clock rate for the CSI\_RX\_MAIN\_CLK is 312.5MHz when DPHY is at max data rate (slower clock permissible when DPHY is at lower data rates). When CSI\_RX\_MAIN\_CLK is operating lower than 312.5MHz, then the clock is essentially limiting the clock rate of the DPHY\_RX. Said another way, CSI\_RX\_MAIN\_CLK must be at least the CSI\_RX\_BYTE\_CLK rate, else FIFOs will overflow. The maximum clock rate is 500MHz 16ffc, 650 Max  $f(\text{CSI\_RX\_BYTE\_CLK}) \times \text{nb\_lanes} / 4$ ; where  $f(\text{CSI\_RX\_BYTE\_CLK})$  is the minimum frequency of CSI\_RX\_BYTE\_CLK.
2. The CSI\_RX\_VBUS\_CLK is the interface configuration clock that runs at half the speed of the CSI\_RX\_MAIN\_CLK.
3. The CSI\_RX\_VP\_CLK is the video port interface clock. It must run at the same speed or higher than CSI\_RX\_MAIN\_CLK. It can be async to CSI\_RX\_MAIN\_CLK clock. It also must be sync to VPAC video clock. Up to 720MHz max rate
4. The CSI\_RX\_BYTE\_CLK is the clock supplied by the DPHY\_RX PLL and is divided down to byte clock. The DPHY\_RX is designed for max of 10gbps. This translates to a max byte clock of 312.5MHz. The clock is inactive when DPHY\_RX is not in HS operation.

Table 12-1525 shows the CSI\_RX\_IF and DPHY\_RX inter-clock dependencies.

**Table 12-1525. CSI\_RX\_IF Inter-clock Dependencies**

	CSI_RX_MAIN_CLK	CSI_RX_VBUS_CLK	CSI_RX_VP_CLK	CSI_RX_BYTE_CLK
<b>Min freq</b>	CSI_RX_BYTE_CLK freq	CSI_RX_MAIN_CLK / 2 freq	CSI_RX_BYTE_CLK freq	N/A
<b>Max freq</b>	500MHz	CSI_RX_MAIN_CLK / 2 freq	720MHz	312.5MHz

#### 12.7.1.4.4 CSI\_RX\_IF Interrupt Events

This section describes the register configuration of the interrupt events that can trigger the several CSI\_RX\_IF interrupt signals. For detailed description and mapping of the interrupt of the device interrupt processors see *CSI\_RX\_IF Integration*.

The interrupts are generally handled within the INTD module of the CSI\_RX\_IF, although there are several interrupt registers in the ECC\_AGGR for ECC errors and in the VBUS2APB for stream monitoring errors/flags.

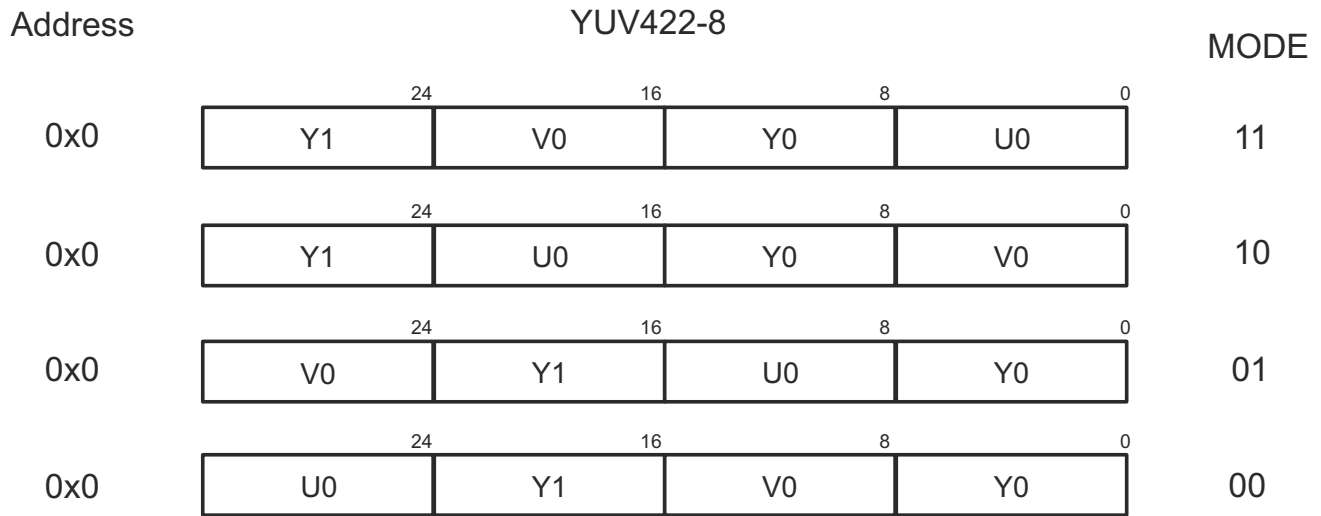
Table 12-1526 lists the event generation and corresponding registers of the CSI\_RX\_IF controller.

**Table 12-1526. CSI\_RX\_IF Interrupt Events Cross Table**

Event	Mask Register	Status Register	Description
CSI_RX_CSI_ERR_IRQ	CSI_RX_IF_VBUS2APB_ERROR_IRQS_MASK_CFG	CSI_RX_IF_VBUS2APB_ERROR_IRQS	Stream error detected. The CSI_RX_IF0 will detect the error condition and capture the associated VC, DT and WC information for the packet header before generating an error interrupt flag on CSI_RX_CSI_ERR_IRQ. Read the status register bitfields to trace the source of the event.
CSI_RX_CSI_IRQ	CSI_RX_IF_CP_INTD_ENABLE_REG_LEVEL_0 CSI_RX_IF_VBUS2APB_ERROR_IRQS_MASK_CFG	CSI_RX_IF_CP_INTD_STATUS_REG_LEVEL_0 CSI_RX_IF_VBUS2APB_ERROR_IRQS	Global functional interrupt that various resynchronized sources converge into interrupt generation.
CSI_RX_CSI_LEVEL	CSI_RX_IF_VBUS2APB_MONITOR_IRQS_MASK_CFG	CSI_RX_IF_VBUS2APB_STREAM0_FIFO_FILL_LVL CSI_RX_IF_VBUS2APB_STREAM1_FIFO_FILL_LVL CSI_RX_IF_VBUS2APB_STREAM2_FIFO_FILL_LVL CSI_RX_IF_VBUS2APB_STREAM3_FIFO_FILL_LVL CSI_RX_IF_VBUS2APB_STREAM0_FIFO_FILL_LVL	PSI_L fifo overflow or VP0/VP1 frame/line mismatch (at the minimum an error interrupt will occur at the end of frame but may be issued within the frame as well). Read the status register bitfields to trace the source of the event.
CSI_RX_CORR_LEVEL	CSI_RX_IF_VBUS2APB_ASF_INT_MASK	CSI_RX_IF_VBUS2APB_ASF_INT_STATUS	This interrupt is for checking the interface signals of the CSI_RX_IF0 controller for parity.
CSI_RX_UNCORR_LEVEL	CSI_RX_IF_VBUS2APB_ASF_INT_MASK	CSI_RX_IF_VBUS2APB_ASF_INT_STATUS	This interrupt is for checking the interface signals of the CSI_RX_IF0 controller for parity.
CSI_RX_CSI_FATAL	CSI_RX_IF_VBUS2APB_ASF_INT_MASK	CSI_RX_IF_VBUS2APB_ASF_INT_STATUS	ASF port fatal interrupt. Level sensitive. Set CSI_RX_IF_VBUS2APB_ASF_FATAL_NONFATAL_SELECT for whether fatal or non-fatal ASF interrupt is triggered. If any of the CSI_RX_IF_VBUS2APB_ASF_INT_STATUS bit is set, the CSI_RX_CSI_FATAL or CSI_RX_CSI_NONFATAL event signal is asserted.
CSI_RX_CSI_NONFATAL	CSI_RX_IF_VBUS2APB_ASF_INT_MASK	CSI_RX_IF_VBUS2APB_ASF_INT_STATUS	ASF port non-fatal interrupt. Level sensitive. Set CSI_RX_IF_VBUS2APB_ASF_FATAL_NONFATAL_SELECT to whether fatal or non-fatal ASF interrupt is triggered. If any of the CSI_RX_IF_VBUS2APB_ASF_INT_STATUS bit is set, the CSI_RX_CSI_FATAL or CSI_RX_CSI_NONFATAL event signal is asserted.

#### 12.7.1.4.5 CSI\_RX\_IF Data Memory Organization Details

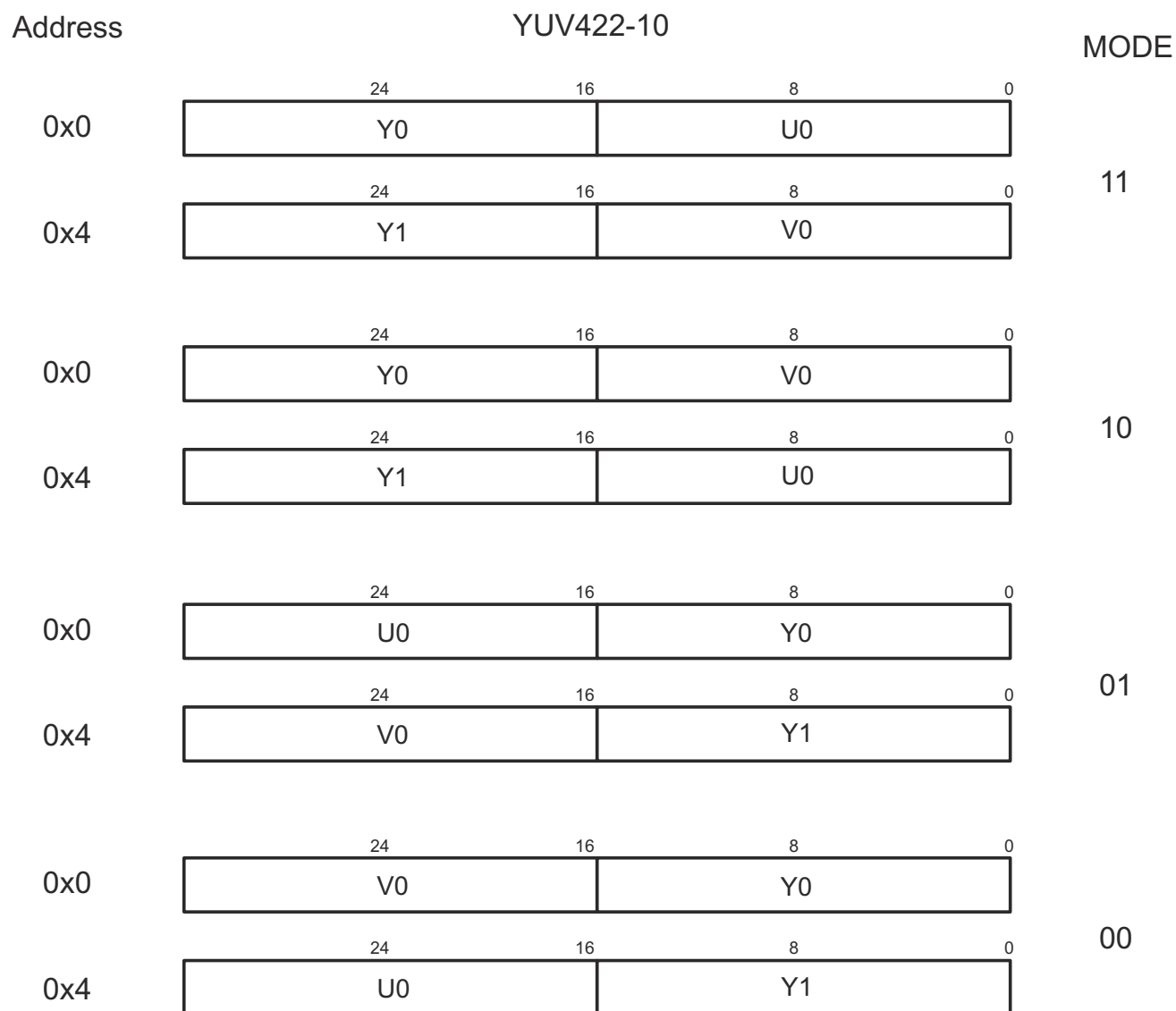
Figure 12-1137 shows the YUV422-8 data organization in memory.



csi\_rx\_if-007

**Figure 12-1137. CSI\_RX\_IF YUV422-8 Memory Data Organization**

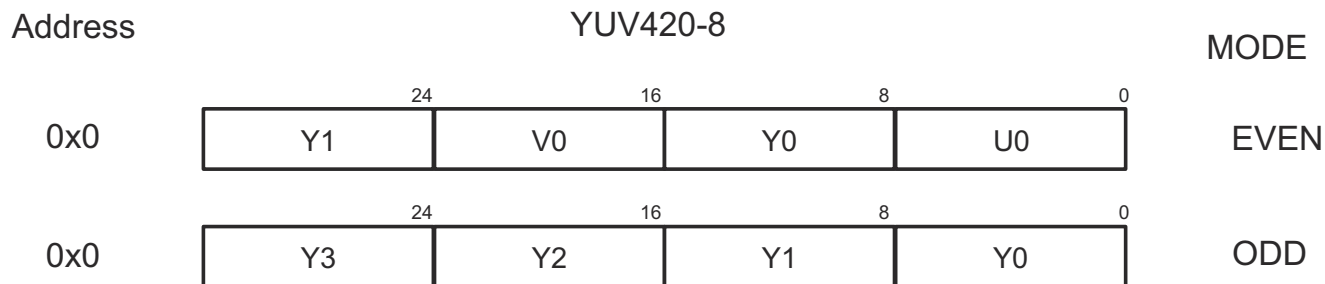
Figure 12-1138 shows the YUV422-10 data organization in memory.



csi\_rx\_if-008

**Figure 12-1138. CSI\_RX\_IF YUV422-10 memory data organization**

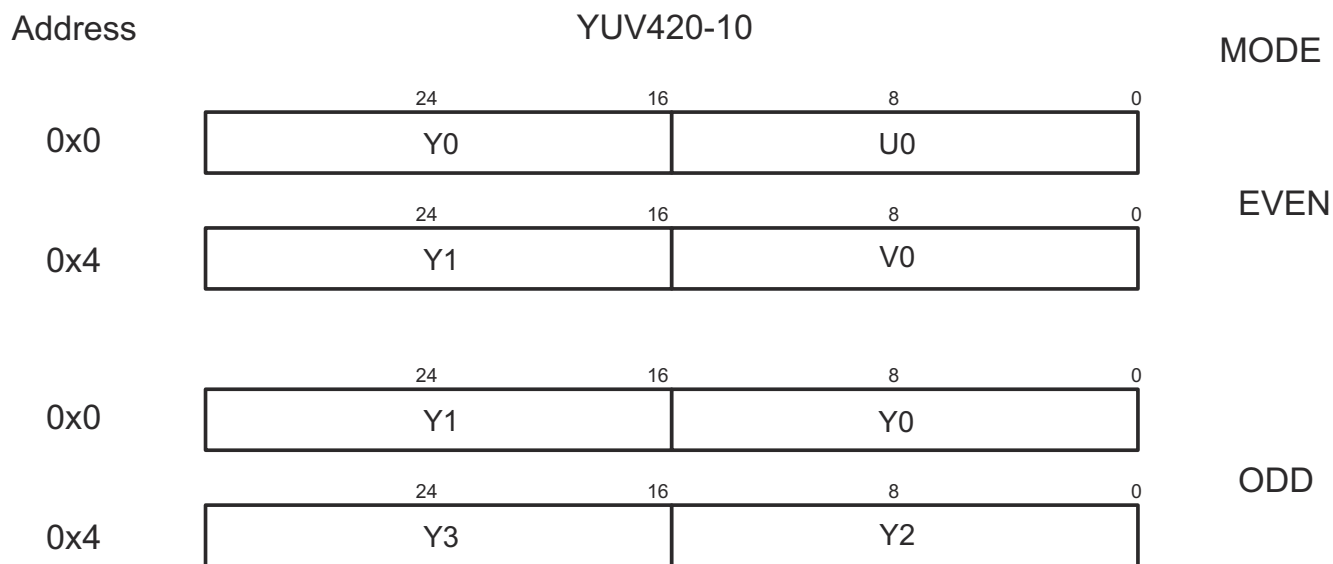
Figure 12-1139 shows the YUV420-8 data organization in memory.



csi\_rx\_if-009

**Figure 12-1139. CSI\_RX\_IF YUV420-8 memory data organization**

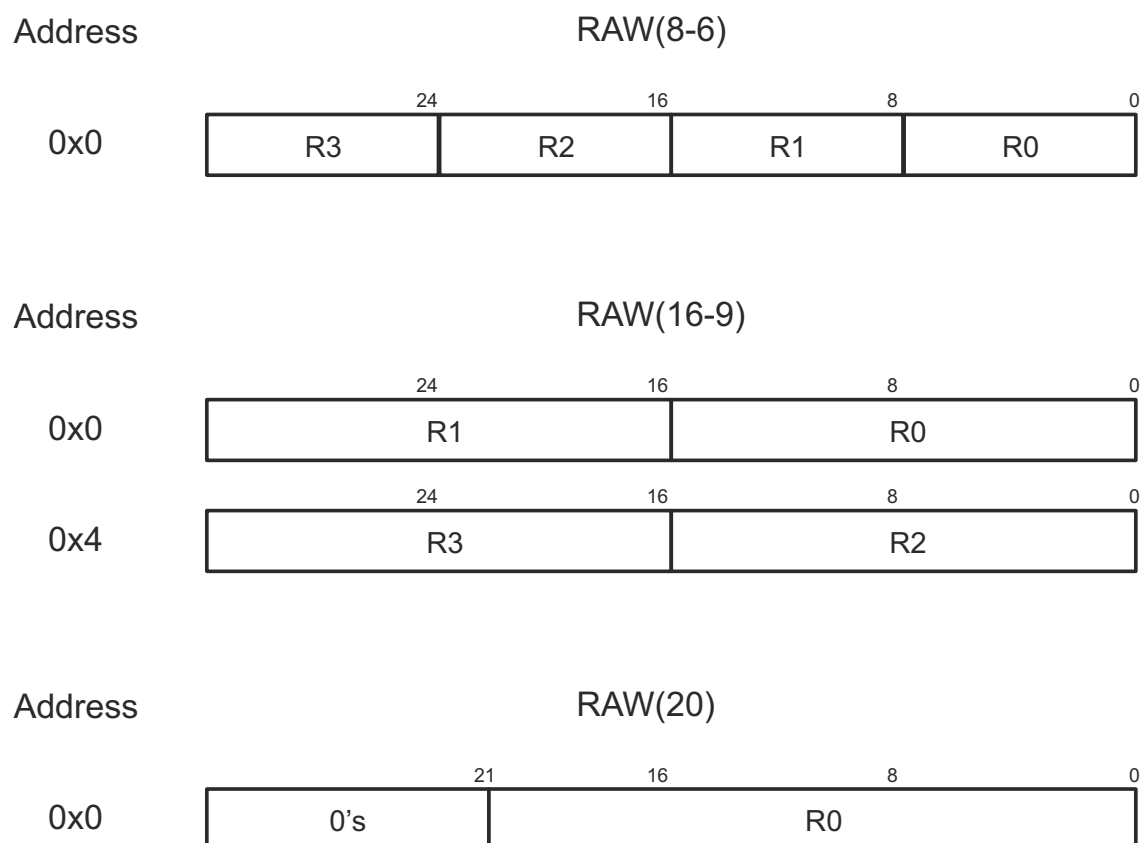
Figure 12-1140 shows the YUV420-10 data organization in memory.



csi\_rx\_if-010

**Figure 12-1140. CSI\_RX\_IF YUV420-10 memory data organization**

Figure 12-1141 shows the RAW data organization in memory.



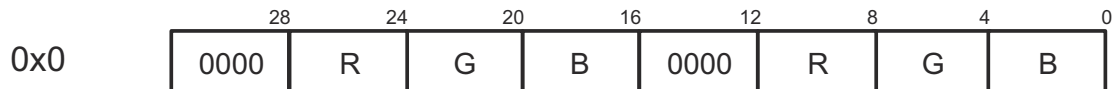
csi\_rx\_if-011

**Figure 12-1141. CSI\_RX\_IF RAW (UNPACKED) memory data organization**

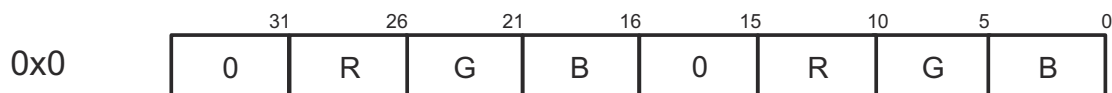
Figure 12-1142 shows the RGB data organization in memory.

Address

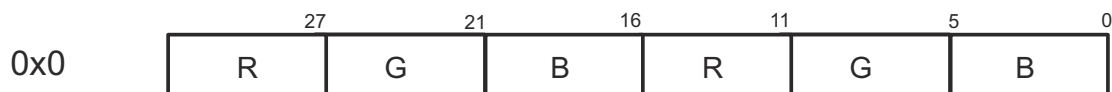
RGB444



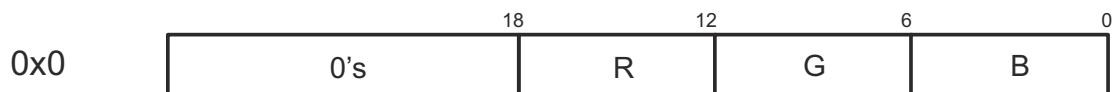
RGB555



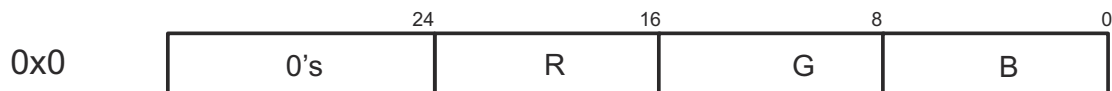
RGB565



RGB666



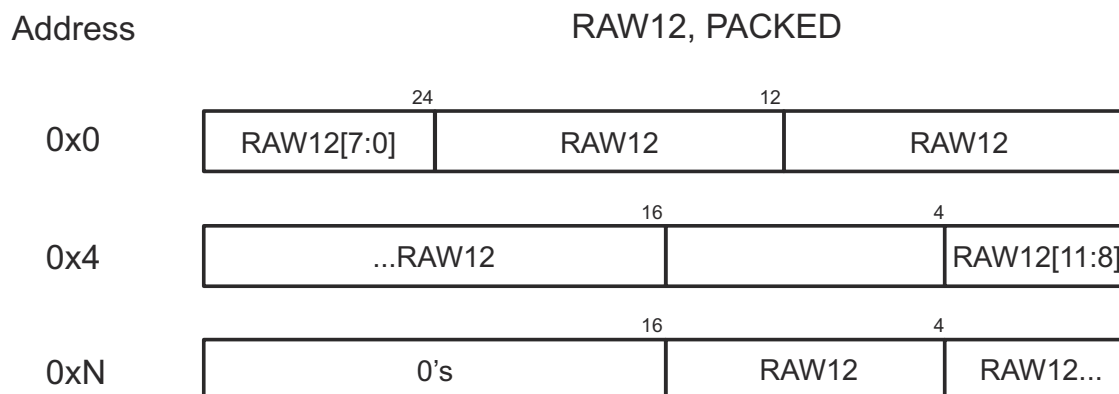
RGB888



csi\_rx\_if-012

**Figure 12-1142. CSI\_RX\_IF RGB memory data organization**

Figure 12-1143 shows the RAW12 (PACKED) data organization in memory.



csi\_rx\_if-012

A. Where data does not fill out to a byte boundary it is zero extended.

**Figure 12-1143. CSI\_RX\_IF RAW12 (PACKED) memory data organization**

#### 12.7.1.4.6 CSI\_RX\_IF PSI\_L (DMA) Interface

##### Note

This section describes CSI\_RX\_IF PSI\_L specific functionality related to the camera interface. The general PSI\_L functional description and registers are described in the PSI\_L specific chapter (see *PSIL Subsystem (PSILSS)*).

##### 12.7.1.4.6.1 PSI\_L DMA framing

The CSI\_RX\_IF stream interface provides the usual VSYNC/HSYNC signals. The signals are per virtual channel where the transition from 1-to-0 or 0-to-1 represents SOF/EOF or SOL/EOL. VSYNC/HSYNC transition is usually not coincident with a valid data phase, though there are use cases where it can be. Because of the nature of CSI protocol, the VSYNC and HSYNC transactions require at least a single clock cycle and can only transition sequentially (important fact in handling of VSYNC events).

There is a rarely used mode of CSI where an entire frame is passed in a single packets without line breaks. In this mode, there are no HSYNC transitions at line starts/ends.

Lastly, there is redundant information provided with each CSI packet where the packet length is supplied in bytes.

To overcome these framing restrictions, the CSI\_RX\_IF PSILSS0 passes metadata through the DMA FIFO. Anytime metadata is inserted into the FIFO, the CSI\_RX\_IF stream is stalled for a clock cycle. The forms of supported meta tags are:

- SOF for a given virtChan
- EOF for a given virtChan
- Packet length in bytes for a given dmaCntx

The PSILSS0 will disregard the HSYNC signal entirely. The long packet beginning is used as SOL and the EOL will be "counted" using the packet length provided. In the special case where packets are of FRAME length, the SOL/EOL handling is identical.

The PSILSS0 solution for VSYNC handling is to create the concept of metadata FIFO entry which is indicated by a metadata bit. The SOF or EOF VSYNC information plus the virtualChan index is fed into the FIFO. On the output of the FIFO, the SOF is saved for each context of that virtualChan type. The very next data phase of that channel context will be marked as SOF. Additionally a state bit per context is maintained indicating the channel is MOP (middle of packet). Once the EOF arrives at the FIFO output, all contexts of that type virtual channel (and currently in MOP state) will receive a PSI\_EOP with zero active bytes of data.



### CAUTION

The line and frame size is not known outside the CSI core (i.e. not passed to the DMA interface). Therefore any mismatches at system level programming will be unknown. If the DMA line/frame size does not match the CSI\_RX\_IF line/frame size, it is assumed the DMA will not result in any adverse side effects as a result of not enough data or too much data.

#### 12.7.1.4.6.2 PSI\_L DMA error handling due to FIFO overflow

The DMA error handling is also called a PSI\_L protocol enforcer. It is intended to prevent hang of the PSILSS0. Unnatural packet size should not cause hang so only SOP/SOL/EOL/EOP framing is enforced. The following list highlights the error handling mechanism:

- For context cleanup the protocol enforcer:
  - cycle through each context index checking if context is in MOL or MOP
  - close out MOP with EOP and MOL&MOP with EOL&EOP
- dropOnFloor SOP if currently in MOPstate
- dropOnFloor EOP if not currently in MOPstate
- dropOnFloor FIFO data
- EOL context if EOP and MOLstate
- After closing out all open contexts the PSILSS0 logic will then wait till end of frame per virtual channel. Once a new frame starts it will then start sending out data from that new frame

#### 12.7.1.4.7 CSI\_RX\_IF ECC Protection Support

ECC is a mechanism for providing increased system reliability (via reduction of memory soft errors) by allowing single bit errors to be detected and corrected and double bit errors to be detected.

The ECC protection on the CSI\_RX\_IF RAM provides Single Error Correction and Double Error Detection (SEC/DED). This logic detects and corrects a single bit error (1 bit error per ECC word or per ECC data segment). For memories that contain critical and/or persistent data, automatic (immediate or delayed) write-back of the corrected data to the corresponding memory address is supported. In addition, the ECC also supports multiple options for partial word writes, such as read-modify-write or multiple ECC code segments per word.

The ECC protection also provides Double Error Detection (DED). This logic only detects (does not correct) double errors (2 bit errors per ECC word or per ECC data segment).

The ECC aggregator in the CSI\_RX\_IF subsystem level consolidates the ECC configuration and status bits for all the ECC supported memories in the subsystem. It provides a single EOI-handshake based interrupt to the interrupt processors (for both single and double error detections) and a standard 32-bit VBUSP interface for configuring and querying the ECC register set, see *CSI\_RX\_IF ECC Registers*. For complete details on the features and functions of the ECC aggregator, see chapter *ECC Aggregator*.

#### 12.7.1.4.8 CSI\_RX\_IF Programming Guide

##### 12.7.1.4.8.1 Overview

This section details specific steps on how to program the CSI\_RX\_IF controller.

##### 12.7.1.4.8.2 Controller Configuration

The CSI\_RX\_IF streams will default to accept all virtual channels and all data types after reset, so the user must program the stream configuration and pixel interface to match the system use case.

### Note

The CSI\_RX\_IF controller can be configured so that the reset values can adopt the users Power\_On state. This can reduce the programming steps required by the system.

The CSI\_RX\_IF controller is designed to operate all the defined streams with all virtual channels and all data types passed to the pixel interface. Also, the connection to the front interface adopts reset values that will allow the connection of the lanes to expect no remapping.

The basic system configuration steps are then programming the number of enabled data lanes and starting the streams.

The system can also decrease the virtual channel and data type processed by the stream by configuring the data config (CSI\_RX\_IF\_VBUS2APB\_STREAM0\_DATA\_CFG - CSI\_RX\_IF\_VBUS2APB\_STREAM3\_DATA\_CFG) registers.

The user must perform a read from the stream config register (CSI\_RX\_IF\_VBUS2APB\_STREAM0\_CFG - CSI\_RX\_IF\_VBUS2APB\_STREAM3\_CFG) at power up to identify the number of streams and pixel interface mode. The stream FIFO depth must be determined from the system configuration information defined at build time to ensure that any programmed [31-16] FIFO\_FILL level is valid for the available FIFO depth.

#### 12.7.1.4.8.3 Power on Configuration

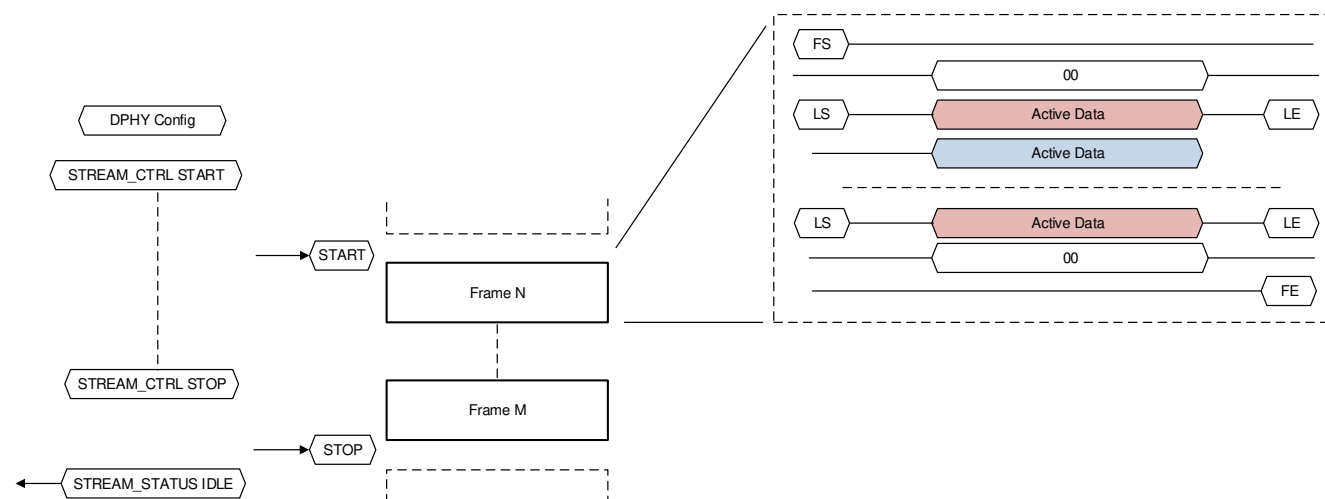
The CSI\_RX\_IF requires the system to perform the configuration of the DPHY\_RX interface before starting the streams of the controller. The default configuration of the controller and each stream can be identified by reading the device ID CSI\_RX\_IF\_VBUS2APB\_DEVICE\_CONFIG register.

The FW must read the CSI\_RX\_IF\_VBUS2APB\_DEVICE\_CONFIG register at power up. This will provide the FW the number of streams that will need to be configured, and all the default system information for the FIFO structures in the available streams.

**Table 12-1527. CSI\_RX\_IF\_VBUS2APB\_DEVICE\_CONFIG bitfield details**

STREAMx_NUM_PIXELS	The width of the pixel interface and the bits per pixel for the selected datatype will determine how many pixels can be output in a single cycle. Default will be 1 pixel per clock. 00 -> 1 pixel per clock 01 -> 2 pixels per clock 10 -> 4 pixels per clock
DATAPATH_SIZE	Internal Datapath width 00 - 32 bit, all other values are reserved.
NUM_STREAMS	Number of Stream interfaces (1-4) = (value+1)
MAX_LANE_NB	Max Number of Lanes (1-4) = (value+1)

Figure 12-1144 shows the minimal sequence of registers that will configure the DPHY\_RX and then start the stream; this will output all the pixel information for all virtual channels and all data types detected in the link data stream.

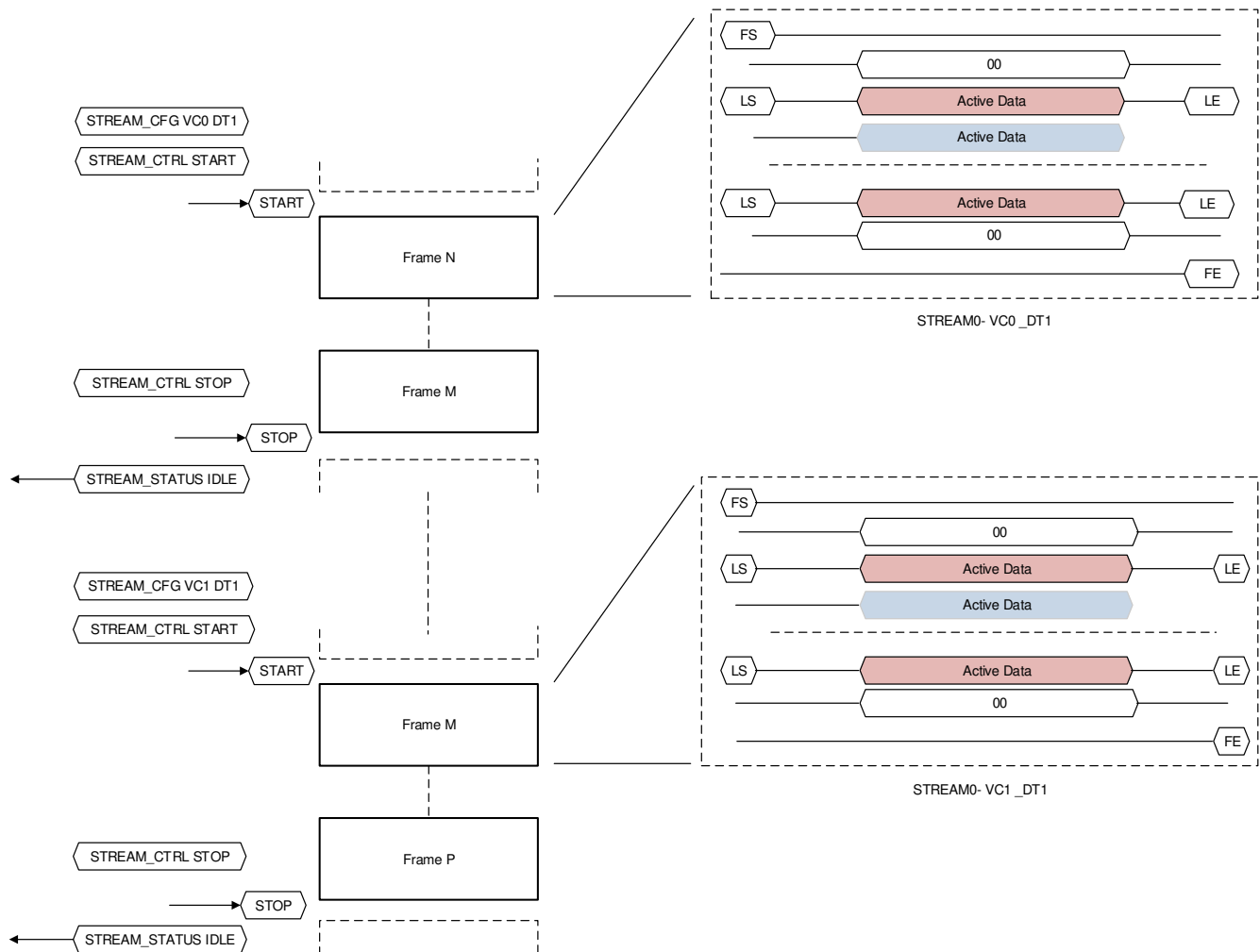


**Figure 12-1144. Minimal Stream Control - Start and Stop**

#### 12.7.1.4.8.4 Stream Start and Stop

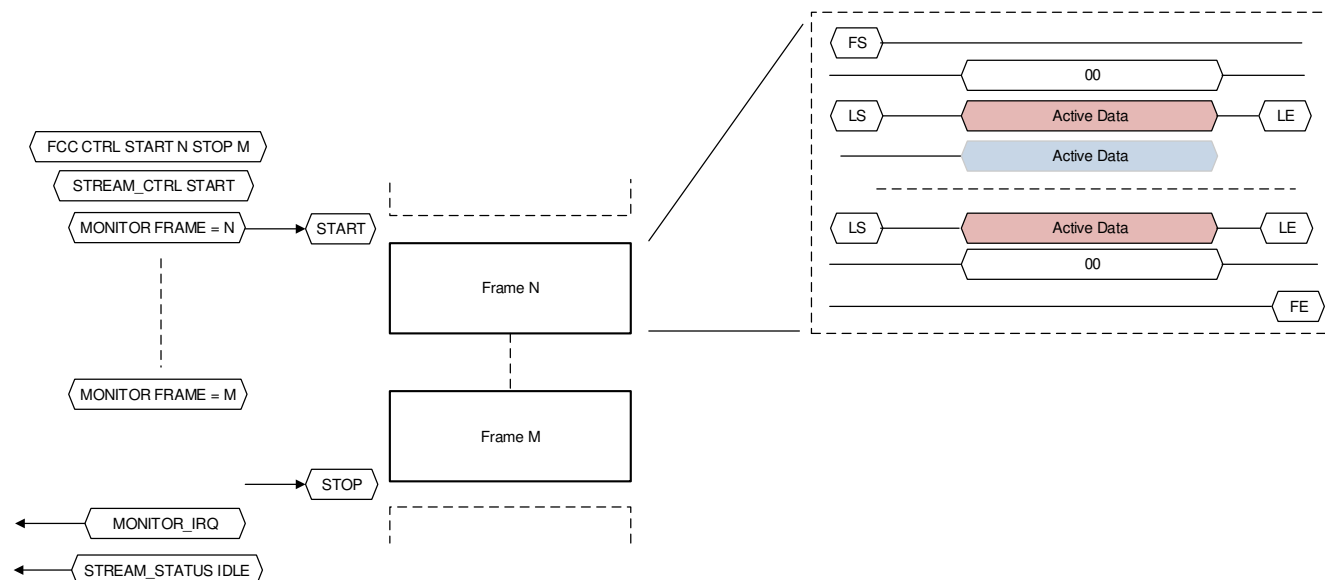
The CSI\_RX\_IF will perform management of the stop and halt control to the pixel stream during functional operation to allow any stream to change the virtual channel or data type information that the stream will process.

- The FW will use the stream control register (CSI\_RX\_IF\_VBUS2APB\_STREAM0\_CTRL - CSI\_RX\_IF\_VBUS2APB\_STREAM3\_CTRL) to perform a stop (set bit 1) and wait for the end of any current frame, and wait for the stream to return its state to IDLE, which can be read from stream status register (CSI\_RX\_IF\_VBUS2APB\_STREAM0\_STATUS - CSI\_RX\_IF\_VBUS2APB\_STREAM3\_STATUS)[31] RUNNING = 1.
- The FW will use the stream control register to perform an abort and wait for the end of any current line, and wait for the stream to return its state to IDLE, which can be read from register STREAM\_STATUS[31] RUNNING = 1.



**Figure 12-1145. Stream Reconfiguration Using Start Stop Start Flow**

The CSI\_RX\_IF will use the monitor control or STOP mechanism to stop the stream processing at the end of the active frame. The stream monitor, configuration and interrupt registers can be redefined and then the stream restarted. The pixel interface will begin processing at the next frame start.



**Figure 12-1146. Stream Start and Stop Using Monitor Control**

#### 12.7.1.4.8.5 Error Control With Soft Resets

The CSI\_RX\_IF will perform control of the soft reset either for error event recovery or to clear a stream FIFO or internal state machine.

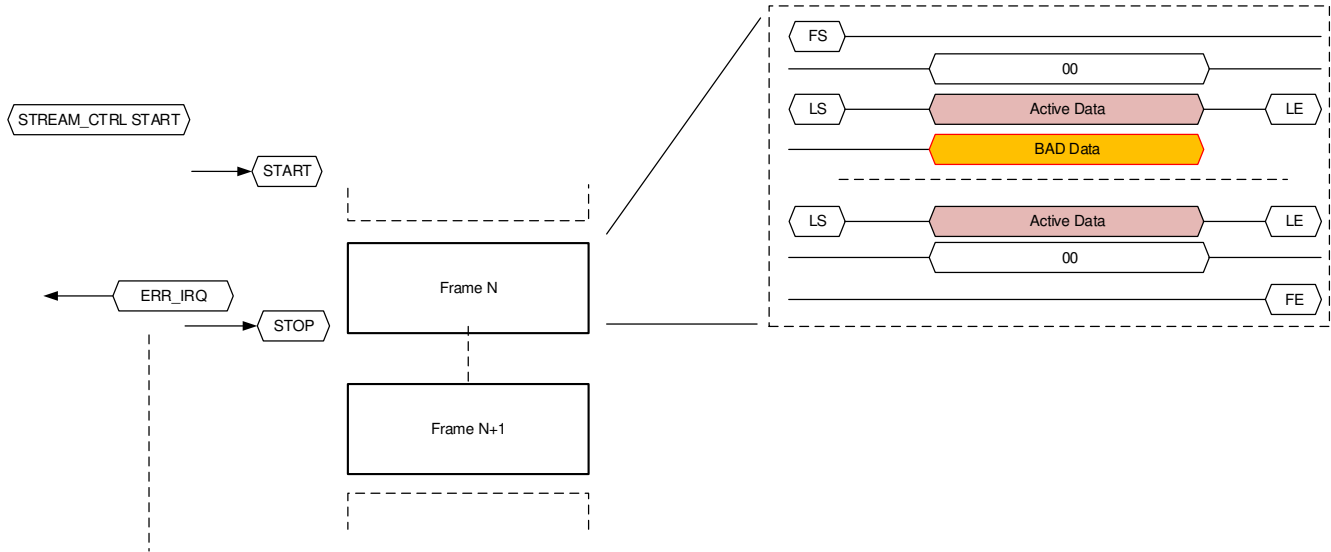
- The FRONT block can be soft reset if the DPHY\_RX becomes unresponsive and the controller wishes to maintain its configuration. In this case the DPHY\_RX resets can be applied and the DPHY\_RX enabled to begin the transfer again.
- The Protocol block can be soft reset if the FRONT soft reset is required, and the protocol is not in the IDLE state.
- The stream soft resets (CSI\_RX\_IF\_VBUS2APB\_STREAM0\_CTRL - CSI\_RX\_IF\_VBUS2APB\_STREAM3\_CTRL)[4] SOFT\_RST can be used to clear the stream to the stop state and reset all the stream state machines and FIFO. If the system has a failure and wishes to clear the stream FIFO and return to a safe state on the pixel interface, the stream soft reset should be asserted.

#### 12.7.1.4.8.6 Stream Error Detected – No Error Bypass Mode

The CSI\_RX\_IF will default to stop processing the frame whenever a header Reserved DT or ECC error is detected, or when a long packet payload CRC is identified.

The CSI\_RX\_IF will detect the error condition and capture the associated VC, DT and WC information for the packet header before generating an error interrupt flag on CSI\_RX\_IF\_VBUS2APB\_ERROR\_IRQS.

The CSI\_RX\_IF will stop processing the current frame and stop the stream.



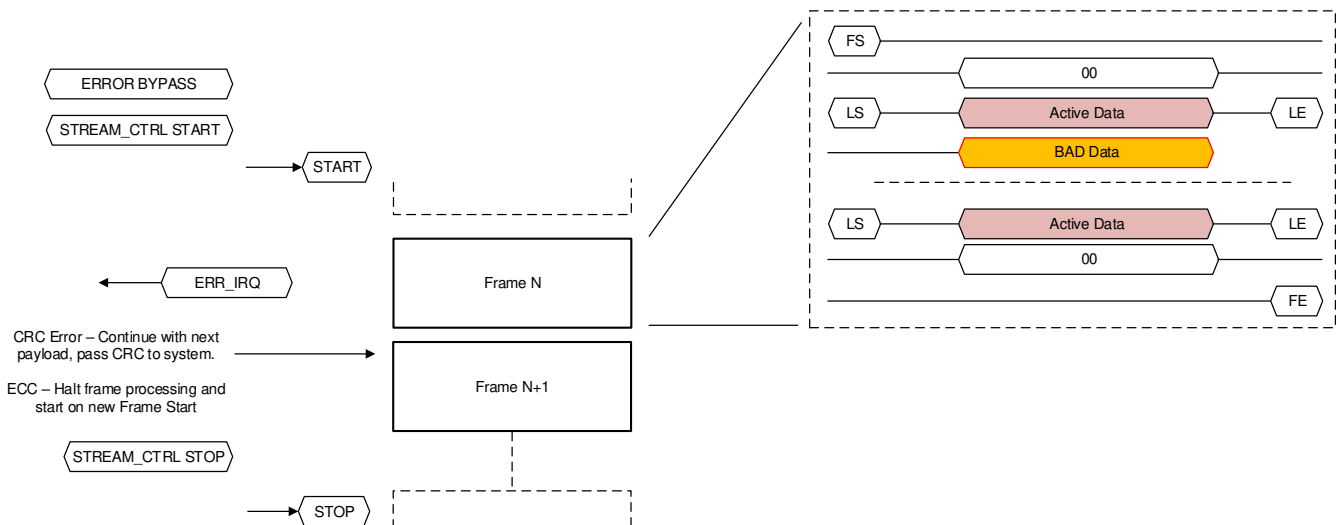
**Figure 12-1147. Stream Error Detection Causing Stop**

#### 12.7.1.4.8.7 Stream Error Detected – Error Bypass Mode

The CSI\_RX\_IF will detect the error condition and capture the associated VC, DT and WC information for the packet header before generating an error interrupt flag on CSI\_RX\_IF\_VBUS2APB\_ERROR\_IRQS.

The CSI\_RX\_IF will continue processing the current frame and continue the stream when a header Reserved DT or a long packet payload CRC error is detected.

The CSI\_RX\_IF will stop processing the current frame and continue the stream when a packet header ECC error is detected, as the current frame synchronisation cannot be maintained.

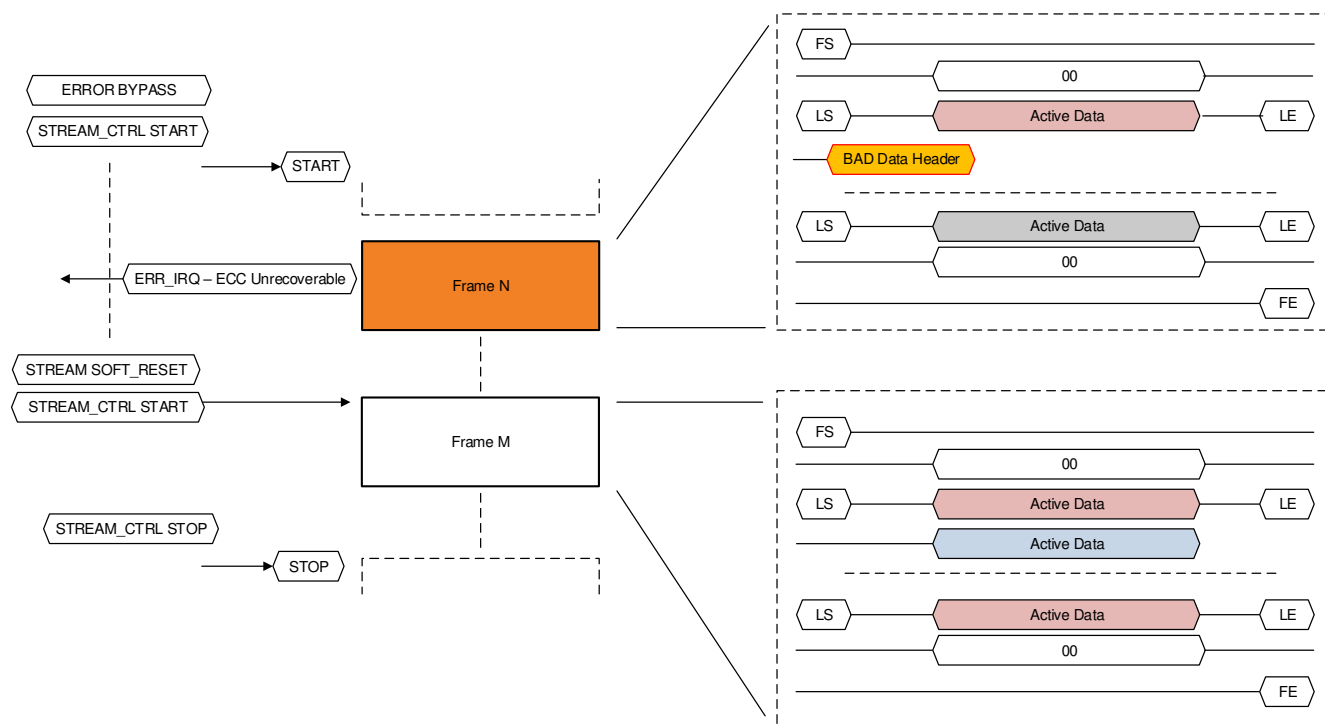


**Figure 12-1148. Stream Error Bypass - CRC Error During Payload**

#### 12.7.1.4.8.8 Stream Error Detected – Soft Reset Recovery

The CSI\_RX\_IF will stop processing the current frame and continue the stream when a packet header ECC error is detected, as the current frame synchronisation cannot be maintained.

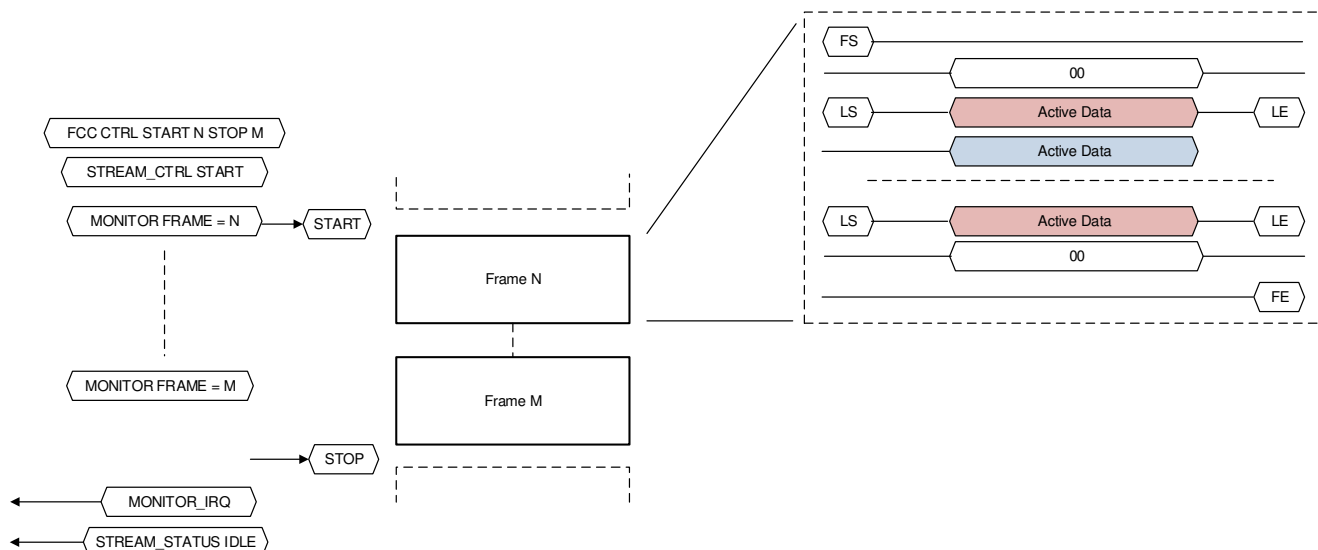
The system can perform a soft reset on the individual stream to clear the pixel interface and payload FIFO and allow the system to restart the stream from the next frame start.



**Figure 12-1149. Stream Soft Reset After ECC Non-Recoverable Error**

#### 12.7.1.4.8.9 Stream Monitor Configuration

The CSI\_RX\_IF will use the monitor control (CSI\_RX\_IF\_VBUS2APB\_STREAM0\_MONITOR\_CTRL - CSI\_RX\_IF\_VBUS2APB\_STREAM3\_MONITOR\_CTRL) or STOP mechanism to stop the stream processing at the end of the active frame. The stream monitor, configuration and interrupt registers can be redefined and then the stream restarted. The pixel interface will begin processing at the next frame start.



**Figure 12-1150. Stream Start and Stop Using Monitor Control**

The FW will use the FCC config control register (CSI\_RX\_IF\_VBUS2APB\_STREAM0\_FCC\_CFG - CSI\_RX\_IF\_VBUS2APB\_STREAM3\_FCC\_CFG) to perform a start and stop as soon as a frame count is detected. The FCC config control register will define the start and stop frame count and the

stream will identify when these values are matched. The stream output will begin when start frame is detected and then stop once the stop frame is reached. The virtual channel can be defined in CSI\_RX\_IF\_VBUS2APB\_STREAM0\_FCC\_CTRL - CSI\_RX\_IF\_VBUS2APB\_STREAM3\_FCC\_CTRL[4:1] FCC\_VC, and must match the virtual channels that are available to the stream in the CSI\_RX\_IF\_VBUS2APB\_STREAM0\_DATA\_CFG - CSI\_RX\_IF\_VBUS2APB\_STREAM3\_DATA\_CFG register.

The frame capture is enabled when CSI\_RX\_IF\_VBUS2APB\_STREAM0\_FCC\_CTRL - CSI\_RX\_IF\_VBUS2APB\_STREAM3\_FCC\_CTRL[0] FCC\_EN is set '1'

The FW can check the current frame counter value from the CSI\_RX\_IF\_VBUS2APB\_STREAM0\_FCC\_CTRL - CSI\_RX\_IF\_VBUS2APB\_STREAM3\_FCC\_CTRL[31:16] FRAME\_COUNTER

#### 12.7.1.4.8.10 Stream Monitor Frame Capture Control

To use Monitor Frame Capture Control Start and Stop operations:

1. Set Stop and Start values in config register (CSI\_RX\_IF\_VBUS2APB\_STREAM0\_FCC\_CFG - CSI\_RX\_IF\_VBUS2APB\_STREAM3\_FCC\_CFG). Set to '0' for free-running.
  - a. [31:16] FRAME\_COUNT\_STOP -> define desired value
  - b. [15:0] FRAME\_COUNT\_START -> define desired value
2. Set the Virtual Channel to the one that is enabled and the enable in register CSI\_RX\_IF\_VBUS2APB\_STREAM0\_FCC\_CTRL - CSI\_RX\_IF\_VBUS2APB\_STREAM3\_FCC\_CTRL.
  - a. [4:1] FCC\_VC -> define VC used
  - b. [0] FCC\_EN -> set to '1'

This will allow visual checking, by setting the Start/Stop as the following examples, this is the behaviour to be observed:

- Stop = 1 / Start = 1: Output (i.e. display) will show a Frame of the Input (i.e. camera)
- Stop = F / Start = 1: Output will show live image from the input during 14 frames and then will freeze in the last one.

Optionally software can read back the frame count value from register (CSI\_RX\_IF\_VBUS2APB\_STREAM0\_FCC\_CTRL - CSI\_RX\_IF\_VBUS2APB\_STREAM3\_FCC\_CTRL).

In order to do that, software will first need to enable the monitor control register.

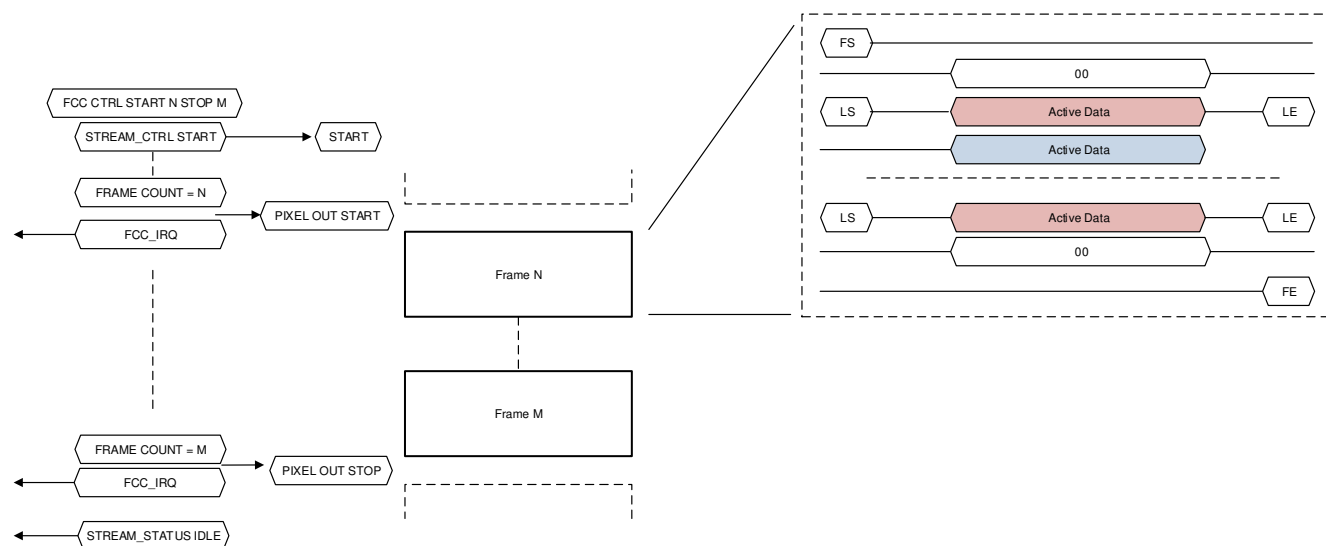
1. Define the same Virtual Channel as above and enable the monitor control register (CSI\_RX\_IF\_VBUS2APB\_STREAM0\_MONITOR\_CTRL - CSI\_RX\_IF\_VBUS2APB\_STREAM3\_MONITOR\_CTRL).
  - a. [15] FRAME\_MON\_EN -> set to '1'
  - b. [14:11] FRAME\_MON\_VC -> define VC used
2. Read frame\_counter containing the current frame number being processed from CSI\_RX\_IF\_VBUS2APB\_STREAM0\_FCC\_CTRL - CSI\_RX\_IF\_VBUS2APB\_STREAM3\_FCC\_CTRL (only when frame numbers are coming from the protocol, not when internal counter)
  - a. [31:16] FRAME\_COUNTER -> read frame number value

The easiest way to check that the frame number matches the Start and/or Stop values defined, is to use the interrupts in CSI\_RX\_IF\_VBUS2APB\_MONITOR\_IRQS register. To be able to do that you will need to allow those interrupts through the mask register CSI\_RX\_IF\_VBUS2APB\_MONITOR\_IRQS\_MASK\_CFG.

1. By default, the CSI\_RX\_IF\_VBUS2APB\_MONITOR\_IRQS\_MASK\_CFG register is set to all '0's, meaning all interrupts are disabled.
  - a. [4] STREAM0\_FCC\_STOP\_IRQM -> set to '1'
  - b. [3] STREAM0\_FCC\_START\_IRQM -> set to '1'
2. Create an interrupt handler to read from CSI\_RX\_IF\_VBUS2APB\_MONITOR\_IRQS.
  - a. [4] STREAM0\_FCC\_STOP\_IRQ -> read, if '1', Stop interrupt triggered
  - b. [3] STREAM0\_FCC\_START\_IRQ -> read, if '1', Start interrupt triggered

To ensure correct functionality, it is highly recommended that the input device (i.e. camera) should be configured after the CSI\_RX\_IF has been configured and enabled if software is using Frame Counter Control features.

Software must make sure all the Virtual Channel fields in the registers explained above are set to the correct VC being used by the stream.



**Figure 12-1151. Stream Frame Capture Control Flow Diagram**

#### 12.7.1.4.8.11 Stream Monitor Timer interrupt

To use the Stream Monitor Timer operations:

1. Set Count value in timer register (CSI\_RX\_IF\_VBUS2APB\_STREAM0\_TIMER - CSI\_RX\_IF\_VBUS2APB\_STREAM3\_TIMER). If set to 0 won't count but interrupt will still trigger every EOF or SOF.
  - a. [24:0] COUNT -> define desired value
2. Set the Virtual Channel to the one that is enabled, define timer\_eof and set enable in the monitor controll register (CSI\_RX\_IF\_VBUS2APB\_STREAM0\_MONITOR\_CTRL - CSI\_RX\_IF\_VBUS2APB\_STREAM3\_MONITOR\_CTRL).
  - a. [15] FRAME\_MON\_EN -> set to '1'
  - b. [14:11] FRAME\_MON\_VC -> define VC used
  - c. [10] TIMER\_EOF -> set to '1' to count from EOF, set to '0' to count from SOF
  - d. [9] TIMER\_EN -> set to '1'
  - e. [8:5] TIMER\_VC -> define VC used

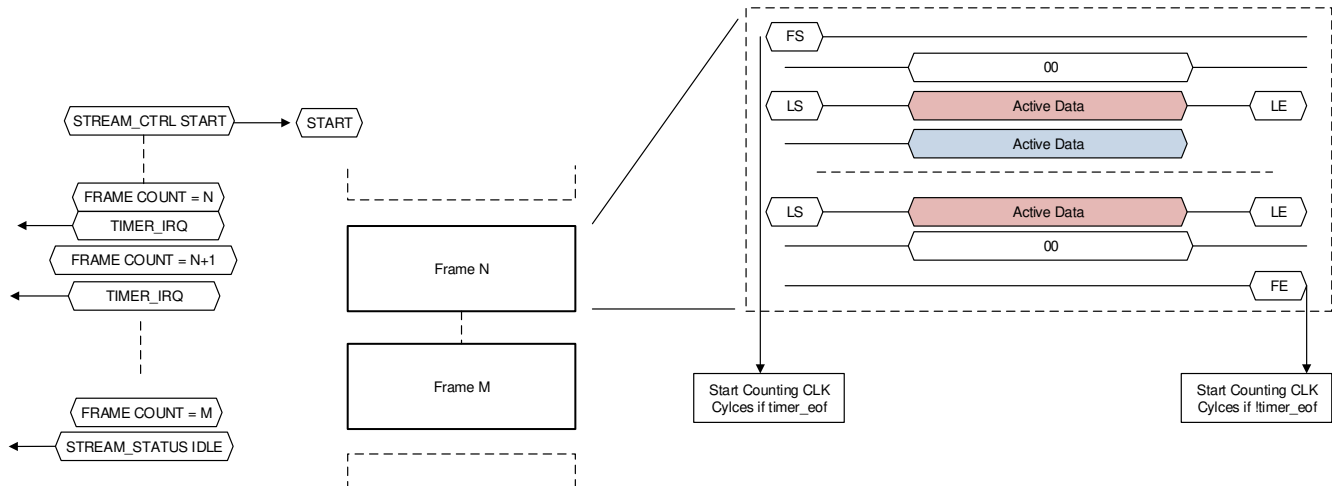
Software should now use the interrupts in register CSI\_RX\_IF\_VBUS2APB\_MONITOR\_IRQS. To be able to do that you will need to allow those interrupts through the mask register CSI\_RX\_IF\_VBUS2APB\_MONITOR\_IRQS\_MASK\_CFG.

1. By default, the CSI\_RX\_IF\_VBUS2APB\_MONITOR\_IRQS\_MASK\_CFG register is set to all '0', meaning all interrupts are disabled.
  - a. [0] STREAM0\_TIMER\_IRQM -> set to '1'
2. Create an interrupt handler to read from monitor\_irqs.
  - a. [0] STREAM0\_TIMER\_IRQ -> read, if '1', timer interrupt is triggered

To ensure correct functionality, it is highly recommended that the input device (i.e. camera) should be configured after the CSI\_RX\_IF has been configured and enabled.

Software must make sure all the Virtual Channel fields in the registers explained above are set to the correct VC being used by the stream.





**Figure 12-1152. Timer Interrupt Flow Diagram**

#### 12.7.1.4.8.12 Stream Monitor Line/Byte Counters Interrupt

To use the stream Monitor Line and Byte counters operations:

1. Set LineCount and ByteCount values in register (CSI\_RX\_IF\_VBUS2APB\_STREAM0\_MONITOR\_LB - CSI\_RX\_IF\_VBUS2APB\_STREAM3\_MONITOR\_LB).
  - a. [31:16] LINE\_COUNT -> define desired value
  - b. [15:0] BYTE\_COUNT -> define desired value
2. Set the Virtual Channel to the one that is enabled, and set enable in register SI\_RX\_IF\_VBUS2APB\_STREAM0\_MONITOR\_CTRL.
  - a. [15] FRAME\_MON\_EN -> set to '1'
  - b. [14:11] FRAME\_MON\_VC -> define VC used
  - c. [4] LB\_EN -> set to '1'
  - d. [3:0] LB\_VC -> set to VC used

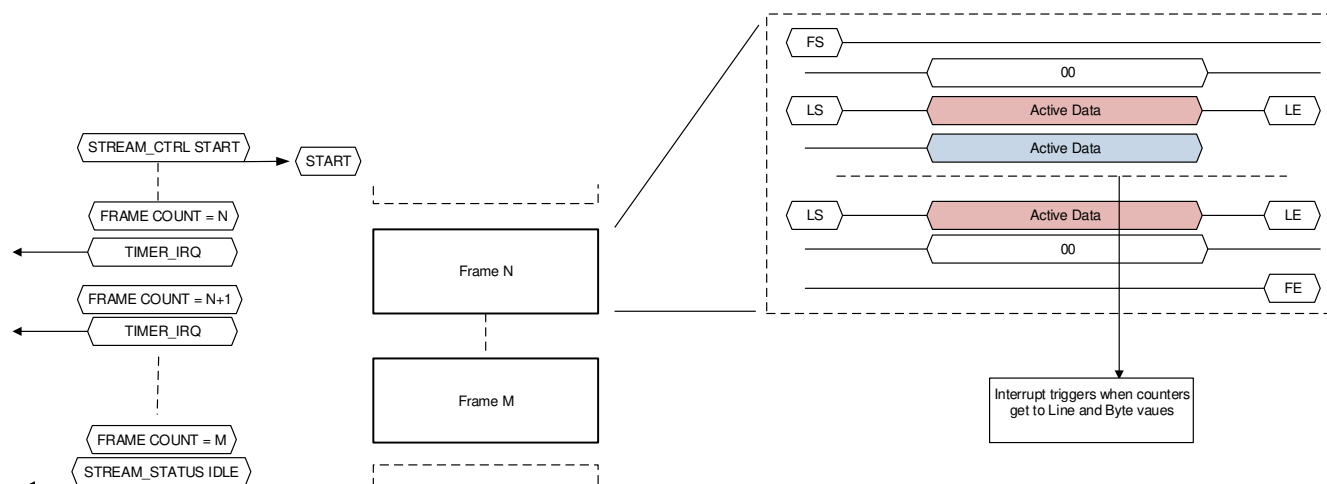
Software should now use the interrupts in register CSI\_RX\_IF\_VBUS2APB\_MONITOR\_IRQS. To be able to do that you will need to allow those interrupts through the mask register CSI\_RX\_IF\_VBUS2APB\_MONITOR\_IRQS\_MASK\_CFG.

1. By default, the CSI\_RX\_IF\_VBUS2APB\_MONITOR\_IRQS\_MASK\_CFG register is set to all '0', meaning all interrupts are disabled.
  - a. [7] STREAM0\_LINE\_CNT\_ERROR\_IRQM -> set to '1'
  - b. STREAM0\_LB\_IRQM [1] -> set to '1'
2. Create an interrupt handler to read from CSI\_RX\_IF\_VBUS2APB\_MONITOR\_IRQS.
  - a. [7] STREAM0\_LINE\_CNT\_ERROR\_IRQ -> read, if '1', line count error interrupt is triggered
  - b. [1] STREAM0\_LB\_IRQ -> read, if '1', line/byte counter interrupt is triggered

Note that the interrupt will trigger on each frame when it reaches byte number BYTE\_COUNT in line number LINE\_COUNT.

To ensure correct functionality, it is highly recommended that the input device (i.e. camera) should be configured after the CSI\_RX\_IF has been configured and enabled.

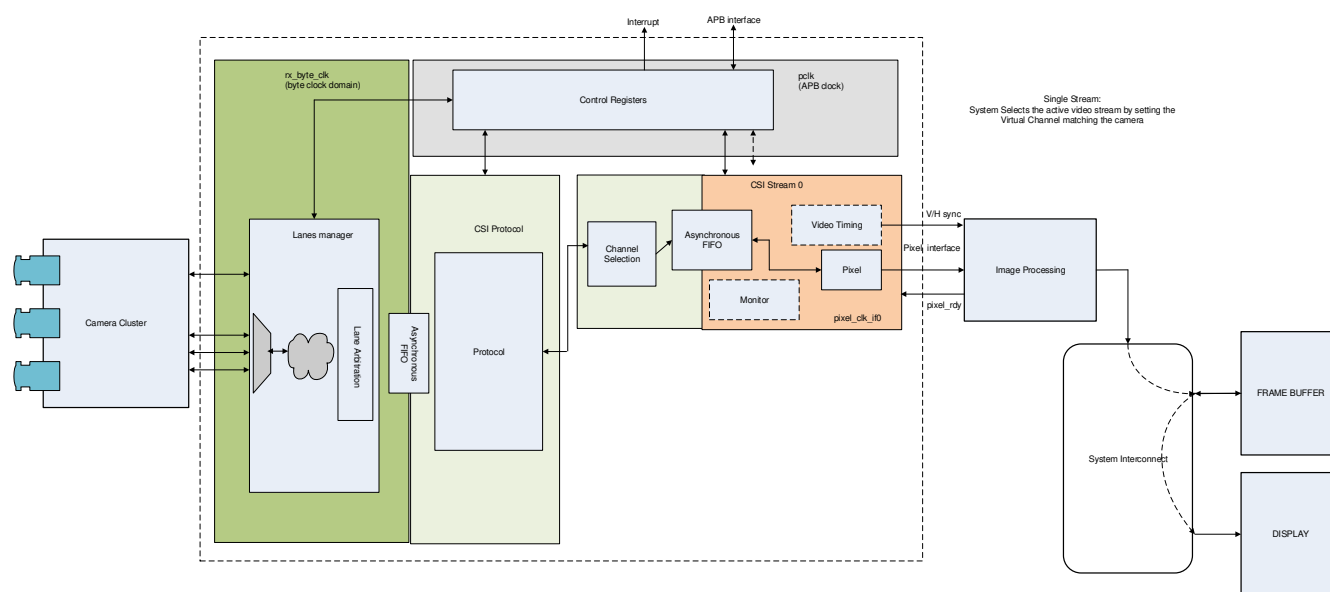
Software must make sure all the Virtual Channel fields in the registers explained above are set to the correct VC being used by the stream.



**Figure 12-1153. Line/Byte Counters Interrupt Flow Diagram**

#### 12.7.1.4.8.13 Example Controller Programming Sequence (Single Stream Operation)

The single stream operation will provide the smallest multi-lane IP configuration with the reduced registers configuration removing some control and status registers. The stream will not require a large FIFO configuration and the clock rates will be matched to simplify the implementation to single pixel transfers using all the available interface bits, (i.e. up to 32).



**Figure 12-1154. Basic Single Stream Use Case Illustration**

For single pixel stream configuration, elastic buffer, RAW8 Data type on VC2, one pixel per cycle:

1. Configure the number of DPHY\_RX lanes:
  - a. See CSI\_RX\_IF\_VBUS2APB\_STATIC\_CFG register
2. Set Error Interrupt mask:
  - a. See CSI\_RX\_IF\_VBUS2APB\_ERROR\_IRQS\_MASK\_CFG register
3. Set the Pixel Interface and FIFO configuration. See CSI\_RX\_IF\_VBUS2APB\_STREAM0\_CFG - CSI\_RX\_IF\_VBUS2APB\_STREAM3\_CFG.
  - a. Set [9-8] FIFO\_MODE to select the elastic buffer configuration -> '1'

- b. Set the FIFO fill level that will determines the FIFO depth that must be reached before data is output -> 0x0000
  - c. Select the number of pixels to be output on each cycle -> '0'.
  - d. Set the type of stream interface to "pixel" mode -> '0'
4. Select the virtual channel and data types to be processed. See CSI\_RX\_IF\_VBUS2APB\_STREAM0\_DATA\_CFG - CSI\_RX\_IF\_VBUS2APB\_STREAM3\_DATA\_CFG register.
  - a. Set [31-16] VC\_SELECT and virtual channel bits if not supporting all virtual channels -> 1h, 2h
  - b. Set [7] ENABLE\_DT0 and [5-0] DATATYPE\_SELECT0 values for one or both data types (default all DT) -> 0h, 1h, 2Ah
5. Enable the stream to begin processing the incoming data from the DPHY\_RX. See CSI\_RX\_IF\_VBUS2APB\_STREAM0\_CTRL - CSI\_RX\_IF\_VBUS2APB\_STREAM3\_CTRL.
  - a. Set [0] START bit -> '1'

The stream will begin to pass pixel data matching the configuration on the next frame start.

#### 12.7.1.4.8.14 CSI\_RX\_IF Programming Restrictions

The following CSI\_RX\_IF programming restrictions apply:

- Full line buffer mode is not supported.
- LS/LE detection has a restriction to be disabled
- Only use Pixel transmit mode can be used, never packet mode
  - In pixel mode only single, dual, or quad mode can be used. It is not allowed to mix these based on data format or virtual channels.
- LSB alignment only
- In CSI\_RX\_IF to CSI\_TX\_IF loop back, the stream needs to have the same mode on both controllers(single, dual, quad). Can not mix dual on CSI\_RX\_IF and single on CSI\_TX\_IF.
- Video port(VP) stream interfaces must be programmed in the CSI\_RX\_IF to meet the following restrictions:
  - Raw 8-16 formats only
  - Dual pixel mode
  - Filtering for single virtual channel and single data type

#### Note

CSI\_RX\_IF has a limitation that any line must fully be exported on its stream interface before a new line or even end of frame is sent on the D-PHY interface. To meet this requirement, software will need to take into account export rates on stream interface(single, dual, quad modes) at 500MHz versus D-PHY 32-bits @ byte clock frequency to know when a new line or end of frame can be sent in after last line was sent over D-PHY interface.

Table 12-1528 shows the CSI\_RX\_IF programming requirements for pixel modes and data sizes.

**Table 12-1528. CSI\_RX\_IF Programming requirements for pixel modes and data sizes**

Format	Pixel transmit mode used	Size	Details
RAW(6-8), user defined 8-bit	Single	0	
RAW(6-8), user defined 8-bit	Dual	1	
RAW(6-8), user defined 8-bit	Quad	2	
RAW(10-16), user defined 16-bit	Single	1	
RAW(10-16), user defined 16-bit	Dual	2	
RAW20	Single	2	
YUV422-8	Single	0	must set dual mode=0
YUV422-8	Dual	0	must set dual mode=1
YUV422-10	Single	1	
YUV420-8	Single	0	must set dual mode=0

**Table 12-1528. CSI\_RX\_IF Programming requirements for pixel modes and data sizes (continued)**

Format	Pixel transmit mode used	Size	Details
YUV420-8	Dual	0	must set dual mode=1
YUV420-10	Single	1	

#### 12.7.1.4.8.15 CSI\_RX\_IF Real-time operating requirements

Care must be taken on blanking requirements for next line, end of frame, or start of frame. The previous packets(long or short) must be fully consumed before any new line, end of frame, or start frame can be sent over the CSI interface. Software will get various error conditions if these are not met. Below are some sample equations to determine this blanking time. Note that this is not blanking data. High level details are that the internal FIFO must be completely empty prior to sending any thing new into it. below are some sample equations to determine how much time is needed to empty the FIFO and blanking time needed. Byte clock rates, pixel data type, and export modes play into the calculation as well.

$$\text{Byte\_clock\_rate} \times 8 \times \#\text{lanes} \times \text{Tcsi} = \text{pixel\_clock\_rate} \times \text{BPP} \times \text{Tpix} = \text{total bits} \quad (27)$$

BPP = Bits Per Pixel. This number is how many bits are sent per pixel clock cycle is based on single, dual, quad modes and data type.

Tcsi = Time for csi interface to complete 1 line

Tpix = Time for pixel interface to complete 1 line

Tpix16 = Tpix+16/pixel\_clock\_rate

(Tpix16-Tcsi) = required blanking time on CSI interface

Tpix16-Tcsi <= 0 implies 0

## 12.7.2 MIPI D-PHY Receiver (DPHY\_RX)

The following sections describe the MIPI D-PHY receiver (DPHY\_RX) modules in the device.

### 12.7.2.1 DPHY\_RX Overview

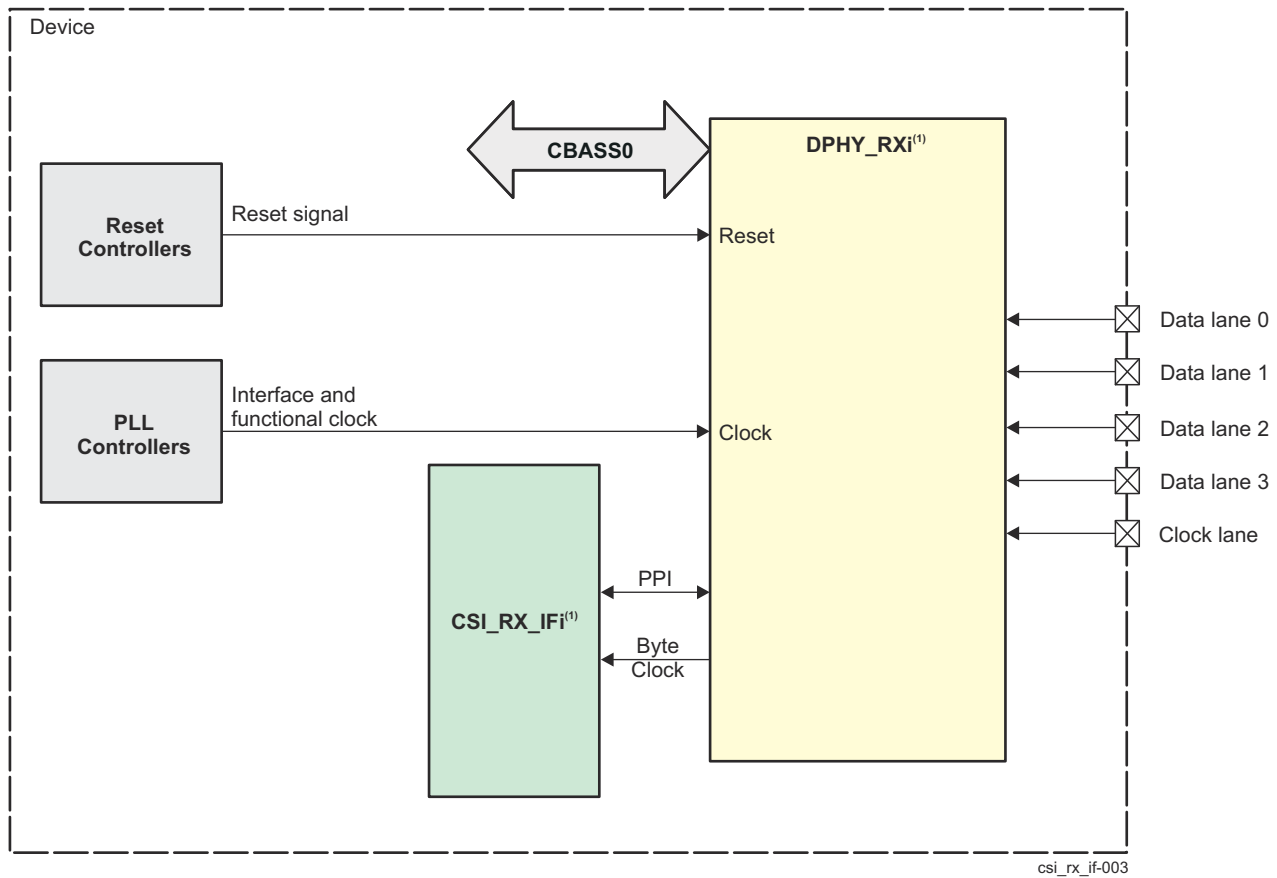
The integration of the DPHY\_RX camera physical port module allows the device to grab video streams from external sensor cameras and other CSI2 compliant sources.

Table 12-1529 shows the DPHY\_RX module allocation across device domains

**Table 12-1529. DPHY\_RX Modules Allocation Across Device Domains**

Module Instance	Domain		
	WKUP	MCU	MAIN
DPHY_RX0	-	-	✓
DPHY_RX1	-	-	✓

Figure 12-1134 shows the DPHY\_RX module overview.



A.  $i = 0$  to  $1$

**Figure 12-1155. DPHY\_RX Module Overview**

#### 12.7.2.1.1 DPHY\_RX Features

The DPHY\_RX module supports the following features:

- Compliant to MIPI D-PHY standard v1.2
- Supports up to 4 data and 1 clock lanes
- Supports up to 2.5 Gbps (with deskew) and 1.5 Gbps (without deskew) per data lane
- Clock lane control / interface logic type is CIL-SCNN for HS and low power receiving:

1. **Slave**
  2. **Clock**
  3. **N/A** forward, **N/A** reverse escape mode features
- Data lane control / interface logic type is CIL-SFAN for HS and low power receiving:
    1. **Slave**
    2. **Forward** direction only for high speed mode
    3. **All** forward direction escape mode features are supported
    4. **No** reverse direction escape mode features are supported
  - Data lanes can be independently operated in HS or ULP mode
  - Swapping of DP/DN signals within each clock/data pair (Facilitated by CSI\_RX\_IF controller)

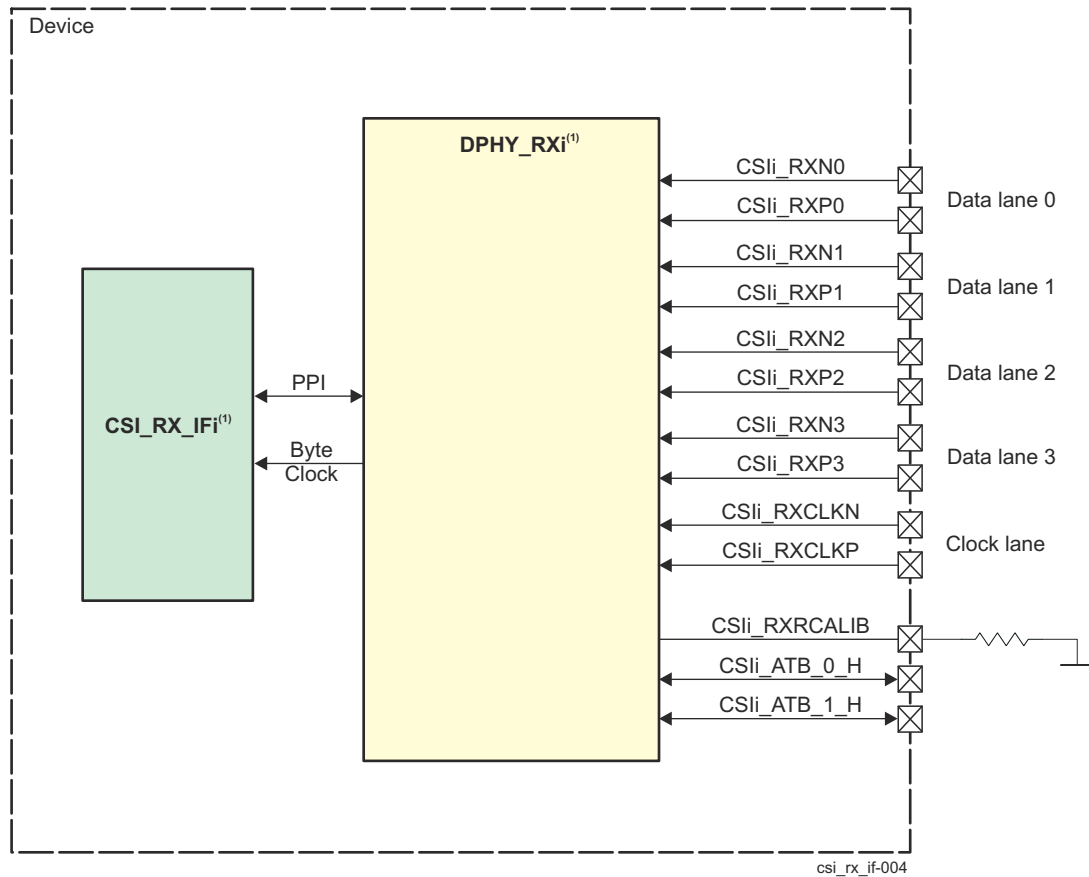
#### **12.7.2.1.2 DPHY\_RX Not Supported Features**

The DPHY\_RX does not support the following features:

- Swapping of clock & data lanes to ease PCB routing

### 12.7.2.2 DPHY\_RX Environment

This section describes the DPHY\_RX application fields from an environment point of view (external connections).



A. i = 0 to 1

**Figure 12-1156. DPHY\_RX Enviroment**

[Table 12-1530](#) describes the external signals of the DPHY\_RX.

**Table 12-1530. DPHY\_RX I/O Signals**

Device Level Signal	I/O	Description
<b>DPHY_RX0</b>		
CSI0_RXN0	I	Lane 0 Receive Differential Data (Negative)
CSI0_RXP0	I	Lane 0 Receive Differential Data (Positive)
CSI0_RXN1	I	Lane 1 Receive Differential Data (Negative)
CSI0_RXP1	I	Lane 1 Receive Differential Data (Positive)
CSI0_RXN2	I	Lane 2 Receive Differential Data (Negative)
CSI0_RXP2	I	Lane 2 Receive Differential Data (Positive)
CSI0_RXN3	I	Lane 3 Receive Differential Data (Negative)
CSI0_RXP3	I	Lane 3 Receive Differential Data (Positive)
CSI0_RXCLKN	I	Lane 3 Receive Differential Clock (Negative)
CSI0_RXCLKP	I	Lane 3 Receive Differential Clock (Positive)
CSI0_RXRCALIB	A	Pin for external calibration resistor. An external resistor must be connected between this pin and package ground. Refer to the device-specific Datasheet for a recommended resistor value.
<b>DPHY_RX1</b>		
CSI1_RXN0	I	Lane 0 Receive Differential Data (Negative)
CSI1_RXP0	I	Lane 0 Receive Differential Data (Positive)
CSI1_RXN1	I	Lane 1 Receive Differential Data (Negative)
CSI1_RXP1	I	Lane 1 Receive Differential Data (Positive)
CSI1_RXN2	I	Lane 2 Receive Differential Data (Negative)
CSI1_RXP2	I	Lane 2 Receive Differential Data (Positive)
CSI1_RXN3	I	Lane 3 Receive Differential Data (Negative)
CSI1_RXP3	I	Lane 3 Receive Differential Data (Positive)
CSI1_RXCLKN	I	Lane 3 Receive Differential Clock (Negative)
CSI1_RXCLKP	I	Lane 3 Receive Differential Clock (Positive)
CSI1_RXRCALIB	A	Pin for external calibration resistor. An external resistor must be connected between this pin and package ground. Refer to the device-specific Datasheet for a recommended resistor value.

### Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

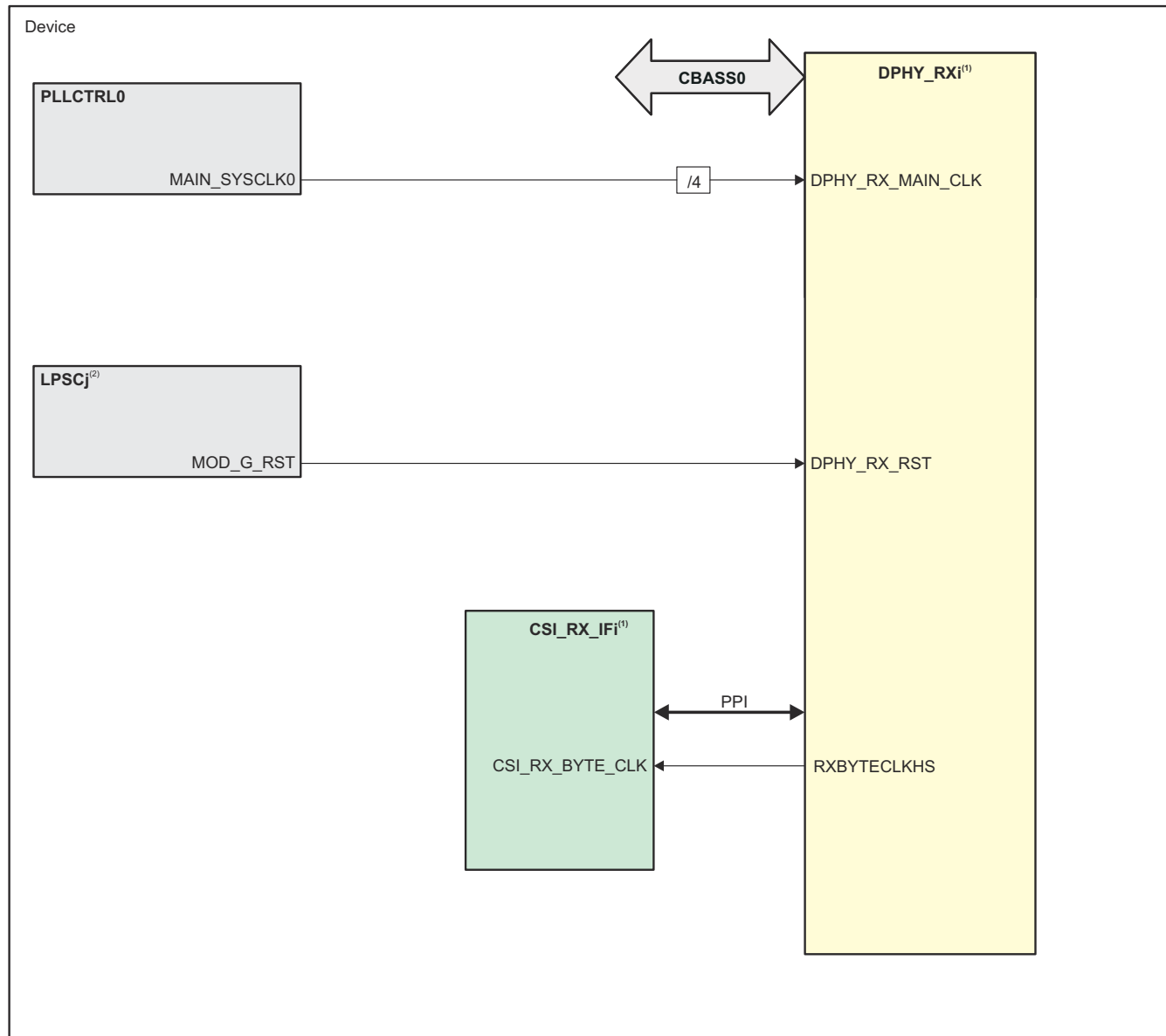


### 12.7.2.3 DPHY\_RX Integration

This section describes the DPHY\_RX integration in the device, including information about clocks, resets, and hardware requests.

#### 12.7.2.3.1 DPHY\_RX Integration in MAIN Domain

There are several DPHY\_RX modules integrated in the device MAIN domain. [Figure 12-1157](#) shows the integration of DPHY\_RX modules.



csi\_rx\_if-005

- A.  $i = 0$  to  $1$
- B.  $j$  = LPSC index, see [Table 12-1531](#)

**Figure 12-1157. DPHY\_RX Integration**

[Table 12-1531](#) through [Table 12-1532](#) summarize the integration of DPHY\_RX in the device MAIN domain.

**Table 12-1531. DPHY\_RX Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
DPHY_RX0	PSC0	PD2	LPSC58	CBASS0
DPHY_RX1	PSC0	PD2	LPSC59	CBASS0

**Table 12-1532. DPHY\_RX Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
DPHY_RX0	DPHY_RX_MAIN_CLK	MAIN_SYSCLK0 / 4	PLLCTRL0	Main functional clock.
	CSI_RX_BYTE_CLK	RXBYTECLKHS	DPHY_RX0	The byte clock is the clock supplied by the DPHY_RX.
DPHY_RX1	DPHY_RX_MAIN_CLK	MAIN_SYSCLK0 / 4	PLLCTRL0	Main functional clock.
	CSI_RX_BYTE_CLK	RXBYTECLKHS	DPHY_RX1	The byte clock is the clock supplied by the DPHY_RX.
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
DPHY_RX0	DPHY_RX_RST	MOD_G_RST	LPSC58	Asynchronous module global reset.
DPHY_RX1	DPHY_RX_RST	MOD_G_RST	LPSC59	Asynchronous module global reset.

## 12.7.2.4 DPHY\_RX Functional Description

### 12.7.2.4.1 DPHY\_RX Programming Guide

#### 12.7.2.4.1.1 Overview

This section details specific steps on how to program the DPHY\_RX

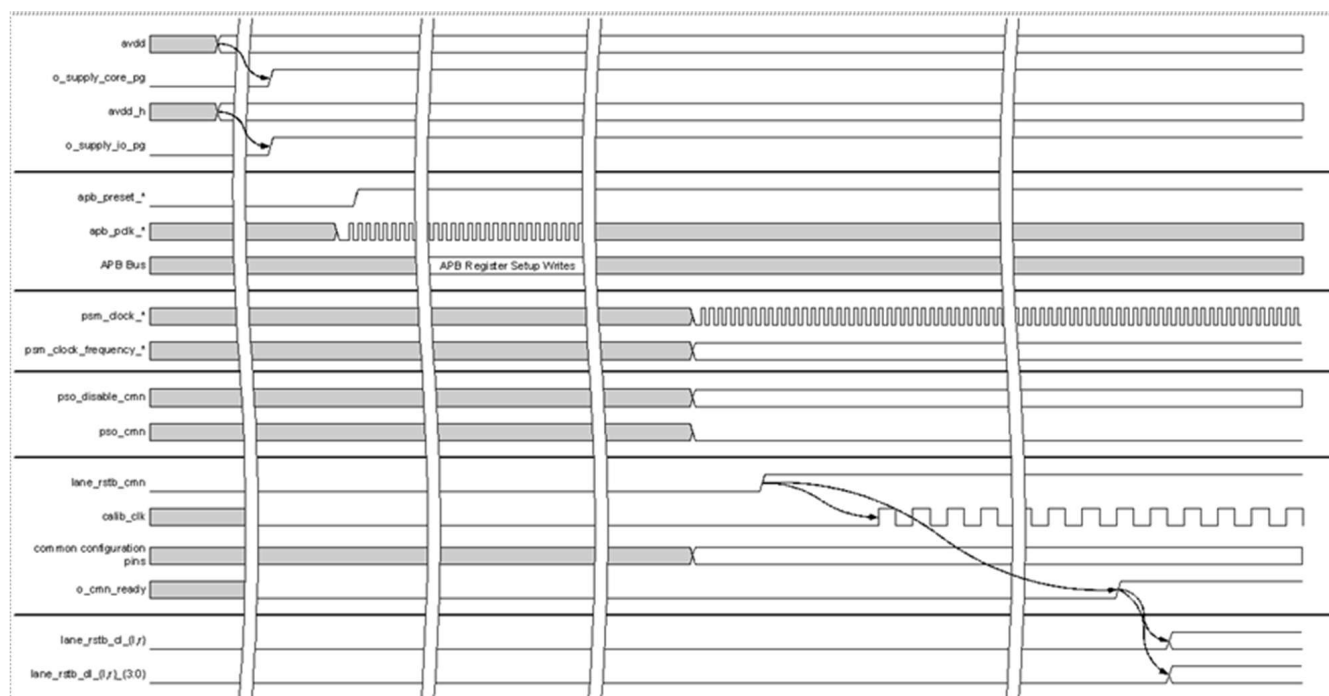
#### 12.7.2.4.1.2 Initial Configuration Programming

The DPHY\_RX operation shows the registers that are written during the startup sequence.

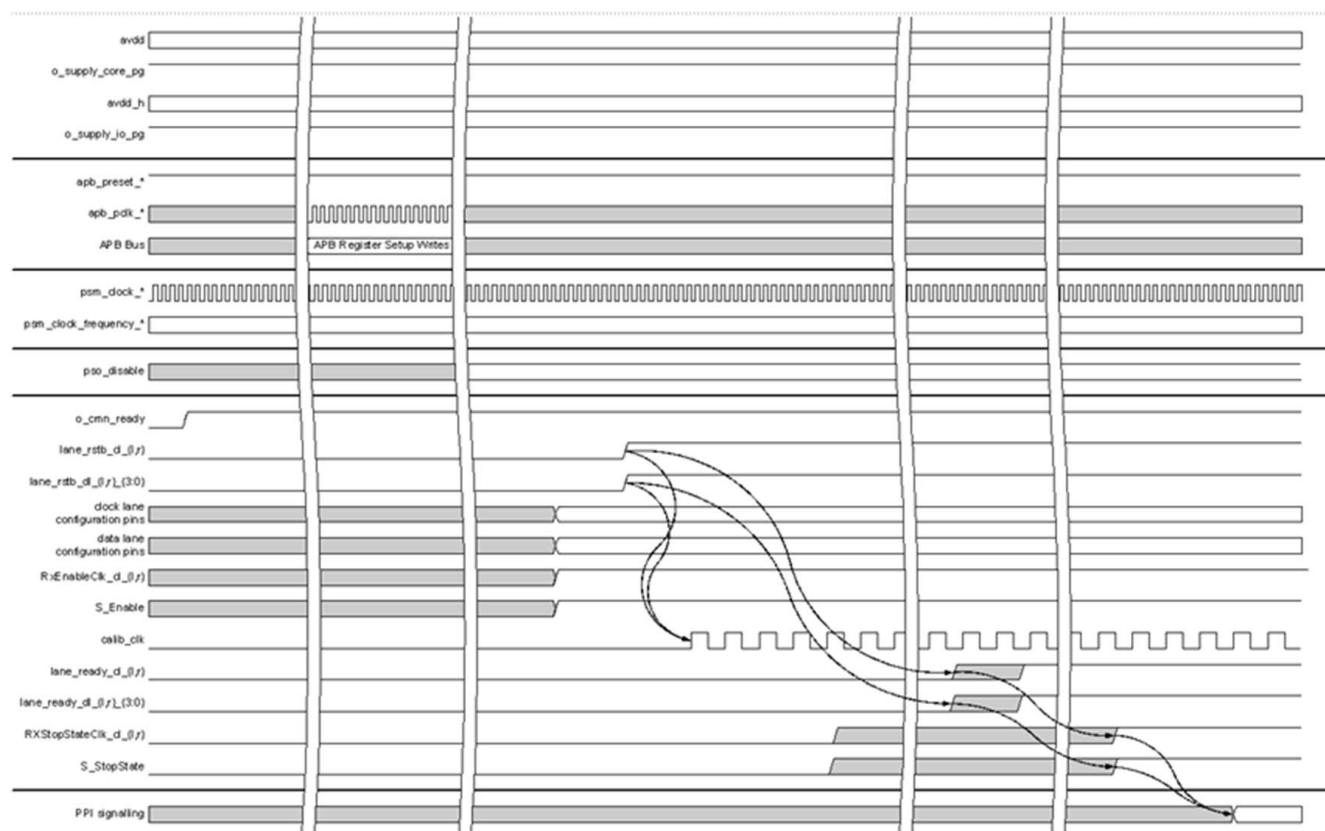
This section provides description and timing diagrams for DPHY\_RX operation. Unless otherwise described in this subsection, the DPHY\_RX PPI interface is compliant with the timing diagrams described in the MIPI D-PHY v1.2 specification.

##### 12.7.2.4.1.2.1 Start-up Sequence Timing Diagram

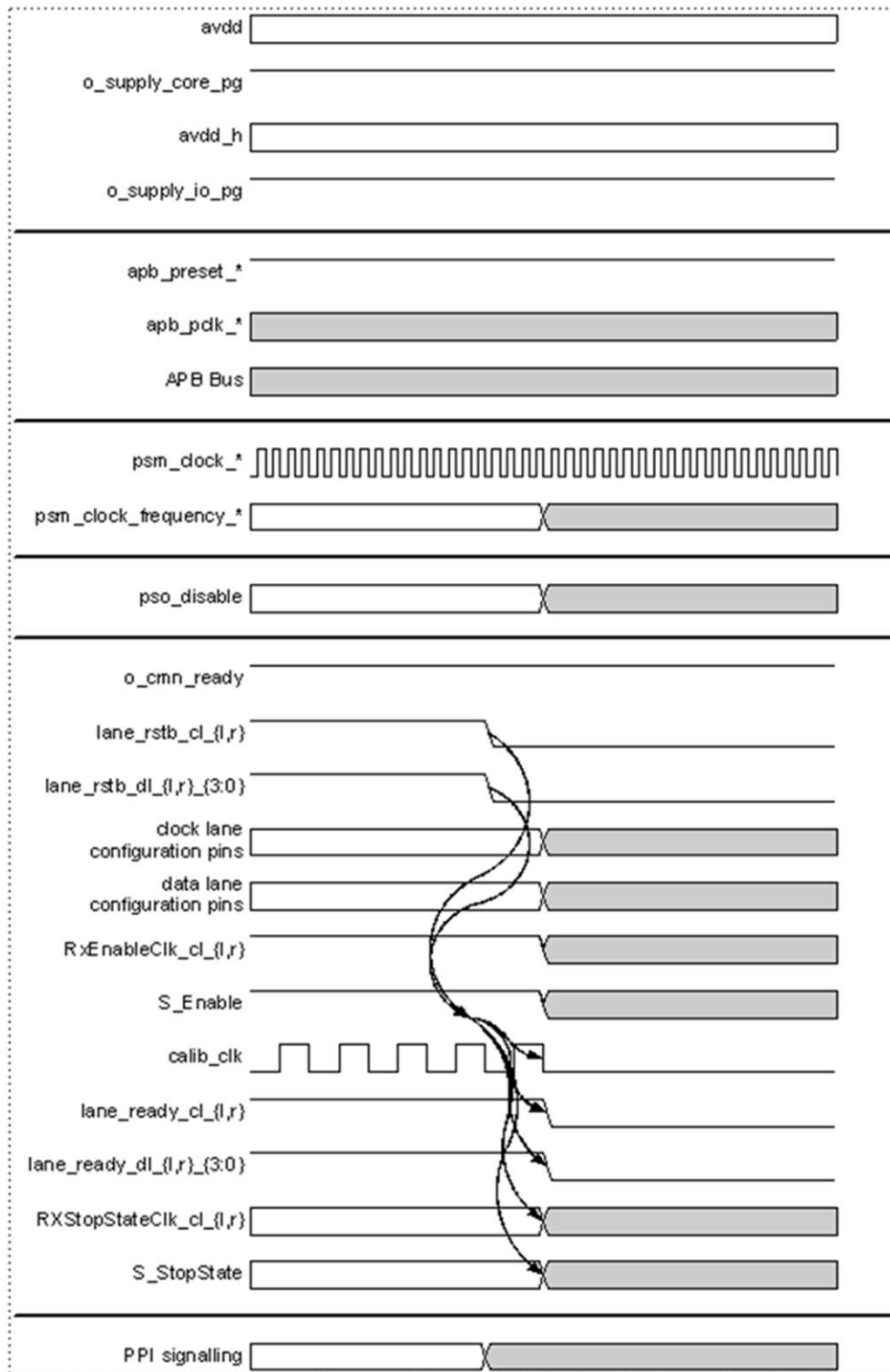
The common configuration pins noted in [Figure 12-1158](#) are ipconfig\_cmnn, pll\_ipdiv, pll\_fbdiv, pll\_opdiv, psm\_clock\_freq\_cmnn. Drive these pins as per the pin description given in PHY pin list.



**Figure 12-1158. Common Power Up and Initialization Timing Diagram**



**Figure 12-1159. Lane Power Up and Initialization Timing Diagram**



**Figure 12-1160. PHY Disable Timing Diagram**

For initial set up, the DPHY\_RX must be configured/set (registers and configuration input pins) for the common module prior to releasing it from reset, and the lane modules prior to releasing them from reset. Registers shall be configured (as required) between releasing the APB from reset and releasing the common / lanes from reset.

Note that in some cases, the option exists to configure a function using either a pin or a register. In such cases, both options will be specified and the you can select the preferred option.

### 12.7.2.4.1.3 Common Configuration

**Table 12-1533. Common Configuration-Related Setup**

Configuration Register / Pin	Configuration Requirement
PHY Pin: psm_clock_freq PHY Register: DPHY_RX_VBUS2APB_PCS_TX_DIG_TBIT 1	Set the PMA state machine clock frequency divider. Set either the pin or the register, as specified in the respective description, for the required PMA state machine clock.
PHY Pin: ipconfig_cmh PHY Register: DPHY_RX_MMR_SLV_LANE[11-9] IPCONFIG_CMN	Set the clock lane configuration. Set as specified in the description for the required clock lane configuration.
PMA Register: DPHY_RX_VBUS2APB_CMN0_CMN_DIG_T BIT2[0] O_CMN_SSM_EN, DPHY_RX_VBUS2APB_CMN0_CMN_DIG_T BIT2[10] O_CMN_RX_MODE_EN	Enable the startup state machines for TX mode of operation. Set both register bits to 1'b1.
PMA Register: DPHY_RX_VBUS2APB_CMN0_CMN_DIG_T BIT35	Set the RX oscillator calibration feedback clock counter start values. Set the register fields, as specified in the description for the required PMA state machine clock and oscillator clock.
PHY Register: DPHY_RX_VBUS2APB_PCS_TX_DIG_TBIT 2	Set the required power island phase 2 time. Set the register to 32'hAAAAAAAA.
PHY Register: DPHY_RX_VBUS2APB_PCS_TX_DIG_TBIT 3[7:4] POWER_SW_2_TIME_CL_R, DPHY_RX_VBUS2APB_PCS_TX_DIG_TBIT 3[3:0] POWER_SW_2_TIME_CL_L	Set the required power island phase 2 time. Set the register to 8'hAA

### 12.7.2.4.1.4 Lane Configuration

**Table 12-1534. Lane Configuration-Related Setup**

Configuration Register / Pin	Configuration Requirement
PHY Register: DPHY_RX_VBUS2APB_PCS_TX_DIG_TBIT 0	Set the lane band control value. Set the register fields, as specified in the register description for the required data rate.
PMA Register (clock lanes): DPHY_RX_VBUS2APB_CLK0_RX_DIG_TBI T2[18:15] RXDA_FREQ_BAND_SEL1 PMA Register (clock lanes): DPHY_RX_VBUS2APB_CLK0_RX_DIG_TBI T2[13:10] RXDA_FREQ_BAND_SEL2	Set the analog frequency band selects, as specified in the register description for the required data rate.
PMA Register (data lanes): DPHY_RX_VBUS2APB_DL0_RX_DIG_TBIT 0 - DPHY_RX_VBUS2APB_DL3_RX_DIG_TBIT 0[21] TM_1P5TO2P5G_MODE_EN	Set the analog data rate select, as specified in the register description for the required data rate.
PMA Register (data lanes): DPHY_RX_VBUS2APB_DL0_RX_DIG_TBIT 0 - DPHY_RX_VBUS2APB_DL3_RX_DIG_TBIT 0[16] TM_SETTLE_COUNT_SEL and [15-9] TM_SETTLE_COUNT	Set the HS-settle counter, as specified in the register description for the required data rate.
PMA Register (data lanes): DPHY_RX_VBUS2APB_DL0_RX_DIG_TBIT 3 - DPHY_RX_VBUS2APB_DL3_RX_DIG_TBIT 3	Set the iteration and initialization wait timer values to create a delay of 500 nSec as a function of the PSM clock rate.

**Table 12-1534. Lane Configuration-Related Setup (continued)**

Configuration Register / Pin	Configuration Requirement
PMA Register (data lanes): DPHY_RX_VBUS2APB_DL0_RX_DIG_TBIT 5 - DPHY_RX_VBUS2APB_DL3_RX_DIG_TBIT 5	Set the iteration and initialization wait timer values to create a delay of 500 nSec as a function of the PSM clock rate.
PMA Register (data lanes): DPHY_RX_VBUS2APB_DL0_RX_DIG_TBIT 7 - DPHY_RX_VBUS2APB_DL3_RX_DIG_TBIT 7	Set the iteration and initialization wait timer values to create a delay of 500 nSec as a function of the PSM clock rate.
PMA Register (data lanes): DPHY_RX_VBUS2APB_DL0_RX_DIG_TBIT 9 - DPHY_RX_VBUS2APB_DL3_RX_DIG_TBIT 9	Set the iteration and initialization wait timer values to create a delay of 500 nSec as a function of the PSM clock rate.
PMA Register (data lanes): DPHY_RX_VBUS2APB_DL0_RX_DIG_TBIT 12 - DPHY_RX_VBUS2APB_DL3_RX_DIG_TBIT 12	Set the iteration and initialization wait timer values to create a delay of 500 nSec as a function of the PSM clock rate.

#### 12.7.2.4.1.5 Procedure: Clock Lane Low Power Analog Receiver Functions Test

##### 12.7.2.4.1.5.1 Description of Procedure

This procedure describes how the low power analog receiver functions (LPRX, and ULPRX) can be tested. The process used in this test is to enable test MUXes for the low power receiver functions, then write registers in common to drive LP signaling states to the lanes, and read registers in the lanes to detect the current state of the analog functions in the lane.

##### 12.7.2.4.1.5.2 Details of the Procedure

- Follow the procedure described in section [Section 12.7.2.4.1.2](#), but do not release the common or lanes from reset.
- Write the following registers.
  - PMA Register (cmn): DPHY\_RX\_VBUS2APB\_CMN0\_CMN\_DIG\_TBIT21[0] O\_RX\_DIG\_BIST\_EN = 1'b1, to enable the analog BIST power island in common.
  - PMA Register (cmn): DPHY\_RX\_VBUS2APB\_CMN0\_CMN\_DIG\_TBIT28[23:20] = 4'b1111, to enable the clock lane diagnostic low power override functions, and drive LP11 from common to the clock lanes.
  - PMA Register (clock lanes): DPHY\_RX\_VBUS2APB\_DL0\_RX\_DIG\_TBIT2[24] = 1'b1, to enable the diagnostic low power override MUXes in the analog.
  - PMA Register (clock lanes): DPHY\_RX\_VBUS2APB\_DL0\_RX\_DIG\_TBIT2[1:0] = 2'b11, to force the analog ULPS receiver to be enabled.
- Release the common from reset, and wait for DPHY\_RX\_VBUS2APB\_ISO\_PHY\_ISO\_CMN\_CTRL[5] O\_CMN\_READY to be driven to 1'b1.
- Release the clock and data lanes from reset, and wait for DPHY\_RX\_VBUS2APB\_ISO\_PHY\_ISO\_CMN\_CTRL[8] LANE\_READY\_CMN to be driven to 1'b1.
- Poll PMA Register (CMN): DPHY\_RX\_VBUS2APB\_CMN0\_CMN\_DIG\_TBIT56[20-11] I\_CMN\_RX\_SSM\_STATE until it is set to 9'b100000000.
- Repeat the following for each of the following values, to drive LP values from common and test the analog functions in the clock lanes: 2'b01, 2'b00, 2'b10, 2'b11.
  - Write the PMA Register (cmn): DPHY\_RX\_VBUS2APB\_CMN0\_CMN\_DIG\_TBIT28[22,20] = value.
  - Read and confirm the PMA Register (clock lanes):  
DPHY\_RX\_VBUS2APB\_CLK0\_RX\_DIG\_TBIT4[13:12] = value.
  - Read and confirm the PMA Register (clock lanes): DPHY\_RX\_VBUS2APB\_CLK0\_RX\_DIG\_TBIT4[9:8] = value.

### 12.7.2.4.1.6 Procedure: Data Lane Low Power Analog Receiver Functions Test

#### 12.7.2.4.1.6.1 Description of Procedure

This procedure describes how the low power analog receiver functions (LPRX, and ULPRX) can be tested. The process used in this test is to enable test MUXes for the low power receiver functions, then write registers in common to drive LP signaling states to the lanes, and read registers in the lanes to detect the current state of the analog functions in the lane.

#### 12.7.2.4.1.6.2 Details of the Procedure

1. Follow the procedure described in section [Section 12.7.2.4.1.2](#), but do not release the common or lanes from reset.
2. Write the following registers.
  - PMA Register (cmn): DPHY\_RX\_VBUS2APB\_CMN0\_CMN\_DIG\_TBIT21[0] O\_RX\_DIG\_BIST\_EN = 1'b1, to enable the analog BIST power island in common.
  - PMA Register (cmn): DPHY\_RX\_VBUS2APB\_CMN0\_CMN\_DIG\_TBIT28[19:16] = 4'b1111, to enable the data lane diagnostic low power override functions, and drive LP11 from common to the data lanes.
  - PMA Register (data lanes): DPHY\_RX\_VBUS2APB\_DL0\_RX\_DIG\_TBIT30[1] TM\_LPRX\_BIST\_EN = 1'b1, to enable the diagnostic low power override MUXes in the analog.
  - PMA Register (data lanes): DPHY\_RX\_VBUS2APB\_DL0\_RX\_DIG\_TBIT1[9:8] = 2'b11, to force the analog ULPS receiver to be enabled.
3. Release the common from reset, and wait for DPHY\_RX\_VBUS2APB\_ISO\_PHY\_ISO\_CMN\_CTRL[5] O\_CMN\_READY to be driven to 1'b1.
4. Release the clock and data lanes from reset, and wait for DPHY\_RX\_VBUS2APB\_ISO\_PHY\_ISO\_CMN\_CTRL[8] LANE\_READY\_CMN to be driven to 1'b1.
5. Poll PMA Register (CMN): DPHY\_RX\_VBUS2APB\_CMN0\_CMN\_DIG\_TBIT56[20-11] I\_CMN\_RX\_SSM\_STATE until it is set to 9'b100000000.
6. Repeat the following for each of the following values, to drive LP values from common and test the analog functions in the data lanes: 2'b01, 2'b00, 2'b10, 2'b11.
  - a. Write the PMA Register (cmn): DPHY\_RX\_VBUS2APB\_CMN0\_CMN\_DIG\_TBIT28[18,16] = value.
  - b. Read and confirm the PMA Register (data lanes): DPHY\_RX\_VBUS2APB\_DL0\_RX\_DIG\_TBIT34[13:12] = value.
  - c. Read and confirm the PMA Register (data lanes): DPHY\_RX\_VBUS2APB\_DL0\_RX\_DIG\_TBIT34[9:8] = value.

### 12.7.2.4.1.7 Procedure: Clock and Data Lane High Speed Receiver BIST Functions Test

#### 12.7.2.4.1.7.1 Description of Procedure

This procedure describes how to use the high speed data BIST generation and checking functions to test the high speed data receiver functions. The process used in this test is to enable test MUXes for the low power and high speed data receiver functions, then write registers to setup, run and check the results of the BIST functions.

#### 12.7.2.4.1.7.2 Details of the Procedure

1. Set up IP for BIST operations.
  - a. Follow the procedure described in section [Section 12.7.2.4.1.2](#), but do not release the common or lanes from reset.
  - b. Write the following registers.
    - PMA Register (cmn): DPHY\_RX\_VBUS2APB\_CMN0\_CMN\_DIG\_TBIT21[0] O\_RX\_DIG\_BIST\_EN = 1'b1, to enable the analog BIST power island in common.
    - PMA Register (clock lanes): DPHY\_RX\_VBUS2APB\_DL0\_RX\_DIG\_TBIT2 - DPHY\_RX\_VBUS2APB\_DL3\_RX\_DIG\_TBIT2 [24] = 1'b1, to enable the diagnostic low power override MUXes in the analog.
    - PMA Register (clock lanes): DPHY\_RX\_VBUS2APB\_DL0\_RX\_DIG\_TBIT2 - DPHY\_RX\_VBUS2APB\_DL3\_RX\_DIG\_TBIT2 [21:20] = 2'b11, to enable the diagnostic high speed override MUXes in the analog.



- PMA Register (data lanes): DPHY\_RX\_VBUS2APB\_DL0\_RX\_DIG\_TBIT30 - DPHY\_RX\_VBUS2APB\_DL3\_RX\_DIG\_TBIT30[1:0] = 2'b11, to enable the diagnostic low power and high speed override MUXes in the analog.
- c. Release the common from reset, and wait for DPHY\_RX\_VBUS2APB\_ISO\_PHY\_ISO\_CMN\_CTRL[5] O\_CMN\_READY to be driven to 1'b1.
- d. Release the clock and data lanes from reset, and wait for DPHY\_RX\_VBUS2APB\_ISO\_PHY\_ISO\_CMN\_CTRL[8] LANE\_READY\_CMN to be driven to 1'b1.
- e. Poll PMA Register (cmn): DPHY\_RX\_VBUS2APB\_CMN0\_CMN\_DIG\_TBIT56[20-11] I\_CMN\_RX\_SSM\_STATE until it is set to 9'b100000000.
- f. When running BIST at 1.5 Gbps, do the following, otherwise skip this if running BIST at 2.5 Gbps.
  - i. Write PMA Register (cmn): DPHY\_RX\_VBUS2APB\_CMN0\_CMN\_DIG\_TBIT36[1] O\_RX\_TM\_SEL\_1P5G\_M\_ODE = 1'b1.
  - ii. Wait 5 uSec.
- 2. Run BIST operations.
  - a. At this point, the BIST generator and checker registers can be written to the values required to generate the desired data patterns. One should see the BIST register descriptions for information about programming the various fields for creating different data patterns. If no registers are written, the BIST default pattern will be generated and checked.
  - b. If running infinite BIST mode is desired, write the PMA Register (cmn): DPHY\_RX\_VBUS2APB\_CMN0\_CMN\_DIG\_TBIT24[15] BIST\_INF\_MODE = 1'b1, to enable infinite BIST mode.
- 3. Write PMA Register (data lanes): DPHY\_RX\_VBUS2APB\_DL0\_RX\_DIG\_TBIT27[0] = 1'b1, to enable the pattern checkers in the data lanes.
- 4. Write PMA Register (cmn): CMN\_DIG\_TBIT22[0] TM\_BIST\_EN = 1'b1, to enable the pattern generator in the common.
- 5. If infinite BIST mode was enabled, do the following.
  - a. Allow the BIST to run as long as required.
  - b. Write the PMA Register (cmn lanes): DPHY\_RX\_VBUS2APB\_CMN0\_CMN\_DIG\_TBIT24[15] BIST\_INF\_MODE = 1'b0, to disable infinite BIST mode.
- 6. Wait for a minimum of 100 nSec.
- 7. Poll PMA Register (cmn): DPHY\_RX\_VBUS2APB\_CMN0\_CMN\_DIG\_TBIT50[1] BIST\_COMPLETE until it is set to 1'b1, indicating that the BIST pattern generation process is complete.
- 8. Wait 1 uSec.
- 9. Read and confirm the PMA Register (data lanes): DPHY\_RX\_VBUS2APB\_DL0\_RX\_DIG\_TBIT48 - DPHY\_RX\_VBUS2APB\_DL3\_RX\_DIG\_TBIT48[0] W\_DRX\_BIST\_PASS = 1'b1, indicating the BIST checker has indicated a pass condition.
- 10. Read and confirm the PMA Register (data lanes): DPHY\_RX\_VBUS2APB\_DL0\_RX\_DIG\_TBIT48 - DPHY\_RX\_VBUS2APB\_DL3\_RX\_DIG\_TBIT48[1] R\_PAT\_CHE\_SYNC = 1'b1, indicating the BIST checker has observed a marker symbol.
- 11. Read and confirm the PMA Register (data lanes): DPHY\_RX\_VBUS2APB\_DL0\_RX\_DIG\_TBIT48 - DPHY\_RX\_VBUS2APB\_DL3\_RX\_DIG\_TBIT48[2] W\_BIST\_ERROR = 1'b0, indicating the BIST checker has not detected any errors.
- 12. Read and confirm the PMA Register (data lanes): DPHY\_RX\_VBUS2APB\_DL0\_RX\_DIG\_TBIT47 - DPHY\_RX\_VBUS2APB\_DL3\_RX\_DIG\_TBIT47[15:0] W\_PAT\_CHE\_PKT\_COUNT = the generated number of packets (15 by default).
- 13. Read and confirm the PMA Register (data lanes): DPHY\_RX\_VBUS2APB\_DL0\_RX\_DIG\_TBIT47 - DPHY\_RX\_VBUS2APB\_DL3\_RX\_DIG\_TBIT47[31:16] W\_PAT\_CHE\_ERROR\_COUNT = 16'h0000, indicating the BIST checker error count is 0.
- 14. Clear the internal state of the BIST pattern checkers by doing the following:
  - a. Read PMA Register (data lanes): DPHY\_RX\_VBUS2APB\_DL0\_RX\_DIG\_TBIT29 - DPHY\_RX\_VBUS2APB\_DL3\_RX\_DIG\_TBIT29[28] TM\_CLEAR\_BIST.
  - b. Write PMA Register (data lanes): DPHY\_RX\_VBUS2APB\_DL0\_RX\_DIG\_TBIT29 - DPHY\_RX\_VBUS2APB\_DL3\_RX\_DIG\_TBIT29[28] TM\_CLEAR\_BIST = inverted value of what was read from this bit in the previous step.

15. Write PMA Register (data lanes): DPHY\_RX\_VBUS2APB\_DL0\_RX\_DIG\_TBIT27 - DPHY\_RX\_VBUS2APB\_DL3\_RX\_DIG\_TBIT27[0] TM\_BIST\_EN = 1'b0, to disable the pattern checkers in the data lanes.
16. Clear the internal state of the BIST pattern generator by doing the following:
  - a. Write the PMA Register (cmn): DPHY\_RX\_VBUS2APB\_CMN0\_CMN\_DIG\_TBIT24[6] = 1'b1,
  - b. Write the PMA Register (cmn): CMN\_DIG\_TBIT24[6] BIST\_CLEAR = 1'b0.
17. Write the PMA Register (cmn): DPHY\_RX\_VBUS2APB\_CMN0\_CMN\_DIG\_TBIT22[0] BIST\_CONTROLLER\_EN = 1'b0, to disable the BIST pattern generator.
18. If desired, the run BIST operations can be repeated without needing to run set up the IP for BIST operations again.

### 12.7.3 Camera Streaming Interface Transmitter (CSI\_TX\_IF)

The following sections describe the camera streaming transmitter interface (CSI\_TX\_IF) module in the device.

#### 12.7.3.1 CSI\_TX\_IF Overview

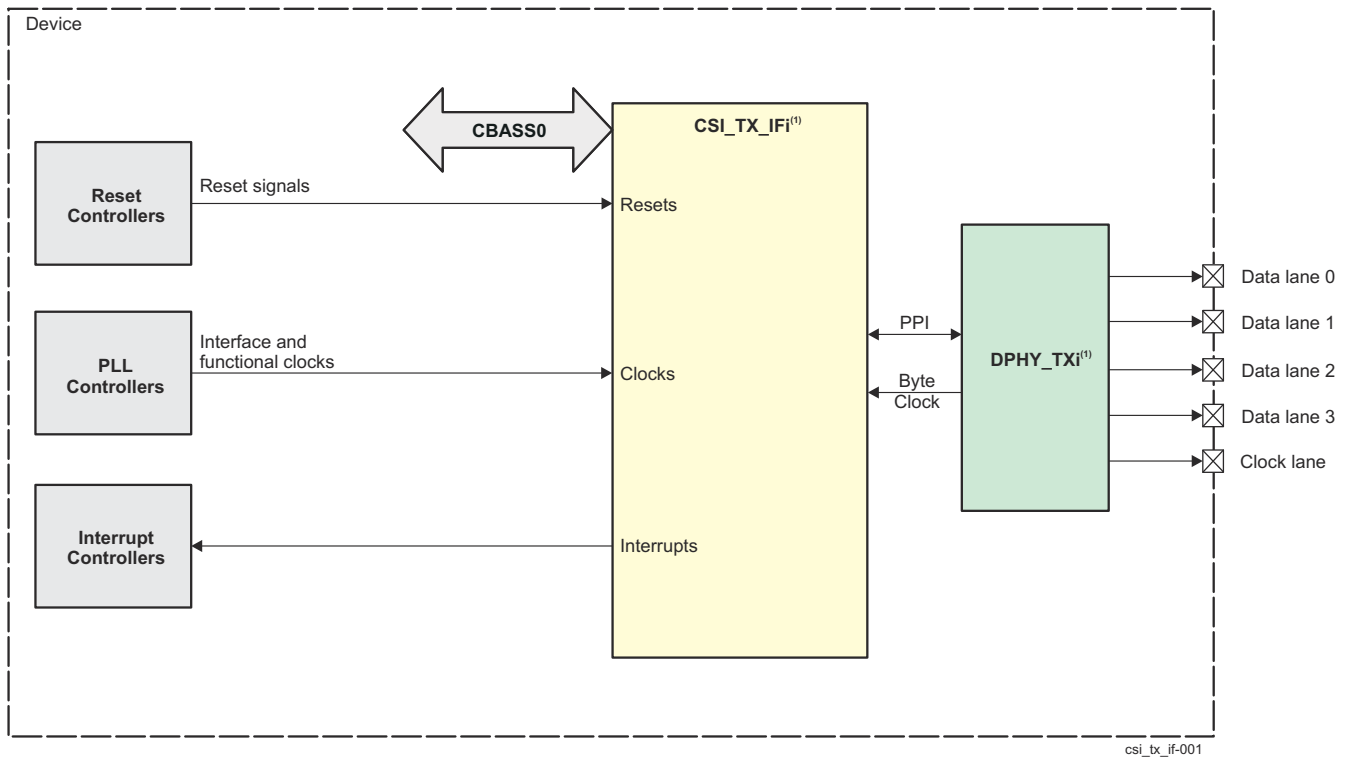
The integration of the CSI\_TX\_IF module allows the device to stream out video data from memory, or retransmit from the CSI receivers as an optional loopback output for diagnostics, debug, and test purposes.

The device has one instance of the CSI\_TX\_IF module. Table 12-1535 shows CSI\_TX\_IF module allocation within device domains.

**Table 12-1535. CSI\_TX\_IF0 Modules Allocation within Device Domains**

Module Instance	Domain		
	WKUP	MCU	MAIN
CSI_TX_IF0	-	-	✓

Figure 12-1161 shows the CSI\_TX\_IF module overview.



A.  $i = 0$

**Figure 12-1161. CSI\_TX\_IF Module Overview**

#### 12.7.3.1.1 CSI\_TX\_IF Features

The CSI\_TX\_IF module supports the following features:

- Compliant to MIPI CSI v1.3 and MIPI D-PHY v1.2
- Data rate up to 2.5 Gbps per lane (wire rate)
- Supports 1, 2, 3, or 4 Data Lane connection to DPHY\_TX
- Programmable formats including YUV420, YUV422, RGB, Raw, and User Defined (over 25 different formats supported); Limitations apply depending on the stream used
- 16 virtual channel support (partial MIPI CSI-2 v2.0 feature, see for limitations)
- Four configurable input streams supported:

- **Stream0:** DMA interface through a 128-bit PSI\_L connection for transfers from memory:
  - 128bit wide pixel data with bursting
  - ByteValid per byte in Last Data Phase (LDP)
  - 32 thread IDs supported (virtual channel and data type combinations); Flexible number of threads (32 Max). Arbitration to next thread is done by lowest thread ID number. Care needs to be taken by software that it does not send too many threads at any given time as that could cause delays for higher thread IDs.
  - Unpacking PSI\_L data and converting to CSI video format (see for limitations)
  - Internal FF based FIFO and external RAM based buffer
- **Stream1:** Color bar video data generator
  - 2 pixel wide
  - YUV422 8-bit format support only
  - Configurable frame/line size via registers
  - 1 programmable virtual channel
  - Internal FF based FIFO; No external buffer.
- **Stream2:** Re-transmit loopback from CSI\_RX\_IF (first VC)
  - 1 virtual channel supported only (see for limitations)
  - RAW formats support only
  - Internal based FIFO, 2k×32
  - Programmable registers for which single VC is used from CSI\_RX\_IF to retransmit
- **Stream3:** Re-transmit loopback from CSI\_RX\_IF (second VC)
  - 1 virtual channel supported only (see for limitations)
  - RAW formats support only
  - Internal based FIFO, 2k×32
  - Programmable registers for which single VC is used from CSI\_RX\_IF to retransmit
- Functional and data path error interrupts
- ECC support on external RAMs

#### 12.7.3.1.2 CSI\_TX\_IF Not Supported Features

The CSI\_TX\_IF does not support the following features:

- MIPI CSI2 v2.0 scrambling
- MIPI CSI2 v2.0 optional no LP (low power state) between packets
- Cropping
- Pixel processing
- YUV420 8-bit legacy not supported
- Limitation on interleaving virtual channels per stream. Only 1 virtual channel supported per stream. The entire frame must be sent. CSI\_TX\_IF will only allow current active thread to go through till end of frame.

#### 12.7.3.2 CSI\_TX\_IF Environment

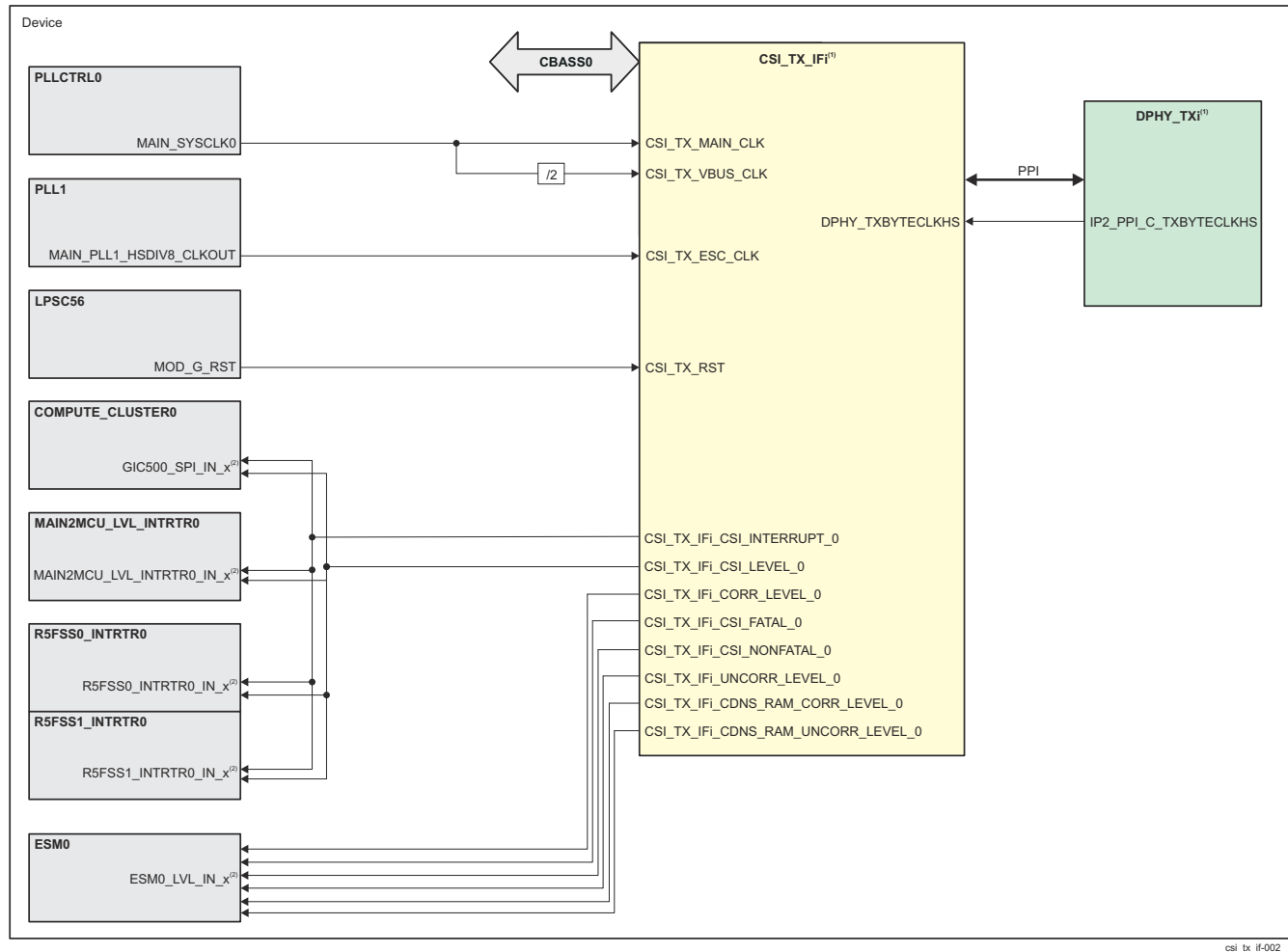
The CSI\_TX\_IF has no dedicated pins. At the device level, the CSI\_TX\_IF video output goes through the shared(with DSS\_DSI0) DPHY\_TX. See chapter [Section 12.8, Shared MIPI D-PHY Transmitter \(DPHY\\_TX\)](#).

#### 12.7.3.3 CSI\_TX\_IF Integration

This section describes the CSI\_TX\_IF integration in the device, including information about clocks, resets, and hardware requests.

#### 12.7.3.3.1 CSI\_TX\_IF Integration in MAIN Domain

There is one CSI\_TX\_IF module integrated in the device MAIN domain. Figure 12-1162 shows the integration of CSI\_TX\_IF module.



- A.  $i = 0$   
B.  $x$  = Interrupt port index, see

**Figure 12-1162. CSI\_TX\_IF Integration**

Table 12-1536 through Table 12-1538 summarize the integration of CSI\_TX\_IF in the device MAIN domain.

**Table 12-1536. CSI\_TX\_IF Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
CSI_TX_IF0	PSC0	PD2	LPSC56	CBASS0

**Table 12-1537. CSI\_TX\_IF Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description

**Table 12-1537. CSI\_TX\_IF Clocks and Resets (continued)**

CSI_TX_IF0	CSI_TX_MAIN_CLK	MAIN_SYSCLK0	PLLCTRL0	Main functional(sometimes referred to as pixel clock) clock.
	CSI_TX_VBUS_CLK	MAIN_SYSCLK0 / 2	PLLCTRL0	The VBUS clock runs at always half the speed of the CSI_TX_MAIN_CLK.
	CSI_TX_ESC_CLK	MAIN_PLL1_HSDIV8_CLKOUT	PLL1	20 MHz max clock input for low speed data transmission and some control signals.
	DPHY_TXBYTECLKHS	IP2_PPI_C_TXBYTECLKHS	DPHY_TX0	The byte clock is the clock supplied by the DPHY_TX.
<b>Resets</b>				
<b>Module Instance</b>	<b>Module Reset Input</b>	<b>Source Reset Signal</b>	<b>Source</b>	<b>Description</b>
CSI_TX_IF0	CSI_TX_RST	MOD_G_RST	LPSC56	Asynchronous module global reset, driving all collateral asynchronous resets of the 4 clock domains to the low state.

**Table 12-1538. CSI\_TX\_IF Hardware Requests**

<b>Interrupt Requests</b>					
<b>Module Instance</b>	<b>Module Interrupt Signal</b>	<b>Destination Interrupt Input</b>	<b>Destination</b>	<b>Description</b>	<b>Type</b>
CSI_TX_IF0	CSI_TX_IF0_CSI_INTERR UPT_0	GIC500_SPI_IN_180	COMPUTE_CLUSTER0	Global interrupt that various re-synchronized sources converge into interrupt generation.	Level
		MAIN2MCU_LVL_INTRTR0_IN_250	MAIN2MCU_LVL_INTRTR0	Global interrupt that various re-synchronized sources converge into interrupt generation.	Level
		R5FSS0_INTRTR0_IN_267	R5FSS0_INTRTR0	Global interrupt that various re-synchronized sources converge into interrupt generation.	Level
		R5FSS1_INTRTR0_IN_267	R5FSS1_INTRTR0	Global interrupt that various re-synchronized sources converge into interrupt generation.	Level
	CSI_TX_IF0_CSI_LEVEL_0	GIC500_SPI_IN_181	COMPUTE_CLUSTER0	Error interrupt that is generated under the following conditions: <ul style="list-style-type: none"> <li>Retransmit error on Stream2 (packet error on the stream interface)</li> <li>Retransmit error on Stream3 (packet error on the stream interface)</li> </ul>	Level
		MAIN2MCU_LVL_INTRTR0_IN_251	MAIN2MCU_LVL_INTRTR0	Error interrupt that is generated under the following conditions: <ul style="list-style-type: none"> <li>Retransmit error on Stream2 (packet error on the stream interface)</li> <li>Retransmit error on Stream3 (packet error on the stream interface)</li> </ul>	Level

**Table 12-1538. CSI\_TX\_IF Hardware Requests (continued)**

	R5FSS0_INTRTR0_IN_268	R5FSS0_INTRTR0	Error interrupt that is generated under the following conditions: <ul style="list-style-type: none"> <li>Retransmit error on Stream2 (packet error on the stream interface)</li> <li>Retransmit error on Stream3 (packet error on the stream interface)</li> </ul>	Level
	R5FSS1_INTRTR0_IN_268	R5FSS1_INTRTR0	Error interrupt that is generated under the following conditions: <ul style="list-style-type: none"> <li>Retransmit error on Stream2 (packet error on the stream interface)</li> <li>Retransmit error on Stream3 (packet error on the stream interface)</li> </ul>	Level
CSI_TX_IF0_CSI_FATAL_0	ESM0_LVL_IN_212	ESM0	ASF port fatal interrupt. Level sensitive.	Level
CSI_TX_IF0_CSI_NONFAT_AL_0	ESM0_LVL_IN_213	ESM0	ASF port non-fatal interrupt. Level sensitive.	Level
CSI_TX_IF0_CDNS_RAM_CORR_LEVEL_0	ESM0_LVL_IN_214	ESM0	Interrupt on internal FIFO RAM	Level
CSI_TX_IF0_CDNS_RAM_UNCORR_LEVEL_0	ESM0_LVL_IN_215	ESM0	Interrupt on internal FIFO RAM	Level
CSI_TX_IF0_CORR_LEVEL_0	ESM0_LVL_IN_216	ESM0	Interrupt on interface parity	Level
CSI_TX_IF0_UNCORR_LEVEL_0	ESM0_LVL_IN_217	ESM0	Interrupt on interface parity	Level

#### 12.7.3.4 CSI\_TX\_IF Functional Description

##### 12.7.3.4.1 CSI\_TX\_IF Block Diagram

Figure 12-1163 depicts a simplified internal block diagram of the CSI\_TX\_IF and its surroundings.

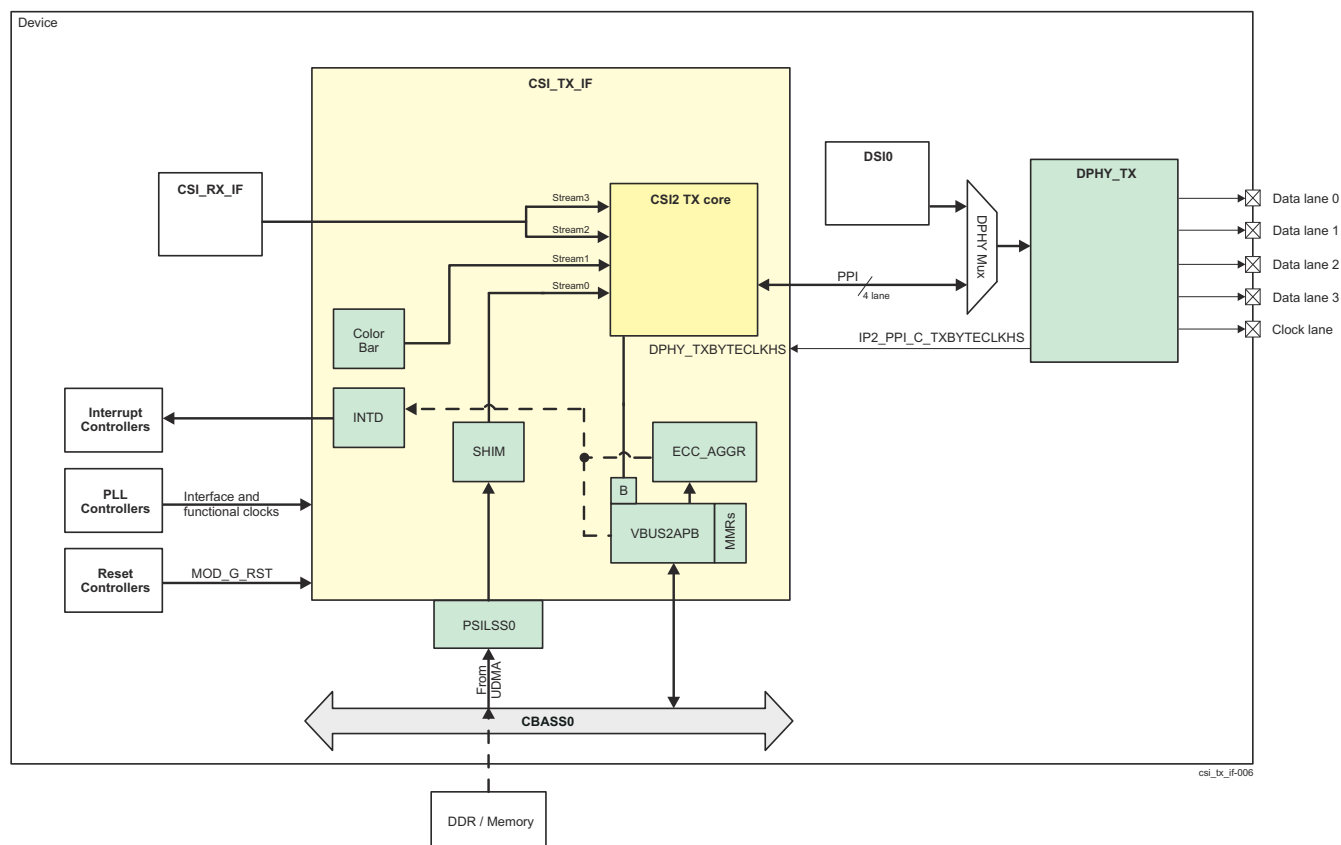


Figure 12-1163. CSI\_TX\_IF Block Diagram

The CSI2 core streams 4 channels of data to the several export paths.

**Stream0** goes through the PSI\_L that converts 128(4x32) bits pixel data and converts it to single 32-bit CSI streaming protocol of pixel data. Each 128 bit set of PSI\_L data is split into N M-bit data transfers based on the SIZE\_CFG bitfield setting in the CSI\_TX\_IF\_DMANTX<sub>j</sub> register for a given thread. See memory organization details in . Via register mapping in the CSI\_TX\_IF\_DMANTX<sub>j</sub> register the PSIL thread ID is mapped to a virtual channel and a data type. The CSI\_TX\_IF\_DMANTX<sub>j</sub> register is the thread filter. There are 32 dma context registers. Figure 12-1164 shows the stream to PSI\_L thread mapping.

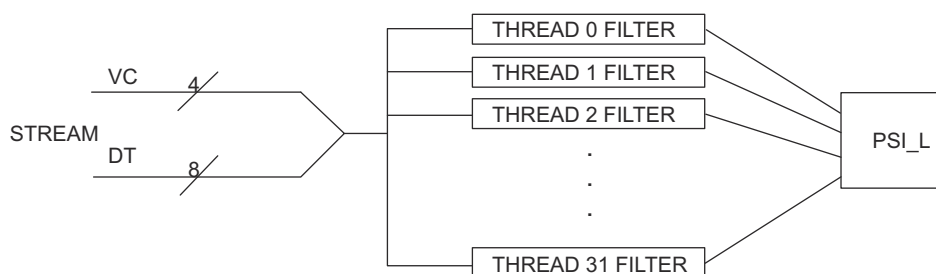


Figure 12-1164. CSI\_TX\_IF PSI\_L thread mapping

Software must take the following notes in consideration

- Current CSI\_TX\_IF does not support interleaved virtual channels. Software must send the entire frame per channel before starting a new channel. CSI\_TX\_IF logic will push back on thread IDs not of the current frame till frame is complete.



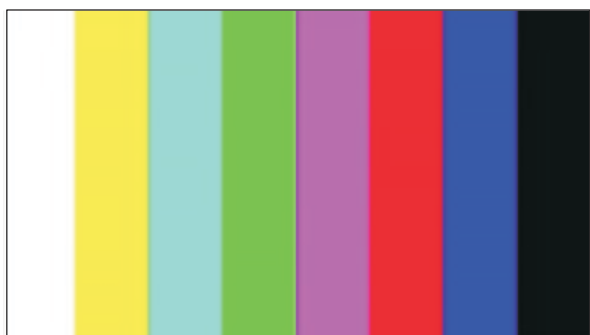
- CSI\_TX\_IF logic will re-arbitrate after completion of thread to lowest available thread IDs ready for transmit, thread ID 0 goes before thread ID1. This could mean that if there are multiple pending threads the higher order ones could be delayed indefinitely.
- DATA is assumed in a specific organization for CSI\_TX\_IF to export data correctly. See memory organization details in

Registers CSI\_TX\_IF\_L2L\_DELAY\_j control the line start to next line start. Also register CSI\_TX\_IF\_F2F\_DELAY\_y control the last line of frame to next start of frame first line start of next frame. Buffering of PSI\_L data is 2k×128 RAM.

**Stream1** streams data from the Color Bar generator. It supports YUV422 8bit format only. CSI\_TX\_IF\_COLOR\_PARAM register is used to control height and width information. Width is divided by 8 for each color section. Colors are defined below. Transfers are done on the Stream1 interface in 32-bit packed 8-bit pixel format(required programming in CSI\_TX\_IF core). Virtual channel and data type is configurable in CSI\_TX\_IF\_COLOR\_CNTL register. Data selection inside CSI\_TX\_IF controller maps data type information. CSI\_TX\_IF\_COLOR\_START\_DELAY register can configure delays from enabled to start. CSI\_TX\_IF\_COLOR\_LINE\_DELAY, and CSI\_TX\_IF\_COLOR\_FRAME\_DELAY register can delay last line to first line of new frame. Once enabled, it will continuously send out frames until disabled. Once disabled, it will stop at end of current frame. [Table 12-1539](#) shows the Color Bar format.

**Table 12-1539. CSI\_TX\_IF Color Bar format details**

Color	Y	CB	CR
White	235	128	128
Yellow	162	44	142
Cyan	131	156	44
Green	112	72	58
Magenta	84	184	198
Red	65	100	212
Blue	35	212	114
Black	16	128	128



csi\_tx\_if-015

**Figure 12-1165. CSI\_TX\_IF PSI\_L Color Bar sample output**

**Stream2 and Stream3** stream data via a loopback path from the CSI\_RX\_IF.

**CAUTION**

Program limitations apply on the CSI\_RX\_IF and CSI\_TX\_IF sides. The CSI\_TX\_IF only supports single, non interleaved virtual channels. Program restriction on CSI\_RX\_IF side must limit this.

The CSI\_TX\_IF will use Stream2 and Stream3 to support 2 virtual channels of CSI\_RX\_IF data. The CSI\_RX\_IF can stream interleaved virtual channels of data. It is limited to programming the CSI\_RX\_IF so that it only sends

data on up to 2 static virtual channels. The VC0 and VC1 fields in CSI\_TX\_IF\_RETRANS\_CNTL register need to be programmed to match which virtual channels are streamed on the CSI\_RX\_IF side so that the CSI\_TX\_IF can stream them. CSI\_TX\_IF can only stream a single virtual channel at any given time. Only RAW data formats can be used for this retransmit interface due to CSI\_TX\_IF limitations.

### Note

It is a requirement that only PSI\_L/DMA, color bar, or loopback streams are not active at the same point in time. The intended use model is not to have multiple combinations active at any given time. The main issue with this is that RAMs are not sized for concurrent operation and will likely overflow resulting in data loss. The 2 loopback streams can be active at the same time just not in addition to the color or DMA streams.

#### 12.7.3.4.2 CSI\_TX\_IF Hardware and Software Reset

An active low asynchronous hardware reset is provided to CSI\_TX\_IF by device LPSC. It is internally re-synchronized to the functional clock domain.

A software reset is triggered by configuring the CSI\_TX\_IF\_TX\_CONF[1] SOFT\_RESET\_REQUEST bit-field for protocol reset and/or module reset.

#### 12.7.3.4.3 CSI\_TX\_IF Clock Configuration

There are four clock domain in the CSI\_TX\_IF.

1. The CSI\_TX\_MAIN\_CLK(or also referred to as pixel clock) runs most of the logic. It needs to be same frequency as CSI\_RX\_MAIN\_CLK main clock (500MHz). When CSI\_TX\_MAIN\_CLK is operating lower than 312.5MHz, then the clock is essentially limiting the clock rate of the DPHY\_TX. Said another way, CSI\_TX\_MAIN\_CLK must be at least the DPHY\_TXBYTECLKHS rate else FIFOs will overflow, crashing the module.
2. The CSI\_TX\_VBUS\_CLK is the interface configuration clock that runs at half the speed of the CSI\_TX\_MAIN\_CLK (250MHz).
3. The DPHY\_TXBYTECLKHS is the clock supplied by the DPHY\_TX PLL and is divided down to byte clock. The DPHY\_TX is designed for max of 10gbps. This translates to a max byte clock of 312.5MHz. The clock is inactive when DPHY\_TX is not in HS operation.
4. The CSI\_TX\_ESC\_CLK escape clock runs at 20 MHz. CSI\_TX\_ESC\_CLK is rarely used by the CSI\_TX\_IF, usually is only to clock in some low speed control signals from DPHY\_TX. The ESC interface is a low speed DPHY common link to the camera/sensor.

Table 12-1540 shows the CSI\_TX\_IF and DPHY\_TX inter-clock dependencies.

**Table 12-1540. CSI\_TX\_IF Inter-clock Dependencies**

	CSI_TX_MAIN_CLK	CSI_TX_VBUS_CLK	DPHY_TXBYTECLKHS	CSI_TX_ESC_CLK
<b>Min freq</b>	DPHY_TXBYTECLKHS freq	CSI_TX_MAIN_CLK / 2 freq	N/A	N/A
<b>Max freq</b>	500 MHz	CSI_TX_MAIN_CLK / 2 freq	312.5MHz	20 MHz

#### 12.7.3.4.4 CSI\_TX\_IF Interrupt Events

This section describes the register configuration of the interrupt events that can trigger the several CSI\_TX\_IF interrupt signals.

The interrupts are generally handled within the INTD module of the CSI\_TX\_IF, although there are several interrupt registers in the ECC\_AGGR for ECC errors and in the VBUS2APB for stream monitoring errors/flags.

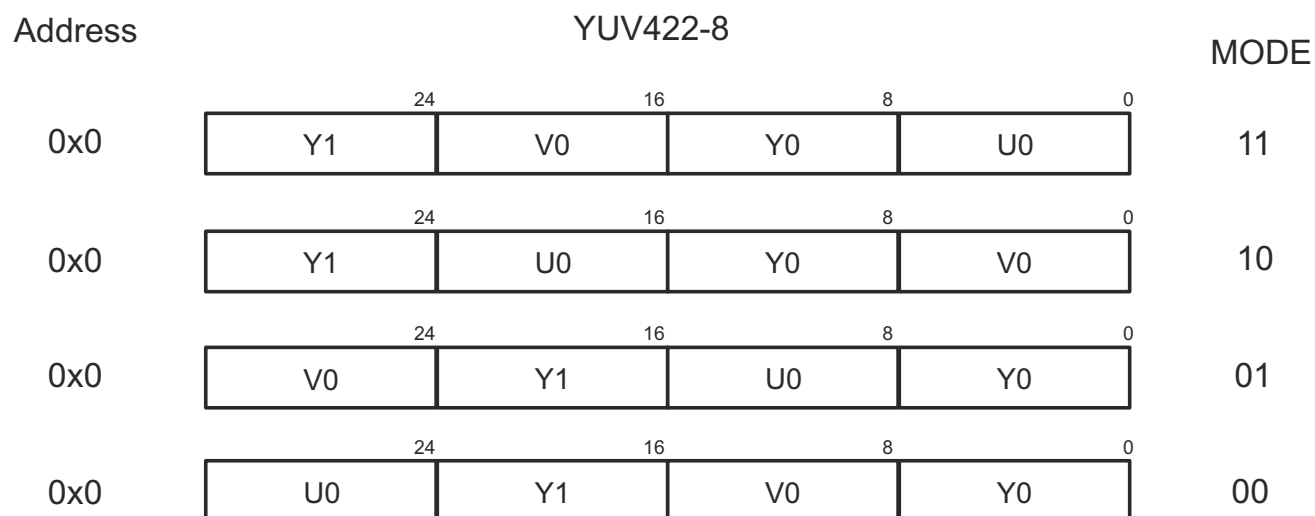
Table 12-1541 lists the event generation and corresponding registers of the CSI\_TX\_IF controller.

**Table 12-1541. CSI\_TX\_IF Interrupt Events Cross Table**

Event	Mask Register	Status Register	Description
CSI_TX_IF_MAIN_0_INTERRUPT	CSI_TX_IF_IRQ_MASK CSI_TX_IF_DPHY_IRQ_MASK	CSI_TX_IF_IRQ CSI_TX_IF_DPHY_IRQ_MASK	Global interrupt that various re-synchronized sources converge into interrupt generation. Read the status register bitfields to trace the source of the event.
CSI_TX_IF_MAIN_0_CSI_LEVEL	CSI_TX_IF_IRQ_MASK CSI_TX_IF_DPHY_IRQ_MASK	CSI_TX_IF_IRQ CSI_TX_IF_DPHY_IRQ_MASK	Error interrupt that is generated under the following conditions: <ul style="list-style-type: none"> <li>Retransmit error on Stream2 (packet error on the stream interface)</li> <li>Retransmit error on Stream3 (packet error on the stream interface)</li> </ul> Read the status register bitfields to trace the source of the event.
CSI_TX_IF_MAIN_0_CORR_LEVEL	CSI_TX_IF_ASF_INT_MASK	CSI_TX_IF_ASF_SRAM_CORR_FAULT_STATUS	Interrupt on internal FIFO RAM. Read the status register bitfields to trace the source of the event.
CSI_TX_IF_MAIN_0_UNCORR_LEVEL	CSI_TX_IF_ASF_INT_MASK	CSI_TX_IF_ASF_SRAM_UNCORR_FAULT_STATUS	Interrupt on internal FIFO RAM. Read the status register bitfields to trace the source of the event.
CSI_TX_IF_MAIN_0_CSI_FATAL	CSI_TX_IF_ASF_INT_MASK	CSI_TX_IF_ASF_INT_STATUS	ASF port fatal interrupt. Level sensitive. Set CSI_TX_IF_ASF_FATAL_NONFATAL_SELECT for whether fatal or non-fatal ASF interrupt is triggered. If any of the CSI_TX_IF_ASF_INT_STATUS bit is set, the CSI_RX_CSI_FATAL or CSI_RX_CSI_NONFATAL event signal is asserted.
CSI_RX_CSI_NONFATAL	CSI_TX_IF_ASF_INT_MASK	CSI_TX_IF_ASF_INT_STATUS	ASF port fatal interrupt. Level sensitive. Set CSI_TX_IF_ASF_FATAL_NONFATAL_SELECT for whether fatal or non-fatal ASF interrupt is triggered. If any of the CSI_TX_IF_ASF_INT_STATUS bit is set, the CSI_RX_CSI_FATAL or CSI_RX_CSI_NONFATAL event signal is asserted.

#### 12.7.3.4.5 CSI\_TX\_IF Data Memory Organization Details

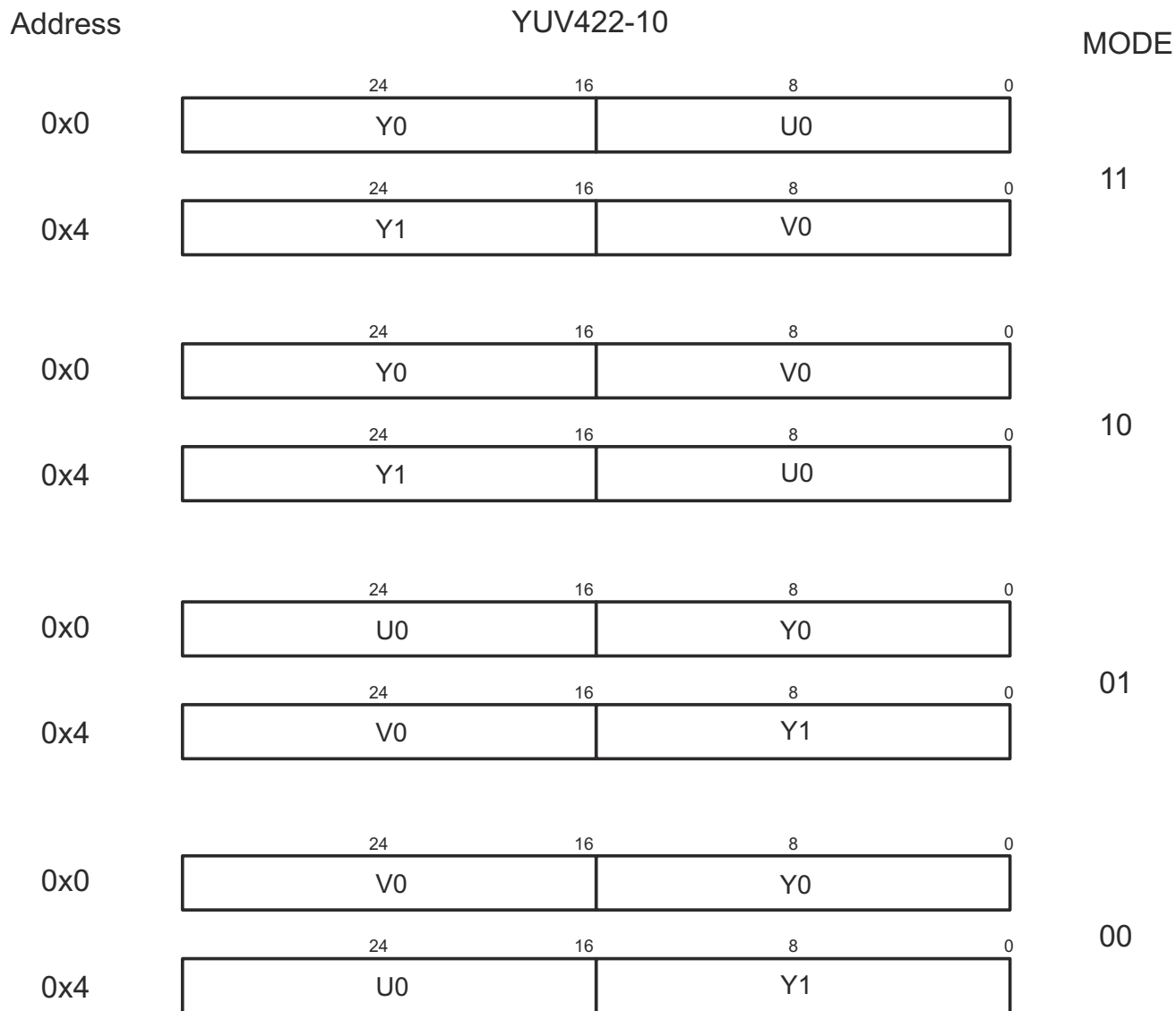
Figure 12-1166 shows the YUV422-8 data organization in memory.



csi\_tx\_if-007

**Figure 12-1166. CSI\_TX\_IF YUV422-8 Memory Data Organization**

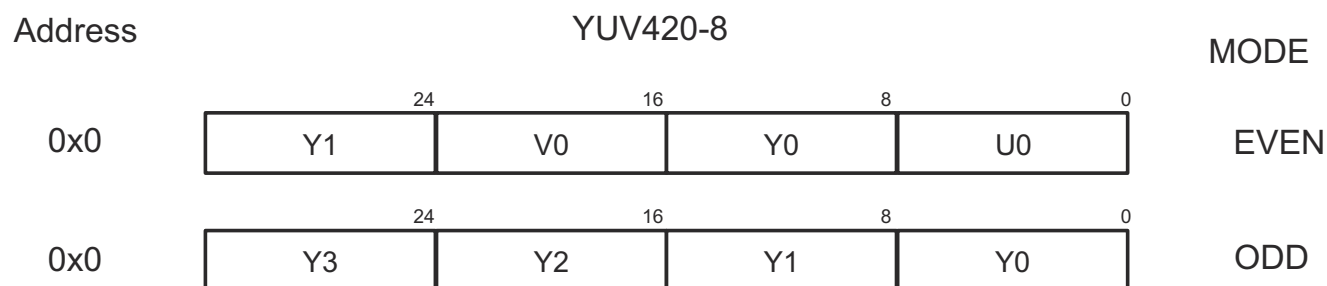
Figure 12-1167 shows the YUV422-10 data organization in memory.



csi\_tx\_if-008

**Figure 12-1167. CSI\_TX\_IF YUV422-10 memory data organization**

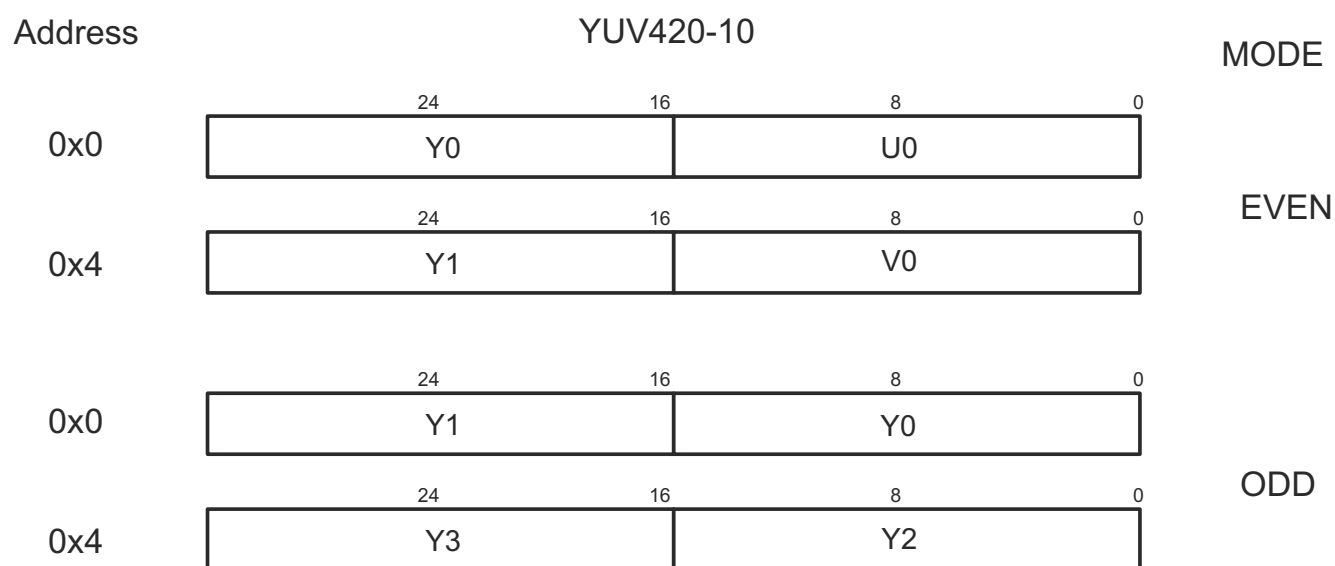
Figure 12-1168 shows the YUV420-8 data organization in memory.



csi\_tx\_if-009

**Figure 12-1168. CSI\_TX\_IF YUV420-8 memory data organization**

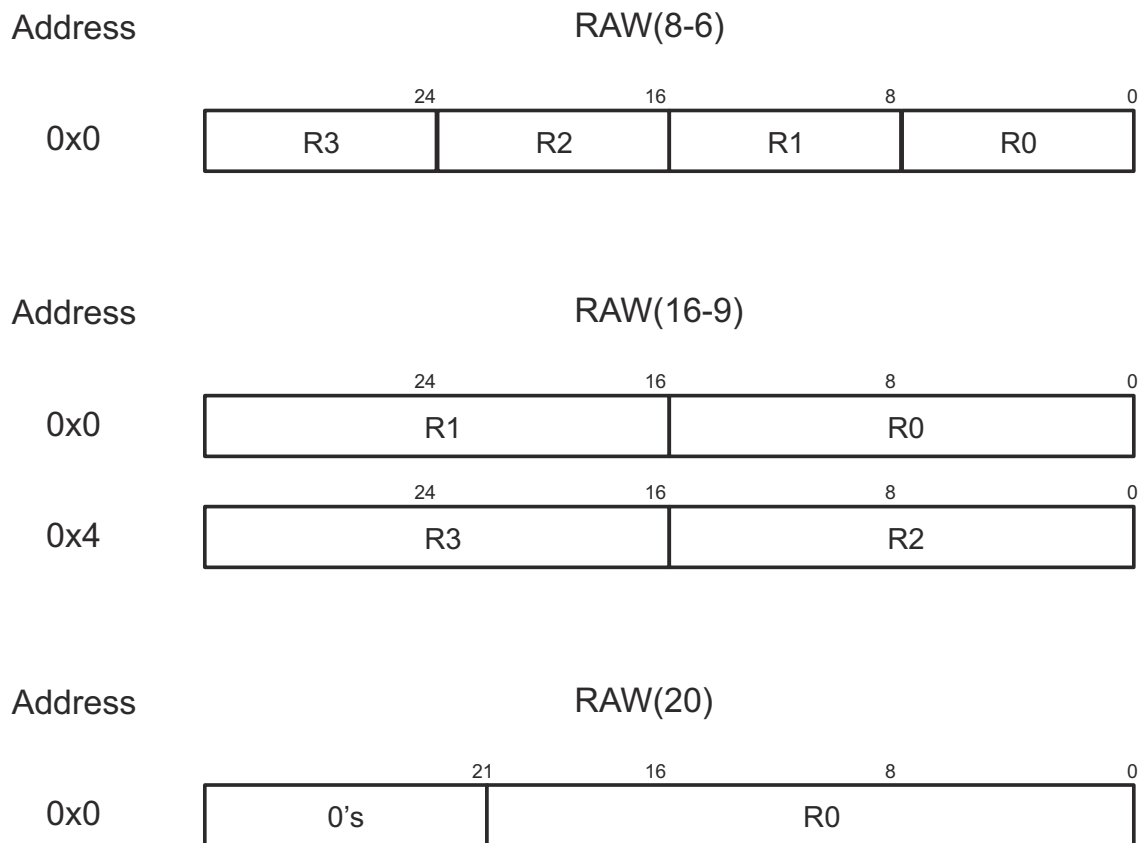
Figure 12-1169 shows the YUV420-10 data organization in memory.



csi\_tx\_if-010

**Figure 12-1169. CSI\_TX\_IF YUV420-10 memory data organization**

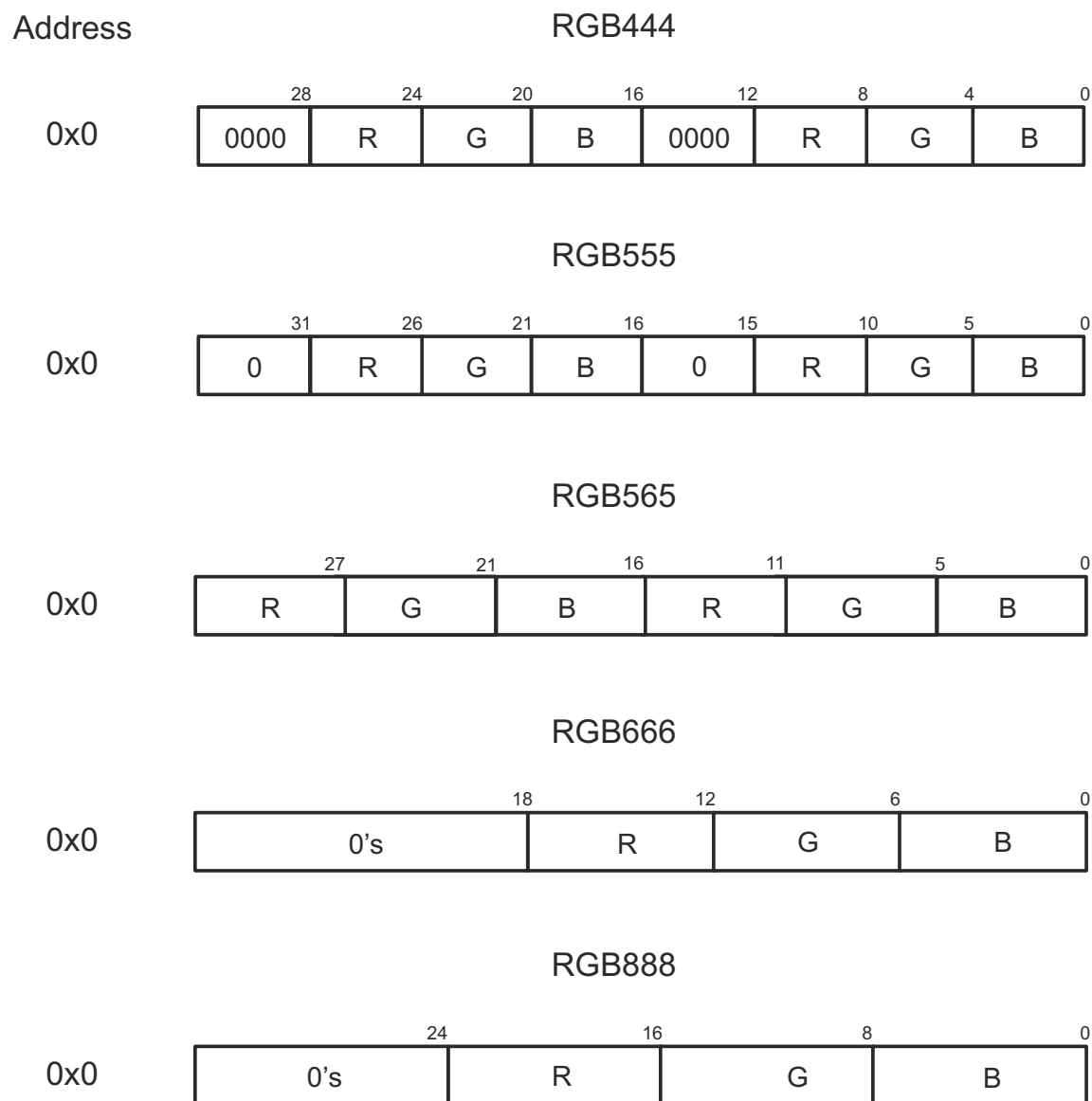
Figure 12-1170 shows the RAW data organization in memory. User defined data types behave the same as RAW data types.



csi\_tx\_if-011

**Figure 12-1170. CSI\_TX\_IF RAW (UNPACKED) memory data organization**

Figure 12-1171 shows the RGB data organization in memory.

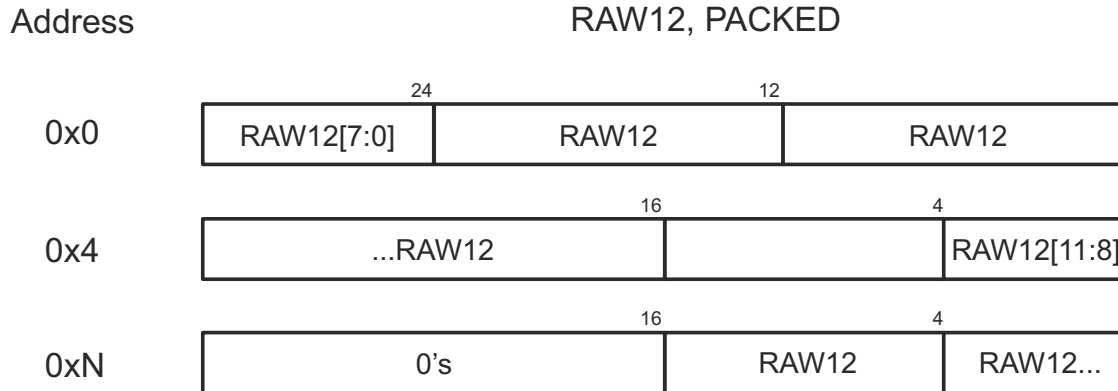


csi\_tx\_if-012

**Figure 12-1171. CSI\_TX\_IF RGB memory data organization**

Figure 12-1172 shows the RAW12 (PACKED) data organization in memory.





csi\_tx\_if-013

A. Where data does not fill out to a byte boundary it is zero extended.

**Figure 12-1172. CSI\_TX\_IF RAW12 (PACKED) memory data organization**

#### 12.7.3.4.6 CSI\_TX\_IF PSI\_L (DMA) Interface

The PSI\_L (DMA) interfaces is common for both CSI\_RX\_IF and CSI\_TX\_IF, see chapter *Camera Streaming Interface Receiver (CSI\_RX\_IF) and MIPI DPHY Receiver (DPHY\_RX)*., subsection *CSI\_RX\_IF PSI\_L (DMA) Interface*

#### 12.7.3.4.7 CSI\_TX\_IF ECC Protection Support

The CSI\_TX\_IF has two ECC aggregators, one for byte clock (DPHY\_TXBYTECLKHS) and the other one for all other clocks. The ECC is a mechanism for providing increased system reliability (via reduction of memory soft errors) by allowing single bit errors to be detected and corrected and double bit errors to be detected.

The ECC protection on the CSI\_TX\_IF RAM provides Single Error Correction and Double Error Detection (SEC/DED). This logic detects and corrects a single bit error (1 bit error per ECC word or per ECC data segment). For memories that contain critical and/or persistent data, automatic (immediate or delayed) write-back of the corrected data to the corresponding memory address is supported. In addition, the ECC also supports multiple options for partial word writes, such as read-modify-write or multiple ECC code segments per word.

The ECC protection also provides Double Error Detection (DED). This logic only detects (does not correct) double errors (2 bit errors per ECC word or per ECC data segment).

The ECC aggregators on the CSI\_TX\_IF subsystem level consolidates the ECC configuration and status bits for all the ECC supported memories in the subsystem. They provide a single EOI-handshake based interrupt to the interrupt processors (for both single and double error detections) and a standard 32-bit VBUS interface for configuring and querying the ECC register set, see *CSI\_RX\_IF ECC Registers*. For complete details on the features and functions of the ECC aggregator, see *ECC Aggregator*.

#### 12.7.3.5 CSI\_TX\_IF Programming Guide

##### Note

Software must keep the CSI\_TX\_IF\_CONTROL1[0] PIXEL\_RESET register in the asserted state in the control register until all CSI\_TX\_IF controller registers are programmed. Only after all CSI\_TX\_IF controller programming can software de-assert CSI\_TX\_IF\_CONTROL1 register. If software has to reprogram any CSI\_TX\_IF controller registers it MUST have the control CSI\_TX\_IF\_CONTROL1 register reset asserted.

#### 12.7.3.5.1 Programming (Configuration Mode)

Although it is always possible to read/write all control registers, it strongly recommended writing to the control registers only while configuration mode is active. To enter configuration mode, the CSI\_TX\_IF\_TX\_CONF[2]

CONFIGURATION\_REQUEST bit must be set. CSI\_TX\_IF enters configuration mode when transmission of any ongoing frame has ended and the request bit is set. Configuration mode is the default state after reset. While configuration mode is active, the CSI\_TX\_IF\_STATUS[2] CONFIGURATION\_ACTIVE bit is set. Traffic at the pixel interface is ignored during configuration mode, and it is safe to write to control registers and change configuration.

To exit configuration mode, the CONFIGURATION\_REQUEST bit in the CSI\_TX\_IF\_TX\_CONF register must be cleared. The CSI\_TX\_IF then starts normal operation, with first new frame incoming after configuration mode exit.

During configuration mode, all interrupts are disabled.

---

#### Note

A mandatory assertion of soft reset occurs following the de-assertion of CONFIGURATION\_REQUEST. This ensures the controller is in a known state and all the parameters set in the configuration registers have been loaded correctly into the internal state. The DPHY\_TXBYTECLKHS must be active before the configuration request is completed.

---

#### 12.7.3.5.2 System Initialization Programming

The power up/reset release of the system will clear all the register values to their reset conditions. Software must perform programming of the virtual channel and data type registers to match the system operation if the reset conditions do not match the required values. Software must guarantee that all the data type configuration registers are programmed with valid values for the pixel\_dt\_sel\_if[:]X parameter used.

The CSI\_TX\_IF registers for all the DPHY\_TX related control and delays must be set before starting the system. The DPHY\_TX delays must be calculated using the system clock periods associated with CSI\_TX\_ESC\_CLK and DPHY\_TXBYTECLKHS. The wait burst time must be calculated using the DPHY\_TX data sheet to ensure that all the lanes begin new requests at the same point.

Software will configure the FIFO fill level control registers if required by the system clock and payload control configurations. When all registers are programmed the configuration can be made active. The pixel streams can then start requests.

---

#### Note

The CSI\_TX\_IF is configured at design time, so that the reset values adopt a generic desired Power\_On state. This reduces the programming steps required by the system for the general use case scenario.

---

#### 12.7.3.5.3 Lane Control Programming

The CSI\_TX\_IF links to the DPHY\_TX clock and data lanes. After power up, software must identify that the DPHY\_TX is powered on and the DPHY\_TX PLL has locked. The DPHY\_TX controls for swapping the DP/DN control can be modified using the CSI\_TX\_IF\_DPHY\_CFG1 register if required.

Software must then program the enable for the clock lane and each data lane that is required. The DPHY\_TX lane signals can be checked using the CSI\_TX\_IF\_DPHY\_STATUS register to identify that the lane is in STOPSTATE and is ready for High Speed transmission. This register can also be used to identify when ULPS is active.

#### 12.7.3.5.4 Line Control

The line control block uses the DPHY\_TXBYTECLKHS from the DPHY\_TX PPI interface to take the line data from the packet interface FIFO and transfer the bytes to the active lanes of the DPHY\_TX. The block detects when the DPHY\_TX is ready to transmit high speed data and the pixel stream has packets available.

The line control block performs pixel stream arbitration using the line control arbitration block when more than one pixel stream is generated in the configuration. The block performs all the clock and data lane control required to activate the high-speed transmission and return the lane to ULPS state as required.

#### 12.7.3.5.5 Line Control Arbitration

The CSI\_TX\_IF controller uses a simple arbitration scheme for configurations with more than one pixel interface.

The arbitration has 2 levels; the first is a round-robin scheme and if no streams are granted by this a priority based scheme is used. The arbitration will select the first stream making a request if there is only one stream waiting or the first in the round robin after a stream has completed transmission. This means that there may be one cycle where the round-robin selector moves across a stream position that is not requesting to transmit.

The re-arbitration boundary is between packets except where line sync is used. In this case the line start packet, long packet and line end are grouped together.

The selected stream will pass the line data from the stream FIFO to the line control block for distribution to the active PPI lanes of the DPHY\_TX.

#### 12.7.3.5.6 Lane Manager FSM

Data distribution to the available lanes is controlled via the lane manager FSM.

The state descriptions can be found in [Table 12-1542](#).

**Table 12-1542. Lane Manager FSM State Description**

State (line_fsm_st_r)	Description
LINE_FSM_IDLE	Wait for new transmission to be ready. When start_hs_transmission_c is high then go to the transmission state per enabled lanes (lanes_enable_r)
LINE_FSM_BURST_1L	Transmit burst data over lane 0 until end of the burst. When end then go to the BURST_END state.
LINE_FSM_BURST_2L	Transmit burst data over lanes 0 and 1 until end of the burst. When end then go to the BURST_END state.
LINE_FSM_BURST_4L	Transmit burst data over all lanes until end of the burst. When end then go to the BURST_END state.
BURST_END	Wait for being ready for new burst transmission. In this state counter counts until WAIT_BURST_TIME value in the register is reached.

#### 12.7.3.5.7 Data Lane Control FSM

Data lanes control uses a simple FSM to sequence entry and exit of Ultra Low Power mode. This FSM observes inputs from the control registers and based on these inputs it generates PPI outputs for ULP mode.

The State descriptions are presented in [Table 12-1543](#).

**Table 12-1543. Data Lanes Control FSM State Description**

State (ulps_data_fsm_st_r)	Description
DATA_LN_IDLE	Idle state. In this state HS transmission can be issued if clock lane is in HS mode.
DATA_LN_ULPS_REQ	Request to enter ULP state. Waiting for activation ULP confirmed by tx_ulps_active_n input.
DATA_LN_ULPS_ACTIVE	ULPS for data lanes is active. Waiting for ulps_req_sync being low to exit ULPS.
DATA_LN_ULPS_EXIT	ULPS exiting, waiting for time defined in ULPS_DATA_WAKEUP_TIME register.

#### 12.7.3.5.8 Clock Lane Control

The clock lane of the DPHY\_TX must always be active and prepared for high speed transmission by moving from ULPS state to high speed clock active. The high speed bit clock PLL should be configured for the desired clock rate and locked before the clock lane exits from ULPS.

### 12.7.3.5.9 Clock Lane Control FSMs

The clock lane is controlled in HS and ULP modes using two simple FSMs which have been implemented in the clock control module. Separate FSMs are required as the HS and ULP signals operate in different clock domains.

A brief description of the states of the HS clock lane FSM is presented in [Table 12-1544](#).

**Table 12-1544. HS Clock Lane Control FSM State Description**

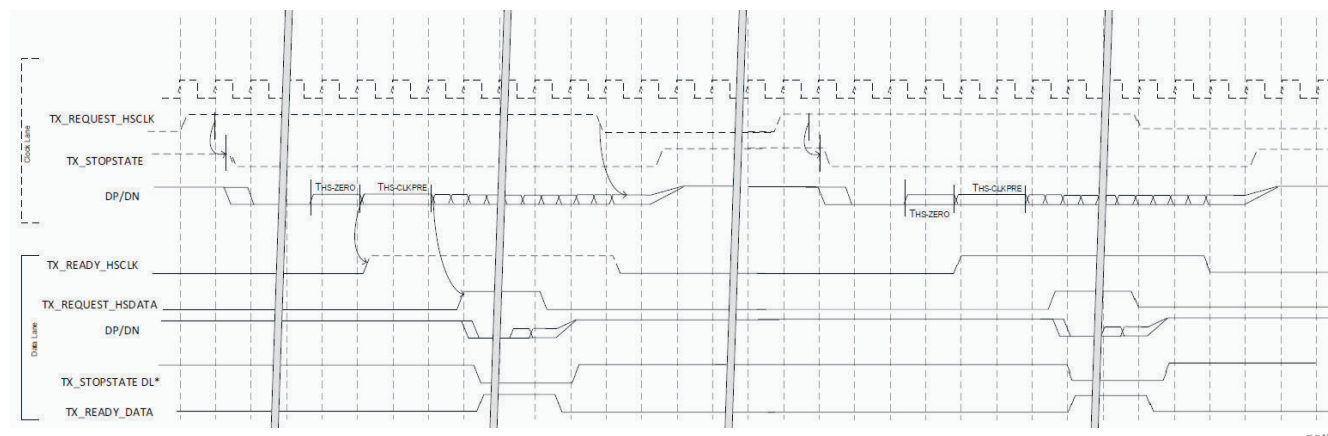
State (clk_fsm_r)	Description
CLK_IDLE	Waiting for HS request and for Clock ULPS FSM to enter IDLE
CLK_HS_REQ	HS mode request, waiting for DPHY_TX HS mode active
CLK_HS_MODE	HS mode active
CLK_HS_EXIT	Exiting HS mode

A brief description of the states of the ULP clock lane FSM is presented in [Table 12-1545](#).

**Table 12-1545. ULPS Clock Lane Control FSM State Description**

State (clk_fsm_r)	Description
ULPS_CLK_IDLE	Waiting for ULP request and for HS Clock Lane FSM to enter IDLE
ULPS_CLK_ULPS_REQ	ULP state request, waiting for DPHY_TX ULPS state active
ULPS_CLK_ULPS_ACTIVE	ULPS state is active
ULPS_CLK_ULPS_EXIT	Exiting ULPS. Waiting for time defined in DPHY_CLK_WAKEUP register

The clock lane will operate in either continuous or non-continuous clock operation based on the bit value on the CSI\_TX\_IF\_DPHY\_CFG[10] DPHY\_CLOCK\_MODE register bitfield. When the non-continuous clock mode is active the clock lane will automatically move between LP and HS modes based on the activity of the streams.



**Figure 12-1173. Clock Lane Control FSMs Timing Diagram**

In this mode of operation, after a High Speed burst, if there are no new HS requests pending, the CSI\_TX\_IF controller will deactivate the Clock Lane request, and the Clock Lane will exit High Speed mode and reach stop state. The Clock Lane will remain in LP while there are no active requests. When a new HS request is received, the Clock Lane will exit LP mode and resume High Speed operation.

The CSI\_TX\_IF controller can request that the DPHY\_TX enters ULPS when there are no active frames, and must return through LP11 when there are new requests to be processed by changing the CSI\_TX\_IF\_DPHY\_CFG[9-8] DPHY\_MODE back to High Speed.

### 12.7.3.5.10 CSI\_TX\_IF Configuration for PSI\_L

Table 12-1546 lists the settings that must be programmed before enabling CSI\_TX\_IF controller and sending PSI\_L data.

**Table 12-1546. CSI\_TX\_IF configuration table for PSI\_L**

Format	Pixel transmit mode used	Size	Details
YUV420-8	single	0	Must set YUV420 mode in CSI_TX_IF_DMACNTX_j
YUV420-10	single	1	Must set YUV420 mode in CSI_TX_IF_DMACNTX_j
YUV422-8	single	0	Must set YUV422 mode in CSI_TX_IF_DMACNTX_j
YUV422-10	single	1	Must set YUV422 mode in CSI_TX_IF_DMACNTX_j
YUV422-8	packed	2	Must set YUV422 mode in CSI_TX_IF_DMACNTX_j
RGB888, RGB666	single	2	
RGB565, RGB555, RGB444	single	1	
RAW6-8	single	0	
RAW8	dual	1	
RAW8	quad	2	
RAW10	single	1	
RAW10	dual	2	
RAW12	single	1	No packing
RAW12	dual	2	No packing
RAW12(packed)	single	0	Must set CSI_TX_IF_DMACNTX_j[23] PACK12_CFG = 1
RAW12(packed)	dual	1	Must set CSI_TX_IF_DMACNTX_j[23] PACK12_CFG = 1
RAW14	single	1	
RAW14	dual	2	
RAW16	single	1	
RAW20	single	2	
User defined 8	single	0	Same as RAW8
User defined 16	single	1	Same as RAW16

### 12.7.3.5.11 CSI\_TX\_IF Configuration for Re-transmit

The following lists the re-transmit configuration requirements:

- Software must program re-transmit related registers before enabling CSI\_RX\_IF to send data.
- In CSI\_RX\_IF to CSI\_TX\_IF loopback needs to have the same mode on both CSI\_RX\_IF and CSI\_TX\_IF controllers(single, dual, quad). Cannot mix for example dual on CSI\_RX\_IF and single on CSI\_TX\_IF.
- Update CSI\_TX\_IF\_RETRANS\_CNTL for virtual channel desired to be transmitted on Stream2/3.
- Enable CSI\_TX\_IF to send out data (Only supports RAW formats)
- Data type select(CSI\_TX\_IF\_DT0\_CFG - CSI\_TX\_IF\_DT7\_CFG) for CSI\_TX\_IF are driven by bottom 3 bits of datatype from CSI\_RX\_IF. CSI\_RX\_IF has the true data type. Software will need to program the appropriate configuration registers based on this.

**Table 12-1547. CSI\_TX\_IF configuration table for PSI\_L**

DataseI(TX)	Datatype(RX)	Description
0	0x28	RAW6
1	0x29	RAW7

**Table 12-1547. CSI\_TX\_IF configuration table for PSI\_L (continued)**

DataseI(TX)	Datatype(RX)	Description
2	0x2A	RAW8
3	0x2B	RAW10
4	0x2C	RAW12
5	0x2D	RAW14
6	0x2E	RAW16
7	0x2F	RAW20

#### 12.7.3.5.12 CSI\_TX\_IF Configuration for Color Bar

The following lists the Color Bar configuration requirements:

- Color bar must be set to packed YUV422-8 format, this is an extended datatype in CSI\_TX\_IF controller.
- Software must program CSI\_TX\_IF controller before enabling the color bar from the CSI\_TX\_IF\_COLOR\_CNTL[0] EN bit-field.
- Program CSI\_TX\_IF\_COLOR\_LINE\_DELAY, CSI\_TX\_IF\_COLOR\_FRAME\_DELAY, CSI\_TX\_IF\_COLOR\_START\_DELAY, and CSI\_TX\_IF\_COLOR\_PARAM registers with appropriate configurations. Update CSI\_TX\_IF\_COLOR\_CNTL with virtual channel and data type desired and enable it.

#### 12.7.3.5.13 CSI\_TX\_IF Error Recovery

When an underflow error occurs, software must reset the entire CSI\_TX\_IF by SoC PSC reset and restart over.

#### 12.7.3.5.14 CSI\_TX\_IF Power Up/Down Sequence

The following lists the power up/down sequence requirements:

1. Software must ensure no more PSI\_L traffic is directed toward the CSI\_TX\_IF module.
2. Software must then go through a tear down process on PSI\_L gasket inside CSI\_TX\_IF wrapper. Refer to chapter *PSI\_L* for tear down process.
3. Software must then check the stream idle values so that they are idle in the CSI\_TX\_IF\_CONTROL1 register.
4. Software must then stop all traffic inside the CSI\_TX\_IF controller and wait till it is idle.
5. After that software can then proceed to do PSC power down routine.

## 12.8 Shared MIPI D-PHY Transmitter (DPHY\_TX)

This section describes the features and functions of the shared MIPI D-PHY Transmitter (DPHY\_TX).

### 12.8.1 DPHY\_TX Subsystem Overview

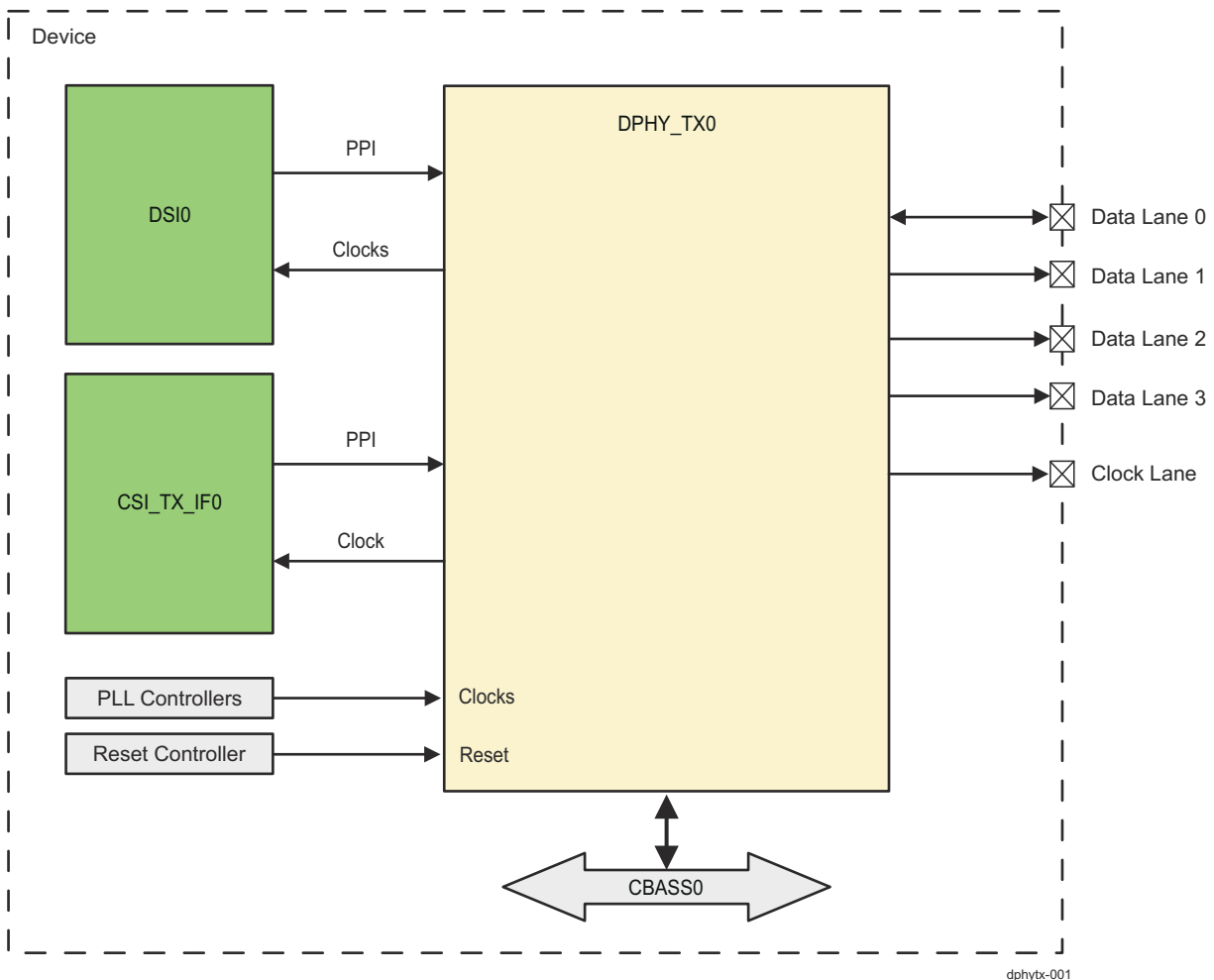
The DPHY\_TX module provides one option for video output interfacing by implementing a four-lane MIPI D-PHY Transmitter.

The device includes one instantiation of DPHY\_TX module. [Table 12-1548](#) shows the DPHY\_TX allocation within device domains:

**Table 12-1548. DPHY\_TX Allocation Accross Device Domains**

Module Instance	Domain		
	WKUP	MCU	MAIN
DPHY_TX0	-	-	✓

[Figure 12-1174](#) provides DPHY\_TX module overview.



**Figure 12-1174. DPHY\_TX Overview**

#### 12.8.1.1 DPHY\_TX Features

The DPHY\_TX module supports the following main features:

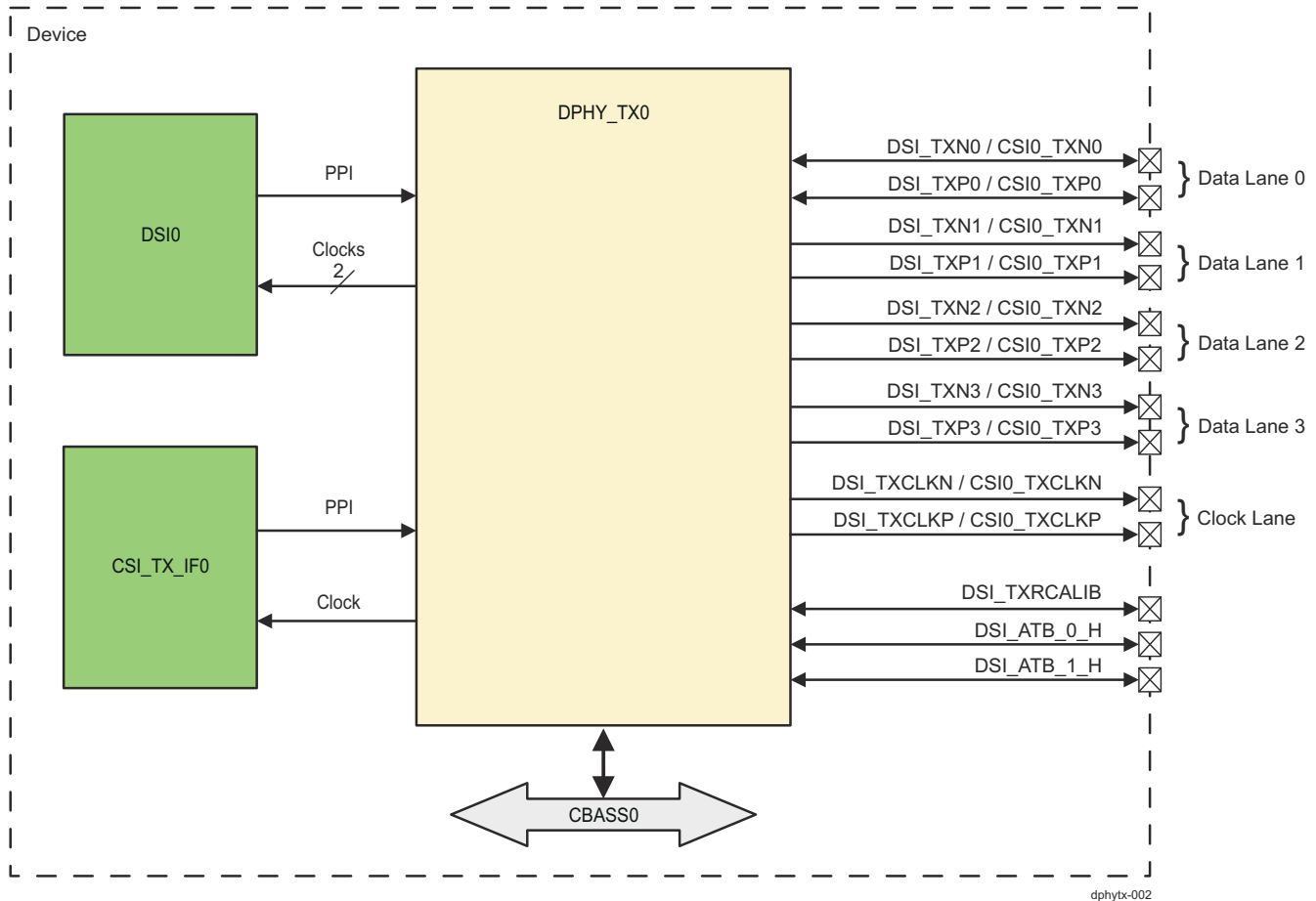
- Compliance with MIPI D-PHY 1.2 physical layer interface specification and features

- 1, 2 or 4 data lanes, in addition to clock signaling
- Maximum data rate up to 2.5 Gbps per data lane (with deskew) and 1.5 Gbps (without deskew)
- Protocol Peripheral Interface (PPI)
- HS (High-Speed) continuous and burst modes
- LP (Low-Power), ULPM (Ultra-Lower Power), and Shutdown modes
- Data lanes can be independently operated in HS or ULP mode.
- Forward direction and reverse direction escape modes (only on Lane-0)
- Automatic termination control in both high-speed and low-power modes
- Fault detection:
  - Contention detection
  - Sequence error detection (corruption on lanes)



## 12.8.2 DPHY\_TX Environment

This section describes the DPHY\_TX application fields from an environment point of view (external connections).



**Figure 12-1175. DPHY\_TX Enviroment**

[Table 12-1549](#) describes the DPHY\_TX signals when transmitting data from a DSI module, and [Table 12-1550](#) describes the DPHY\_TX signals when transmitting data from a CSI\_TX\_IF module.

The CTRLMMR\_DPHY\_TX0\_CTRL[1-0] LANE\_FUNC\_SEL register field in the device Control Module selects which module, DSI0 or CSI\_TX\_IF0, uses the DPHY\_TX0 lanes.

**Table 12-1549. DPHY\_TX I/O Signals in DSI Mode**

Device Level Signal	I/O <sup>(1)</sup>	Description
<b>DPHY_TX0</b>		
DSI_TXN0	I/O	DPHY_TX0 Lane 0 Transmit Differential Data (Negative)
DSI_TXP0	I/O	DPHY_TX0 Lane 0 Transmit Differential Data (Positive)
DSI_TXN1	O	DPHY_TX0 Lane 1 Transmit Differential Data (Negative)
DSI_TXP1	O	DPHY_TX0 Lane 1 Transmit Differential Data (Positive)
DSI_TXN2	O	DPHY_TX0 Lane 2 Transmit Differential Data (Negative)
DSI_TXP2	O	DPHY_TX0 Lane 2 Transmit Differential Data (Positive)
DSI_TXN3	O	DPHY_TX0 Lane 3 Transmit Differential Data (Negative)
DSI_TXP3	O	DPHY_TX0 Lane 3 Transmit Differential Data (Positive)
DSI_TXCLKN	O	DPHY_TX0 Transmit Differential Clock Lane (Negative)
DSI_TXCLKP	O	DPHY_TX0 Transmit Differential Clock Lane (Positive)
DSI_TXRCALIB	A	DPHY_TX0 pin for external calibration resistor. Refer to the device-specific Datasheet for a recommended resistor value.
DSI_ATB_0_H	I/O	DPHY_TX0 Analog Test Bus 0
DSI_ATB_1_H	I/O	DPHY_TX0 Analog Test Bus 1

(1) I = Input; O = Output.

**Table 12-1550. DPHY\_TX I/O Signals in CSI Mode**

Device Level Signal	I/O	Description
<b>DPHY_TX0</b>		
CSI0_TXN0	O	DPHY_TX0 Lane 0 Transmit Differential Data (Negative)
CSI0_TXP0	O	DPHY_TX0 Lane 0 Transmit Differential Data (Positive)
CSI0_TXN1	O	DPHY_TX0 Lane 1 Transmit Differential Data (Negative)
CSI0_TXP1	O	DPHY_TX0 Lane 1 Transmit Differential Data (Positive)
CSI0_TXN2	O	DPHY_TX0 Lane 2 Transmit Differential Data (Negative)
CSI0_TXP2	O	DPHY_TX0 Lane 2 Transmit Differential Data (Positive)
CSI0_TXN3	O	DPHY_TX0 Lane 3 Transmit Differential Data (Negative)
CSI0_TXP3	O	DPHY_TX0 Lane 3 Transmit Differential Data (Positive)
CSI0_TXCLKN	O	DPHY_TX0 Transmit Differential Clock Lane (Negative)
CSI0_TXCLKP	O	DPHY_TX0 Transmit Differential Clock Lane (Positive)

### Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

### 12.8.3 DPHY\_TX Integration

This section describes the DPHY\_TX integration in the device MAIN domain, including information about clocks and resets.

There is one DPHY\_TX integrated in the device MAIN domain - DPHY\_TX0. Figure 12-1176 shows the integration of DPHY\_TX.

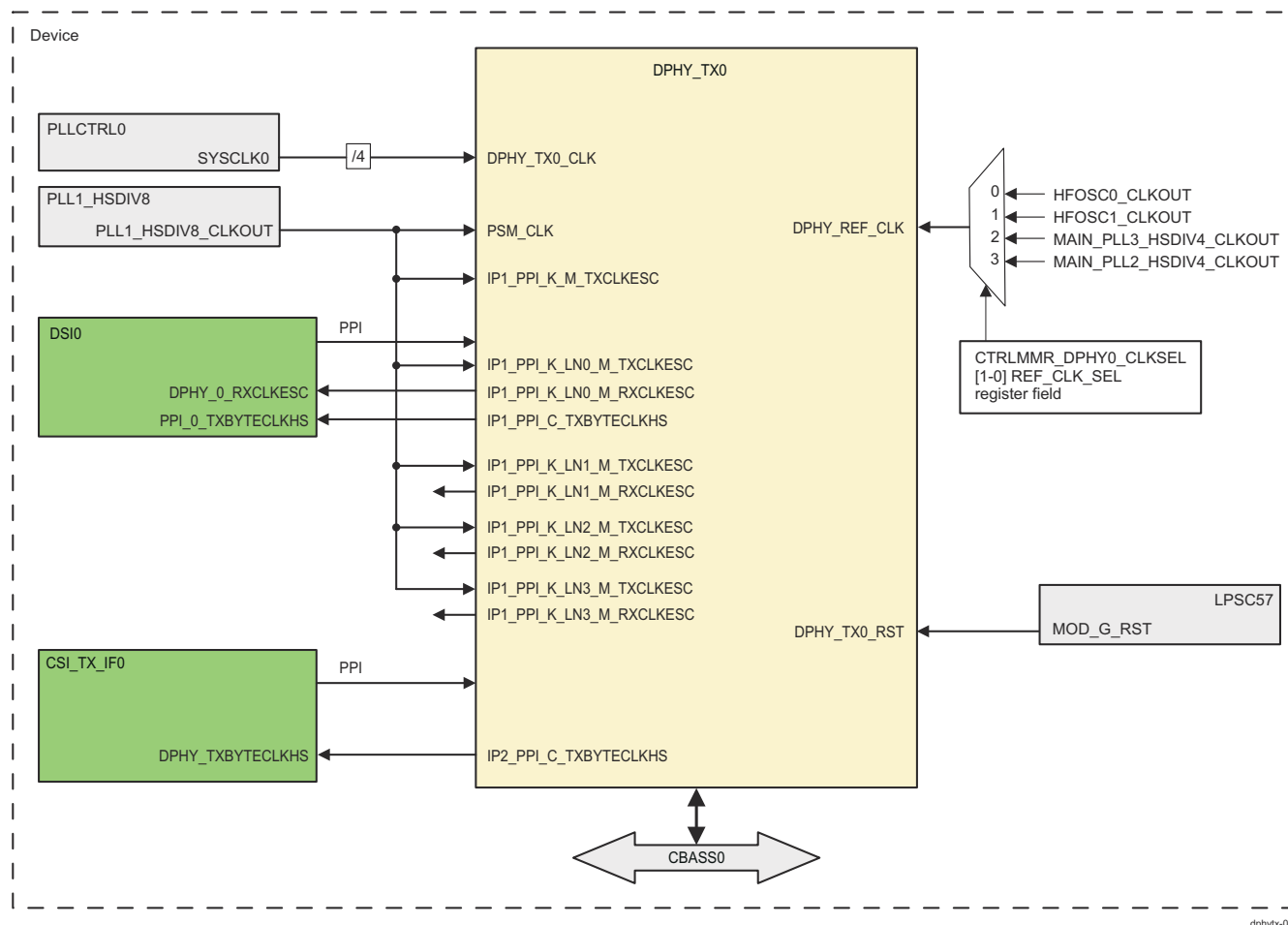


Figure 12-1176. DPHY\_TX Integration

Table 12-1551 and Table 12-1552 summarize the integration of DPHY\_TX in device MAIN domain.

Table 12-1551. DPHY\_TX Integration Attributes

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
DPHY_TX0	PSC0	PD2	LPSC57	CBASS0

Table 12-1552. DPHY\_TX Clocks and Resets

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
DPHY_TX0	DPHY_TX0_CLK	SYSClk0 / 4	PLLCTRL0	DPHY_TX0 interface clock

**Table 12-1552. DPHY\_TX Clocks and Resets (continued)**

	PSM_CLK	PLL1_HSDIV8_CLKOUT	PLL1_HSDIV8	DPHY_TX0 reference clock
	IP1_PPI_K_M_TXCLKE SC			DPHY_TX0 escape mode clock
	IP1_PPI_K_LN0_M_TX CLKESC			DPHY_TX0 escape mode clock for lane 0 for IP1_PPI
	IP1_PPI_K_LN1_M_TX CLKESC			DPHY_TX0 escape mode clock for lane 1 for IP1_PPI
	IP1_PPI_K_LN2_M_TX CLKESC			DPHY_TX0 escape mode clock for lane 2 for IP1_PPI
	IP1_PPI_K_LN3_M_TX CLKESC			DPHY_TX0 escape mode clock for lane 3 for IP1_PPI
	DPHY_REF_CLK	HFOSC0_CLKOUT	HFOSC0	DPHY_TX0 reference clock.
		HFOSC1_CLKOUT	HFOSC1	The selection of the
		MAIN_PLL3_HSDIV4_CLKOUT	MAIN_PLL3_HSDIV 4	source clock (see , <i>DPHY_TX Integration</i> ) is provided via the CTRLMMR_DPHY0_CLKSEL[1:0]
		MAIN_PLL2_HSDIV4_CLKOUT	MAIN_PLL2_HSDIV 4	REF_CLK_SEL register bit-field in device Control Module.
DSI0	PPI_0_TXBYTECLKHS	IP1_PPI_C_TXBYTECLKHS	DPHY_TX0	DPHY_TX0 byte clock for IP1_PPI
	DPHY_0_RXCLKESC	IP1_PPI_K_LN0_M_RXCLKES C		DPHY_TX0 reverse escape mode recovered clock from lane 0
CSI_TX_IF0	DPHY_TXBYTECLKHS	IP2_PPI_C_TXBYTECLKHS	DPHY_TX0	DPHY_TX0 byte clock for IP2_PPI
<b>Resets</b>				
<b>Module Instance</b>	<b>Module Reset Input</b>	<b>Source Reset Signal</b>	<b>Source</b>	<b>Description</b>
DPHY_TX0	DPHY_TX0_RST	MOD_G_RST	LPSC57	DPHY_TX0 reset

### Note

For more information on the interconnects in device MAIN domain, see [Chapter 3, System Interconnect](#).

For more information on the power, reset and clock management in device MAIN domain, see the corresponding sections within [Chapter 5, Device Configuration](#).

## 12.9 Video Processing Front End (VPFE)

This section describes the Video Processing Front End (VPFE) module for the device.

### 12.9.1 VPFE Overview

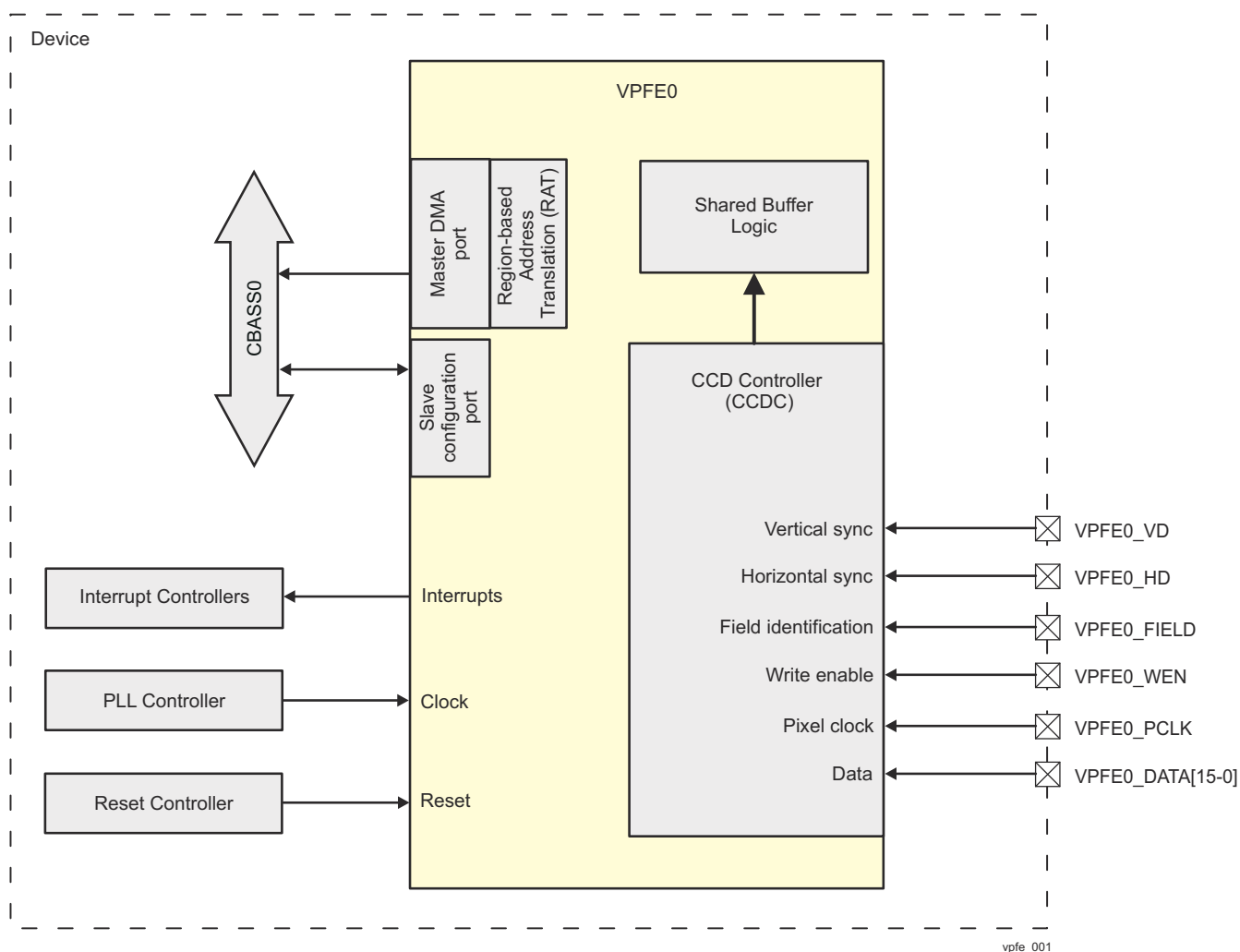
The Video Processing Front End (VPFE) is an input interface module that receives raw (unprocessed) image/video data or YUV digital video data from external imaging peripherals (such as image sensors, video decoders, etc) and performs DMA transfers to store the captured data in the system DDR memory.

Table 12-1553 shows the VPFE allocation within device domains.

**Table 12-1553. VPFE Modules Allocation within Device Domains**

Module Instance	Domain		
	WKUP	MCU	MAIN
VPFE0	-	-	✓

Figure 12-1177 is an overview of the VPFE.



**Figure 12-1177. VPFE Overview**

#### 12.9.1.1 VPFE Features

The VPFE module supports the following features:

- 
- Encompasses a buffer memory for interfacing to the DMA at the chip level and preventing the charge-coupled device (CCD) Controller from overflowing.
- Support for conventional Bayer pattern and Foveon sensor formats.
- Generates HD/VD timing signals and field ID to an external timing generator or can synchronize to the external timing generator
- Support for progressive (non-interlaced) and interlaced sensors (hardware support for up to 2 fields and firmware support for higher number of fields, typically 3-, 4-, and 5-field sensors).
- Support for up to 110-MHz sensor clock.
- Support for REC656/CCIR-656 standard (YCbCr 422 format, either 8- or 16-bit).
- Support for YCbCr 422 format, either 8- or 16-bit with discrete HSYNC and VSYNC signals.
- Support for up to 16-bit input.
- Generates optical black clamping signals.
- Support for digital clamping and black level compensation.
- Support for 10-bit to 8-bit A-law compression.
- Support for a low-pass filter prior to writing to DDR. If this filter is enabled, 2 pixels each in the left and right edges of each line are cropped from the output.
- Support for generating output to range from 16-bits to 8-bits wide (8-bits wide allows for 50% saving in storage area).
- Support for downsampling via programmable culling patterns.
- Ability to control output to the DDR via an external write enable signal.
- Support for up to 16K pixels (image size) in both the horizontal and vertical directions.
- Region-based Address Translation (RAT) module for converting legacy 32-bit to 48-bit addressing scheme in the write DMA path.

#### 12.9.1.2 VPFE Not Supported Features

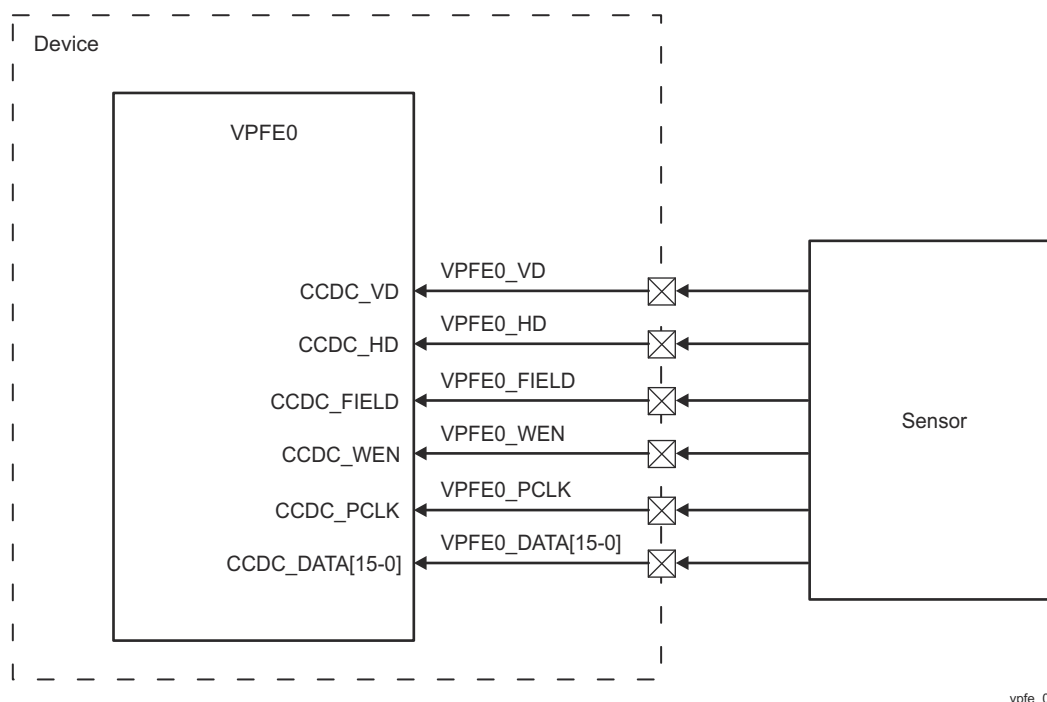
The following features are not supported on this family of devices:

- 
- Hardware de-interlace (only field identification is supported).

## 12.9.2 VPFE Environment

This section describes the VPFE external connections (environment).

### 12.9.2.1 VPFE External System Interface



vpfe\_002

**Figure 12-1178. VPFE External System Interface**

**Table 12-1554. VPFE I/O Signals**

Module Pin	Device Level Signal	I/O <sup>(1)</sup>	Description	Module Pin Reset Value
<b>VPFE0</b>				
CCDC_DATA0	VPFE_DATA0	I	CCDC data input 0	HiZ
CCDC_DATA1	VPFE_DATA1	I	CCDC data input 1	HiZ
CCDC_DATA2	VPFE_DATA2	I	CCDC data input 2	HiZ
CCDC_DATA3	VPFE_DATA3	I	CCDC data input 3	HiZ
CCDC_DATA4	VPFE_DATA4	I	CCDC data input 4	HiZ
CCDC_DATA5	VPFE_DATA5	I	CCDC data input 5	HiZ
CCDC_DATA6	VPFE_DATA6	I	CCDC data input 6	HiZ
CCDC_DATA7	VPFE_DATA7	I	CCDC data input 7	HiZ
CCDC_DATA8	VPFE_DATA8	I	CCDC data input 8	HiZ
CCDC_DATA9	VPFE_DATA9	I	CCDC data input 9	HiZ
CCDC_DATA10	VPFE_DATA10	I	CCDC data input 10	HiZ
CCDC_DATA11	VPFE_DATA11	I	CCDC data input 11	HiZ
CCDC_DATA12	VPFE_DATA12	I	CCDC data input 12	HiZ
CCDC_DATA13	VPFE_DATA13	I	CCDC data input 13	HiZ
CCDC_DATA14	VPFE_DATA14	I	CCDC data input 14	HiZ
CCDC_DATA15	VPFE_DATA15	I	CCDC data input 15	HiZ
CCDC_VD	VPFE_VD	I	Vertical synchronization signal (VSYNC)	HiZ
CCDC_HD	VPFE_HD	I	Horizontal synchronization signal (HSYNC)	HiZ



**Table 12-1554. VPFE I/O Signals (continued)**

Module Pin	Device Level Signal	I/O <sup>(1)</sup>	Description	Module Pin Reset Value
CCDC_FIELD	VPFE_FIELD	I	Field identification signal	HiZ
CCDC_WEN	VPFE_WEN	I	Write enable signal	HiZ
CCDC_PCLK	VPFE_PCLK	I	Pixel clock	HiZ

(1) When configured for that function; I = Input, O = Output

#### Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

### 12.9.3 VPFE Integration

This section describes module integration in the device, including information about clocks, resets, and hardware requests.

### 12.9.3.1 VPFE Integration in MAIN Domain

There is one VPFE module integrated in the device MAIN domain - VPFE0. Figure 12-1179 shows its integration in the device.

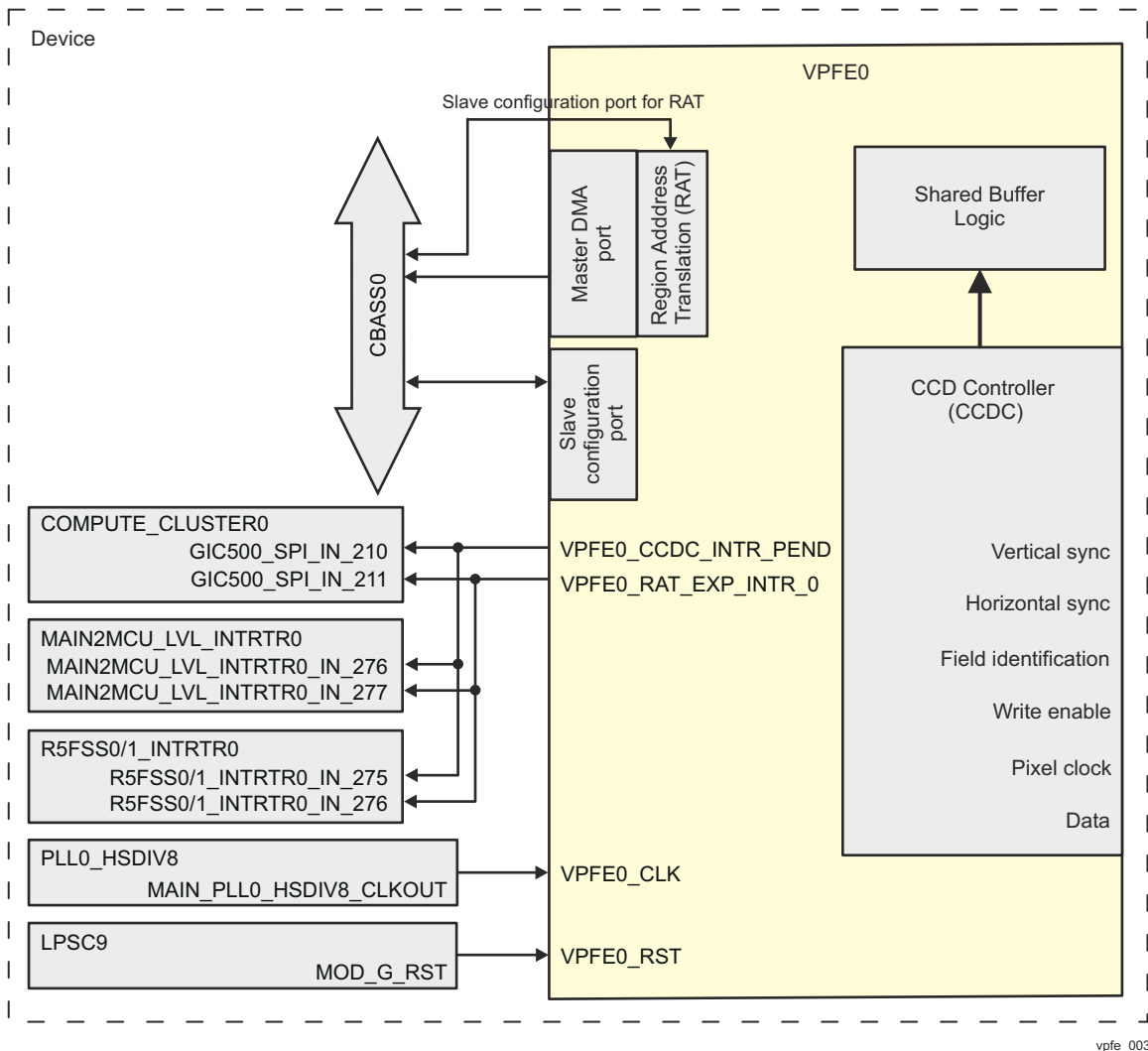


Figure 12-1179. VPFE Integration

Table 12-1555 through Table 12-1557 summarize the integration of VPFE0 in device MAIN domain.

Table 12-1555. VPFE0 Integration Attributes

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
VPFE0	PSC0	PD0	LPSC9	CBASS0

Table 12-1556. VPFE0 Clocks and Resets

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
VPFE0	VPFE0_CLK	MAIN_PLL0_HSDIV8_CLKO UT	PLL0_HSDIV8	VPFE0 clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description

**Table 12-1556. VPFE0 Clocks and Resets (continued)**

VPFE0	VPFE0_RST	MOD_G_RST	LPSC9	VPFE0 reset
-------	-----------	-----------	-------	-------------

**Table 12-1557. VPFE0 Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
VPFE0	VPFE0_CCDC_INTR_PEND	GIC500_SPI_IN_210	COMPUTE_CLUSTER0	VPFE interrupt to COMPUTE_CLUSTER0	Level
		MAIN2MCU_LVL_INTRTR0_IN_276	MAIN2MCU_LVL_INTRTR0	VPFE interrupt to MAIN2MCU_LVL_INTRTR0 router	
		R5FSS0_INTRTR0_IN_275	R5FSS0_INTRTR0	VPFE interrupt to R5FSS0_INTRTR0	
		R5FSS1_INTRTR0_IN_275	R5FSS1_INTRTR0	VPFE interrupt to R5FSS1_INTRTR0	
	VPFE0_RAT_EXP_IN_TR_0	GIC500_SPI_IN_211	COMPUTE_CLUSTER0	VPFE RAT exception interrupt to COMPUTE_CLUSTER0	Level
		MAIN2MCU_LVL_INTRTR0_IN_277	MAIN2MCU_LVL_INTRTR0	VPFE RAT exception interrupt to MAIN2MCU_LVL_INTRTR0 router	
		R5FSS0_INTRTR0_IN_276	R5FSS0_INTRTR0	VPFE RAT exception interrupt to R5FSS0_INTRTR0	
		R5FSS1_INTRTR0_IN_276	R5FSS1_INTRTR0	VPFE RAT exception interrupt to R5FSS1_INTRTR0	

### Note

VPFE interrupts are further described in [Section 12.9.4.3, VPFE Interrupts](#).

For more information on the interconnects, see [Chapter 3, System Interconnect](#).

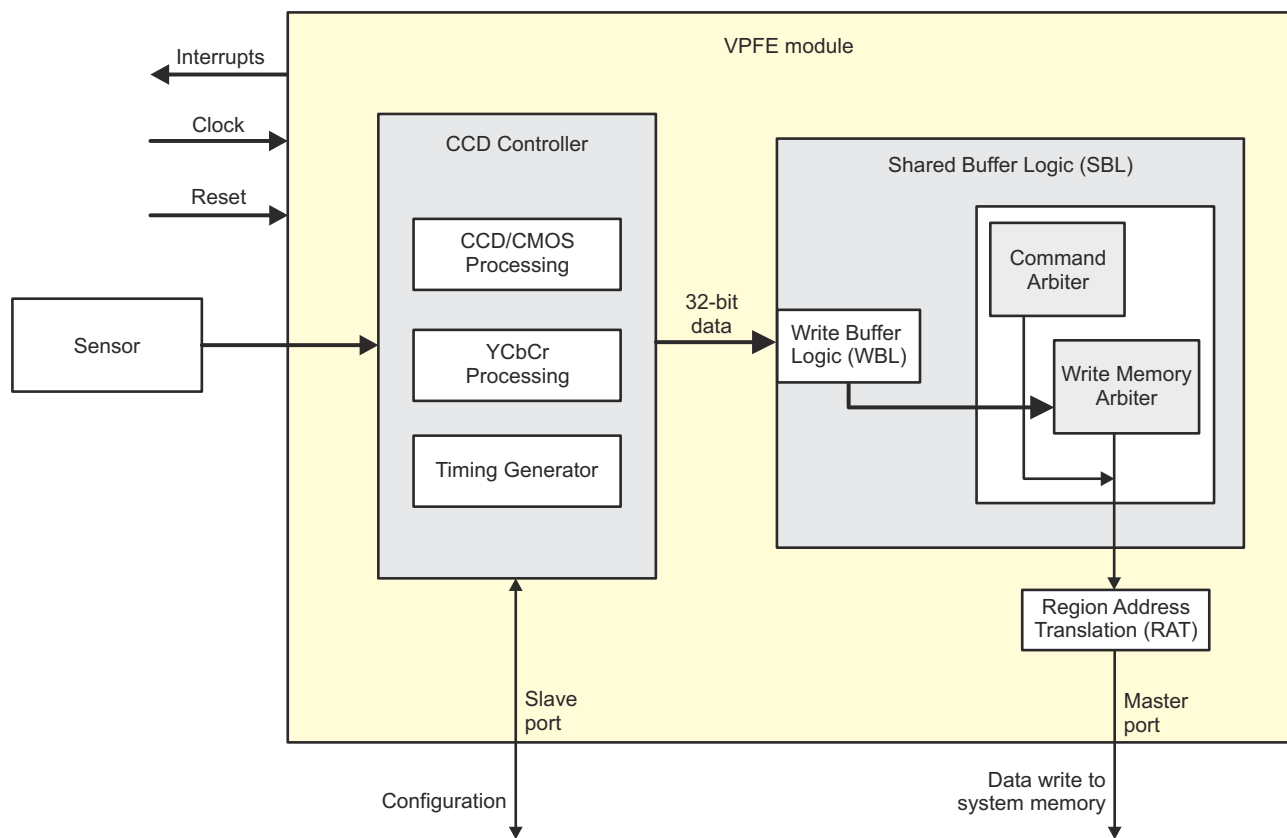
For more information on the power, reset and clock management, see the corresponding sections within [Chapter 5, Device Configuration](#).

For more information on the device interrupt controllers, see [Section 9.2, Interrupt Controllers](#).

## 12.9.4 VPFE Functional Description

### 12.9.4.1 VPFE Block Diagram

Figure 12-1180 shows the simplified internal block diagram of VPFE.



vpfe\_004

**Figure 12-1180. VPFE Block Diagram**

The VPFE module is mainly comprised of two processing blocks:

- CCD controller (CCDC) for data capture.
- Shared Buffer Logic (SBL) for data transfer to DDR.

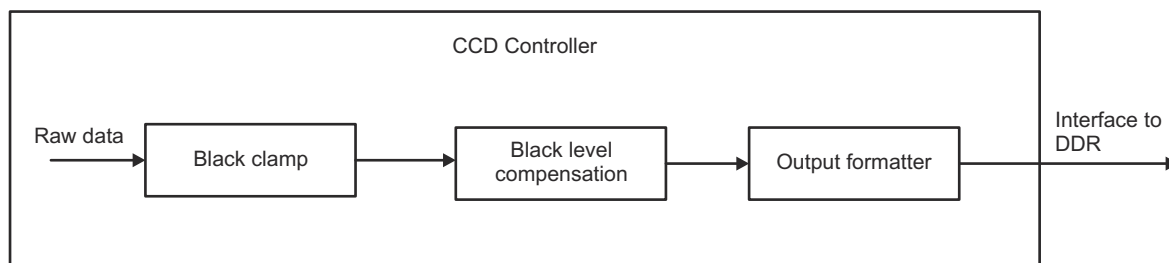
Additionally, a Region-based Address Translation (RAT) block is included to extent DMA address map space.

The VPFE module has the following two system data bus interfaces:

- Slave interface for VPFE configuration by a host processor.
- Master interface for writing data to the system memory by VPFE.

#### 12.9.4.1.1 CCD Controller (CCDC)

The CCDC accepts raw image/video data from an external CCD/CMOS sensor and performs minimal image processing before it outputs the data to DDR, see Figure 12-1181. The CCDC can also accept REC656/CCIR-656 input data.



vpfe\_005

**Figure 12-1181. CCD/CMOS Processing**

#### 12.9.4.1.2 Shared Buffer Logic (SBL)

The shared buffer logic/memory provides local buffering of the captured video data from CCDC module and performs bandwidth efficient burst DMA transfers to DDR memory via a 32-bit wide high bandwidth system data bus. The shared buffer logic/memory (divided into the write buffer and arbitration logic) is capable of performing the following functions:

- Interface to the CCDC module. Receives data from the CCDC and provides a zero latency interface.
- Write buffer logic (WBL) block to interface between the respective module/write port and the write buffer memory.
- A command arbiter to generate the larger VBUSP commands.
- Make appropriate VBUSP requests to the DMA interface to send data to the DDR. The data resides in the write buffer memory until enough data exists to make a large enough burst to be efficient.

#### 12.9.4.1.3 Region-based Address Translation

The region-based address translation works by translating a 32-bit input address into a 48-bit output address. Any input transaction that starts inside of a programmed region will have its address translated, if the region is enabled. Any disabled region is ignored from the translation lookup. For more information about RAT, see [Section 8.4, Region-based Address Translation \(RAT\) Module](#).

#### 12.9.4.2 VPFE Power Management

The following software sequence enables the controller in the VPFE module (CCDC) to receive video or image capture data.

1. Power-up the VPFE module using the device Power Sleep Controller (PSC).
2. Disable the CCDC by setting VPFE\_PCR[1] ENABLE bit to 0.
3. Configure the VPFE registers to match the system requirements, including interrupt setup.
4. Enable the CCDC by setting VPFE\_PCR[1] ENABLE bit to 1.

The CCDC should be enabled prior to data transmission from the external device to avoid data loss. Once enabled, it starts to process input data continuously. No user intervention is necessary to re-start the process. As a result, the VPFE interrupt must be set up correctly in order to operate properly. User can terminate its operation by disabling the CCDC.

#### 12.9.4.3 VPFE Interrupts

The VPFE module can generate one interrupt signal (VPFE0\_CCDC\_INTR\_PEND) to the device host interrupt controllers and one interrupt signal from the RAT in the module (VPFE0\_RAT\_EXP\_INTR\_0).

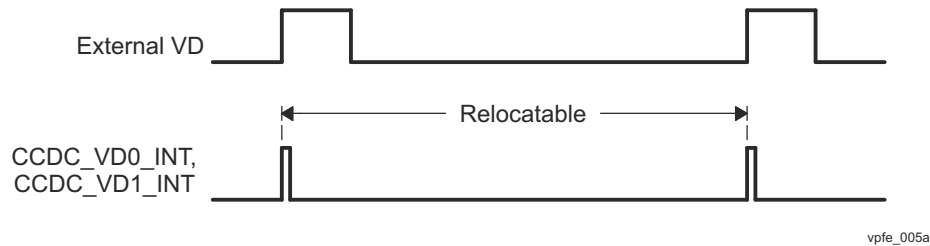
The CCDC controller can generate three interrupts: CCDC\_VD0\_INT, CCDC\_VD1\_INT, and CCDC\_VD2\_INT. These signals are passed through interrupt generator to generate the CCDC\_PEND\_INTR.

#### Note

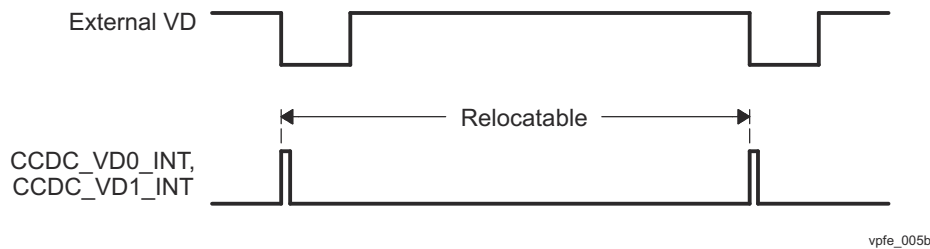
Enable the VPFE\_SYNMODE[16] VDHEN to receive any of the VPFE controller interrupts.

The CCDC\_VD0\_INT and CCDC\_VD1\_INT interrupts occur relative to the VD pulse, as shown in [Figure 12-1182](#) and [Figure 12-1183](#). Note that VPFE\_SYNMODE[2] VDPOL bit changes the trigger timing.

- If VPFE\_SYNMODE[2] VDPOL = 1, the VDINT0 and VDINT1 counters begin counting CCDC\_HD pulses from the falling edge of the external VD.
- If VPFE\_SYNMODE[2] VDPOL = 0, the VDINT0 and VDINT1 counters begin counting CCDC\_HD pulses from the rising edge of the external VD.
- Interrupts CCDC\_VD0\_INT and CCDC\_VD1\_INT occur after receiving the number of horizontal lines (CCDC\_HD pulse signals) set in the VPFE\_VDINT[30-16] VDINT0 and VPFE\_VDINT[14-0] VDINT1 bitfields, respectively.

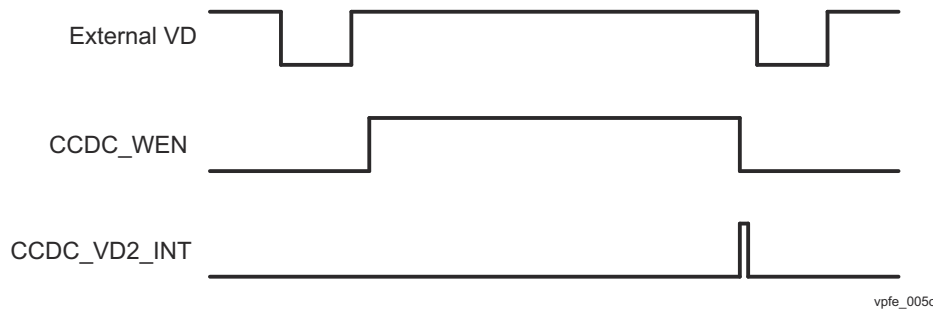


**Figure 12-1182. CCDC\_VD0\_INT/CCDC\_VD1\_INT Interrupt Behavior when VDPOL = 0**



**Figure 12-1183. CCDC\_VD0\_INT/CCDC\_VD1\_INT Interrupt Behavior when VDPOL = 1**

The CCDC\_VD2\_INT interrupt always occurs at the falling edge of the CCDC\_WEN signal (via an external pin), as shown in [Figure 12-1184](#). There are no registers in the VPFE module to configure this interrupt.



**Figure 12-1184. CCDC\_VD2\_INT Interrupt Behavior**

The required software behavior for handling interrupts is as follows:

- Program the interrupt mask by writing a '1' into the correct bitfields of VPFE\_IRQ\_ENABLE\_SET.
- An interrupt is detected by the interrupt controller .
- Read the VPFE\_IRQ\_STATUS register and handle the different interrupt status bits.
- Write into VPFE\_IRQ\_STATUS the value read previously to specify the bits which have been handled.
- Write into the VPFE\_IRQ\_EOI register to enable the interrupt to propagate through the interrupt controllers.

For more details on programmable configuration of the module interrupt signals mapping to the different device host interrupt controllers, see *Interrupt Controllers*.

### 12.9.4.4 VPFE Register Configuration

#### 12.9.4.4.1 General Register Setup

Table 12-1558 lists the minimum register bitfields that must be configured.

**Table 12-1558. Basic Configuration of VPFE Registers**

Register	Bit and Bitfield Name	Function
VPFE_SYNMODE	[16] VDHDEN [7] FLDMODE [6] DATAPOL [5] EXWEN [4] FLDPOL [3] HDPOL [2] VDPOL	External signal configuration (This includes signals CCDC_VD, CCDC_HD, CCDC_FIELD, and CCDC_WEN)
VPFE_CCDCFG	[15] VDLC	Latching function on VSYNC
VPFE_REC656IF	[0] R656ON	REC656 interface
VPFE_SYNMODE	[13-12] INPMOD	Input data mode
VPFE_COLPTN	All bitfields	Color pattern
VPFE_BLKCOMP	All bitfields	Black compensation
VPFE_SYNMODE	[17] WEN	Data path configuration

#### 12.9.4.4.2 Status

The status bit (VPFE\_PCR[1] BUSY) is set when the start of frame occurs (if the VPFE\_PCR[0] ENABLE bit is 1 at that time). It automatically resets to 0 at the end of a frame.

#### 12.9.4.4.3 CCDC\_VD Latched Registers

The VPFE\_CCDCFG[15] VDLC bit affects the access of the following registers and register bits.

- VPFE\_PCR register
- VPFE\_HORZ\_INFO register
- VPFE\_VERT\_START register
- VPFE\_VERT\_LINES register
- VPFE\_CULLING register
- VPFE\_HSIZE\_OFF register
- VPFE\_SDOFST register
- VPFE\_SDR\_ADDR register
- VPFE\_SYNMODE[17] WEN bit
- VPFE\_SYNMODE[14] LPF bit
- VPFE\_CLAMP[31] CLAMPEN bit
- VPFE\_CCDCFG[4] YCINSWP bit

#### Note

If the VPFE\_CCDCFG[15] VDLC is 0, changes to the above registers/register bits takes effect immediately.

If the VPFE\_CCDCFG[15] VDLC is 1, changes to the above registers/register bits takes effect at the start of a new frame (which means the changes are latched by the CCDC\_VD signal). Reads from these registers/register bits returns the most recent write value (even though these values may not take effect).

#### 12.9.4.4.3.1 Inter-Frame Operations

VPFE\_PCR and VPFE\_SDR\_ADDR registers are latched by the CCDC\_VD signal if the VPFE\_CCDCFG[15] VDLC is 0 (see *VPFE CCDC\_VD Latched Registers* for more information). This important feature provides a reliable way to enable/disable the VPFE module and/or modify the memory pointers in-between frames. For example, the VPFE interrupt service routine can program the VPFE\_SDR\_ADDR register to a new value before



the end of the current frame. By doing so, the current frame is received without any interruption and the new external memory address (VPFE\_SDR\_ADDR register) is used for receiving the next frame.

#### 12.9.4.5 VPFE Limitations

The major limitations of the VPFE module are as follows:

- The CCDC\_PCLK (pixel clock) signal shall be up to 110 MHz.
- Both VPFE\_SDR\_ADDR and VPFE\_HSIZE\_OFF registers should be programmed to values with 5 LBS bits as 0; namely, the memory space they point to should be on a 32-byte boundary.
- The VPFE\_COLPTN register should be set to 0 in YCbCr and BT.656 modes.
- The VPFE\_BLKCOMP register should be set to 0 in YCbCr and BT.656 modes.
- Low-pass filter through VPFE\_SYNMODE[14] LPF should be disabled in YCbCr and BT.656 modes.
- A-law transformation should be disabled through VPFE\_ALAW register in YCbCr and BT.656 modes.

#### 12.9.4.6 VPFE Interfaces

The following subsections explain how to connect VPFE signal pins to various input devices.

##### 12.9.4.6.1 Interfaces Summary

[Table 12-1559](#) summarizes the VPFE signal pins required by common input devices.

**Table 12-1559. Summary of VPFE Signal Pins and Common Input Devices**

Signal Name	Type <sup>(1)</sup>	Description	Raw Data Mode 16-bit	Raw Data Mode 8-bit	Raw Data Mode, no FIELD or WEN 16-bit	Raw Data Mode, no FIELD or WEN 8-bit	BT.656 Mode 16-bit	BT.656 Mode 8-bit	Digital YCbCr Mode 16-bit	Digital YCbCr Mode 8-bit
CCDC_HD	I	Horizontal sync	✓	✓	✓	✓	-	-	✓	✓
CCDC_VD	I	Vertical sync	✓	✓	✓	✓	-	-	✓	✓
CCDC_FIELD	I	Field identification	✓	✓	-	-	-	-	✓	✓
CCDC_WEN	I	Write enable	✓	✓	-	-	-	-	✓	✓
CCDC_PCLK	I	Pixel clock	✓	✓	✓	✓	✓	✓	✓	✓
CCDC_DATA0 (YIN[0])	I	VPFE data (bit 0)	✓	✓	✓	✓	✓	✓	✓	✓
CCDC_DATA1 (YIN[1])	I	VPFE data (bit 1)	✓	✓	✓	✓	✓	✓	✓	✓
CCDC_DATA2 (YIN[2])	I	VPFE data (bit 2)	✓	✓	✓	✓	✓	✓	✓	✓
CCDC_DATA3 (YIN[3])	I	VPFE data (bit 3)	✓	✓	✓	✓	✓	✓	✓	✓
CCDC_DATA4 (YIN[74])	I	VPFE data (bit 4)	✓	✓	✓	✓	✓	✓	✓	✓
CCDC_DATA5 (YIN[5])	I	VPFE data (bit 5)	✓	✓	✓	✓	✓	✓	✓	✓
CCDC_DATA6 (YIN[6])	I	VPFE data (bit 6)	✓	✓	✓	✓	✓	✓	✓	✓
CCDC_DATA7 (YIN[7])	I	VPFE data (bit 7)	✓	✓	✓	✓	✓	✓	✓	✓
CCDC_DATA8 (CIN[0])	I	VPFE data (bit 8)	✓	-	✓	-	✓	-	✓	-
CCDC_DATA9 (CIN[1])	I	VPFE data (bit 9)	✓	-	✓	-	✓	-	✓	-
CCDC_DATA10 (CIN[2])	I	VPFE data (bit 10)	✓	-	✓	-	✓	-	✓	-
CCDC_DATA11 (CIN[3])	I	VPFE data (bit 11)	✓	-	✓	-	✓	-	✓	-
CCDC_DATA12 (CIN[4])	I	VPFE data (bit 12)	✓	-	✓	-	✓	-	✓	-

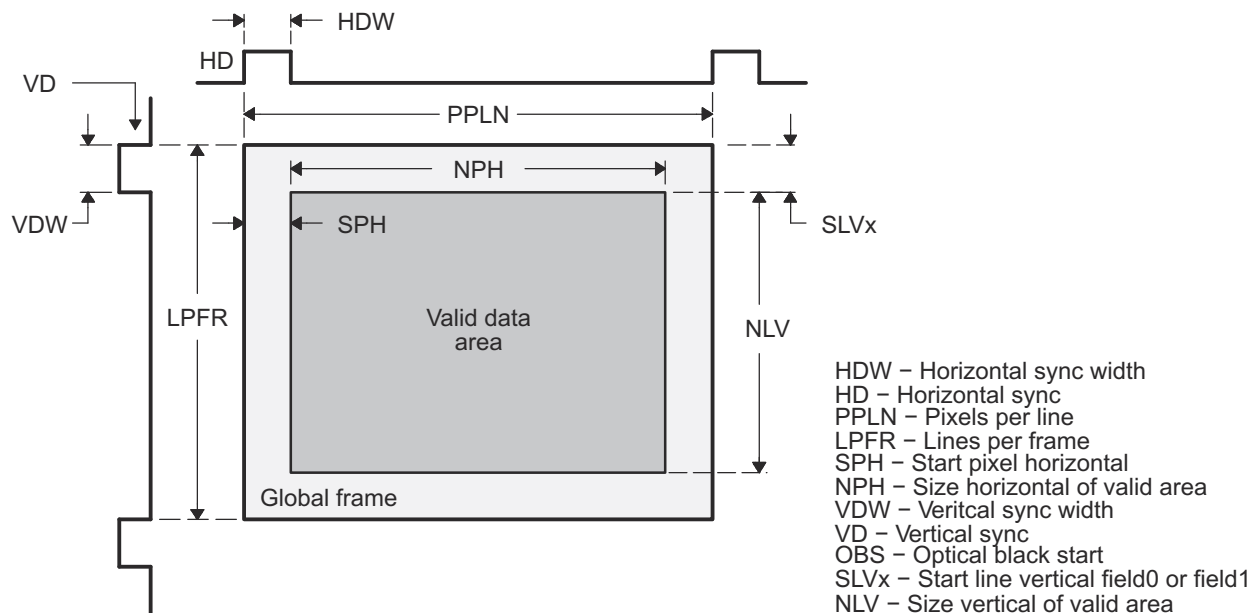
**Table 12-1559. Summary of VPFE Signal Pins and Common Input Devices (continued)**

Signal Name	Type <sup>(1)</sup>	Description	Raw Data Mode 16-bit	Raw Data Mode 8-bit	Raw Data Mode, no FIELD or WEN 16-bit	Raw Data Mode, no FIELD or WEN 8-bit	BT.656 Mode 16-bit	BT.656 Mode 8-bit	Digital YCbCr Mode 16-bit	Digital YCbCr Mode 8-bit
CCDC_DATA13 (CIN[5])	I	VPFE data (bit 13)	✓	-	✓	-	✓	-	✓	-
CCDC_DATA14 (CIN[6])	I	VPFE data (bit 14)	✓	-	✓	-	✓	-	✓	-
CCDC_DATA15 (CIN[7])	I	VPFE data (bit 15)	✓	-	✓	-	✓	-	✓	-

(1) I = Input, O = Output

#### 12.9.4.6.2 Timing Generator and Frame Settings

The timing generator in the CCDC either enables the use of external sync signals (HD/VD) or internal generated timing signals. The CPU controls width, polarity, position and direction of internal generated signals. [Figure 12-1185](#) shows various CCDC register settings related to the timing. The shaded area is the physical imager size and the gray area is the valid data area. The image data in this area is processed and stored to external DDR. The vertical start position for even and odd fields are configured independently.



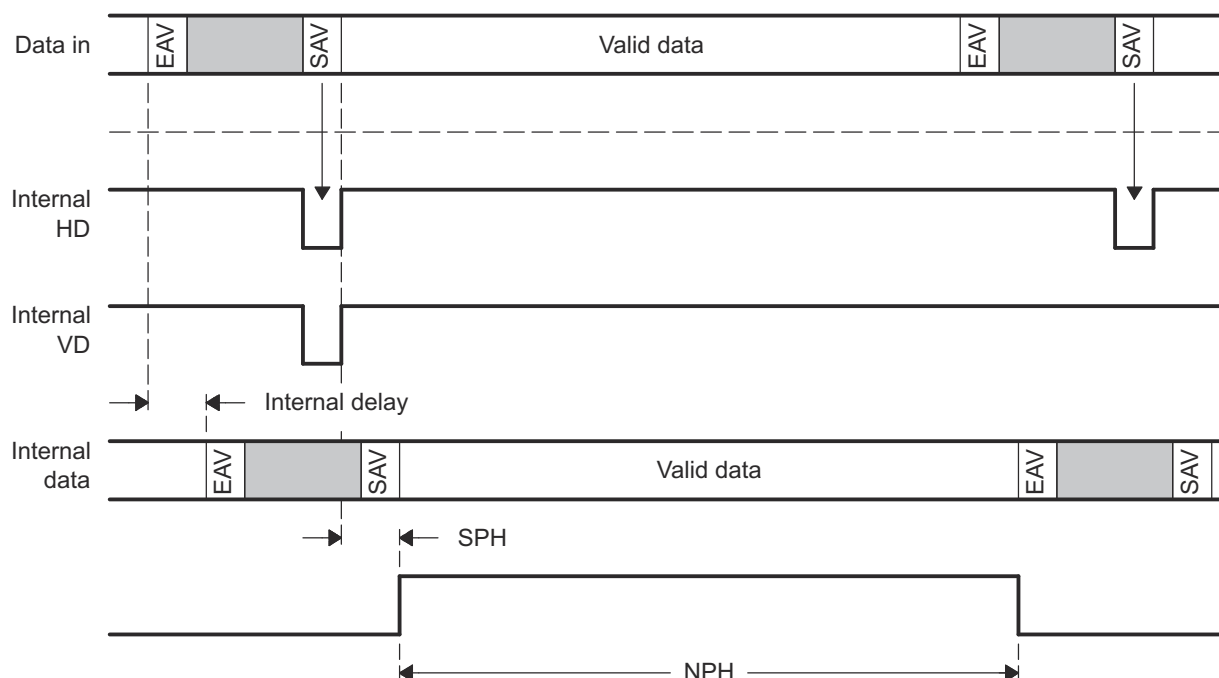
**Figure 12-1185. CCD Controller Frame Settings and Format**

#### 12.9.4.6.3 ITU-R BT.656 Interface

The ITU-R BT.656 interface supports either 8-bit or 16-bit processing of input video YCbCr data. See [Table 12-1559](#) for signals used to configure this mode.

Since the data synchronization information is carried along with the data lines, no synchronization signals (that is CCDC\_HD, CCDC\_VD, and CCDC\_FIELD) are necessary in this mode.

Two timing reference codes are transmitted as the synchronization signal. At the start and end of each video data block, two unique codes are sent, respectively. The start code is called the Start of Active Video Signal (SAV), and the end code is called the End of Active Video Signal (EAV). The SAV and EAV codes proceed and follow valid data, as shown in [Figure 12-1186](#). The controller in the VPFE internally bases on SAV and EAV codes to generate the necessary synchronization signals, that is, horizontal sync, vertical sync, and field ID.



vpfe\_007

**Figure 12-1186. BT.656 Signal Interface**

Both timing reference signals, SAV and EAV, consist of a four-word sequence in the following format: FF 00 00 XY, where FF 00 00 are a set preamble and the fourth word defines the field identification, the state of vertical field blanking, the state of horizontal line blanking, and protection (error correction) codes. The bit format of the fourth word is shown in [Table 12-1560](#) and the definitions for bits, F, V, and H, are given in [Table 12-1561](#). F, V, and H are used in place of the usual horizontal sync, vertical sync, and blank timing control signals. Bits P3, P2, P1, and P0 are protection (error correction) bits for F, V, and H. The relationship between F, V, and H and the protection (error correction) bits is given in [Table 12-1562](#). To enable error correction, set the VPFE\_REC656IF[1] ECCFVH bit. The controller in the VPFE automatically detects and applies error correction when the VPFE\_REC656IF[1] ECCFVH bit is set.

**Table 12-1560. Video Timing Reference Codes for SAV and EAV**

Data Bit Number	First Word (FF)	Second Word (00)	Third Word (00)	Fourth Word (XY)
9 (MSB)	1	0	0	1
8	1	0	0	F
7	1	0	0	V
6	1	0	0	H
5	1	0	0	P3
4	1	0	0	P2
3	1	0	0	P1
2	1	0	0	P0
1	1	0	0	0
0	1	0	0	0

**Table 12-1561. F, V, H Signal Descriptions**

Signal	Value	Command
F	0	Field 1
	1	Field 2

**Table 12-1561. F, V, H Signal Descriptions  
(continued)**

Signal	Value	Command
V	0	0
	1	Vertical blank
H	0	SAV
	1	EAV

**Table 12-1562. F, H, V Protection (Error Correction)  
Bits**

F	V	H	P3	P2	P1	P0
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	1	1
1	0	1	1	0	1	0
1	1	0	1	1	0	0
1	1	1	0	0	0	1

#### Note

The controller in the VPFE outputs the XY code in the SAV and EAV into the external memory. In order to eliminate this, set the VPFE\_HORZ\_INFO[30-16] SPH field to +1. Also set the VPFE\_HORZ\_INFO[14-0] NPH bitfield to accurately represent the number of active pixels.

#### 12.9.4.6.4 Digital YCbCr Interface

The digital YCbCr interface supports either 8-bit or 16-bit devices. See [Table 12-1559](#) for signals used to configure this mode.

Unlike the BT.656 mode, discrete horizontal sync (CCDC\_HD) and vertical sync (CCDC\_VD) signals are required. An NTSC/PAL decoder is an example device that can be connected to the YCbCr interface.

Data lines YIN[7-0] or CIN[7-0] are used for input in 8-bit mode. Alternately, two separate devices can be connected; however, only one can be active at any given time. Setting the VPFE\_CCDCFG[4] YCINSWP bit determines which set of 8-bit inputs are active.

Data lines YIN[7-0] and CIN[7-0] are used for input in 16-bit mode. Use the VPFE\_CCDCFG[4] YCINSWP bit to swap the Y and Cr/Cb data lines.

### 12.9.4.7 VPFE Data / Image Processing

This section describes the data / image processing of each module in the VPFE in more detail. [Section 12.9.4.7.1](#) describes the Raw Data mode and [Section 12.9.4.7.2](#) explains the YCbCr and BT.656 modes.

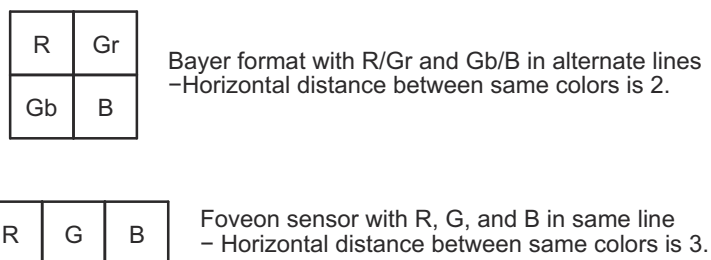
#### 12.9.4.7.1 Raw Data Mode

Raw data mode is enabled by setting VPFE\_SYNMODE[13-12] INPMOD bitfield to 0 and setting VPFE\_REC656IF[0] R656ON bit to 0. [Figure 12-1187](#) shows the corresponding data processing diagram.



**Figure 12-1187. Data Processing for Raw Data Mode**

Typically, raw sensor data is one color per pixel in a color filter array (CFA). The color filter array applied is typically a Bayer pattern, as shown in [Figure 12-1188](#) for RGB color space. Alternatively, the special Foveon X3-family sensors capture R, G, and B lights at each pixel location. Both Bayer and Foveon sensors are supported by the controller in the VPFE module.

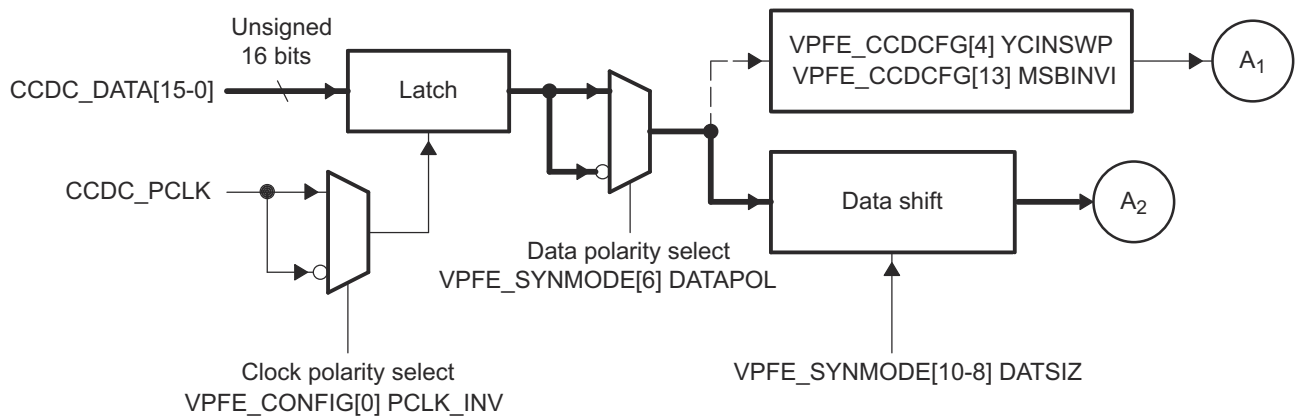


**Figure 12-1188. Color Patterns**

#### 12.9.4.7.1.1 Input Sampling and Formatting for Raw Data Mode

The data path of raw data mode is shown in the thicker lines in [Figure 12-1189](#) (that is A2). Data path A1 is applicable to YUV input mode only.

- The pixel clock (CCDC\_PCLK) latches the data.
- Pixel clock polarity can be either rising or falling edge and is set in VPFE clock control register (VPFE\_CONFIG[0] PCLK\_INV).
- VPFE\_SYNMODE[6] DATAPOL bit affects the data representation.
- Data is right-shifted to align the data in the least significant bits of the data bus and provide the maximum dynamic range for the remainder of the processing (VPFE\_SYNMODE[10-8] DATSIZ bitfield). This also sets the maximum data size allowed in subsequent clipping/limiting operations and is the output data alignment when it is written to external memory.

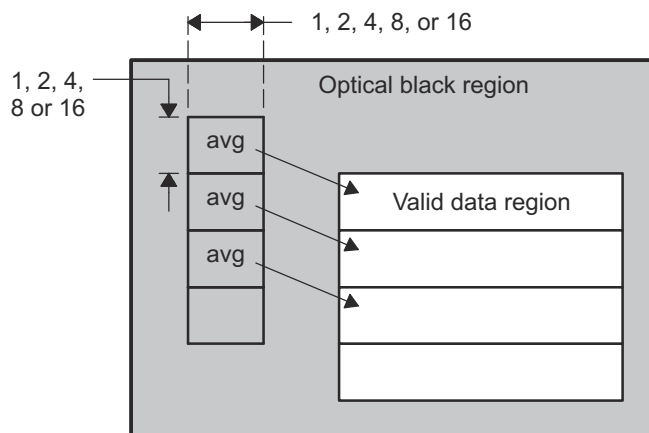


vpfe\_010

**Figure 12-1189. Input Sampling Block Diagram for Raw Data Mode**

#### 12.9.4.7.1.2 Optical Black Clamping for Raw Data Mode

Some sensors provide few optically masked pixels at the beginning/end of each line which allow user to determine the noise floor on any given frame of data. The optical black clamping function provides a means to average the optically black pixels and subtract that value from each input pixel as the first step in reducing the noise on the input pixels.



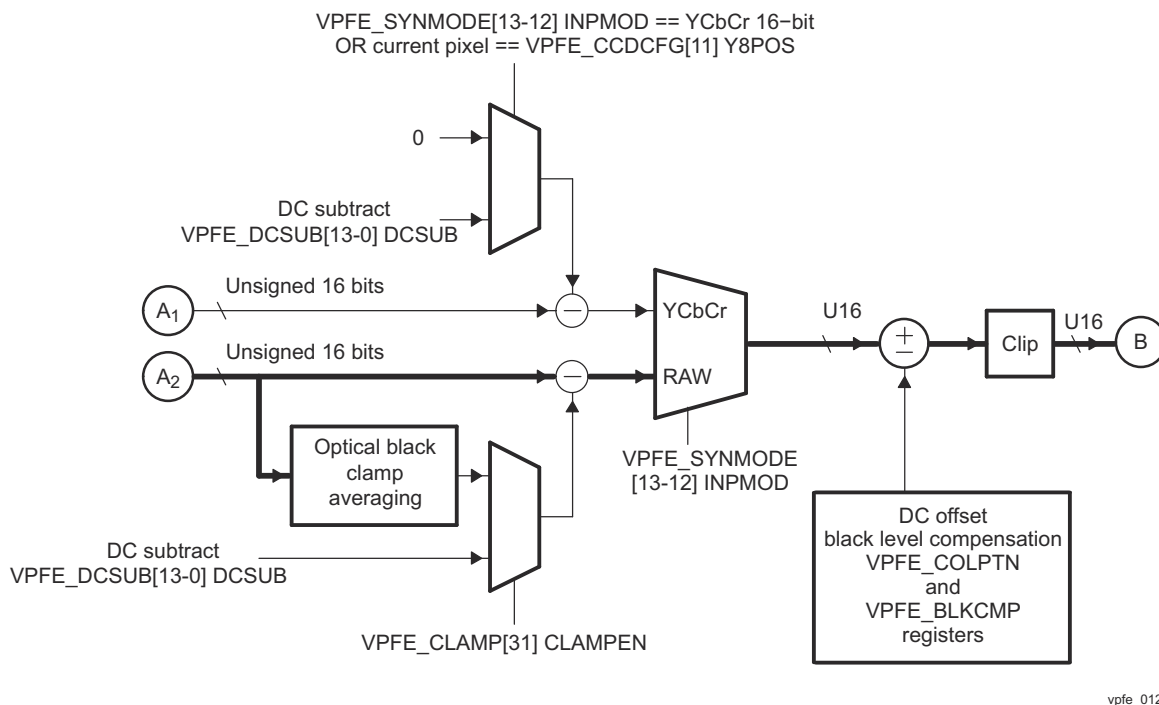
**Figure 12-1190. Optical Black Averaging and Application for Raw Data Mode**

The averaging circuit takes an average of masked (black) pixel values from the image sensor, averaging pixels at the start (in the VPFE\_CLAMP[24-10] OBST bitfield) of each line (in the VPFE\_CLAMP[30-28] OBSLEN bitfield) and for the number of indicated lines (VPFE\_CLAMP[27-25] OBSLN bitfield), plus an optional gain adjustment (VPFE\_CLAMP[4-0] OBGAIN bitfield). The resultant value is subtracted from the image data at the succeeding line. User can control the position of the black pixels, the number of pixels (1, 2, 4, 8, or 16) in each line that are averaged, and the number of lines (1, 2, 4, 8, or 16) that are averaged.

Alternately, user can disable black clamp averaging (VPFE\_CLAMP[31] CLAMPEN bit) and select a constant black value for subtraction (VPFE\_DCSUB[13-0] DCSUB bit) instead of using the calculated average value.

The corresponding data path is shown in the thicker lines in [Figure 12-1191](#).





vpfe\_012

**Figure 12-1191. Black Clamping and Black Level Compensation for Raw Data Mode**

#### 12.9.4.7.1.3 Black Level Compensation

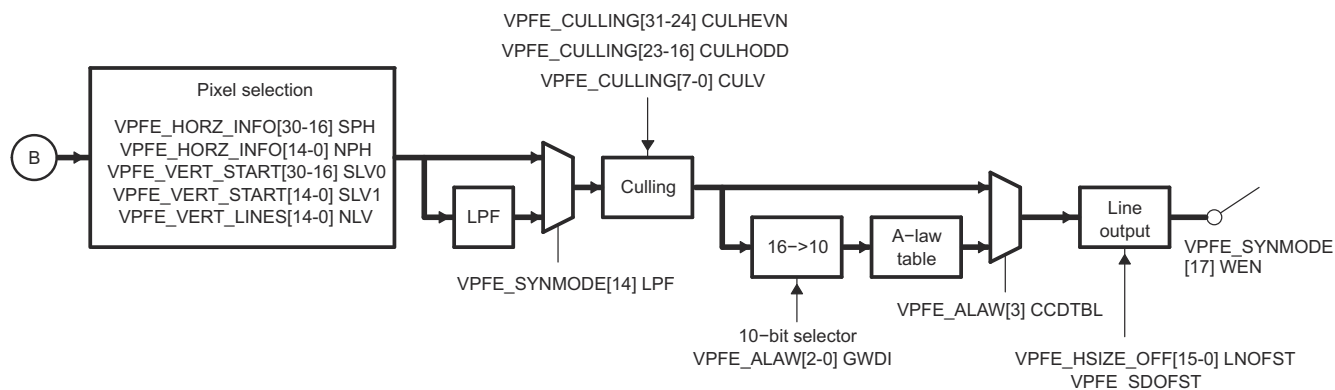
After the digital clamping is applied to the data, black level compensation is applied (see [Figure 12-1191](#)). In this operation, a fixed value is subtracted from the data depending on the color (that is R/Ye, Gr/Cy, Gb/G, and B/Mg). The offset (VPFE\_BLKCOMP register) that is applied to each data sample is selected according to the 0/1/2/3 phase and the color (0/1/2/3) specified for each phase (VPFE\_COLPTN). The color pattern definition is very flexible in order to accommodate many different capture devices, including normal Bayer CFA sampling, Foveon sensors, and VGA movie mode CCDs, whose VGA draft mode output does not follow the typical Bayer pattern.

#### 12.9.4.7.1.4 Output Formatter for Raw Data Mode

The final stage of VPFE processing is the output formatter, shown on [Figure 12-1192](#). A framing selection is applied to limit the processing area by the settings in the VPFE\_HORZ\_INFO, VPFE\_VERT\_START and VPFE\_VERT\_LINES registers.

#### Note

In addition to the framing applied at the beginning and end of the data formatter operation, user must apply a framing selection to limit the processing area. User must ensure that the settings are relative to that frame.



vpfe\_013

**Figure 12-1192. Output Formatter for Raw Data Mode**

#### 12.9.4.7.1.4.1 Low Pass Filter (LPF)

Use the VPFE\_SYNMODE[14] LPF bit to apply an optional low-pass filter after the reframing. The low-pass filter consists of a simple 3-tap ( $\frac{1}{4}$ ,  $\frac{1}{2}$ ,  $\frac{1}{4}$ ) filter. Two pixels on the left and two pixels on the right of each line are cropped if the filter is enabled.

#### 12.9.4.7.1.4.2 Culling

Use the VPFE\_CULLING[31-24] CULHVEN and VPFE\_CULLING[23-16] CULHODD bitfields (8-bit repeating mask, one per field) to enable an optional culling operation, which culls (deletes) selected pixel data from a line. Use the VPFE\_CULLING[7-0] CULV bitfield to select lines from a frame.

[Table 12-1563](#) illustrates how the register values apply the decimation pattern to the data. The pixels in white will be saved to external memory and the shaded pixels are discarded.

In this case:

- VPFE\_CULLING = 59C4 0066h, with
- VPFE\_CULLING[31-24] CULHVEN = 59h,
- VPFE\_CULLING[23-16] CULHODD = C4h, and
- VPFE\_CULLING[7-0] CULV = 66h.

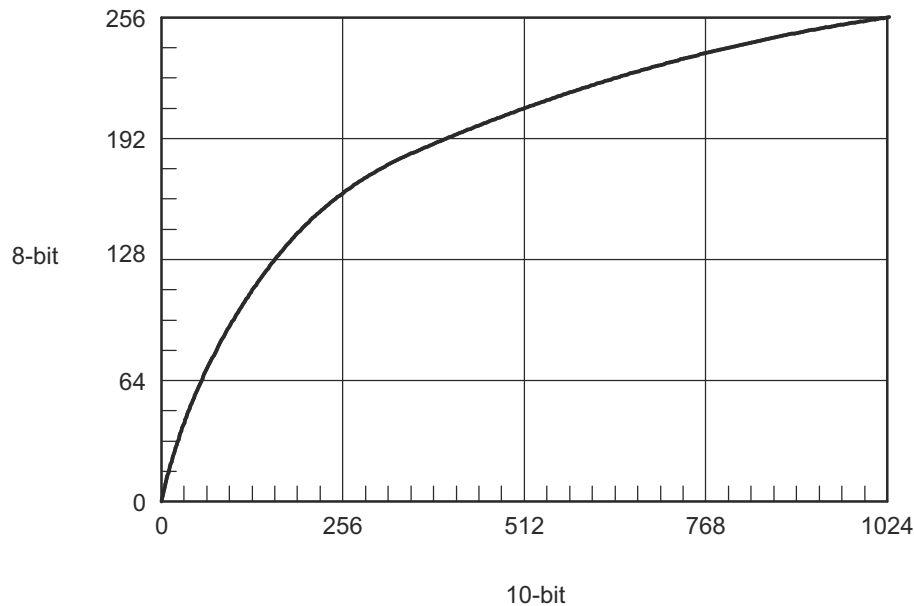
**Table 12-1563. Example for Decimation Pattern**

	MSB							LSB	
CULHEVN	0	1	0	1	1	0	0	1	
CULHODD	1	1	0	0	0	1	0	0	
0th line									0
1st line									1
2nd line									1
3rd line									0
4th line									0
5th line									1
6th line									1
7th line									0
									CULV

#### 12.9.4.7.1.4.3 A-Law Transformation

Use the VPFE\_ALAW[3] CCDBTL bit to apply an optional 10-to-8-bit A-Law transformation using a fixed A-Law table (see [Figure 12-1193](#), [Table 12-1564](#) and [Table 12-1565](#)) as the final processing stage. Using this causes the data width to be reduced to 8 bits and allows packing to 8 bits/pixel when saving to external memory. Since

the data resolution can be greater than 10 bits at this stage, user must select the 10 bits for input to the A-Law operation. Use the VPFE\_ALAW[2-0] GWDI register to select the 10 bits for input.



**Figure 12-1193. A-Law Table**

**Table 12-1564. A-Law Table – Part 1**

Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law
0	0	64	64	128	112	192	140	256	161	320	176	384	189	448	200
1	1	65	65	129	113	193	141	257	161	321	176	385	189	449	200
2	2	66	66	130	113	194	141	258	161	322	177	386	189	450	200
3	3	67	67	131	114	195	142	259	161	323	177	387	189	451	200
4	4	68	68	132	114	196	142	260	162	324	177	388	190	452	200
5	5	69	69	133	115	197	142	261	162	325	177	389	190	453	200
6	6	70	70	134	115	198	143	262	162	326	177	390	190	454	201
7	7	71	71	135	116	199	143	263	162	327	178	391	190	455	201
8	8	72	72	136	116	200	143	264	163	328	178	392	190	456	201
9	9	73	73	137	117	201	144	265	163	329	178	393	190	457	201
10	10	74	74	138	117	202	144	266	163	330	178	394	191	458	201
11	11	75	75	139	118	203	144	267	163	331	178	395	191	459	201
12	12	76	76	140	118	204	145	268	164	332	179	396	191	460	201
13	13	77	77	141	119	205	145	269	164	333	179	397	191	461	202
14	14	78	78	142	119	206	145	270	164	334	179	398	191	462	202
15	15	79	78	143	120	207	146	271	164	335	179	399	191	463	202
16	16	80	79	144	120	208	146	272	165	336	179	400	192	464	202
17	17	81	80	145	121	209	146	273	165	337	180	401	192	465	202
18	18	82	81	146	121	210	147	274	165	338	180	402	192	466	202
19	19	83	82	147	122	211	147	275	166	339	180	403	192	467	202
20	20	84	83	148	122	212	147	276	166	340	180	404	192	468	203
21	21	85	84	149	123	213	148	277	166	341	181	405	193	469	203
22	22	86	84	150	123	214	148	278	166	342	181	406	193	470	203
23	23	87	85	151	124	215	148	279	167	343	181	407	193	471	203

**Table 12-1564. A-Law Table – Part 1 (continued)**

Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law
24	24	88	86	152	124	216	149	280	167	344	181	408	193	472	203
25	25	89	87	153	125	217	149	281	167	345	181	409	193	473	203
26	26	90	88	154	125	218	149	282	167	346	182	410	193	474	204
27	27	91	88	155	125	219	150	283	168	347	182	411	194	475	204
28	28	92	89	156	126	220	150	284	168	348	182	412	194	476	204
29	29	93	90	157	126	221	150	285	168	349	182	413	194	477	204
30	30	94	91	158	127	222	151	286	168	350	182	414	194	478	204
31	31	95	91	159	127	223	151	287	168	351	183	415	194	479	204
32	32	96	92	160	128	224	151	288	169	352	183	416	194	480	204
33	33	97	93	161	128	225	152	289	169	353	183	417	195	481	205
34	34	98	93	162	129	226	152	290	169	354	183	418	195	482	205
35	35	99	94	163	129	227	152	291	169	355	183	419	195	483	205
36	36	100	95	164	129	228	152	292	170	356	184	420	195	484	205
37	37	101	96	165	130	229	153	293	170	357	184	421	195	485	205
38	38	102	96	166	130	230	153	294	170	358	184	422	195	486	205
39	39	103	97	167	131	231	153	295	170	359	184	423	196	487	205
40	40	104	98	168	131	232	154	296	171	360	184	424	196	488	206
41	41	105	98	169	132	233	154	297	171	361	185	425	196	489	206
42	42	106	99	170	132	234	154	298	171	362	185	426	196	490	206
43	43	107	100	171	132	235	155	299	171	363	185	427	196	491	206
44	44	108	100	172	133	236	155	300	172	364	185	428	196	492	206
45	45	109	101	173	133	237	155	301	172	365	185	429	197	493	206
46	46	110	102	174	134	238	155	302	172	366	185	430	197	494	206
47	47	111	102	175	134	239	156	303	172	367	186	431	197	495	207
48	48	112	103	176	134	240	156	304	173	368	186	432	197	496	207
49	49	113	103	177	135	241	156	305	173	369	186	433	197	497	207
50	50	114	104	178	135	242	157	306	173	370	186	434	197	498	207
51	51	115	105	179	136	243	157	307	173	371	186	435	198	499	207
52	52	116	105	180	136	244	157	308	173	372	187	436	198	500	207
53	53	117	106	181	136	245	157	309	174	373	187	437	198	501	207
54	54	118	106	182	137	246	158	310	174	374	187	438	198	502	208
55	55	119	107	183	137	247	158	311	174	375	187	439	198	503	208
56	56	120	108	184	137	248	158	312	174	376	187	440	198	504	208
57	57	121	108	185	138	249	159	313	175	377	188	441	198	505	208
58	58	122	109	186	138	250	159	314	175	378	188	442	199	506	208
59	59	123	109	187	139	251	159	315	175	379	188	443	199	507	208
60	60	124	110	188	139	252	159	316	175	380	188	444	199	508	208
61	61	125	110	189	139	253	160	317	175	381	188	445	199	509	208
62	62	126	111	190	140	254	160	318	176	382	188	446	199	510	209
63	63	127	112	191	140	255	160	319	176	383	189	447	199	511	209

**Table 12-1565. A-Law Table – Part 2**

Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law
512	209	576	217	640	224	704	231	768	237	832	243	896	248	960	253
513	209	577	217	641	225	705	231	769	237	833	243	897	248	961	253

**Table 12-1565. A-Law Table – Part 2 (continued)**

Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law
514	209	578	217	642	225	706	231	770	237	834	243	898	248	962	253
515	209	579	217	643	225	707	231	771	237	835	243	899	248	963	253
516	209	580	218	644	225	708	232	772	238	836	243	900	248	964	253
517	210	581	218	645	225	709	232	773	238	837	243	901	248	965	253
518	210	582	218	646	225	710	232	774	238	838	243	902	248	966	253
519	210	583	218	647	225	711	232	775	238	839	243	903	249	967	253
520	210	584	218	648	225	712	232	776	238	840	243	904	249	968	253
521	210	585	218	649	225	713	232	777	238	841	244	905	249	969	253
522	210	586	218	650	226	714	232	778	238	842	244	906	249	970	254
523	210	587	218	651	226	715	232	779	238	843	244	907	249	971	254
524	211	588	219	652	226	716	232	780	238	844	244	908	249	972	254
525	211	589	219	653	226	717	232	781	238	845	244	909	249	973	254
526	211	590	219	654	226	718	233	782	238	846	244	910	249	974	254
527	211	591	219	655	226	719	233	783	239	847	244	911	249	975	254
528	211	592	219	656	226	720	233	784	239	848	244	912	249	976	254
529	211	593	219	657	226	721	233	785	239	849	244	913	249	977	254
530	211	594	219	658	226	722	233	786	239	850	244	914	249	97.8	254
531	211	595	219	659	227	723	233	787	239	851	244	915	249	979	254
532	212	596	220	660	227	724	233	788	239	852	244	916	250	980	254
533	212	597	220	661	227	725	233	789	239	853	245	917	250	981	254
534	212	598	220	662	227	726	233	790	239	854	245	918	250	982	254
535	212	599	220	663	227	727	233	791	239	855	245	919	250	983	254
536	212	600	220	664	227	728	233	792	239	856	245	920	250	984	255
537	212	601	220	665	227	729	234	793	239	857	245	921	250	985	255
538	212	602	220	666	227	730	234	794	240	858	245	922	250	986	255
539	212	603	220	667	227	731	234	795	240	859	245	923	250	987	255
540	213	604	220	668	227	732	234	796	240	860	245	924	250	988	255
541	213	605	221	669	228	733	234	797	240	861	245	925	250	989	255
542	213	606	221	670	228	734	234	798	240	862	245	926	250	990	255
543	213	607	221	671	228	735	234	799	240	863	245	927	250	991	255
544	213	608	221	672	228	736	234	800	240	864	245	928	250	992	255
545	213	609	221	673	228	737	234	801	240	865	246	929	250	993	255
546	213	610	221	674	228	738	234	802	240	866	246	930	251	994	255
547	214	611	221	675	228	739	235	803	240	867	246	931	251	995	255
548	214	612	221	676	228	740	235	804	240	868	246	932	251	996	255
549	214	613	221	677	228	741	235	805	240	869	246	933	251	997	255
550	214	614	222	678	229	742	235	806	241	870	246	934	251	998	255
551	214	615	222	679	229	743	235	807	241	871	246	935	251	999	255
552	214	616	222	680	229	744	235	808	241	872	246	936	251	1000	255
553	214	617	222	681	229	745	235	809	241	873	246	937	251	1001	255
554	214	618	222	682	229	746	235	810	241	874	246	938	251	1002	255
555	215	619	222	683	229	747	235	811	241	875	246	939	251	1003	255
556	215	620	222	684	229	748	235	812	241	876	246	940	251	1004	255
557	215	621	222	685	229	749	235	813	241	877	246	941	251	1005	255
558	215	622	222	686	229	750	236	814	241	878	247	942	251	1006	255

**Table 12-1565. A-Law Table – Part 2 (continued)**

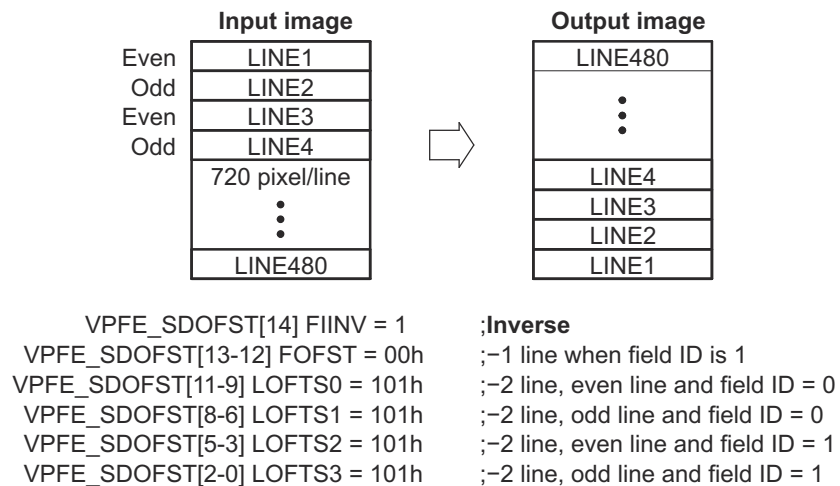
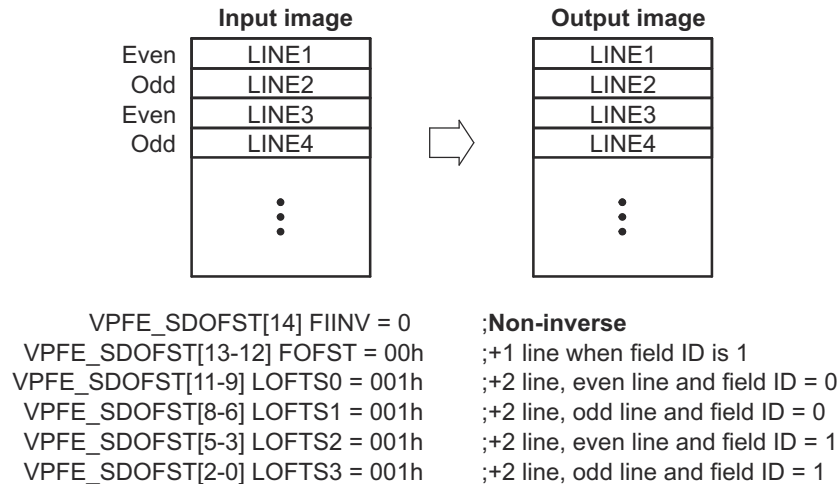
Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law
559	215	623	223	687	229	751	236	815	241	879	247	943	252	1007	255
560	215	624	223	688	230	752	236	816	241	880	247	944	252	1008	255
561	215	625	223	689	230	753	236	817	242	881	247	945	252	1009	255
562	215	626	223	690	230	754	236	818	242	882	247	946	252	1010	255
563	216	627	223	691	230	755	236	819	242	883	247	947	252	1011	255
564	216	628	223	692	230	756	236	820	242	884	247	948	252	1012	255
565	216	629	223	693	230	757	236	821	242	885	247	949	252	1013	255
566	216	630	223	694	230	758	236	822	242	886	247	950	252	1014	255
567	216	631	223	695	230	759	236	823	242	887	247	951	252	1015	255
568	216	632	224	696	230	760	236	824	242	888	247	952	252	1016	255
569	216	633	224	697	230	761	237	825	242	889	247	953	252	1017	255
570	216	634	224	698	231	762	237	826	242	890	247	954	252	1018	255
571	217	635	224	699	231	763	237	827	242	891	248	955	252	1019	255
572	217	636	224	700	231	764	237	828	242	892	248	956	252	1020	255
573	217	637	224	701	231	765	237	829	243	893	248	957	253	1021	255
574	217	638	224	702	231	766	237	830	243	894	248	958	253	1022	255
575	217	639	224	703	231	767	237	831	243	895	248	959	253	1023	255

#### 12.9.4.7.1.4.4 Line Output Control

The final stage of the raw data mode is the line output control, which controls how the input sensor lines are written to external memory. The value of the bits in the VPFE\_SDR\_ADDR register defines the starting address where the frame should be written in external memory. The value of the VPFE\_HSIZE\_OFF[15-0] LNOFST bitfield defines the distance between the beginning of output lines in bytes. Both the starting address and line offset must be aligned to 32-byte boundaries (that is, either 16 or 32 pixels, depending on the VPFE\_SYNMODE[11] PACK8 bit). Use the VPFE\_SDOFST register to define additional offsets, depending on the field ID and the even/odd line numbers.

Figure 12-1194 shows example of input and output images in the two formats - non-inversed and inversed.

**Figure 12-1194. Input and Output Image in Non-Inversed vs Inversed Formats**



vpfe\_015

#### 12.9.4.7.1.4.5 Output Format in External Memory for Raw Data Mode

The pixel data format in external memory is shown in [Table 12-1567](#).

- If pixel data format is 8-bit or if the A-Law compression is applied, every 16-bit word in external memory stores two pixel data with VPFE\_SYNMODE[11] PACK8 bit set.
- If pixel data format is greater than 8-bit, every 16-bit word in external memory stores one pixel data, and the unused bits are MSB which are filled with 0.

**Table 12-1566. Storage Format in External Memory for Raw Data Mode**

Upper Word																Lower Word																			
	MSB (31)									LSB (16)									MSB (15)									LSB (0)							
16-bit									Pixel1									Pixel0																	
15-bit	0								Pixel1	0								Pixel0																	
14-bit		0							Pixel1		0							Pixel0		0															
13-bit			0						Pixel1			0						Pixel0			0														
12-bit				0					Pixel1				0					Pixel0				0													
11-bit					0				Pixel1					0				Pixel0					0												
10-bit						0			Pixel1						0			Pixel0						0											
9-bit							0		Pixel1							0		Pixel0							0										
8-bit								0	Pixel1								0	Pixel0								0									

**Table 12-1566. Storage Format in External Memory for Raw Data Mode (continued)**

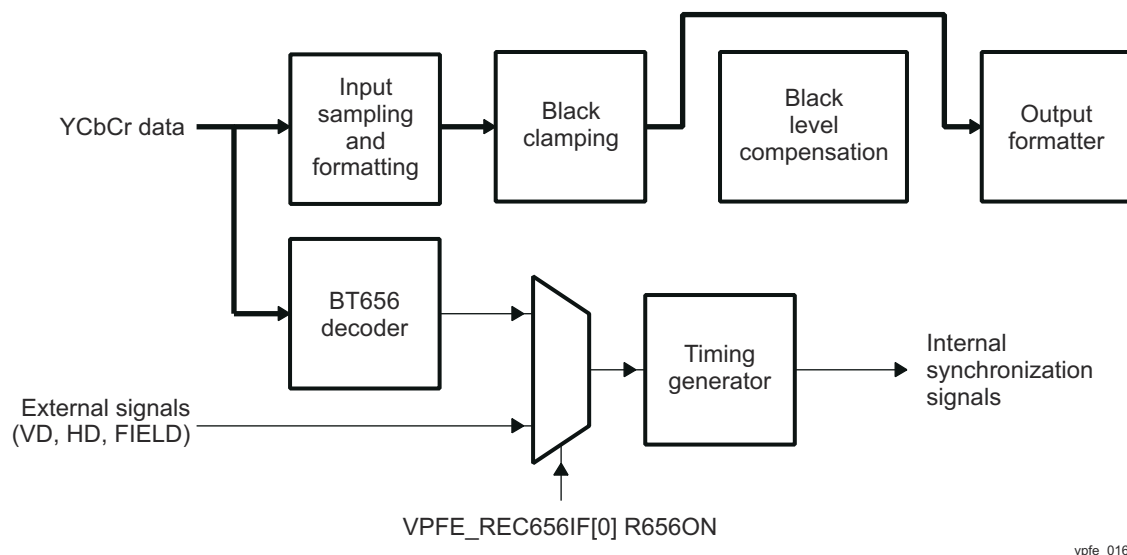
Upper Word		Lower Word	
8-bit pack	Pixel3	Pixel2	Pixel1
			Pixel0



#### 12.9.4.7.2 YCbCr and BT.656 Modes

YCbCr mode and BT.656 mode are similar in operation, as shown in [Figure 12-1195](#). The additional logic used in BT.656 mode is the CCIR-656 decoder, which extracts synchronization information from input YCbCr data and regenerates the corresponding timing for internal operation.

- YCbCr mode is enabled by setting VPFE\_SYNMODE[13-12] INPMOD bitfield to 1h or 2h and clearing VPFE\_REC656IF[0] R656ON bit.
- BT.656 mode is enabled by setting VPFE\_REC656IF[0] R656ON to 1.

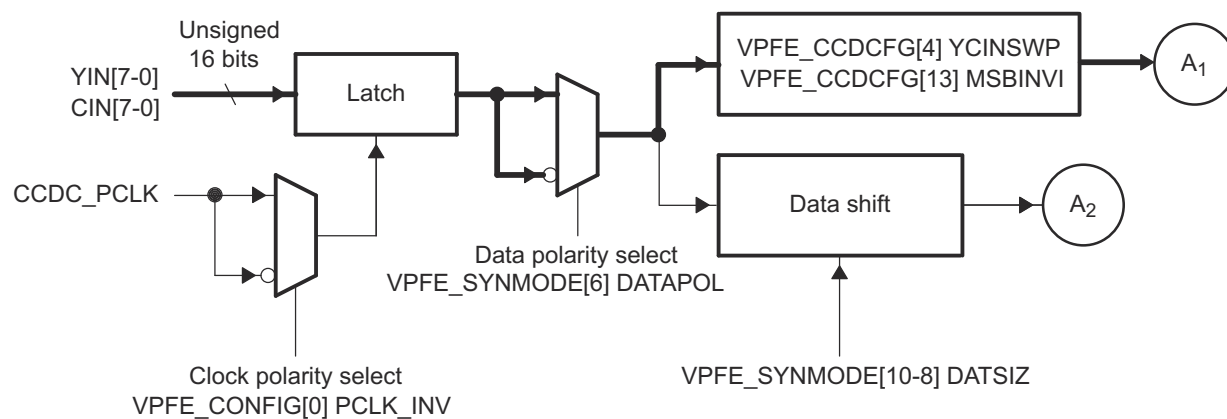


**Figure 12-1195. Data Processing for YCbCr/BT.656 Modes**

##### 12.9.4.7.2.1 Input Sampling and Formatting for YCbCr and BT.656 Modes

The data path of YCbCr/BT.656 modes is shown in the thicker lines in [Figure 12-1196](#) (that is A1). Data path A2 is applicable to raw data mode only.

- The pixel clock (CCDC\_PCLK) latches data.
- Pixel clock polarity can be either rising or falling edge and is set in VPFE clock control register (VPFE\_CONFIG[0] PCLK\_INV bit).
- The VPFE\_SYNMODE[6] DATAPOL bit affects the data representation.
- The VPFE\_CCDCFG[4] YCINSWP bit can be used to swap the upper and lower portions of the 16-bit YCbCr data bus.
  - In 16-bit YCbCr mode, this swap bit determines which part of the bus luma and chroma samples occupy respectively.
  - In 8-bit mode, this swap bit determines which part of the bus is the effective 8-bit input.
- The VPFE\_CCDCFG[13] MSBINVI bit can be used to invert the MSB of the chroma signal.



vpfe\_017

**Figure 12-1196. Input Sampling Block Diagram for YCbCr/BT.656 Modes**

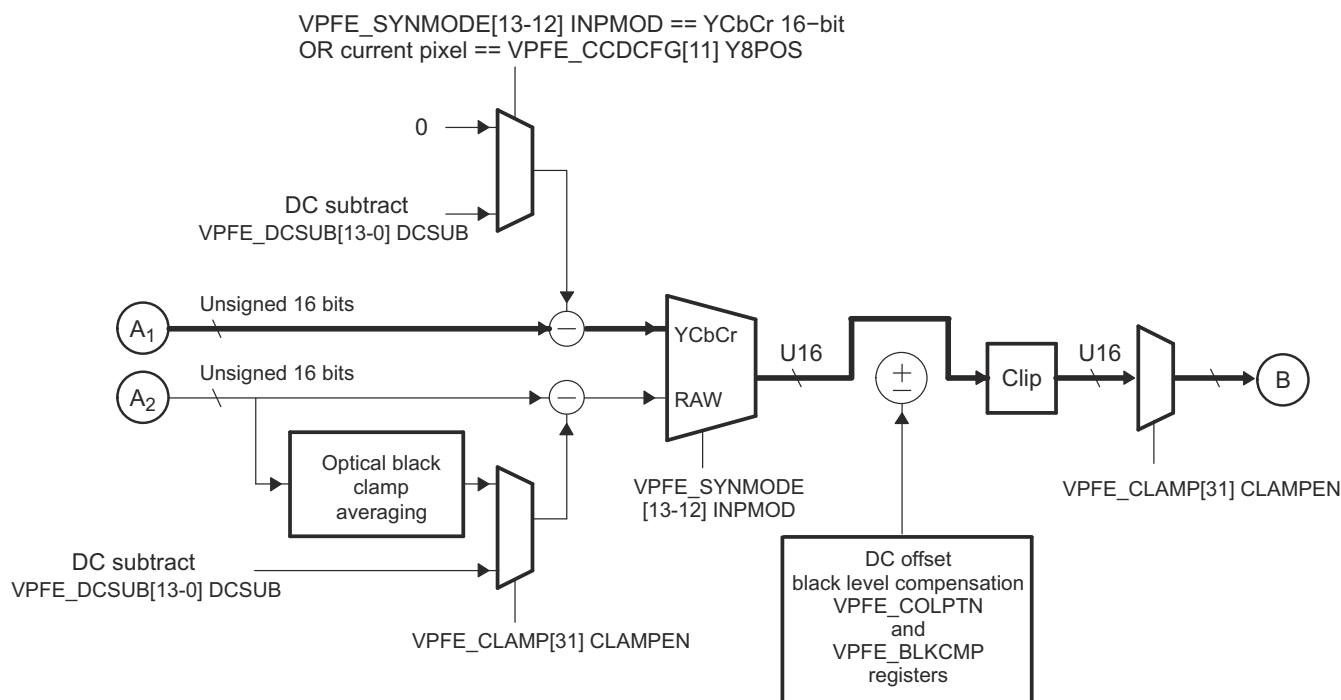
#### 12.9.4.7.2.2 Black Clamping for YCbCr and BT.656 Modes

The second step in BT.656/YCbCr processing is black clamping.

The VPFE\_DCSUB[13-0] DCSUB bitfield is used to subtract a fixed value from the luma sample for YUV data. Set the subtraction value to 0 to disable the operation. For more information, see [Figure 12-1197](#).

#### Note

Black level compensation (VPFE\_BLKCOMP register) is not used in YCbCr and BT.656 modes.

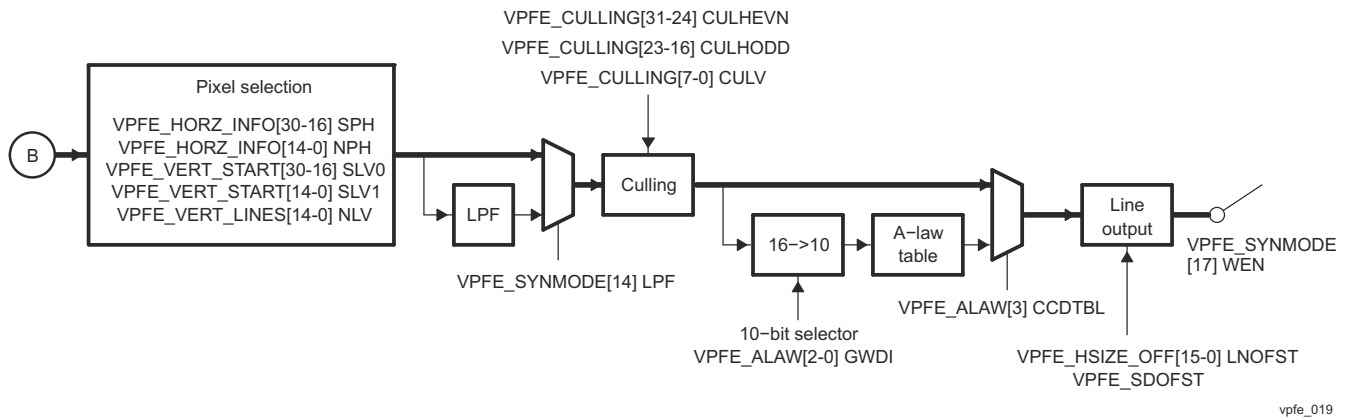


vpfe\_018

**Figure 12-1197. Black Clamping for YCbCr/BT.656 Modes**

#### 12.9.4.7.2.3 Output Formatter for YCbCr and BT.656 Modes

The final stage of VPFE processing is the output formatter, for YCbCr and BT.656 modes it is shown on [Figure 12-1198](#). A framing selection is applied to limit the processing area by the settings in the VPFE\_HORZ\_INFO, VPFE\_VERT\_START and VPFE\_VERT\_LINES registers.



**Figure 12-1198. Output Formatter for YCbCr/BT.656 Modes**

- Set the VPFE\_SYNMODE[14] LFP bit to 0 to disable the Low Pass Filter.
- Culling can be used in YCbCr and BT.656 modes, however user must take care to preserve the 422 output format.
- Set the VPFE\_ALAW[3] CCDTBL bit to 0 as A-Law transformation is not used in YCbCr or BT.656 modes.

#### 12.9.4.7.2.3.1 Output Format in External Memory for YCbCr and BT.656 Modes

The pixel data in external memory is shown in [Table 12-1567](#).

**Table 12-1567. Storage Format in External Memory for YCbCr/BT.656 Modes**

External memory	Upper word		Lower word	
Address	MSB (31)	LSB (16)	MSB (15)	LSB (0)
N	Y1	Cr0	Y0	Cb0
N + 1	Y3	Cr2	Y2	Cb2
N + 2	Y5	Cr4	Y4	Cb4

## 12.10 Timer Modules

This section describes the timer modules in the device.

### 12.10.1 Global Timebase Counter (GTC)

This section describes the Global Timebase Counter (GTC) in the device.

#### 12.10.1.1 GTC Overview

The GTC module provides a continuous running counter that can be used for time synchronization and debug trace time stamping.

The device includes one GTC instance named GTC0. [Table 12-1568](#) shows GTC module allocation within device domains.

**Table 12-1568. GTC Module Allocation within Device Domains**

Module Instance	Domain		
	WKUP	MCU	MAIN
GTC0	–	–	✓

#### 12.10.1.1.1 GTC Features

The GTC supports the following features:

- 64-bit up counter
- No rollover during the lifetime of the device
- Compatible with ARMv8 system counter requirements:
  - Disabled at power-up
  - Register definition and memory map aligned to ARMv8 definition
  - Implements memory-mapped counter control and status frames
- Outputs reflected binary (Gray) encoded timer value for system timer bus distribution to other modules
- Selectable counter bit output as a push event that can be used by CPTS modules, timers or interface protocols

#### 12.10.1.1.2 GTC Not Supported Features

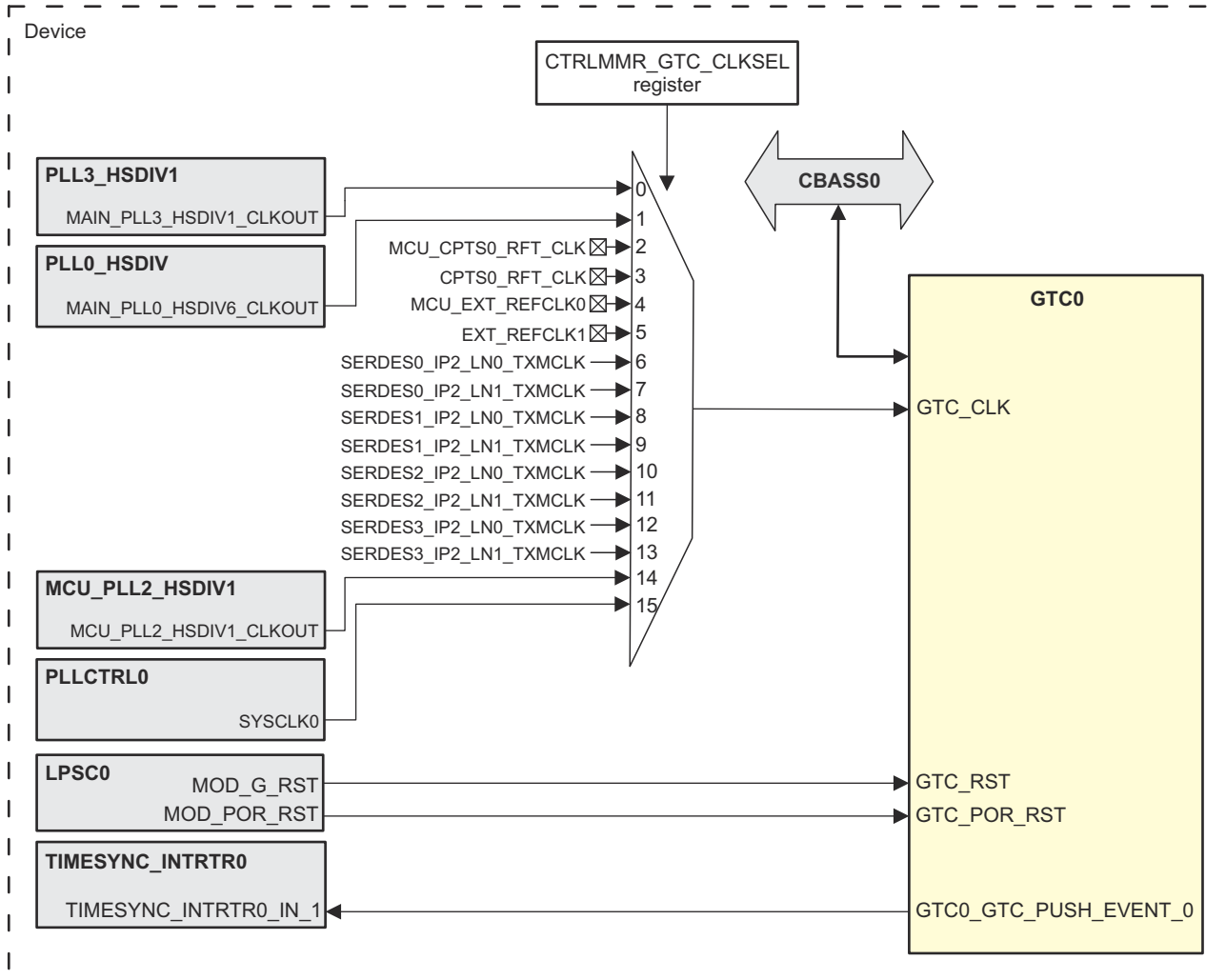
The GTC module does *not* support:

- System counter frequency change for reduced power operation
- ARMv8 memory-mapped timer
- Register write locks (MMR kick protection)

### 12.10.1.2 GTC Integration

This section describes the GTC integration in the device, including information about clocks, resets, and hardware requests.

Figure 12-1199 shows the GTC integration.



**Figure 12-1199. GTC Integration**

Table 12-1569 through Table 12-1571 summarize the GTC integration.

**Table 12-1569. GTC Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
GTC0	PSC0	PD0	LPSC0	CBASS0

**Table 12-1570. GTC Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description

**Table 12-1570. GTC Clocks and Resets (continued)**

GTC0	GTC_CLK	MAIN_PLL3_HSDIV1_CLK OUT (default)	PLL3_HSDIV1	GTC0 functional and interface clock. Source clock selection is done through a mux which is controlled via CTRLMMR_GTC_CLKSEL[2-0] CLK_SEL register bit field (see <a href="#">Figure 12-1199</a> ).
		MAIN_PLL0_HSDIV6_CLK OUT	PLL0_HSDIV6	
		MCU_CPTS_RFT_CLK	I/O pin	
		CPTS_RFT_CLK	I/O pin	
		MCU_EXT_REFCLK0	I/O pin	
		EXT_REFCLK1	I/O pin	
		SERDES0_IP2_LN0_TXMC LK	SERDES0, lane 0	
		SERDES0_IP2_LN1_TXMC LK	SERDES0, lane 1	
		SERDES1_IP2_LN0_TXMC LK	SERDES1, lane 0	
		SERDES1_IP2_LN1_TXMC LK	SERDES1, lane 1	
		SERDES2_IP2_LN0_TXMC LK	SERDES2, lane 0	
		SERDES2_IP2_LN1_TXMC LK	SERDES2, lane 1	
		SERDES3_IP2_LN0_TXMC LK	SERDES3, lane 0	
		SERDES3_IP2_LN1_TXMC LK	SERDES3, lane 1	
		MCU_PLL2_HSDIV1_CLK OUT	MCU_PLL2_HSDI V1	
		SYSCLK0	PLLCTRL0	

Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
GTC0	GTC_POR_RST	MOD_POR_RST	LPSC0	GTC0 power-on-reset (affects GTC0 counter logic)
	GTC_RST	MOD_G_RST	LPSC0	GTC0 MMR reset (affects GTC0 registers)

**Table 12-1571. GTC Hardware Requests**

Time Sync Events					
Module Instance	Module Sync Output	Destination Sync Input	Destination	Description	Type
GTC0	GTC0_GTC_PUSH_EVE NT_0	TIMESYNC_INTRTR0_IN_ 1	TIMESYNC_INTRT R0	GTC hardware push event	Pulse

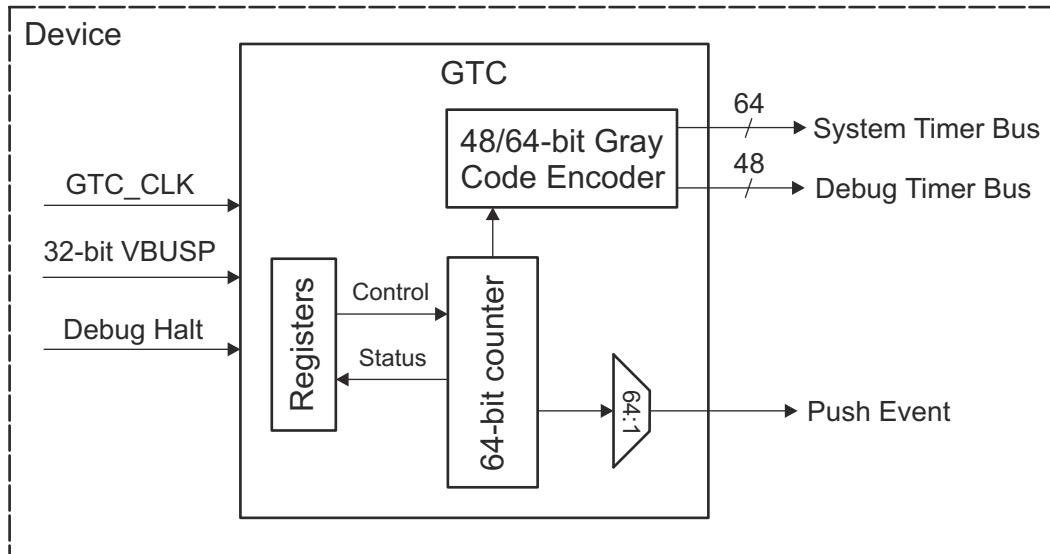
### Note

GTC hardware push event is further described in [Section 12.10.1.3.4, GTC Push Event Generation](#).

### 12.10.1.3 GTC Functional Description

#### 12.10.1.3.1 GTC Block Diagram

The GTC is essentially comprised of a 64-bit up counter, a Gray encoder, a 64-bit multiplexer, and memory-mapped control registers. [Figure 12-1200](#) shows a high-level block diagram of the GTC.



**Figure 12-1200. GTC Block Diagram**

#### 12.10.1.3.2 GTC Counter

The counter is a 64-bit binary up counter that increments on every GTC\_CLK rising edge. The source for GTC\_CLK is selected through a mux which is controlled via the CTRLMMR\_GTC\_CLKSEL[2-0] CLK\_SEL register bit field.

The counter is disabled by default (at power-on-reset) and increments only when enabled by setting the GTC\_CNTCR[0] EN bit to '1'. The current counter value is readable via the combined COUNTVALUE bit field of both GTC\_CNTCV\_HI (upper 32 bits) and GTC\_CNTCV\_LO (lower 32 bits) registers.

When counting is disabled, a new counter value can be loaded via a software write to the combined COUNTVALUE bit field. In this case, when the counter gets re-enabled, it will start counting from the last value written to this field.

The counter can be optionally configured to stop incrementing when a debug halt signal is issued, by writing a '1' to the GTC\_CNTCR[1] HDBG bit. This condition is indicated through the GTC\_CNTSR[1] DBGH status bit.

#### 12.10.1.3.3 GTC Gray Encoder

The Gray encoder performs a realtime conversion of the binary counter value into a 64-bit Gray code. This value is exported as the system timer bus for distributing the counter value to:

- A72SS global timestamp inputs
- GPU timestamp input

A second realtime conversion of the 48 LSBs of the binary counter value into a 48-bit Gray code value is also performed and exported to:

- A72SS debug timestamp inputs
- DMPAC timestamp input
- VPAC timestamp input

#### 12.10.1.3.4 GTC Push Event Generation

The GTC can generate periodic push events for global time synchronization at up to half the GTC\_CLK frequency. The frequency of the events is determined by selecting one of the counter bits through the internal [64:1] mux, which is controlled through the GTC\_PUSHEVT[5-0] EXPBIT\_SEL bit field.

#### 12.10.1.3.5 GTC Register Partitioning

The ARMv8 architecture spec requires specific system-level components, each with one or two register frames. To accommodate this, the GTC memory-mapped registers (MMRs) are divided into four separate regions (4KB each):

- Peripheral MMRs (GTC0\_GTC\_CFG0): This region includes standard peripheral identification registers and any other control registers not associated with the ARMv8 system timer functions.
- Counter Control MMRs (GTC0\_GTC\_CFG1): This region includes registers that provide control over the operation of the system timer.
- Counter Status MMRs (GTC0\_GTC\_CFG2): This region includes registers that provide a mechanism for reading the status of the system timer.
- Timer Control MMRs (GTC0\_GTC\_CFG3): This region includes registers that identify and provide a control mechanism for memory-mapped timer implementations. For this device, no memory-mapped timers are implemented and only the minimum registers required to indicate the presence of no timers are necessary.



### 12.10.2 Windowed Watchdog Timer (WWDT)

This section describes the Windowed Watchdog Timer (WWDT), implemented by using the Digital Windowed Watchdog (DWWD) function of the Real Time Interrupt (RTI) module in the device.

#### CAUTION

The RTI module shall be used to serve only as a Digital Windowed Watchdog (DWWD).

The Real Time Interrupt module provides timer functionality for operating systems and for benchmarking code. The module incorporates several counters, which define the timebases needed for scheduling in the operating system.

This module is specifically designed to fulfill the requirements for OSEK (“Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug”; “Open Systems and the Corresponding Interfaces for Automotive Electronics”) as well as OSEK/Time compliant operating systems.

The timers also provide the ability to benchmark certain areas of code by reading the counter contents at the beginning and the end of the desired code range and calculating the difference between the values.

#### 12.10.2.1 RTI Overview

There are twelve RTI modules in the device. [Table 12-1572](#) shows the RTI allocation within device domains.

**Table 12-1572. RTI Modules Allocation within Device Domains**

Module Instance	Domain		
	WKUP	MCU	MAIN
MCU_RTI0	-	✓	-
MCU_RTI1	-	✓	-
RTI0	-	-	✓
RTI1	-	-	✓
RTI15	-	-	✓
RTI16	-	-	✓
RTI24	-	-	✓
RTI25	-	-	✓
RTI28	-	-	✓
RTI29	-	-	✓
RTI30	-	-	✓
RTI31	-	-	✓

Instances in the MCU domain:

- MCU\_RTI0 is dedicated to the MCU cluster (MCU\_R5FSS0) in lockstep and when unlocked serves as a Windowed Watchdog for the first R5F CPU core in the MCU domain (MCU\_R5FSS0\_CORE0).
- MCU\_RTI1 is dedicated to the second R5F CPU core of the MCU cluster (MCU\_R5FSS0\_CORE1) when unlocked.

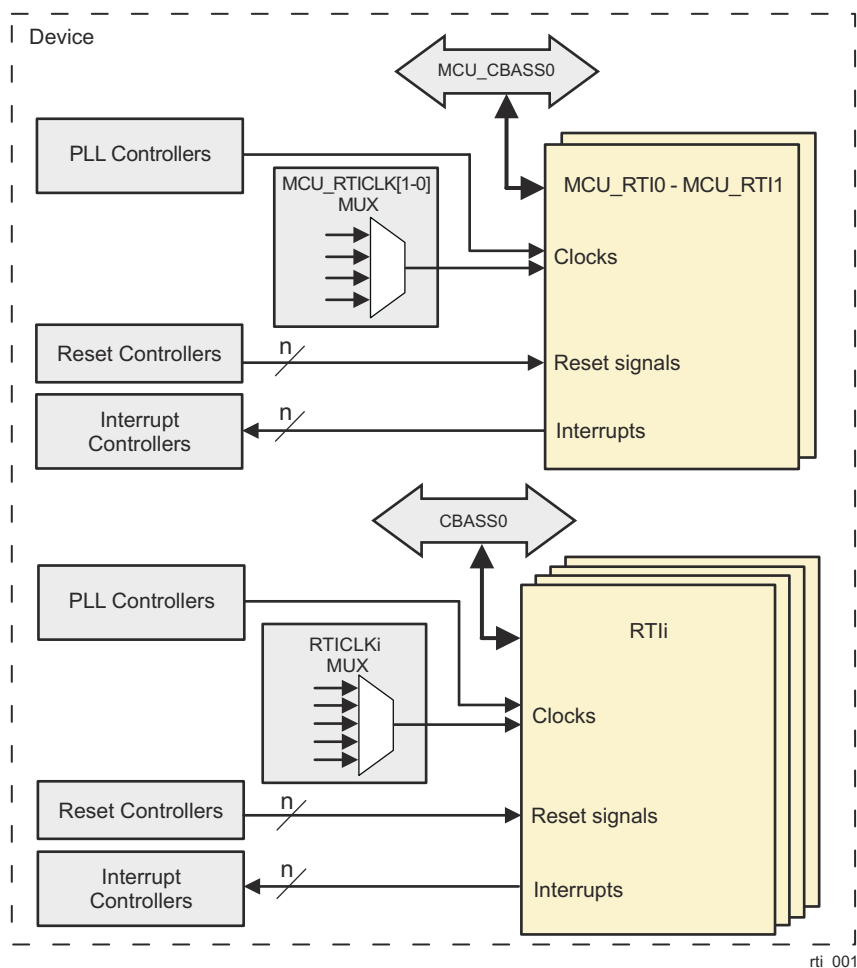
Instances in the MAIN domain: They are intended to function as a Digital Windowed Watchdog for the CPU core that they are associated with, that is:

- RTI0 is dedicated to the first A72 CPU core in the A72 cluster (A72SS0\_CORE0)
- RTI1 is dedicated to the second A72 CPU core in the A72 cluster (A72SS0\_CORE1)
- RTI15 is dedicated to the GPU
- RTI16 is dedicated to the C7x DSP
- RTI24 is dedicated to the first C66x DSP core (C66SS0\_CORE0)
- RTI25 is dedicated to the second C66x DSP core (C66SS1\_CORE0)

- RTI28 is dedicated to the first R5F CPU core in the Main domain (R5FSS0\_CORE0)
- RTI29 is dedicated to the second R5F CPU core in the Main domain (R5FSS0\_CORE1)
- RTI30 is dedicated to the third R5F CPU core in the Main domain (R5FSS1\_CORE0)
- RTI31 is dedicated to the fourth R5F CPU core in the Main domain (R5FSS1\_CORE1)

It is not intended to use an RTI that is provisioned for a particular CPU core with a different CPU core.

Figure 12-1201 shows the RTI overview.



i = 0, 1, 15, 16, 24, 25, 28, 29, 30, 31

**Figure 12-1201. RTI Overview**

#### 12.10.2.1.1 RTI Features

The RTI modules include the following main features:

- Windowed Watchdog Timer (WWDT) feature.
- Two independent 64 bit counter blocks (counter block0 or counter block1). Each block consists of
  - One 32 bit up counter
  - One 32 bit free running counter
  - Two capture registers for capturing the prescale and free running counter on a special event.
- Free running counter 0 can be incremented by either the internal prescale counter or by an external event.
- Four configurable compare registers for generating operating system ticks. Each event can be driven by either counter block0 or counter block1.
- Fast enabling/disabling of events.

- RTI clock input derived from any of the available clock sources, selectable in the System Module
- Optional capability to drive a pulse-width modulated signal out on an interrupt line.

#### **12.10.2.1.2 RTI Not Supported Features**

The following features are not supported on this family of devices:

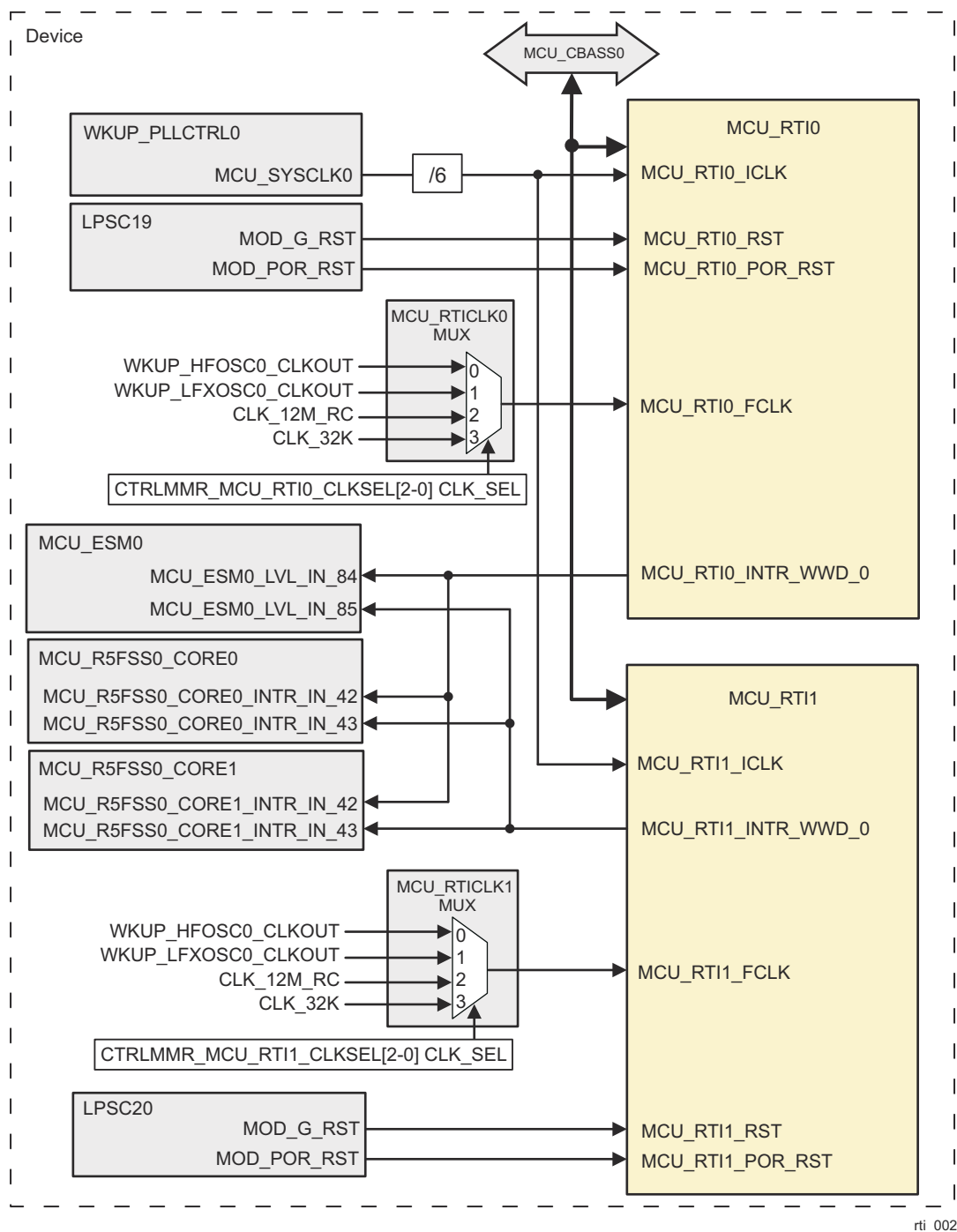
- Real time interrupt functionality
- Analog and Digital Watchdog (DWD) Timers
- Periodic interrupt / DMA events
- Capture and compare events not used
- Counter operation is not supported and counter overflow interrupts are not used.

### 12.10.2.2 RTI Integration

This section describes module integration in the device, including information about clocks, resets, and hardware requests.

### 12.10.2.2.1 RTI Integration in MCU Domain

There are two RTI modules integrated in the device MCU domain - MCU\_RTIO and MCU\_RT11. [Figure 12-1202](#) shows their integration in the device.



**Figure 12-1202. MCU RTI Integration**

[Table 12-1573](#) through [Table 12-1575](#) summarize the integration of MCU\_RTIO and MCU\_RT11 in device MCU domain.

Each MCU\_RT1 instance is supplied by dedicated MCU\_RTICLK[1-0] clock mux.

**Table 12-1573. MCU\_RTI Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
MCU_RTI0	WKUP_PSC0	PD1	LPSC19	MCU_CBASS0
MCU_RTI1	WKUP_PSC0	PD1	LPSC20	MCU_CBASS0

**Table 12-1574. MCU\_RTI Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
MCU_RTI0	MCU_RTI0_ICLK	MCU_SYSCCLK0/6	WKUP_PLLCTRL0	MCU_RTI0 Interface Clock
	MCU_RTI0_FCLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	MCU_RTI0 Functional Clock. For more information about clock multiplexing in MCU_RTICLK0 MUX, see CTRLMMR_MCU_RTI0_CLKSEL[2-0] CLK_SEL in <i>Control Module (CTRL_MMR)</i> .
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CLK_12M_RC	WKUP_RC_OSC_12M	
	CLK_32K			
MCU_RTI1	MCU_RTI1_ICLK	MCU_SYSCCLK0/6	WKUP_PLLCTRL0	MCU_RTI1 Interface Clock
	MCU_RTI1_FCLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	MCU_RTI1 Functional Clock. For more information about clock multiplexing in MCU_RTICLK1 MUX, see CTRLMMR_MCU_RTI1_CLKSEL[2-0] CLK_SEL in <i>Control Module (CTRL_MMR)</i> .
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CLK_12M_RC	WKUP_RC_OSC_12M	
	CLK_32K			
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
MCU_RTI0	MCU_RTI0_RST	MOD_G_RST	LPSC19	MCU_RTI0 Asynchronous Reset
	MCU_RTI0_POR_RST	MOD_POR_RST	LPSC19	MCU_RTI0 Power-On Reset
MCU_RTI1	MCU_RTI1_RST	MOD_G_RST	LPSC20	MCU_RTI1 Asynchronous Reset
	MCU_RTI1_POR_RST	MOD_POR_RST	LPSC20	MCU_RTI1 Power-On Reset

**Table 12-1575. MCU\_RTI Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCU_RTI0	MCU_RTI0_INTR_WD_0	MCU_R5FSS0_CORE0_INT_R_IN_42	MCU_R5FSS0_CO_RE0	MCU_RTI0 window watchdog violation interrupt	Level
		MCU_R5FSS0_CORE1_INT_R_IN_42	MCU_R5FSS0_CO_RE1	MCU_RTI0 window watchdog violation interrupt	Level
		MCU_ESM0_LVL_IN_84	MCU_ESM0	MCU_RTI0 window watchdog violation interrupt	Level
MCU_RTI1	MCU_RTI1_INTR_WD_0	MCU_R5FSS0_CORE0_INT_R_IN_43	MCU_R5FSS0_CO_RE0	MCU_RTI1 window watchdog violation interrupt	Level
		MCU_R5FSS0_CORE1_INT_R_IN_43	MCU_R5FSS0_CO_RE1	MCU_RTI1 window watchdog violation interrupt	Level
		MCU_ESM0_LVL_IN_85	MCU_ESM0	MCU_RTI1 window watchdog violation interrupt	Level

#### 12.10.2.2.2 RTI Integration in MAIN Domain

There are ten RTI modules integrated in the device MAIN domain. [Figure 12-1203](#) shows their integration in the device.

Device

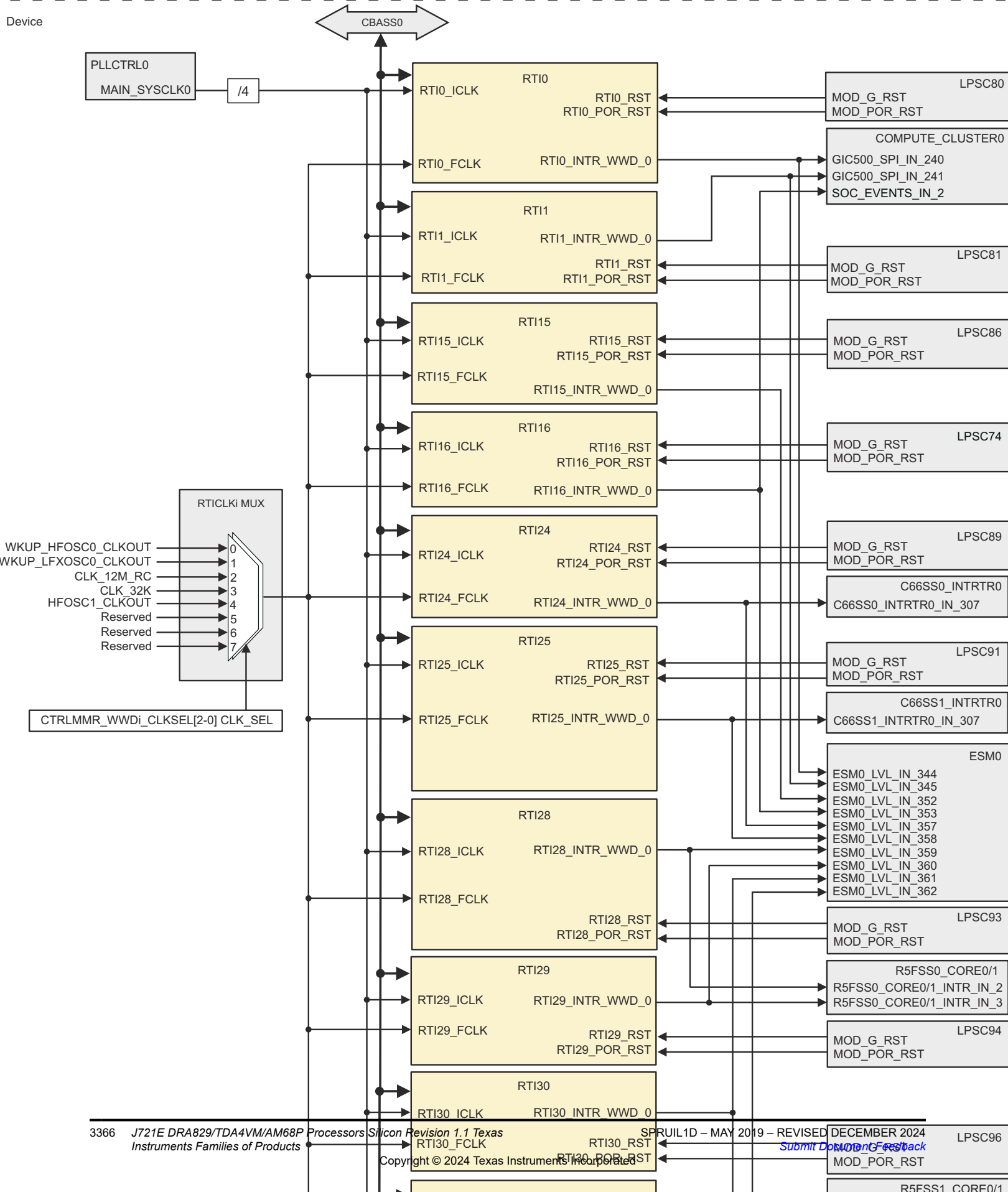




Table 12-1576 through Table 12-1578 summarize the integration of RTIi (where i = 0, 1, 15, 16, 24, 25, 28, 29, 30, 31) in device MAIN domain.

Each RTI instance is supplied by dedicated RTICLK<sub>i</sub> mux.

**Table 12-1576. RTI Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
RTI0	PSC0	PD15	LPSC80	CBASS0
RTI1	PSC0	PD16	LPSC81	CBASS0
RTI15	PSC0	PD20	LPSC86	CBASS0
RTI16	PSC0	PD12	LPSC74	CBASS0
RTI24	PSC0	PD22	LPSC89	CBASS0
RTI25	PSC0	PD23	LPSC91	CBASS0
RTI28	PSC0	PD24	LPSC93	CBASS0
RTI29	PSC0	PD24	LPSC94	CBASS0
RTI30	PSC0	PD25	LPSC96	CBASS0
RTI31	PSC0	PD25	LPSC97	CBASS0

**Table 12-1577. RTI Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
RTI0	RTI0_ICLK	MAIN_SYSCLK0/4	PLLCTRL0	RTI0 Interface Clock
	RTI0_FCLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	RTI0 Functional Clock. For more information about clock multiplexing in RTICLK0 MUX, see CTRLMMR_WWD0_CLKSEL[2-0] CLK_SEL in <i>Control Module (CTRL_MMR)</i> .
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CLK_12M_RC	WKUP_RC_OSC_12M	
		CLK_32K		
RTI1	RTI1_FCLK	HFOSC1_CLKOUT	HFOSC1	RTI1 Functional Clock. For more information about clock multiplexing in RTICLK1 MUX, see CTRLMMR_WWD1_CLKSEL[2-0] CLK_SEL in <i>Control Module (CTRL_MMR)</i> .
		MAIN_SYSCLK0/4	PLLCTRL0	
		WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CLK_12M_RC	WKUP_RC_OSC_12M	
RTI15	RTI15_FCLK	CLK_32K		RTI15 Functional Clock. For more information about clock multiplexing in RTICLK15 MUX, see CTRLMMR_WWD15_CLKSEL[2-0] CLK_SEL in <i>Control Module (CTRL_MMR)</i> .
		HFOSC1_CLKOUT	HFOSC1	
		MAIN_SYSCLK0/4	PLLCTRL0	
		WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
RTI16	RTI16_FCLK	CLK_12M_RC	WKUP_RC_OSC_12M	RTI16 Functional Clock. For more information about clock multiplexing in RTICLK16 MUX, see CTRLMMR_WWD16_CLKSEL[2-0] CLK_SEL in <i>Control Module (CTRL_MMR)</i> .
		CLK_32K		
		HFOSC1_CLKOUT	HFOSC1	
		MAIN_SYSCLK0/4	PLLCTRL0	
		WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	
RTI24	RTI24_FCLK	WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	RTI24 Functional Clock. For more information about clock multiplexing in RTICLK24 MUX, see CTRLMMR_WWD24_CLKSEL[2-0] CLK_SEL in <i>Control Module (CTRL_MMR)</i> .
		CLK_12M_RC	WKUP_RC_OSC_12M	
		CLK_32K		
		HFOSC1_CLKOUT	HFOSC1	
		MAIN_SYSCLK0/4	PLLCTRL0	
RTI25	RTI25_FCLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	RTI25 Functional Clock. For more information about clock multiplexing in RTICLK25 MUX, see CTRLMMR_WWD25_CLKSEL[2-0] CLK_SEL in <i>Control Module (CTRL_MMR)</i> .
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CLK_12M_RC	WKUP_RC_OSC_12M	
		CLK_32K		
		MAIN_SYSCLK0/4	PLLCTRL0	
RTI28	RTI28_FCLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	RTI28 Functional Clock. For more information about clock multiplexing in RTICLK28 MUX, see CTRLMMR_WWD28_CLKSEL[2-0] CLK_SEL in <i>Control Module (CTRL_MMR)</i> .
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CLK_12M_RC	WKUP_RC_OSC_12M	
		CLK_32K		
		MAIN_SYSCLK0/4	PLLCTRL0	
RTI29	RTI29_FCLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	RTI29 Functional Clock. For more information about clock multiplexing in RTICLK29 MUX, see CTRLMMR_WWD29_CLKSEL[2-0] CLK_SEL in <i>Control Module (CTRL_MMR)</i> .
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CLK_12M_RC	WKUP_RC_OSC_12M	
		CLK_32K		
		MAIN_SYSCLK0/4	PLLCTRL0	
RTI30	RTI30_FCLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	RTI30 Functional Clock. For more information about clock multiplexing in RTICLK30 MUX, see CTRLMMR_WWD30_CLKSEL[2-0] CLK_SEL in <i>Control Module (CTRL_MMR)</i> .
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CLK_12M_RC	WKUP_RC_OSC_12M	
		CLK_32K		
		MAIN_SYSCLK0/4	PLLCTRL0	
RTI31	RTI31_FCLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	RTI31 Functional Clock. For more information about clock multiplexing in RTICLK31 MUX, see CTRLMMR_WWD31_CLKSEL[2-0] CLK_SEL in <i>Control Module (CTRL_MMR)</i> .
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CLK_12M_RC	WKUP_RC_OSC_12M	
		CLK_32K		
		MAIN_SYSCLK0/4	PLLCTRL0	

**Table 12-1577. RTI Clocks and Resets (continued)**

RTI16	FCLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	RTI16 Functional Clock. For more information about clock multiplexing in RTICK16 MUX, see CTRLMMR_WWD16_CLKSEL[2-0] CLK_SEL in <i>Control Module</i> (CTRL_MMR).
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CLK_12M_RC	WKUP_RC_OSC_12M	
		CLK_32K		
RTI24	FCLK	HFOSC1_CLKOUT	HFOSC1	RTI24 Interface Clock
		MAIN_SYSCLK0/4	PLLCTRL0	
		WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
RTI24	FCLK	CLK_12M_RC	WKUP_RC_OSC_12M	RTI24 Functional Clock. For more information about clock multiplexing in RTICK24 MUX, see CTRLMMR_WWD24_CLKSEL[2-0] CLK_SEL in <i>Control Module</i> (CTRL_MMR).
		CLK_32K		
		HFOSC1_CLKOUT	HFOSC1	
		MAIN_SYSCLK0/4	PLLCTRL0	
RTI25	FCLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	RTI25 Interface Clock
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CLK_12M_RC	WKUP_RC_OSC_12M	
		CLK_32K		
RTI25	FCLK	HFOSC1_CLKOUT	HFOSC1	RTI25 Functional Clock. For more information about clock multiplexing in RTICK25 MUX, see CTRLMMR_WWD25_CLKSEL[2-0] CLK_SEL in <i>Control Module</i> (CTRL_MMR).
		MAIN_SYSCLK0/4	PLLCTRL0	
		WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
RTI28	FCLK	CLK_12M_RC	WKUP_RC_OSC_12M	RTI28 Interface Clock
		CLK_32K		
		HFOSC1_CLKOUT	HFOSC1	
		MAIN_SYSCLK0/4	PLLCTRL0	
RTI28	FCLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	RTI28 Functional Clock. For more information about clock multiplexing in RTICK28 MUX, see CTRLMMR_WWD28_CLKSEL[2-0] CLK_SEL in <i>Control Module</i> (CTRL_MMR).
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CLK_12M_RC	WKUP_RC_OSC_12M	
		CLK_32K		
RTI29	FCLK	HFOSC1_CLKOUT	HFOSC1	RTI29 Interface Clock
		MAIN_SYSCLK0/4	PLLCTRL0	
		WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
RTI29	FCLK	CLK_12M_RC	WKUP_RC_OSC_12M	RTI29 Functional Clock.. For more information about clock multiplexing in RTICK29 MUX, see CTRLMMR_WWD29_CLKSEL[2-0] CLK_SEL in <i>Control Module</i> (CTRL_MMR).
		CLK_32K		
		HFOSC1_CLKOUT	HFOSC1	
		MAIN_SYSCLK0/4	PLLCTRL0	
RTI30	FCLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	RTI30 Interface Clock
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CLK_12M_RC	WKUP_RC_OSC_12M	
		CLK_32K		
RTI30	FCLK	HFOSC1_CLKOUT	HFOSC1	RTI30 Functional Clock. For more information about clock multiplexing in RTICK30 MUX, see CTRLMMR_WWD30_CLKSEL[2-0] CLK_SEL in <i>Control Module</i> (CTRL_MMR).
		MAIN_SYSCLK0/4	PLLCTRL0	
		WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
RTI31	FCLK	CLK_12M_RC	WKUP_RC_OSC_12M	RTI31 Interface Clock
		CLK_32K		
		HFOSC1_CLKOUT	HFOSC1	
		MAIN_SYSCLK0/4	PLLCTRL0	

**Table 12-1577. RTI Clocks and Resets (continued)**

RTI31_FCLK		WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	RTI31 Functional Clock. For more information about clock multiplexing in RTICLK31 MUX, see CTRLMMR_WWD31_CLKSEL[2-0] CLK_SEL in <i>Control Module (CTRL_MMR)</i> .
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CLK_12M_RC	WKUP_RC_OSC_12M	
		CLK_32K		
		HFOSC1_CLKOUT	HFOSC1	
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
RTI0	RTI0_RST	MOD_G_RST	LPSC80	RTI0 Asynchronous Reset
	RTI0_POR_RST	MOD_POR_RST	LPSC80	RTI0 Power-On Reset
RTI1	RTI1_RST	MOD_G_RST	LPSC81	RTI1 Asynchronous Reset
	RTI1_POR_RST	MOD_POR_RST	LPSC81	RTI1 Power-On Reset
RTI15	RTI15_RST	MOD_G_RST	LPSC86	RTI15 Asynchronous Reset
	RTI15_POR_RST	MOD_POR_RST	LPSC86	RTI15 Power-On Reset
RTI16	RTI16_RST	MOD_G_RST	LPSC74	RTI16 Asynchronous Reset
	RTI16_POR_RST	MOD_POR_RST	LPSC74	RTI16 Power-On Reset
RTI24	RTI24_RST	MOD_G_RST	LPSC89	RTI24 Asynchronous Reset
	RTI24_POR_RST	MOD_POR_RST	LPSC89	RTI24 Power-On Reset
RTI25	RTI25_RST	MOD_G_RST	LPSC91	RTI25 Asynchronous Reset
	RTI25_POR_RST	MOD_POR_RST	LPSC91	RTI25 Power-On Reset
RTI28	RTI28_RST	MOD_G_RST	LPSC93	RTI28 Asynchronous Reset
	RTI28_POR_RST	MOD_POR_RST	LPSC93	RTI28 Power-On Reset
RTI29	RTI29_RST	MOD_G_RST	LPSC94	RTI29 Asynchronous Reset
	RTI29_POR_RST	MOD_POR_RST	LPSC94	RTI29 Power-On Reset
RTI30	RTI30_RST	MOD_G_RST	LPSC96	RTI30 Asynchronous Reset
	RTI30_POR_RST	MOD_POR_RST	LPSC96	RTI30 Power-On Reset
RTI31	RTI31_RST	MOD_G_RST	LPSC97	RTI31 Asynchronous Reset
	RTI31_POR_RST	MOD_POR_RST	LPSC97	RTI31 Power-On Reset

**Table 12-1578. RTI Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
RTI0	RTI0_INTR_WWD_0	GIC500_SPI_IN_240	COMPUTE_CLUSTER0	RTI0 window watchdog violation interrupt	Level
		ESM0_LVL_IN_344	ESM0	RTI0 window watchdog violation interrupt	Level
RTI1	RTI1_INTR_WWD_0	GIC500_SPI_IN_241	COMPUTE_CLUSTER0	RTI1 window watchdog violation interrupt	Level
		ESM0_LVL_IN_345	ESM0	RTI1 window watchdog violation interrupt	Level
RTI15	RTI15_INTR_WWD_0	ESM0_LVL_IN_352	ESM0	RTI15 window watchdog violation interrupt	Level
RTI16	RTI16_INTR_WWD_0	SOC_EVENTS_IN_2	COMPUTE_CLUSTER0	RTI16 window watchdog violation interrupt	Level
		ESM0_LVL_IN_353	ESM0	RTI16 window watchdog violation interrupt	Level
RTI24	RTI24_INTR_WWD_0	C66SS0_INTRTR0_IN_307	C66SS0_INTRTR0	RTI24 window watchdog violation interrupt	Level
		ESM0_LVL_IN_357	ESM0	RTI24 window watchdog violation interrupt	Level
RTI25	RTI25_INTR_WWD_0	C66SS1_INTRTR0_IN_307	C66SS1_INTRTR0	RTI25 window watchdog violation interrupt	Level
		ESM0_LVL_IN_358	ESM0	RTI25 window watchdog violation interrupt	Level
RTI28	RTI28_INTR_WWD_0	R5FSS0_CORE0_INTR_IN_2	R5FSS0_CORE0	RTI28 window watchdog violation interrupt	Level
		R5FSS0_CORE1_INTR_IN_2	R5FSS0_CORE1	RTI28 window watchdog violation interrupt	Level
		ESM0_LVL_IN_359	ESM0	RTI28 window watchdog violation interrupt	Level
RTI29	RTI29_INTR_WWD_0	R5FSS0_CORE0_INTR_IN_3	R5FSS0_CORE0	RTI29 window watchdog violation interrupt	Level
		R5FSS0_CORE1_INTR_IN_3	R5FSS0_CORE1	RTI29 window watchdog violation interrupt	Level
		ESM0_LVL_IN_360	ESM0	RTI29 window watchdog violation interrupt	Level
RTI30	RTI30_INTR_WWD_0	R5FSS1_CORE0_INTR_IN_2	R5FSS1_CORE0	RTI30 window watchdog violation interrupt	Level
		R5FSS1_CORE1_INTR_IN_2	R5FSS1_CORE1	RTI30 window watchdog violation interrupt	Level
		ESM0_LVL_IN_361	ESM0	RTI30 window watchdog violation interrupt	Level
RTI31	RTI31_INTR_WWD_0	R5FSS1_CORE0_INTR_IN_3	R5FSS1_CORE0	RTI31 window watchdog violation interrupt	Level
		R5FSS1_CORE1_INTR_IN_3	R5FSS1_CORE1	RTI31 window watchdog violation interrupt	Level
		ESM0_LVL_IN_362	ESM0	RTI31 window watchdog violation interrupt	Level

### 12.10.2.3 RTI Functional Description

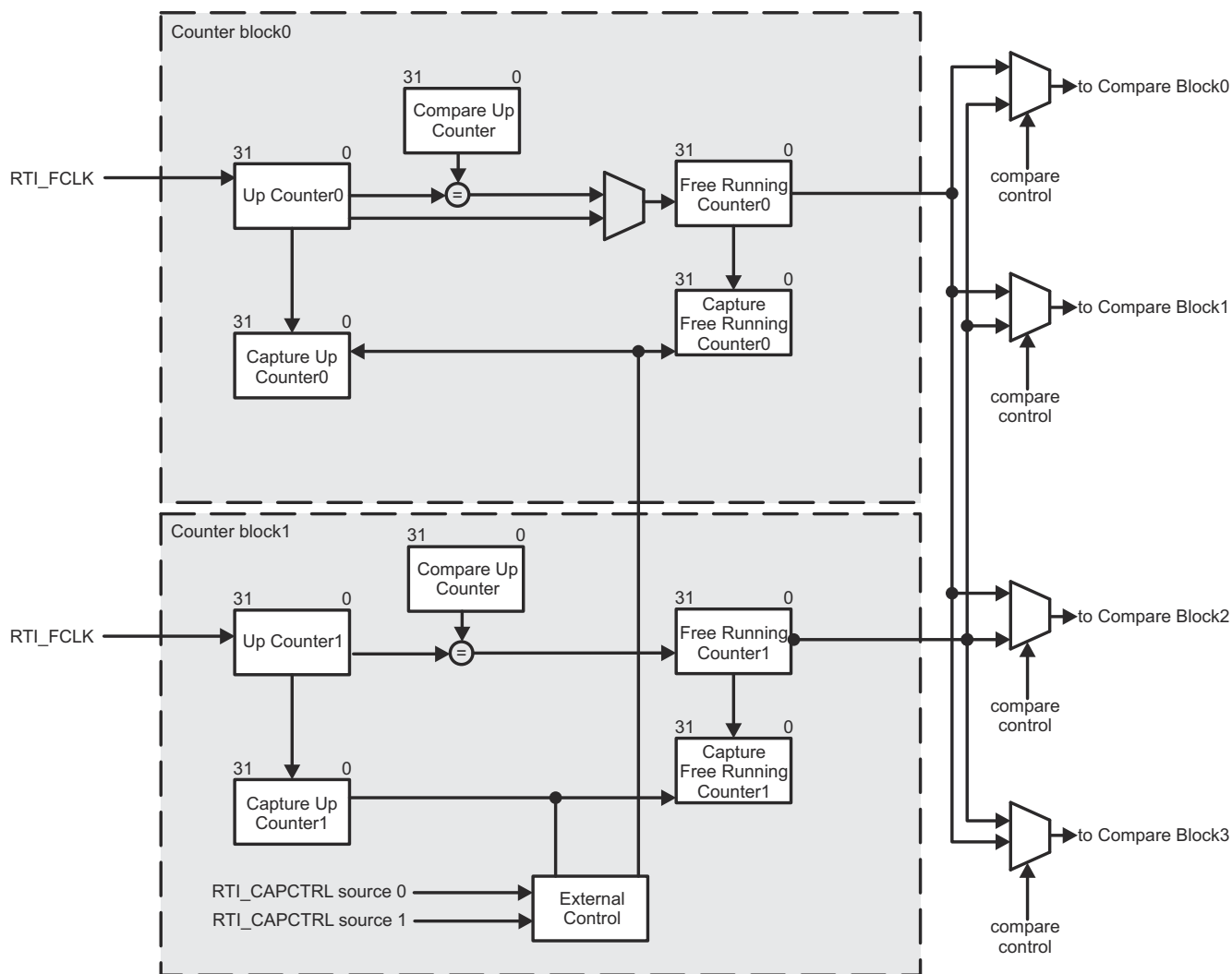
The MCU\_RTI[1-0] and RTIi (where i = 0, 1, 15, 16, 24, 25, 28, 29, 30, 31) modules are hereinafter referred to as RTI module.

#### 12.10.2.3.1 RTI Counter Operation

##### Note

Some of the RTI features described in this section may not be supported on this family of devices. For more information, see *RTI Not Supported Features*.

Figure 12-1204 shows the RTI module counter blocks. The RTI module supports two counter blocks.



rti-004

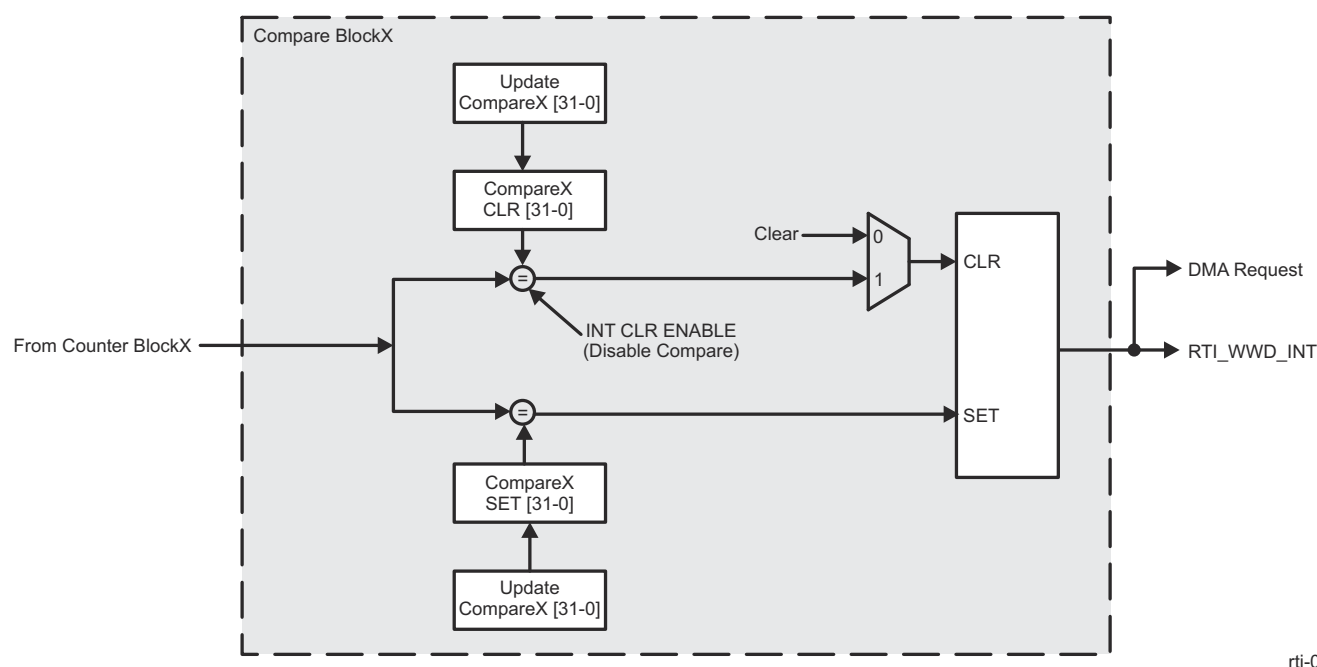
**Figure 12-1204. RTI Counters Block Diagram**

Each block consists of two 32-bit up counters: Up Counter (UC), and Free Running Counter (FRC). The Up Counter (RTI\_UC0 or RTI\_UC1 register) is driven by the RTI\_FCLK, and counts up until the compare value in the Compare Up Counter register (RTI\_CPUC0 or RTI\_CPUC1) is reached. When the compare matches, the second counter (RTI\_FRC0 or RTI\_FRC1 register), which is a free running counter, is incremented. At the same time UCx is reset to zero.

To ensure the consistency of the counters, when both counter values have to be determined, read the Free Running Counter first. This makes sure that at the time when the counter register is read, the Up Counter value has been stored into the counter register. The second read is then performed on the Up Counter register, which holds then the value of the counter cycle of the previous read on the Free Running Counter register.

Both blocks provide also a capture feature on external events. Two capture sources can trigger the capture event. Which event triggers block 0 or block 1 is configurable from the RTI\_CAPCTRL register. The event sources come from the interrupt manager, enabling the device to generate a capture event when a peripheral module generates an interrupt. The peripheral which generates an RTI capture event is configured in the interrupt manager. When the event is detected, UCx and FRCx are stored in Capture Up Counter (RTI\_CAUC0 or RTI\_CAUC1) and Capture Free Running Counter (RTI\_CAFRC0 or RTI\_CAFRC1) registers. The read order of the captured values must be in the same order as the counter register reads. So, the CAFRCx must be read first, and then the CAUCx registers are read after the CAFRCx value has been determined. While CAFRCx is read, the CAUCx value is loaded into a shadow register to maintain data consistency, in case a capture event happens during the two reads. If the application fails to read the two registers before a second capture event happens, the previous data is overwritten.

Figure 12-1205 shows the block diagram for one compare block. The RTI module supports four compare blocks.



rti-005

**Figure 12-1205. RTI Compare Block Diagram**

In order to generate interrupt requests to the interrupt manager, there are four compare registers (RTI\_COMP0, RTI\_COMP1, RTI\_COMP2, and RTI\_COMP3). Each of the compare registers can be configured to work either on FRC0 (Counter block0) or FRC1 (Counter block1). When the counter value matches the compare value, an interrupt is generated. This sets an interrupt request line to the interrupt manager. The compare value gets updated automatically with the value stored in Update Compare (RTI\_UDCP0, RTI\_UDCP1, RTI\_UDCP2, and RTI\_UDCP3) registers when the compare matches. This gives the ability to generate periodic interrupts/DMA requests without having to update the compare value by software.

An optional feature allows an application to program another compare value which is then used to clear the interrupt request line. This feature is supported by four compare clear registers (RTI\_COMP0CLR, RTI\_COMP1CLR, RTI\_COMP2CLR, and RTI\_COMP3CLR). When the counter value matches the compare clear value, the interrupt line is cleared. This clears the interrupt request line to the interrupt manager. The

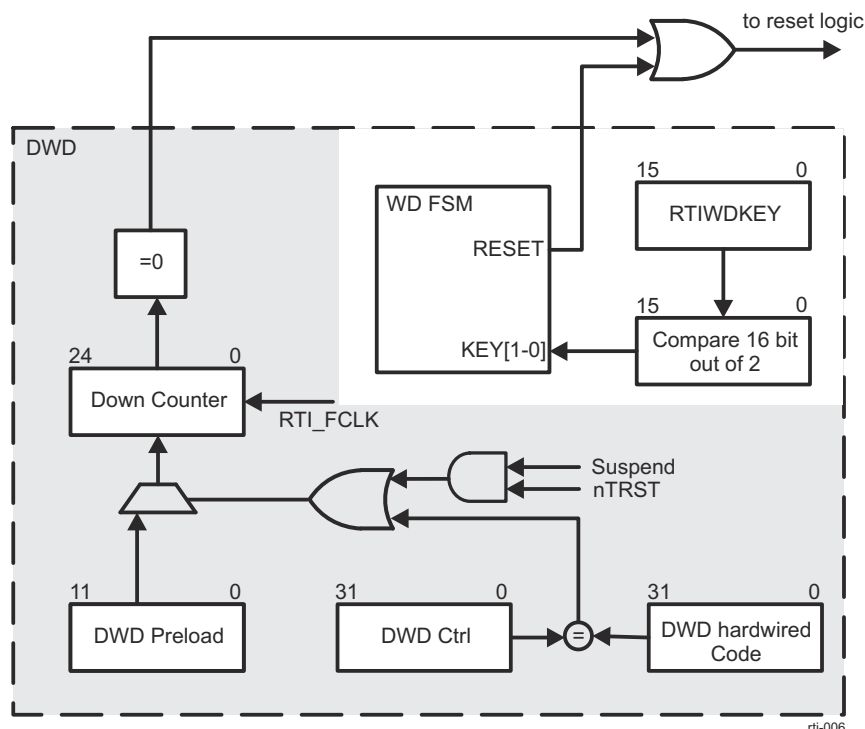
compare clear value gets updated automatically with the value stored in Update Compare (RTI\_UDCPx) registers when the compare matches.

### 12.10.2.3.2 RTI Digital Watchdog

#### Note

Some of the RTI features described in this section may not be supported on this family of devices. For more information, see *RTI Not Supported Features*.

Some applications might use a digital watchdog (DWD) integrated in the RTI module. The digital watchdog generates resets after a programmable period, if no correct key sequence is written to the RTI\_WDKEY register. Figure 12-1206 shows the digital watchdog functional block.



**Figure 12-1206. RTI Digital Watchdog Functional Block Diagram**

The digital watchdog functionality is implemented such that it can be enabled by software.

The DWD starts counting down from the reset value of the RTI\_DWDCNTR (DWD Counter Register). The DWD preload register can be configured at any time by the application according to the desired time-out period.

When enabled by software, the digital watchdog is disabled after system reset. If it should be used, it has to be enabled by writing A98559DAh to the RTI\_DWDCTRL register. The DWD timeout period must be configured using the DWD preload register before the DWD is enabled. The DWD cannot be disabled by the application once it is enabled.

#### Note

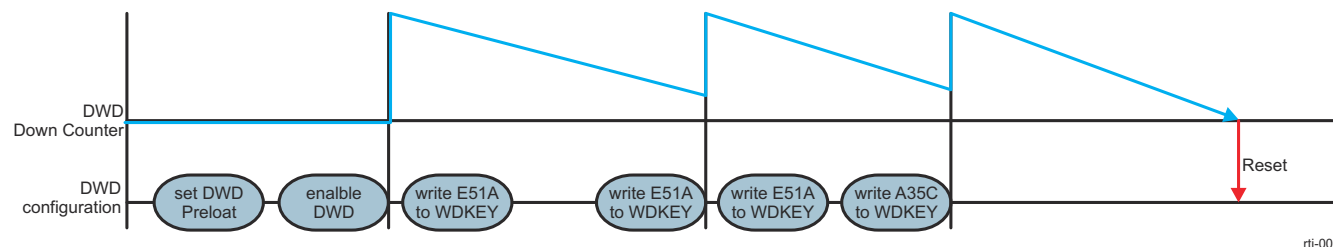
When the DWD is enabled by software, any system reset will disable the DWD. This reset could have been generated by the watchdog itself.

If the correct key sequence is written to the RTI\_WDKEY register (E51Ah followed by A35Ch), the 25-bit DWD Down Counter is reloaded with the 12-bit preload value stored in RTI\_DWDPRLD register. If any incorrect value

is written to the RTI\_WDKEY register, a watchdog reset will occur immediately. A reset will also be generated, when the DWD Down Counter is decremented to 0.

The user has to take into account that the write to the RTI\_WDKEY register takes 3 RTI\_ICLK cycles. This needs to be considered for the DWD expiration calculation.

The DWD Down Counter will be decremented with RTI\_FCLK frequency. If the RTI\_FCLK is switched off via the disable registers of the Clock management, the DWD counter stops decrementing. The DWD module cannot generate a reset under this condition.



**Figure 12-1207. RTI Digital Watchdog Operation**

The expiration time of the DWD Down Counter can be determined with following equation:

$$t_{exp} = (RTI\_DWDPRLD + 1) \times 2^{13} / RTI\_FCLK \quad (28)$$

where RTI\_DWDPRLD = 0...4095

### 12.10.2.3.3 RTI Digital Windowed Watchdog

#### Note

Some of the RTI features described in this section may not be supported on this family of devices. For more information, see *RTI Not Supported Features*.

#### Note

Digital windowed watchdog (DWWD) timer is implemented using the digital windowed watchdog function of the RTI modules. Real time interrupt functionality is not supported. In this mode, the timer should default to disabled and user can adjust the period as desired before enabling the watchdog.

In addition to the time-out boundary configurable via the digital watchdog (DWD), some applications may also want to configure the start-time boundary of the watchdog. This is enabled by the digital windowed watchdog (DWWD) feature.

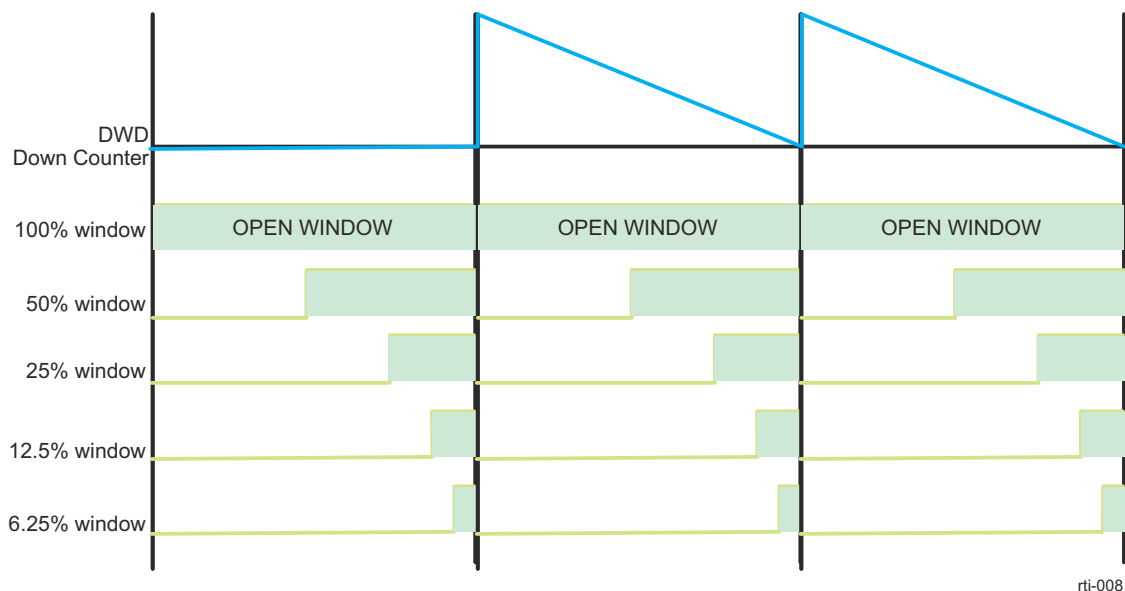
### Functional Behavior

The DWWD opens a configurable time window in which the watchdog must be serviced. Any attempt to service the watchdog outside this time window, or a failure to service the watchdog in this time window, will cause the watchdog to generate either a reset or a non-maskable interrupt to the CPU. This is controlled by configuring the RTI\_WWDRXNCTRL register. As stated earlier, when the watchdog needs to be enabled by software, the watchdog counter is disabled on a system reset. When the DWWD is configured to generate a non-maskable interrupt on a window violation, the watchdog counter continues to count down. The RTI\_INTR\_WWD interrupt handler needs to clear the watchdog violation status flag(s) and then service the watchdog by writing the correct sequence in the watchdog key RTI\_WDKEY register. This service will cause the watchdog counter to get reloaded from the preload value and start counting down. If the RTI\_INTR\_WWD handler does not service the watchdog in time, it could count down all the way to zero and wrap around. No second exception for a time out is generated in this case.

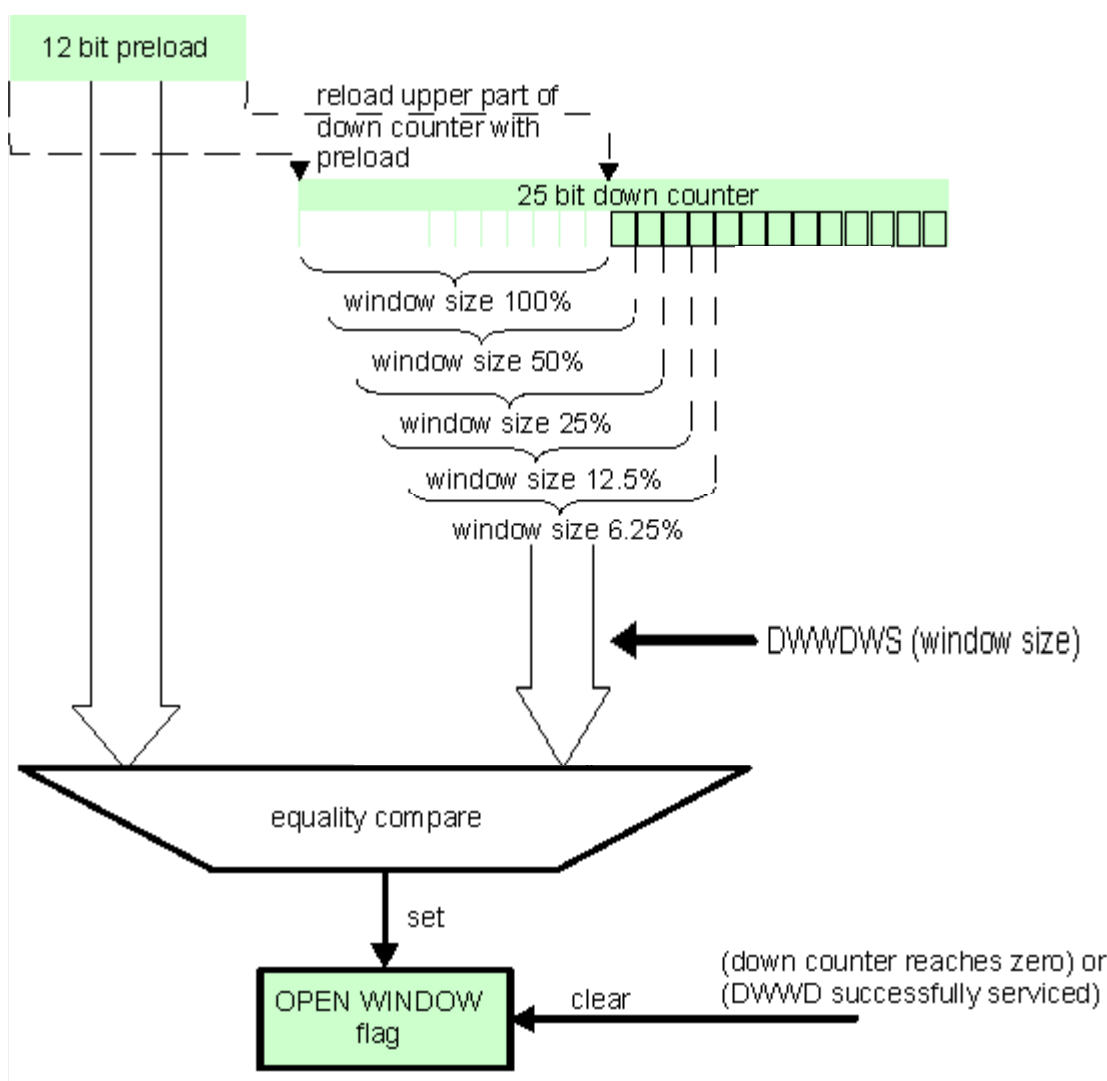
### Configuration of DWWD



The DWWD preload value (same as DWD preload) can only be configured when the DWWD counter is disabled. The window size and watchdog reaction to a violation can be configured even after the watchdog has been enabled. Any changes to the window size and watchdog reaction configurations will only take effect after the next servicing of the DWWD.



**Figure 12-1208. RTI Digital Windowed Watchdog Timing Example**



**Figure 12-1209. RTI Digital Windowed Watchdog Operation Block Diagram**

#### 12.10.2.3.4 RTI Low Power Mode Operation

The operation of the RTI module is guaranteed in run, doze and snooze mode. In sleep or hibernate mode all clocks will be switched off and the RTI module will not work.

In doze and snooze modes all parts of the RTI are active, since it has to be able to wake up the device with compare and timebases interrupts. Capturing events generated by the interrupt module is also possible since in both modes the peripheral modules are able to generate interrupts, which can trigger capture events. The RTI module will generate compare and timebases interrupts. The compare interrupts will periodically wake up the device.

#### Note

In the special case of doze mode with DPLL off, RTI\_FCLK might have a different period than with DPLL enabled, since RTI\_FCLK will be derived from the oscillator output. It has to be ensured that the RTI\_ICLK to RTI\_FCLK ratio is at least 3:1.

The DWD/DWWD remains active when the device enters low power mode as long as the RTI\_FCLK is kept active.

Whenever the LPSC that controls an RTI is in any state other than Enable (see *Module States*), the RTI cannot count or generate interrupts. For more information, see *Power Control Modules*.

During standard SoC warm reset the RTI and the rest of the SoC are reset. In this case, the RTI counters are stopped and interrupts are not issued. During reset-isolated warm reset, if an R5FSS is put in reset-isolation, then the associated RTI also becomes reset isolated. As such, the R5FSS and the RTI are not reset, but RTI stops counting and cannot generate interrupts until R5FSS is taken out of CLKSTOP (brought to Enable state).

#### **12.10.2.3.5 RTI Debug Mode Behavior**

---

##### **Note**

Some of the RTI features described in this section may not be supported on this family of devices. For more information, see *RTI Not Supported Features*.

---

Once the system enters debug mode, the behavior of the RTI depends on the RTI\_GCTRL[15] COS bit. If the bit is cleared and debug mode is active, all counters will stop operation. If the bit is set to one, all counters will be clocked normally and the RTI will work like in normal mode.

The DWD counter will not decrement in debug mode and will hold its current value, regardless of the RTI\_GCTRL[15] COS bit.

---

##### **Note**

The user must not service the watchdog while in debug mode.

---

## 12.10.3 Timers

This section describes the Timer modules for the device.

### 12.10.3.1 Timers Overview

There are thirty timer modules in the device. [Table 12-1579](#) shows the timers allocation across device domains.

**Table 12-1579. Timers Allocation Across Device Domains**

Instance	Domain		
	WKUP	MCU	MAIN
MCU_TIMER0	-	✓	-
MCU_TIMER1	-	✓	-
MCU_TIMER2	-	✓	-
MCU_TIMER3	-	✓	-
MCU_TIMER4	-	✓	-
MCU_TIMER5	-	✓	-
MCU_TIMER6	-	✓	-
MCU_TIMER7	-	✓	-
MCU_TIMER8	-	✓	-
MCU_TIMER9	-	✓	-
TIMER0	-	-	✓
TIMER1	-	-	✓
TIMER2	-	-	✓
TIMER3	-	-	✓
TIMER4	-	-	✓
TIMER5	-	-	✓
TIMER6	-	-	✓
TIMER7	-	-	✓
TIMER8	-	-	✓
TIMER9	-	-	✓
TIMER10	-	-	✓
TIMER11	-	-	✓
TIMER12	-	-	✓
TIMER13	-	-	✓
TIMER14	-	-	✓
TIMER15	-	-	✓
TIMER16	-	-	✓
TIMER17	-	-	✓
TIMER18	-	-	✓
TIMER19	-	-	✓

All timers include specific functions to generate accurate tick interrupts to the operating system.

Each timer can be clocked from several different independent clocks. The selection of clock source is made from registers in the MCU\_CTRL\_MMR0/CTRL\_MMR0.

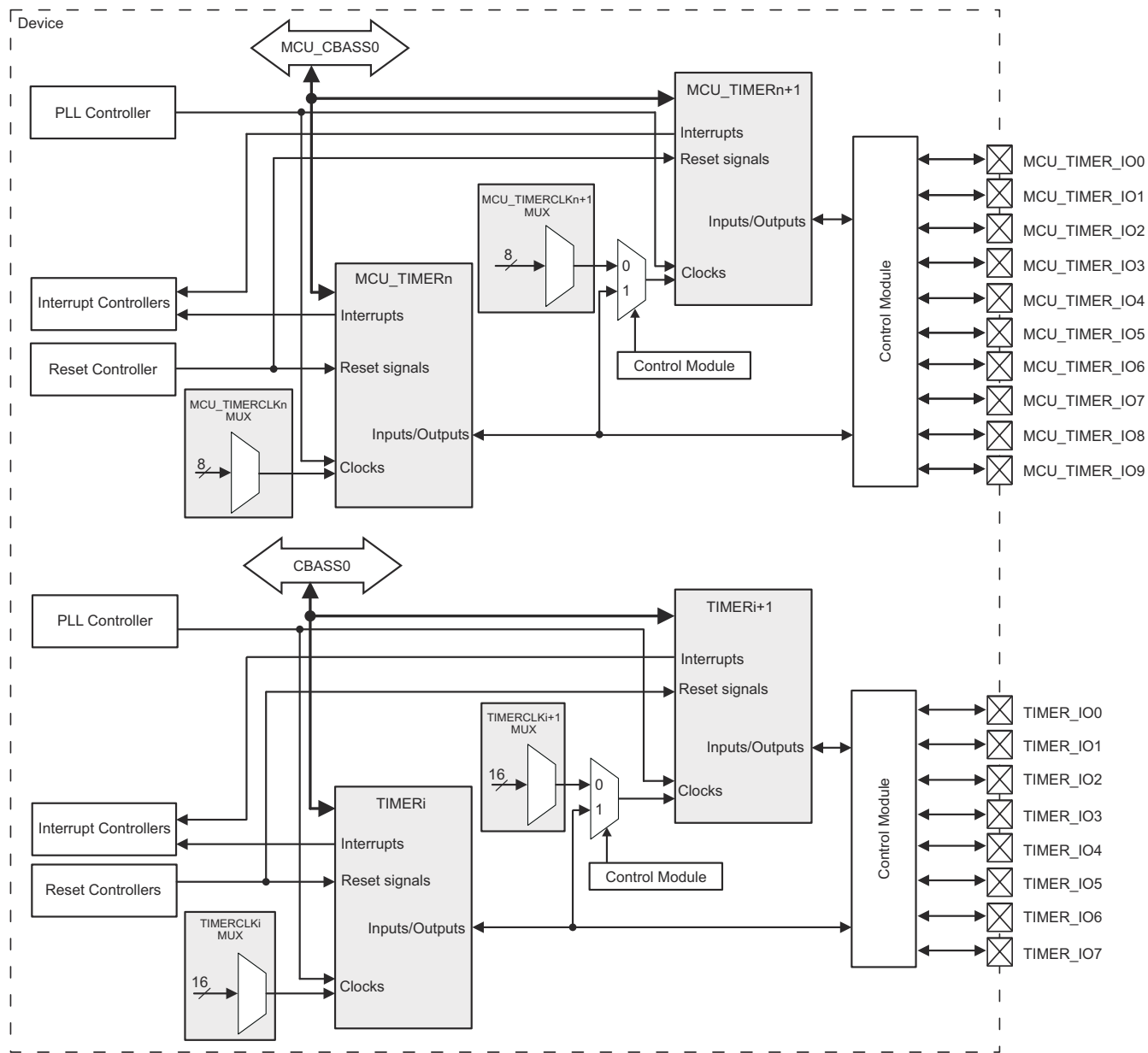
In the MCU domain the device provides 10 timer pins to be used as MCU Timer Capture inputs or as MCU Timer PWM outputs. In order to provide maximum flexibility, these 10 pins may be used with any of MCU\_TIMER0 through MCU\_TIMER9 instances. System level muxes are used to control the capture source pin for each MCU\_TIMER[9-0] and the MCU\_TIMER[9-0] source for each MCU\_TIMER\_IO[9-0] PWM output.

In the MAIN domain the device provides 8 timer pins to be used as Timer Capture inputs or as Timer PWM outputs. For maximum flexibility, these 8 pins may be used with any of TIMER0 through TIMER19 instances. System level muxes are used to control the capture source pin for each TIMER[19-0] and the TIMER[19-0] source for each TIMER\_IO[7-0] PWM output.

Each odd numbered timer instance from each of the domains may be optionally cascaded with the previous even numbered timer instance from the same domain to form up to a 64-bit timer. For example, TIMER1 may be cascaded to TIMER0, MCU\_TIMER1 may be cascaded to MCU\_TIMER0, etc.

When cascaded,  $TIMER_i$  acts as a 32-bit prescaler to  $TIMER_{i+1}$ , as well as  $MCU\_TIMER_n$  acts as a 32-bit prescaler to  $MCU\_TIMER_{n+1}$ .  $TIMER_i$  /  $MCU\_TIMER_n$  must be configured to generate a PWM output edge at the desired rate to increment the  $TIMER_{i+1}$  /  $MCU\_TIMER_{n+1}$  counter.

Figure 12-1210 is an overview of the timers.



$i = 0, 2, 4, 6, 8, 10, 12, 14, 16, 18$

timers\_001

n = 0, 2, 4, 6, 8

## Figure 12-1210. Timers Overview

### 12.10.3.1.1 Timers Features

The following are the main features of the timer controllers:

- Slave interface supports:
  - 32-bit data bus width
  - 32-bit access supported
  - 10-bit address bus width
  - Write nonposted transaction mode supported
- Interrupts generated on overflow, compare, and capture
- Free-running 32-bit upward counter
- Compare and capture modes
- Autoreload mode
- Start/stop mode
- Programmable divider clock source ( $2^n$ , where  $n = [0-8]$ )
- Dedicated input trigger for capture mode and dedicated output trigger/PWM signal
- On-the-fly read/write register (while counting)
- Generates a 1-ms tick clock when functional clock is 32.768 kHz

### 12.10.3.1.2 Timers Not Supported Features

The following features are not supported on this family of devices:

- Atomic 64-bit timer value read of cascaded timers is not supported.
- Smart-idle with wake-up mode is not supported.

### 12.10.3.2 Timers Environment

The MCU\_TIMER[9-0] and TIMER[19-0] modules are hereinafter referred to as timer module.

This section describes the timers external connections (environment).

#### 12.10.3.2.1 Timer External System Interface

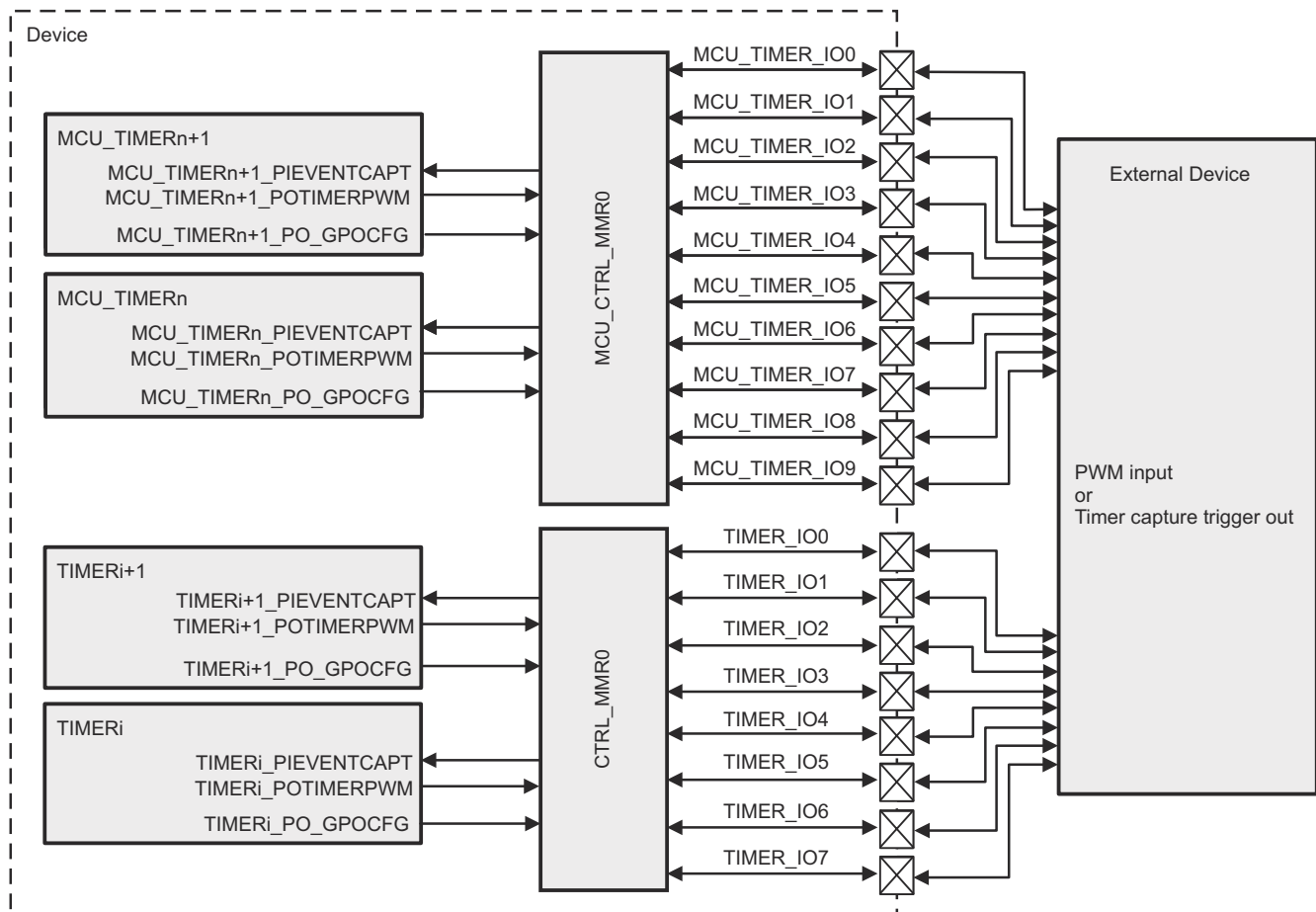
Each timer can send or receive stimulus to/from the external (off-chip) system. In the device all timers are configured to output a PWM pulse or receive an external event signal used as a trigger to capture the current timer count.

Figure 12-1211 shows the external system interface for the timers, and Table 12-1580 describes the timer inputs and outputs.

#### Note

Software control (MCU\_CTRL\_MMR0 and CTRL\_MMR0) must ensure that MUX mode is configured to select the MCU\_TIMER[9-0] signals on ten different pads and to select the TIMER0 through TIMER19 signals on eight different pads. For more information about registers in MCU\_CTRL\_MMR0 and CTRL\_MMR0, see *Timer IO Muxing Control Registers* and *Timer IO Muxing Control Registers in Control Module (CTRL\_MMR)*.

For more information about the configuration of the MCU\_TIMER\_IO[9-0] and TIMER\_IO[7-0] pads, see *Pad Configuration Registers*.



timers-002

i = 0, 2, 4, 6, 8, 10, 12, 14, 16, 18

n = 0, 2, 4, 6, 8

**Figure 12-1211. Timers External System Interface**
**Table 12-1580. Timer I/O Signals**

Module Pin	Device Level Signal	I/O <sup>(1)</sup>	Description	Module Pin Reset Value <sup>(2)</sup>
<b>MCU_TIMER[9-0]</b>				
MCU_TIMER_IO0	MCU_TIMER[9-0]_PIEVENTCAPT MCU_TIMER[9-0]_POTIMERPWM	I/O	MCU_TIMER[9-0] trigger input or MCU_TIMER[9-0] output	0
MCU_TIMER_IO1	MCU_TIMER[9-0]_PIEVENTCAPT MCU_TIMER[9-0]_POTIMERPWM	I/O	MCU_TIMER[9-0] trigger input or MCU_TIMER[9-0] output	0
MCU_TIMER_IO2	MCU_TIMER[9-0]_PIEVENTCAPT MCU_TIMER[9-0]_POTIMERPWM	I/O	MCU_TIMER[9-0] trigger input or MCU_TIMER[9-0] output	0
MCU_TIMER_IO3	MCU_TIMER[9-0]_PIEVENTCAPT MCU_TIMER[9-0]_POTIMERPWM	I/O	MCU_TIMER[9-0] trigger input or MCU_TIMER[9-0] output	0
MCU_TIMER_IO4	MCU_TIMER[9-0]_PIEVENTCAPT MCU_TIMER[9-0]_POTIMERPWM	I/O	MCU_TIMER[9-0] trigger input or MCU_TIMER[9-0] output	0
MCU_TIMER_IO5	MCU_TIMER[9-0]_PIEVENTCAPT MCU_TIMER[9-0]_POTIMERPWM	I/O	MCU_TIMER[9-0] trigger input or MCU_TIMER[9-0] output	0
MCU_TIMER_IO6	MCU_TIMER[9-0]_PIEVENTCAPT MCU_TIMER[9-0]_POTIMERPWM	I/O	MCU_TIMER[9-0] trigger input or MCU_TIMER[9-0] output	0
MCU_TIMER_IO7	MCU_TIMER[9-0]_PIEVENTCAPT MCU_TIMER[9-0]_POTIMERPWM	I/O	MCU_TIMER[9-0] trigger input or MCU_TIMER[9-0] output	0
MCU_TIMER_IO8	MCU_TIMER[9-0]_PIEVENTCAPT MCU_TIMER[9-0]_POTIMERPWM	I/O	MCU_TIMER[9-0] trigger input or MCU_TIMER[9-0] output	0
MCU_TIMER_IO9	MCU_TIMER[9-0]_PIEVENTCAPT MCU_TIMER[9-0]_POTIMERPWM	I/O	MCU_TIMER[9-0] trigger input or MCU_TIMER[9-0] output	0
<b>TIMER[19-0]</b>				
TIMER_IO0	TIMER[19-0]_PIEVENTCAPT TIMER[19-0]_POTIMERPWM	I/O	TIMER[19-0] trigger input or TIMER[19-0] PWM output	0
TIMER_IO1	TIMER[19-0]_PIEVENTCAPT TIMER[19-0]_POTIMERPWM	I/O	TIMER[19-0] trigger input or TIMER[19-0] PWM output	0
TIMER_IO2	TIMER[19-0]_PIEVENTCAPT TIMER[19-0]_POTIMERPWM	I/O	TIMER[19-0] trigger input or TIMER[19-0] PWM output	0
TIMER_IO3	TIMER[19-0]_PIEVENTCAPT TIMER[19-0]_POTIMERPWM	I/O	TIMER[19-0] trigger input or TIMER[19-0] PWM output	0
TIMER_IO4	TIMER[19-0]_PIEVENTCAPT TIMER[19-0]_POTIMERPWM	I/O	TIMER[19-0] trigger input or TIMER[19-0] PWM output	0
TIMER_IO5	TIMER[19-0]_PIEVENTCAPT TIMER[19-0]_POTIMERPWM	I/O	TIMER[19-0] trigger input or TIMER[19-0] PWM output	0
TIMER_IO6	TIMER[19-0]_PIEVENTCAPT TIMER[19-0]_POTIMERPWM	I/O	TIMER[19-0] trigger input or TIMER[19-0] PWM output	0
TIMER_IO7	TIMER[19-0]_PIEVENTCAPT TIMER[19-0]_POTIMERPWM	I/O	TIMER[19-0] trigger input or TIMER[19-0] PWM output	0

(1) When configured for that function; I = Input, O = Output



---

**Note**

Each MCU\_TIMER\_PO\_GPOCFG and TIMER\_PO\_GPOCFG signals are used as an output enable to control the function of the MCU\_TIMER\_IO[9-0], and respectively TIMER\_IO[7-0], as the PWM output (PO\_GPOCFG = 0) or capture input (PO\_GPOCFG = 1).

---

---

**Note**

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

---

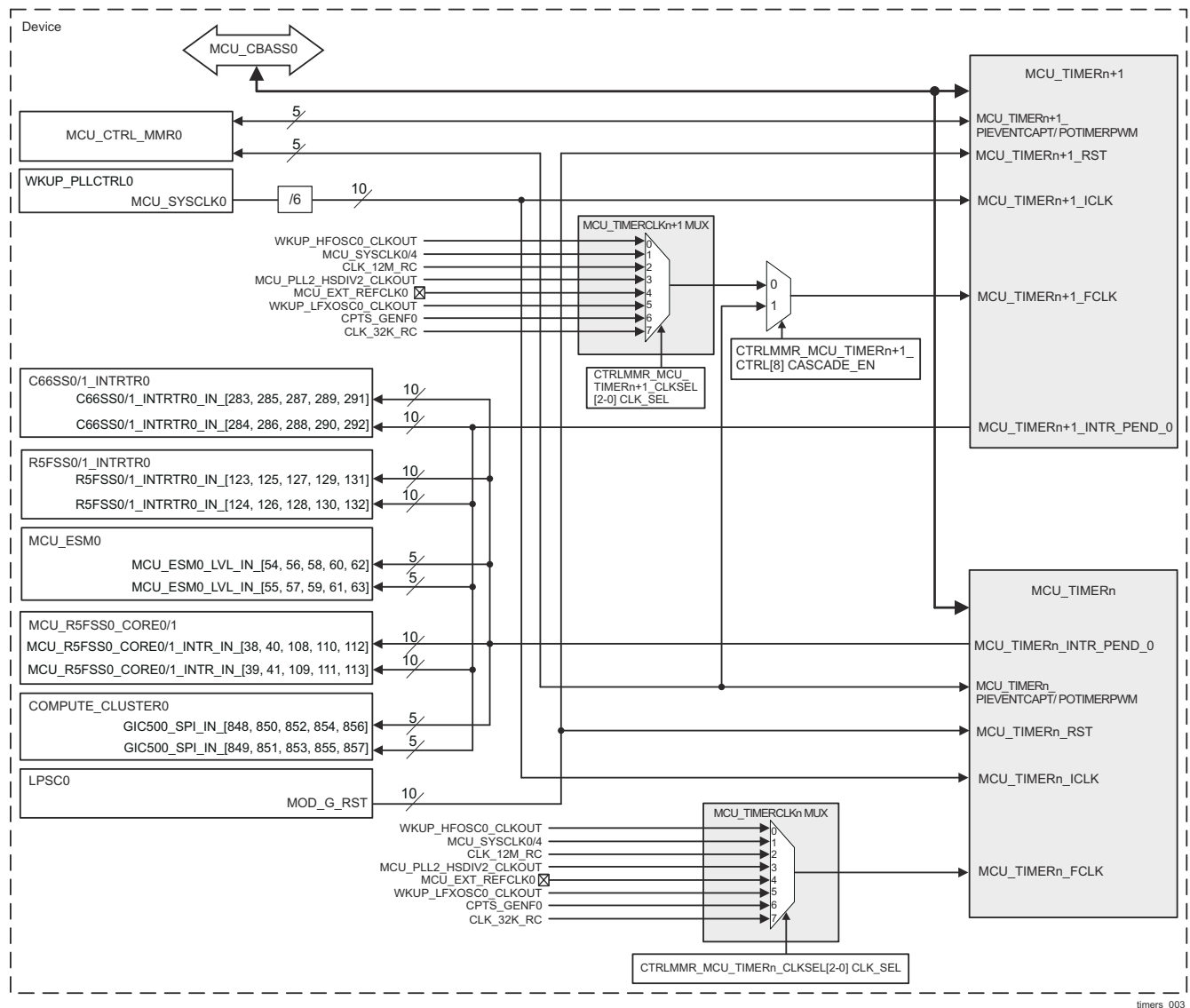
### 12.10.3.3 Timers Integration

This section describes module integration in the device, including information about clocks, resets, and hardware requests.

### 12.10.3.3.1 Timers Integration in MCU Domain

There are ten timer modules integrated in the device MCU domain - MCU\_TIMER0 through MCU\_TIMER9. Figure 12-1212 shows their integration in the device.

Each timer instance is supplied by dedicated TIMERCLKn clock mux. For MCU\_TIMERn+1 the TIMERCLKn output is further muxed with the TIMERn\_POTIMERPWM output.



n = 0, 2, 4, 6, 8

**Figure 12-1212. MCU\_TIMER Integration**

Table 12-1581 through Table 12-1583 summarize the integration of MCU\_TIMER0 through MCU\_TIMER9 in device MCU domain.

**Table 12-1581. MCU\_TIMER Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
MCU_TIMER0	WKUP_PSC0	PD0	LPSC0	MCU_CBASS0
MCU_TIMER1	WKUP_PSC0	PD0	LPSC0	MCU_CBASS0

**Table 12-1581. MCU\_TIMER Integration Attributes (continued)**

MCU_TIMER2	WKUP_PSC0	PD0	LPSC0	MCU_CBASS0
MCU_TIMER3	WKUP_PSC0	PD0	LPSC0	MCU_CBASS0
MCU_TIMER4	WKUP_PSC0	PD0	LPSC0	MCU_CBASS0
MCU_TIMER5	WKUP_PSC0	PD0	LPSC0	MCU_CBASS0
MCU_TIMER6	WKUP_PSC0	PD0	LPSC0	MCU_CBASS0
MCU_TIMER7	WKUP_PSC0	PD0	LPSC0	MCU_CBASS0
MCU_TIMER8	WKUP_PSC0	PD0	LPSC0	MCU_CBASS0
MCU_TIMER9	WKUP_PSC0	PD0	LPSC0	MCU_CBASS0

**Table 12-1582. MCU\_TIMER Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
MCU_TIMER0	MCU_TIMER0_ICLK	MCU_SYSCCLK0/6	WKUP_PLLCTRL0	MCU_TIMER0 Interface Clock
	MCU_TIMER0_FCLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	MCU_TIMER0 Functional Clock. Output of multiplexor
		MCU_SYSCCLK0/4	PLLCTRL0	MCU_TIMERCLK0 MUX controlled by CTRLMMR_MCU_TIMER0_CLKSEL[2-0] CLK_SEL in <i>Control Module (CTRL_MMR)</i>
		CLK_12M_RC	WKUP_RC_OSC_12M	
		MCU_PLL2_HSDIV2_CLK OUT	MCU_PLL2_HSDIV2	
		MCU_EXT_REFCLK0	I/O pin	
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CPTS_GENF0	MCU_CPSW0	
MCU_TIMER1	MCU_TIMER1_ICLK	MCU_SYSCCLK0/6	WKUP_PLLCTRL0	MCU_TIMER1 Interface Clock
	MCU_TIMER1_FCLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	MCU_TIMER1 Functional Clock. Output of multiplexor
		MCU_SYSCCLK0/4	PLLCTRL0	MCU_TIMERCLK1 MUX controlled by CTRLMMR_MCU_TIMER1_CLKSEL[2-0] CLK_SEL or
		CLK_12M_RC	WKUP_RC_OSC_12M	MCU_TIMER0_POTIMERPWM in <i>Control Module (CTRL_MMR)</i>
		MCU_PLL2_HSDIV2_CLK OUT	MCU_PLL2_HSDIV2	
		MCU_EXT_REFCLK0	I/O pin	
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CPTS_GENF0	MCU_CPSW0	
MCU_TIMER2	MCU_TIMER2_ICLK	MCU_SYSCCLK0/6	WKUP_PLLCTRL0	MCU_TIMER2 Interface Clock
	MCU_TIMER2_FCLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	MCU_TIMER2 Functional Clock. Output of multiplexor
		MCU_SYSCCLK0/4	PLLCTRL0	MCU_TIMERCLK2 MUX controlled by CTRLMMR_MCU_TIMER2_CLKSEL[2-0] CLK_SEL in <i>Control Module (CTRL_MMR)</i>
		CLK_12M_RC	WKUP_RC_OSC_12M	
		MCU_PLL2_HSDIV2_CLK OUT	MCU_PLL2_HSDIV2	
		MCU_EXT_REFCLK0	I/O pin	
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CPTS_GENF0	MCU_CPSW0	
MCU_TIMER3	MCU_TIMER3_ICLK	CLK_32K_RC	WKUP_RC_OSC_12M	
		MCU_SYSCCLK0/6	WKUP_PLLCTRL0	MCU_TIMER3 Interface Clock

**Table 12-1582. MCU\_TIMER Clocks and Resets (continued)**

MCU_TIMER3_FCLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	MCU_TIMER3 Functional Clock.
	MCU_SYSCLK0/4	PLLCTRL0	Output of multiplexor
	CLK_12M_RC	WKUP_RC_OSC_12M	MCU_TIMERCLK3 MUX controlled by
	MCU_PLL2_HSDIV2_CLK OUT	MCU_PLL2_HSDIV2	CTRLMMR_MCU_TIMER3_CLKSEL[ 2-0] CLK_SEL or
	MCU_EXT_REFCLK0	I/O pin	MCU_TIMER2_POTIMERPWM in
	WKUP_LFXOSC0_CLKOU T	WKUP_LFXOSC0	Control Module (CTRL_MMR)
	CPTS_GENF0	MCU_CPSW0	
MCU_TIMER4	CLK_32K_RC	WKUP_RC_OSC_12M	
	MCU_TIMER4_ICLK	MCU_SYSCLK0/6	WKUP_PLLCTRL0
	MCU_TIMER4_FCLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0
		MCU_SYSCLK0/4	PLLCTRL0
		CLK_12M_RC	WKUP_RC_OSC_12M
		MCU_PLL2_HSDIV2_CLK OUT	MCU_PLL2_HSDIV2
		MCU_EXT_REFCLK0	I/O pin
MCU_TIMER5		WKUP_LFXOSC0_CLKOU T	WKUP_LFXOSC0
		CPTS_GENF0	MCU_CPSW0
		CLK_32K_RC	WKUP_RC_OSC_12M
	MCU_TIMER5_ICLK	MCU_SYSCLK0/6	WKUP_PLLCTRL0
	MCU_TIMER5_FCLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0
		MCU_SYSCLK0/4	PLLCTRL0
		CLK_12M_RC	WKUP_RC_OSC_12M
MCU_TIMER6		MCU_PLL2_HSDIV2_CLK OUT	MCU_PLL2_HSDIV2
		MCU_EXT_REFCLK0	I/O pin
		WKUP_LFXOSC0_CLKOU T	WKUP_LFXOSC0
		CPTS_GENF0	MCU_CPSW0
		CLK_32K_RC	WKUP_RC_OSC_12M
	MCU_TIMER6_ICLK	MCU_SYSCLK0/6	WKUP_PLLCTRL0
	MCU_TIMER6_FCLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0
MCU_TIMER7		MCU_SYSCLK0/4	PLLCTRL0
		CLK_12M_RC	WKUP_RC_OSC_12M
		MCU_PLL2_HSDIV2_CLK OUT	MCU_PLL2_HSDIV2
		MCU_EXT_REFCLK0	I/O pin
		WKUP_LFXOSC0_CLKOU T	WKUP_LFXOSC0
		CPTS_GENF0	MCU_CPSW0
		CLK_32K_RC	WKUP_RC_OSC_12M
MCU_TIMER7	MCU_TIMER7_ICLK	MCU_SYSCLK0/6	WKUP_PLLCTRL0

**Table 12-1582. MCU\_TIMER Clocks and Resets (continued)**

MCU_TIMER7_FCLK		WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	MCU_TIMER7 Functional Clock. Output of multiplexor MCU_TIMERCLK7 MUX controlled by CTRLMMR_MCU_TIMER7_CLKSEL[ 2-0] CLK_SEL or MCU_TIMER6_POTIMERPWM in Control Module (CTRL_MMR)
		MCU_SYSCLK0/4	PLLCTRL0	
		CLK_12M_RC	WKUP_RC_OSC_12M	
		MCU_PLL2_HSDIV2_CLK OUT	MCU_PLL2_HSDIV2	
		MCU_EXT_REFCLK0	I/O pin	
		WKUP_LFXOSC0_CLKOU T	WKUP_LFXOSC0	
		CPTS_GENF0	MCU_CPSW0	
MCU_TIMER8	MCU_TIMER8_FCLK	CLK_32K_RC	WKUP_RC_OSC_12M	MCU_TIMER8 Interface Clock
		MCU_TIMER8_ICLK	MCU_SYSCLK0/6	
		WKUP_PLLCTRL0	WKUP_PLLCTRL0	
		WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	
		MCU_SYSCLK0/4	PLLCTRL0	
		CLK_12M_RC	WKUP_RC_OSC_12M	
		MCU_PLL2_HSDIV2_CLK OUT	MCU_PLL2_HSDIV2	
MCU_TIMER8	MCU_TIMER8_FCLK	MCU_EXT_REFCLK0	I/O pin	MCU_TIMER8 Functional Clock. Output of multiplexor MCU_TIMERCLK8 MUX controlled by CTRLMMR_MCU_TIMER8_CLKSEL[ 2-0] CLK_SEL in Control Module (CTRL_MMR)
		WKUP_LFXOSC0_CLKOU T	WKUP_LFXOSC0	
		CPTS_GENF0	MCU_CPSW0	
		CLK_32K_RC	WKUP_RC_OSC_12M	
		MCU_TIMER9_ICLK	MCU_SYSCLK0/6	
		WKUP_PLLCTRL0	WKUP_PLLCTRL0	
		WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	
MCU_TIMER9	MCU_TIMER9_FCLK	MCU_SYSCLK0/4	PLLCTRL0	MCU_TIMER9 Interface Clock
		CLK_12M_RC	WKUP_RC_OSC_12M	
		MCU_PLL2_HSDIV2_CLK OUT	MCU_PLL2_HSDIV2	
		MCU_EXT_REFCLK0	I/O pin	
		WKUP_LFXOSC0_CLKOU T	WKUP_LFXOSC0	
		CPTS_GENF0	MCU_CPSW0	
		CLK_32K_RC	WKUP_RC_OSC_12M	

Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
MCU_TIMER0	MCU_TIMER0_RST	MOD_G_RST	LPSC0	Asynchronous Reset to MCU_TIMER0
MCU_TIMER1	MCU_TIMER1_RST	MOD_G_RST	LPSC0	Asynchronous Reset to MCU_TIMER1
MCU_TIMER2	MCU_TIMER2_RST	MOD_G_RST	LPSC0	Asynchronous Reset to MCU_TIMER2
MCU_TIMER3	MCU_TIMER3_RST	MOD_G_RST	LPSC0	Asynchronous Reset to MCU_TIMER3
MCU_TIMER4	MCU_TIMER4_RST	MOD_G_RST	LPSC0	Asynchronous Reset to MCU_TIMER4
MCU_TIMER5	MCU_TIMER5_RST	MOD_G_RST	LPSC0	Asynchronous Reset to MCU_TIMER5
MCU_TIMER6	MCU_TIMER6_RST	MOD_G_RST	LPSC0	Asynchronous Reset to MCU_TIMER6
MCU_TIMER7	MCU_TIMER7_RST	MOD_G_RST	LPSC0	Asynchronous Reset to MCU_TIMER7
MCU_TIMER8	MCU_TIMER8_RST	MOD_G_RST	LPSC0	Asynchronous Reset to MCU_TIMER8
MCU_TIMER9	MCU_TIMER9_RST	MOD_G_RST	LPSC0	Asynchronous Reset to MCU_TIMER9

**Table 12-1583. MCU\_TIMER Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type

**Table 12-1583. MCU\_TIMER Hardware Requests (continued)**

MCU_TIMER0	MCU_TIMER0_INTR_P END_0	C66SS0_INTRTR0_IN_283	C66SS0_INTRTR0	MCU_TIMER0 Interrupt Request	Level
		C66SS1_INTRTR0_IN_283	C66SS1_INTRTR0	MCU_TIMER0 Interrupt Request	Level
		MCU_R5FSS0_CORE0_INT R_IN_38	MCU_R5FSS0_CORE 0	MCU_TIMER0 Interrupt Request	Level
		MCU_R5FSS0_CORE1_INT R_IN_38	MCU_R5FSS0_CORE 1	MCU_TIMER0 Interrupt Request	Level
		MCU_ESM0_LVL_IN_54	MCU_ESM0	MCU_TIMER0 Interrupt Request	Level
		R5FSS0_INTRTR0_IN_123	R5FSS0_INTRTR0	MCU_TIMER0 Interrupt Request	Level
		R5FSS1_INTRTR0_IN_123	R5FSS1_INTRTR0	MCU_TIMER0 Interrupt Request	Level
		GIC500_SPI_IN_848	COMPUTE_CLUSTER R0	MCU_TIMER0 Interrupt Request	Level
MCU_TIMER1	MCU_TIMER1_INTR_P END_0	C66SS0_INTRTR0_IN_284	C66SS0_INTRTR0	MCU_TIMER1 Interrupt Request	Level
		C66SS1_INTRTR0_IN_284	C66SS1_INTRTR0	MCU_TIMER1 Interrupt Request	Level
		MCU_R5FSS0_CORE0_INT R_IN_39	MCU_R5FSS0_CORE 0	MCU_TIMER1 Interrupt Request	Level
		MCU_R5FSS0_CORE1_INT R_IN_39	MCU_R5FSS0_CORE 1	MCU_TIMER1 Interrupt Request	Level
		MCU_ESM0_LVL_IN_55	MCU_ESM0	MCU_TIMER1 Interrupt Request	Level
		R5FSS0_INTRTR0_IN_124	R5FSS0_INTRTR0	MCU_TIMER1 Interrupt Request	Level
		R5FSS1_INTRTR0_IN_124	R5FSS1_INTRTR0	MCU_TIMER1 Interrupt Request	Level
		GIC500_SPI_IN_849	COMPUTE_CLUSTER R0	MCU_TIMER1 Interrupt Request	Level
MCU_TIMER2	MCU_TIMER2_INTR_P END_0	C66SS0_INTRTR0_IN_285	C66SS0_INTRTR0	MCU_TIMER2 Interrupt Request	Level
		C66SS1_INTRTR0_IN_285	C66SS1_INTRTR0	MCU_TIMER2 Interrupt Request	Level
		MCU_R5FSS0_CORE0_INT R_IN_40	MCU_R5FSS0_CORE 0	MCU_TIMER2 Interrupt Request	Level
		MCU_R5FSS0_CORE1_INT R_IN_40	MCU_R5FSS0_CORE 1	MCU_TIMER2 Interrupt Request	Level
		MCU_ESM0_LVL_IN_56	MCU_ESM0	MCU_TIMER2 Interrupt Request	Level
		R5FSS0_INTRTR0_IN_125	R5FSS0_INTRTR0	MCU_TIMER2 Interrupt Request	Level
		R5FSS1_INTRTR0_IN_125	R5FSS1_INTRTR0	MCU_TIMER2 Interrupt Request	Level
		GIC500_SPI_IN_850	COMPUTE_CLUSTER R0	MCU_TIMER2 Interrupt Request	Level
MCU_TIMER3	MCU_TIMER3_INTR_P END_0	C66SS0_INTRTR0_IN_286	C66SS0_INTRTR0	MCU_TIMER3 Interrupt Request	Level
		C66SS1_INTRTR0_IN_286	C66SS1_INTRTR0	MCU_TIMER3 Interrupt Request	Level
		MCU_R5FSS0_CORE0_INT R_IN_41	MCU_R5FSS0_CORE 0	MCU_TIMER3 Interrupt Request	Level

**Table 12-1583. MCU\_TIMER Hardware Requests (continued)**

		MCU_R5FSS0_CORE1_INT R_IN_41	MCU_R5FSS0_CORE 1	MCU_TIMER3 Interrupt Request	Level
		MCU_ESM0_LVL_IN_57	MCU_ESM0	MCU_TIMER3 Interrupt Request	Level
		R5FSS0_INTRTR0_IN_126	R5FSS0_INTRTR0	MCU_TIMER3 Interrupt Request	Level
		R5FSS1_INTRTR0_IN_126	R5FSS1_INTRTR0	MCU_TIMER3 Interrupt Request	Level
		GIC500_SPI_IN_851	COMPUTE_CLUSTER R0	MCU_TIMER3 Interrupt Request	Level
MCU_TIMER4	MCU_TIMER4_INTR_P END_0	C66SS0_INTRTR0_IN_287	C66SS0_INTRTR0	MCU_TIMER4 Interrupt Request	Level
		C66SS1_INTRTR0_IN_287	C66SS1_INTRTR0	MCU_TIMER4 Interrupt Request	Level
		MCU_R5FSS0_CORE0_INT R_IN_108	MCU_R5FSS0_CORE 0	MCU_TIMER4 Interrupt Request	Level
		MCU_R5FSS0_CORE1_INT R_IN_108	MCU_R5FSS0_CORE 1	MCU_TIMER4 Interrupt Request	Level
		MCU_ESM0_LVL_IN_58	MCU_ESM0	MCU_TIMER4 Interrupt Request	Level
		R5FSS0_INTRTR0_IN_127	R5FSS0_INTRTR0	MCU_TIMER4 Interrupt Request	Level
		R5FSS1_INTRTR0_IN_127	R5FSS1_INTRTR0	MCU_TIMER4 Interrupt Request	Level
		GIC500_SPI_IN_852	COMPUTE_CLUSTER R0	MCU_TIMER4 Interrupt Request	Level
MCU_TIMER5	MCU_TIMER5_INTR_P END_0	C66SS0_INTRTR0_IN_288	C66SS0_INTRTR0	MCU_TIMER5 Interrupt Request	Level
		C66SS1_INTRTR0_IN_288	C66SS1_INTRTR0	MCU_TIMER5 Interrupt Request	Level
		MCU_R5FSS0_CORE0_INT R_IN_109	MCU_R5FSS0_CORE 0	MCU_TIMER5 Interrupt Request	Level
		MCU_R5FSS0_CORE1_INT R_IN_109	MCU_R5FSS0_CORE 1	MCU_TIMER5 Interrupt Request	Level
		MCU_ESM0_LVL_IN_59	MCU_ESM0	MCU_TIMER5 Interrupt Request	Level
		R5FSS0_INTRTR0_IN_128	R5FSS0_INTRTR0	MCU_TIMER5 Interrupt Request	Level
		R5FSS1_INTRTR0_IN_128	R5FSS1_INTRTR0	MCU_TIMER5 Interrupt Request	Level
		GIC500_SPI_IN_853	COMPUTE_CLUSTER R0	MCU_TIMER5 Interrupt Request	Level
MCU_TIMER6	MCU_TIMER6_INTR_P END_0	C66SS0_INTRTR0_IN_289	C66SS0_INTRTR0	MCU_TIMER6 Interrupt Request	Level
		C66SS1_INTRTR0_IN_289	C66SS1_INTRTR0	MCU_TIMER6 Interrupt Request	Level
		MCU_R5FSS0_CORE0_INT R_IN_110	MCU_R5FSS0_CORE 0	MCU_TIMER6 Interrupt Request	Level
		MCU_R5FSS0_CORE1_INT R_IN_110	MCU_R5FSS0_CORE 1	MCU_TIMER6 Interrupt Request	Level
		MCU_ESM0_LVL_IN_60	MCU_ESM0	MCU_TIMER6 Interrupt Request	Level
		R5FSS0_INTRTR0_IN_129	R5FSS0_INTRTR0	MCU_TIMER6 Interrupt Request	Level



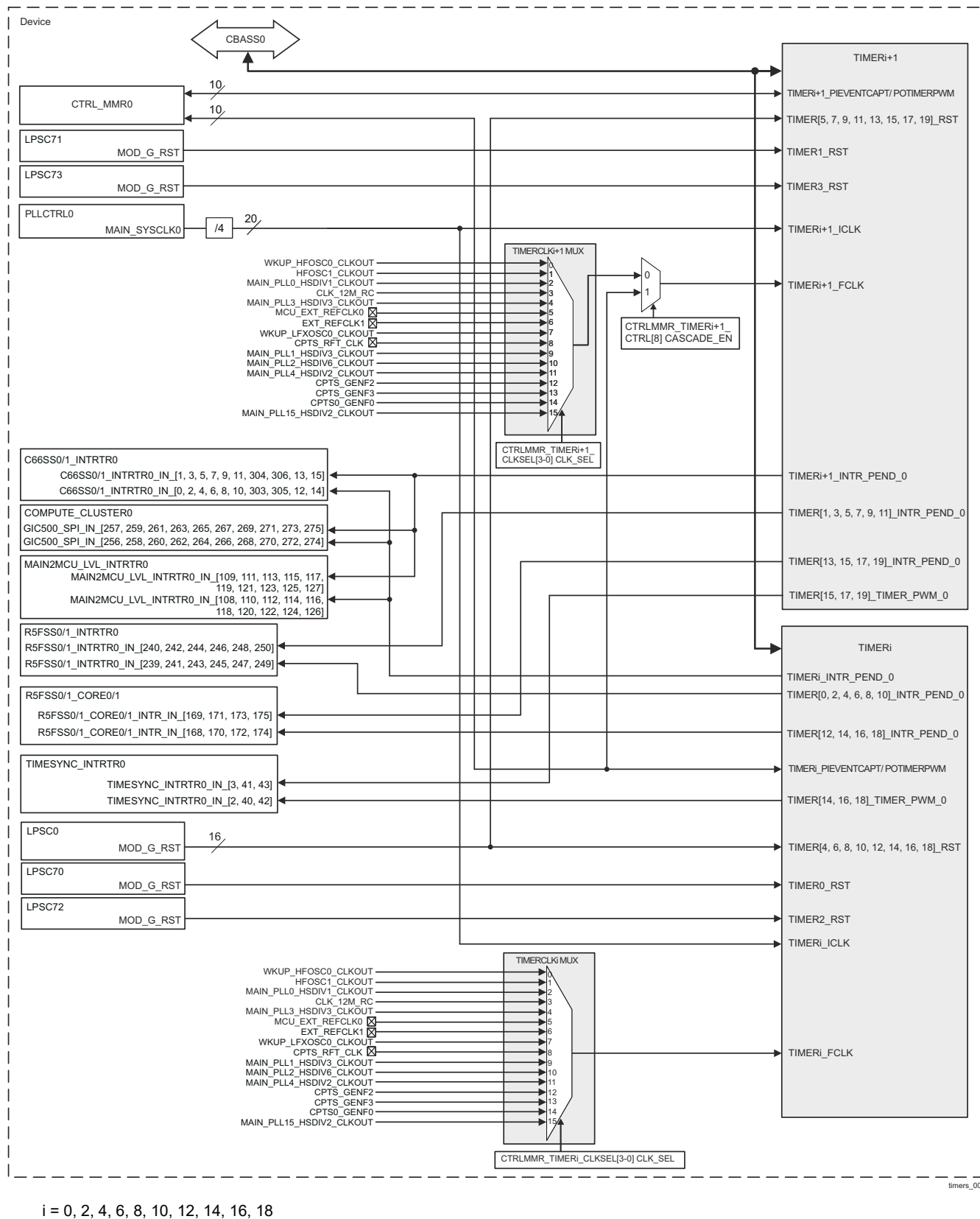
**Table 12-1583. MCU\_TIMER Hardware Requests (continued)**

MCU_TIMER7	MCU_TIMER7_INTR_P END_0	R5FSS1_INTRTR0_IN_129	R5FSS1_INTRTR0	MCU_TIMER6 Interrupt Request	Level
		GIC500_SPI_IN_854	COMPUTE_CLUSTER0	MCU_TIMER6 Interrupt Request	Level
		C66SS0_INTRTR0_IN_290	C66SS0_INTRTR0	MCU_TIMER7 Interrupt Request	Level
		C66SS1_INTRTR0_IN_290	C66SS1_INTRTR0	MCU_TIMER7 Interrupt Request	Level
		MCU_R5FSS0_CORE0_INT_R_IN_111	MCU_R5FSS0_CORE0	MCU_TIMER7 Interrupt Request	Level
		MCU_R5FSS0_CORE1_INT_R_IN_111	MCU_R5FSS0_CORE1	MCU_TIMER7 Interrupt Request	Level
		MCU_ESM0_LVL_IN_61	MCU_ESM0	MCU_TIMER7 Interrupt Request	Level
		R5FSS0_INTRTR0_IN_130	R5FSS0_INTRTR0	MCU_TIMER7 Interrupt Request	Level
		R5FSS1_INTRTR0_IN_130	R5FSS1_INTRTR0	MCU_TIMER7 Interrupt Request	Level
MCU_TIMER8	MCU_TIMER8_INTR_P END_0	GIC500_SPI_IN_855	COMPUTE_CLUSTER0	MCU_TIMER7 Interrupt Request	Level
		C66SS0_INTRTR0_IN_291	C66SS0_INTRTR0	MCU_TIMER8 Interrupt Request	Level
		C66SS1_INTRTR0_IN_291	C66SS1_INTRTR0	MCU_TIMER8 Interrupt Request	Level
		MCU_R5FSS0_CORE0_INT_R_IN_112	MCU_R5FSS0_CORE0	MCU_TIMER8 Interrupt Request	Level
		MCU_R5FSS0_CORE1_INT_R_IN_112	MCU_R5FSS0_CORE1	MCU_TIMER8 Interrupt Request	Level
		MCU_ESM0_LVL_IN_62	MCU_ESM0	MCU_TIMER8 Interrupt Request	Level
		R5FSS0_INTRTR0_IN_131	R5FSS0_INTRTR0	MCU_TIMER8 Interrupt Request	Level
		R5FSS1_INTRTR0_IN_131	R5FSS1_INTRTR0	MCU_TIMER8 Interrupt Request	Level
		GIC500_SPI_IN_856	COMPUTE_CLUSTER0	MCU_TIMER8 Interrupt Request	Level
MCU_TIMER9	MCU_TIMER9_INTR_P END_0	C66SS0_INTRTR0_IN_292	C66SS0_INTRTR0	MCU_TIMER9 Interrupt Request	Level
		C66SS1_INTRTR0_IN_292	C66SS1_INTRTR0	MCU_TIMER9 Interrupt Request	Level
		MCU_R5FSS0_CORE0_INT_R_IN_113	MCU_R5FSS0_CORE0	MCU_TIMER9 Interrupt Request	Level
		MCU_R5FSS0_CORE1_INT_R_IN_113	MCU_R5FSS0_CORE1	MCU_TIMER9 Interrupt Request	Level
		MCU_ESM0_LVL_IN_63	MCU_ESM0	MCU_TIMER9 Interrupt Request	Level
		R5FSS0_INTRTR0_IN_132	R5FSS0_INTRTR0	MCU_TIMER9 Interrupt Request	Level
		R5FSS1_INTRTR0_IN_132	R5FSS1_INTRTR0	MCU_TIMER9 Interrupt Request	Level
		GIC500_SPI_IN_857	COMPUTE_CLUSTER0	MCU_TIMER9 Interrupt Request	Level

#### 12.10.3.3.2 Timers Integration in MAIN Domain

There are twenty timer modules integrated in the device MAIN domain - TIMER0 through TIMER19. [Figure 12-1213](#) shows their integration in the device.

Each timer instance is supplied by dedicated TIMERCLKi clock mux. For TIMERi+1 the TIMERCLKi output is further muxed with the TIMERi\_POTIMERPWM output.



**Figure 12-1213. TIMER Integration**

Table 12-1584 through Table 12-1586 summarize the integration of TIMER0 through TIMER19 in device MAIN

domain.

**Table 12-1584. TIMER Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
TIMER0	PSC0	PD11	LPSC70	CBASS0
TIMER1	PSC0	PD11	LPSC71	CBASS0
TIMER2	PSC0	PD11	LPSC72	CBASS0
TIMER3	PSC0	PD11	LPSC73	CBASS0
TIMER4	PSC0	PD0	LPSC0	CBASS0
TIMER5	PSC0	PD0	LPSC0	CBASS0
TIMER6	PSC0	PD0	LPSC0	CBASS0
TIMER7	PSC0	PD0	LPSC0	CBASS0
TIMER8	PSC0	PD0	LPSC0	CBASS0
TIMER9	PSC0	PD0	LPSC0	CBASS0
TIMER10	PSC0	PD0	LPSC0	CBASS0
TIMER11	PSC0	PD0	LPSC0	CBASS0
TIMER12	PSC0	PD0	LPSC0	CBASS0
TIMER13	PSC0	PD0	LPSC0	CBASS0
TIMER14	PSC0	PD0	LPSC0	CBASS0
TIMER15	PSC0	PD0	LPSC0	CBASS0
TIMER16	PSC0	PD0	LPSC0	CBASS0
TIMER17	PSC0	PD0	LPSC0	CBASS0
TIMER18	PSC0	PD0	LPSC0	CBASS0
TIMER19	PSC0	PD0	LPSC0	CBASS0

**Table 12-1585. TIMER Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
TIMER0	TIMER0_ICLK	MAIN_SYSCCLK0/4	PLLCTRL0	TIMER0 Interface Clock

**Table 12-1585. TIMER Clocks and Resets (continued)**

TIMER0_FCLK		WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	TIMER0 Functional Clock. Output of multiplexor TIMERCLK0 MUX controlled by CTRLMMR_TIMER0_CLKSEL[3-0] CLK_SEL in <i>Control Module (CTRL_MMR)</i>
		HFOSC1_CLKOUT	HFOSC1	
		MAIN_PLL0_HSDIV1_CLKO UT	PLL0_HSDIV1	
		CLK_12M_RC	WKUP_RC_OSC_12M	
		MAIN_PLL3_HSDIV3_CLKO UT	PLL3_HSDIV3	
		MCU_EXT_REFCLK0	I/O pin	
		EXT_REFCLK1	I/O pin	
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CPTS_RFT_CLK	I/O pin	
		MAIN_PLL1_HSDIV3_CLKO UT	PLL1_HSDIV3	
		MAIN_PLL2_HSDIV6_CLKO UT	PLL4_HSDIV6	
		MAIN_PLL4_HSDIV2_CLKO UT	PLL4_HSDIV2	
		CPTS_GENF2	NAVSS0_CPTS0	
		CPTS_GENF3	NAVSS0_CPTS0	
		CPTS0_GENF0	CPSW0	
		MAIN_PLL15_HSDIV2_CLK OUT	PLL15_HSDIV2	
TIMER1	TIMER1_ICLK	MAIN_SYSCCLK0/4	PLLCTRL0	TIMER1 Interface Clock
	TIMER1_FCLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	TIMER1 Functional Clock. Output of multiplexor TIMERCLK1 MUX controlled by CTRLMMR_TIMER1_CLKSEL[3-0] CLK_SEL or TIMER0_POTIMERPWM in <i>Control Module (CTRL_MMR)</i>
		HFOSC1_CLKOUT	HFOSC1	
		MAIN_PLL0_HSDIV1_CLKO UT	PLL0_HSDIV1	
		CLK_12M_RC	WKUP_RC_OSC_12M	
		MAIN_PLL3_HSDIV3_CLKO UT	PLL3_HSDIV3	
		MCU_EXT_REFCLK0	I/O pin	
		EXT_REFCLK1	I/O pin	
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CPTS_RFT_CLK	I/O pin	
		MAIN_PLL1_HSDIV3_CLKO UT	PLL1_HSDIV3	
		MAIN_PLL2_HSDIV6_CLKO UT	PLL4_HSDIV6	
		MAIN_PLL4_HSDIV2_CLKO UT	PLL4_HSDIV2	
		CPTS_GENF2	NAVSS0_CPTS0	
		CPTS_GENF3	NAVSS0_CPTS0	
		CPTS0_GENF0	CPSW0	
		MAIN_PLL15_HSDIV2_CLK OUT	PLL15_HSDIV2	
TIMER2	TIMER2_ICLK	MAIN_SYSCCLK0/4	PLLCTRL0	TIMER2 Interface Clock

**Table 12-1585. TIMER Clocks and Resets (continued)**

TIMER2_FCLK		WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	TIMER2 Functional Clock. Output of multiplexor TIMERCLK2 MUX controlled by CTRLMMR_TIMER2_CLKSEL[3-0] CLK_SEL in <i>Control Module (CTRL_MMR)</i>
		HFOSC1_CLKOUT	HFOSC1	
		MAIN_PLL0_HSDIV1_CLKO UT	PLL0_HSDIV1	
		CLK_12M_RC	WKUP_RC_OSC_12M	
		MAIN_PLL3_HSDIV3_CLKO UT	PLL3_HSDIV3	
		MCU_EXT_REFCLK0	I/O pin	
		EXT_REFCLK1	I/O pin	
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CPTS_RFT_CLK	I/O pin	
		MAIN_PLL1_HSDIV3_CLKO UT	PLL1_HSDIV3	
		MAIN_PLL2_HSDIV6_CLKO UT	PLL4_HSDIV6	
		MAIN_PLL4_HSDIV2_CLKO UT	PLL4_HSDIV2	
		CPTS_GENF2	NAVSS0_CPTS0	
		CPTS_GENF3	NAVSS0_CPTS0	
		CPTS0_GENF0	CPSW0	
		MAIN_PLL15_HSDIV2_CLK OUT	PLL15_HSDIV2	
TIMER3	TIMER3_ICLK	MAIN_SYSCCLK0/4	PLLCTRL0	TIMER3 Interface Clock
	TIMER3_FCLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	TIMER3 Functional Clock. Output of multiplexor TIMERCLK3 MUX controlled by CTRLMMR_TIMER3_CLKSEL[3-0] CLK_SEL or TIMER2_POTIMERPWM in <i>Control Module (CTRL_MMR)</i>
		HFOSC1_CLKOUT	HFOSC1	
		MAIN_PLL0_HSDIV1_CLKO UT	PLL0_HSDIV1	
		CLK_12M_RC	WKUP_RC_OSC_12M	
		MAIN_PLL3_HSDIV3_CLKO UT	PLL3_HSDIV3	
		MCU_EXT_REFCLK0	I/O pin	
		EXT_REFCLK1	I/O pin	
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CPTS_RFT_CLK	I/O pin	
		MAIN_PLL1_HSDIV3_CLKO UT	PLL1_HSDIV3	
		MAIN_PLL2_HSDIV6_CLKO UT	PLL4_HSDIV6	
		MAIN_PLL4_HSDIV2_CLKO UT	PLL4_HSDIV2	
		CPTS_GENF2	NAVSS0_CPTS0	
		CPTS_GENF3	NAVSS0_CPTS0	
		CPTS0_GENF0	CPSW0	
		MAIN_PLL15_HSDIV2_CLK OUT	PLL15_HSDIV2	
TIMER4	TIMER4_ICLK	MAIN_SYSCCLK0/4	PLLCTRL0	TIMER4 Interface Clock

**Table 12-1585. TIMER Clocks and Resets (continued)**

TIMER4_FCLK		WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	TIMER4 Functional Clock. Output of multiplexor TIMERCLK4 MUX controlled by CTRLMMR_TIMER4_CLKSEL[3-0] CLK_SEL in <i>Control Module (CTRL_MMR)</i>
		HFOSC1_CLKOUT	HFOSC1	
		MAIN_PLL0_HSDIV1_CLKO UT	PLL0_HSDIV1	
		CLK_12M_RC	WKUP_RC_OSC_12M	
		MAIN_PLL3_HSDIV3_CLKO UT	PLL3_HSDIV3	
		MCU_EXT_REFCLK0	I/O pin	
		EXT_REFCLK1	I/O pin	
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CPTS_RFT_CLK	I/O pin	
		MAIN_PLL1_HSDIV3_CLKO UT	PLL1_HSDIV3	
		MAIN_PLL2_HSDIV6_CLKO UT	PLL4_HSDIV6	
		MAIN_PLL4_HSDIV2_CLKO UT	PLL4_HSDIV2	
		CPTS_GENF2	NAVSS0_CPTS0	
		CPTS_GENF3	NAVSS0_CPTS0	
		CPTS0_GENF0	CPSW0	
		MAIN_PLL15_HSDIV2_CLK OUT	PLL15_HSDIV2	
TIMER5	TIMER5_ICLK	MAIN_SYSCCLK0/4	PLLCTRL0	TIMER5 Interface Clock
	TIMER5_FCLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	TIMER5 Functional Clock. Output of multiplexor TIMERCLK5 MUX controlled by CTRLMMR_TIMER5_CLKSEL[3-0] CLK_SEL or TIMER4_POTIMERPWM in <i>Control Module (CTRL_MMR)</i>
		HFOSC1_CLKOUT	HFOSC1	
		MAIN_PLL0_HSDIV1_CLKO UT	PLL0_HSDIV1	
		CLK_12M_RC	WKUP_RC_OSC_12M	
		MAIN_PLL3_HSDIV3_CLKO UT	PLL3_HSDIV3	
		MCU_EXT_REFCLK0	I/O pin	
		EXT_REFCLK1	I/O pin	
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CPTS_RFT_CLK	I/O pin	
		MAIN_PLL1_HSDIV3_CLKO UT	PLL1_HSDIV3	
		MAIN_PLL2_HSDIV6_CLKO UT	PLL4_HSDIV6	
		MAIN_PLL4_HSDIV2_CLKO UT	PLL4_HSDIV2	
		CPTS_GENF2	NAVSS0_CPTS0	
		CPTS_GENF3	NAVSS0_CPTS0	
		CPTS0_GENF0	CPSW0	
		MAIN_PLL15_HSDIV2_CLK OUT	PLL15_HSDIV2	
TIMER6	TIMER6_ICLK	MAIN_SYSCCLK0/4	PLLCTRL0	TIMER6 Interface Clock

**Table 12-1585. TIMER Clocks and Resets (continued)**

TIMER6_FCLK		WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	TIMER6 Functional Clock. Output of multiplexor TIMERCLK6 MUX controlled by CTRLMMR_TIMER6_CLKSEL[3-0] CLK_SEL in <i>Control Module (CTRL_MMR)</i>
		HFOSC1_CLKOUT	HFOSC1	
		MAIN_PLL0_HSDIV1_CLKO UT	PLL0_HSDIV1	
		CLK_12M_RC	WKUP_RC_OSC_12M	
		MAIN_PLL3_HSDIV3_CLKO UT	PLL3_HSDIV3	
		MCU_EXT_REFCLK0	I/O pin	
		EXT_REFCLK1	I/O pin	
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CPTS_RFT_CLK	I/O pin	
		MAIN_PLL1_HSDIV3_CLKO UT	PLL1_HSDIV3	
		MAIN_PLL2_HSDIV6_CLKO UT	PLL4_HSDIV6	
		MAIN_PLL4_HSDIV2_CLKO UT	PLL4_HSDIV2	
		CPTS_GENF2	NAVSS0_CPTS0	
		CPTS_GENF3	NAVSS0_CPTS0	
		CPTS0_GENF0	CPSW0	
		MAIN_PLL15_HSDIV2_CLK OUT	PLL15_HSDIV2	
TIMER7	TIMER7_ICLK	MAIN_SYSCCLK0/4	PLLCTRL0	TIMER7 Interface Clock
	TIMER7_FCLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	TIMER7 Functional Clock. Output of multiplexor TIMERCLK7 MUX controlled by CTRLMMR_TIMER7_CLKSEL[3-0] CLK_SEL or TIMER6_POTIMERPWM in <i>Control Module (CTRL_MMR)</i>
		HFOSC1_CLKOUT	HFOSC1	
		MAIN_PLL0_HSDIV1_CLKO UT	PLL0_HSDIV1	
		CLK_12M_RC	WKUP_RC_OSC_12M	
		MAIN_PLL3_HSDIV3_CLKO UT	PLL3_HSDIV3	
		MCU_EXT_REFCLK0	I/O pin	
		EXT_REFCLK1	I/O pin	
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CPTS_RFT_CLK	I/O pin	
		MAIN_PLL1_HSDIV3_CLKO UT	PLL1_HSDIV3	
		MAIN_PLL2_HSDIV6_CLKO UT	PLL4_HSDIV6	
		MAIN_PLL4_HSDIV2_CLKO UT	PLL4_HSDIV2	
		CPTS_GENF2	NAVSS0_CPTS0	
		CPTS_GENF3	NAVSS0_CPTS0	
		CPTS0_GENF0	CPSW0	
		MAIN_PLL15_HSDIV2_CLK OUT	PLL15_HSDIV2	
TIMER8	TIMER8_ICLK	MAIN_SYSCCLK0/4	PLLCTRL0	TIMER8 Interface Clock



**Table 12-1585. TIMER Clocks and Resets (continued)**

TIMER8_FCLK		WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	TIMER8 Functional Clock. Output of multiplexor TIMERCLK8 MUX controlled by CTRLMMR_TIMER8_CLKSEL[3-0] CLK_SEL in <i>Control Module (CTRL_MMR)</i>
		HFOSC1_CLKOUT	HFOSC1	
		MAIN_PLL0_HSDIV1_CLKO UT	PLL0_HSDIV1	
		CLK_12M_RC	WKUP_RC_OSC_12M	
		MAIN_PLL3_HSDIV3_CLKO UT	PLL3_HSDIV3	
		MCU_EXT_REFCLK0	I/O pin	
		EXT_REFCLK1	I/O pin	
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CPTS_RFT_CLK	I/O pin	
		MAIN_PLL1_HSDIV3_CLKO UT	PLL1_HSDIV3	
		MAIN_PLL2_HSDIV6_CLKO UT	PLL4_HSDIV6	
		MAIN_PLL4_HSDIV2_CLKO UT	PLL4_HSDIV2	
		CPTS_GENF2	NAVSS0_CPTS0	
		CPTS_GENF3	NAVSS0_CPTS0	
		CPTS0_GENF0	CPSW0	
		MAIN_PLL15_HSDIV2_CLK OUT	PLL15_HSDIV2	
TIMER9	TIMER9_ICLK	MAIN_SYSCCLK0/4	PLLCTRL0	TIMER9 Interface Clock
	TIMER9_FCLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	TIMER9 Functional Clock. Output of multiplexor TIMERCLK9 MUX controlled by CTRLMMR_TIMER9_CLKSEL[3-0] CLK_SEL or TIMER8_POTIMERPWM in <i>Control Module (CTRL_MMR)</i>
		HFOSC1_CLKOUT	HFOSC1	
		MAIN_PLL0_HSDIV1_CLKO UT	PLL0_HSDIV1	
		CLK_12M_RC	WKUP_RC_OSC_12M	
		MAIN_PLL3_HSDIV3_CLKO UT	PLL3_HSDIV3	
		MCU_EXT_REFCLK0	I/O pin	
		EXT_REFCLK1	I/O pin	
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CPTS_RFT_CLK	I/O pin	
		MAIN_PLL1_HSDIV3_CLKO UT	PLL1_HSDIV3	
		MAIN_PLL2_HSDIV6_CLKO UT	PLL4_HSDIV6	
		MAIN_PLL4_HSDIV2_CLKO UT	PLL4_HSDIV2	
		CPTS_GENF2	NAVSS0_CPTS0	
		CPTS_GENF3	NAVSS0_CPTS0	
		CPTS0_GENF0	CPSW0	
		MAIN_PLL15_HSDIV2_CLK OUT	PLL15_HSDIV2	
TIMER10	TIMER10_ICLK	MAIN_SYSCCLK0/4	PLLCTRL0	TIMER10 Interface Clock

**Table 12-1585. TIMER Clocks and Resets (continued)**

TIMER10_FCLK		WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	TIMER10 Functional Clock. Output of multiplexor TIMERCLK10 MUX controlled by CTRLMMR_TIMER10_CLKSEL[3-0] CLK_SEL in Control Module (CTRL_MMR)
		HFOSC1_CLKOUT	HFOSC1	
		MAIN_PLL0_HSDIV1_CLKO UT	PLL0_HSDIV1	
		CLK_12M_RC	WKUP_RC_OSC_12M	
		MAIN_PLL3_HSDIV3_CLKO UT	PLL3_HSDIV3	
		MCU_EXT_REFCLK0	I/O pin	
		EXT_REFCLK1	I/O pin	
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CPTS_RFT_CLK	I/O pin	
		MAIN_PLL1_HSDIV3_CLKO UT	PLL1_HSDIV3	
		MAIN_PLL2_HSDIV6_CLKO UT	PLL4_HSDIV6	
		MAIN_PLL4_HSDIV2_CLKO UT	PLL4_HSDIV2	
		CPTS_GENF2	NAVSS0_CPTS0	
		CPTS_GENF3	NAVSS0_CPTS0	
		CPTS0_GENF0	CPSW0	
		MAIN_PLL15_HSDIV2_CLK OUT	PLL15_HSDIV2	
TIMER11	TIMER11_ICLK	MAIN_SYSCCLK0/4	PLLCTRL0	TIMER11 Interface Clock
	TIMER11_FCLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	TIMER11 Functional Clock. Output of multiplexor TIMERCLK11 MUX controlled by CTRLMMR_TIMER11_CLKSEL[3-0] CLK_SEL or TIMER10_POTIMERPWM in Control Module (CTRL_MMR)
		HFOSC1_CLKOUT	HFOSC1	
		MAIN_PLL0_HSDIV1_CLKO UT	PLL0_HSDIV1	
		CLK_12M_RC	WKUP_RC_OSC_12M	
		MAIN_PLL3_HSDIV3_CLKO UT	PLL3_HSDIV3	
		MCU_EXT_REFCLK0	I/O pin	
		EXT_REFCLK1	I/O pin	
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CPTS_RFT_CLK	I/O pin	
		MAIN_PLL1_HSDIV3_CLKO UT	PLL1_HSDIV3	
		MAIN_PLL2_HSDIV6_CLKO UT	PLL4_HSDIV6	
		MAIN_PLL4_HSDIV2_CLKO UT	PLL4_HSDIV2	
		CPTS_GENF2	NAVSS0_CPTS0	
		CPTS_GENF3	NAVSS0_CPTS0	
		CPTS0_GENF0	CPSW0	
		MAIN_PLL15_HSDIV2_CLK OUT	PLL15_HSDIV2	
TIMER12	TIMER12_ICLK	MAIN_SYSCCLK0/4	PLLCTRL0	TIMER12 Interface Clock

**Table 12-1585. TIMER Clocks and Resets (continued)**

TIMER12	FCLK	WKP_HFOSC0_CLKOUT	WKP_HFOSC0	TIMER12 Functional Clock. Output of multiplexor TIMERCLK12 MUX controlled by CTRLMMR_TIMER12_CLKSEL[3-0] CLK_SEL in Control Module (CTRL_MMR)
		HFOSC1_CLKOUT	HFOSC1	
		MAIN_PLL0_HSDIV1_CLKO	PLL0_HSDIV1	
		UT		
		CLK_12M_RC	WKP_RC_OSC_12M	
		MAIN_PLL3_HSDIV3_CLKO	PLL3_HSDIV3	
		UT		
		MCU_EXT_REFCLK0	I/O pin	
		EXT_REFCLK1	I/O pin	
		WKP_LFXOSC0_CLKOUT	WKP_LFXOSC0	
		CPTS_RFT_CLK	I/O pin	
		MAIN_PLL1_HSDIV3_CLKO	PLL1_HSDIV3	
		UT		
		MAIN_PLL2_HSDIV6_CLKO	PLL4_HSDIV6	
		UT		
		MAIN_PLL4_HSDIV2_CLKO	PLL4_HSDIV2	
		UT		
		CPTS_GENF2	NAVSS0_CPTS0	
		CPTS_GENF3	NAVSS0_CPTS0	
		CPTS0_GENF0	CPSW0	
		MAIN_PLL15_HSDIV2_CLK	PLL15_HSDIV2	
		OUT		
TIMER13	ICLK	MAIN_SYSCCLK0/4	PLLCTRL0	TIMER13 Interface Clock
	FCLK	WKP_HFOSC0_CLKOUT	WKP_HFOSC0	TIMER13 Functional Clock. Output of multiplexor TIMERCLK13 MUX controlled by CTRLMMR_TIMER13_CLKSEL[3-0] CLK_SEL or TIMER12_POTIMERPWM in Control Module (CTRL_MMR)
		HFOSC1_CLKOUT	HFOSC1	
		MAIN_PLL0_HSDIV1_CLKO	PLL0_HSDIV1	
		UT		
		CLK_12M_RC	WKP_RC_OSC_12M	
		MAIN_PLL3_HSDIV3_CLKO	PLL3_HSDIV3	
		UT		
		MCU_EXT_REFCLK0	I/O pin	
		EXT_REFCLK1	I/O pin	
		WKP_LFXOSC0_CLKOUT	WKP_LFXOSC0	
		CPTS_RFT_CLK	I/O pin	
		MAIN_PLL1_HSDIV3_CLKO	PLL1_HSDIV3	
		UT		
		MAIN_PLL2_HSDIV6_CLKO	PLL4_HSDIV6	
		UT		
		MAIN_PLL4_HSDIV2_CLKO	PLL4_HSDIV2	
		UT		
		CPTS_GENF2	NAVSS0_CPTS0	
		CPTS_GENF3	NAVSS0_CPTS0	
		CPTS0_GENF0	CPSW0	
		MAIN_PLL15_HSDIV2_CLK	PLL15_HSDIV2	
		OUT		
TIMER14	ICLK	MAIN_SYSCCLK0/4	PLLCTRL0	TIMER14 Interface Clock

**Table 12-1585. TIMER Clocks and Resets (continued)**

TIMER14_FCLK		WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	TIMER14 Functional Clock. Output of multiplexor TIMERCLK14 MUX controlled by CTRLMMR_TIMER14_CLKSEL[3-0] CLK_SEL in Control Module (CTRL_MMR)
		HFOSC1_CLKOUT	HFOSC1	
		MAIN_PLL0_HSDIV1_CLKO UT	PLL0_HSDIV1	
		CLK_12M_RC	WKUP_RC_OSC_12M	
		MAIN_PLL3_HSDIV3_CLKO UT	PLL3_HSDIV3	
		MCU_EXT_REFCLK0	I/O pin	
		EXT_REFCLK1	I/O pin	
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CPTS_RFT_CLK	I/O pin	
		MAIN_PLL1_HSDIV3_CLKO UT	PLL1_HSDIV3	
		MAIN_PLL2_HSDIV6_CLKO UT	PLL4_HSDIV6	
		MAIN_PLL4_HSDIV2_CLKO UT	PLL4_HSDIV2	
		CPTS_GENF2	NAVSS0_CPTS0	
		CPTS_GENF3	NAVSS0_CPTS0	
		CPTS0_GENF0	CPSW0	
		MAIN_PLL15_HSDIV2_CLK OUT	PLL15_HSDIV2	
TIMER15	TIMER15_ICLK	MAIN_SYSCCLK0/4	PLLCTRL0	TIMER15 Interface Clock
	TIMER15_FCLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	TIMER15 Functional Clock. Output of multiplexor TIMERCLK15 MUX controlled by CTRLMMR_TIMER15_CLKSEL[3-0] CLK_SEL or TIMER14_POTIMERPWM in Control Module (CTRL_MMR)
		HFOSC1_CLKOUT	HFOSC1	
		MAIN_PLL0_HSDIV1_CLKO UT	PLL0_HSDIV1	
		CLK_12M_RC	WKUP_RC_OSC_12M	
		MAIN_PLL3_HSDIV3_CLKO UT	PLL3_HSDIV3	
		MCU_EXT_REFCLK0	I/O pin	
		EXT_REFCLK1	I/O pin	
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CPTS_RFT_CLK	I/O pin	
		MAIN_PLL1_HSDIV3_CLKO UT	PLL1_HSDIV3	
		MAIN_PLL2_HSDIV6_CLKO UT	PLL4_HSDIV6	
		MAIN_PLL4_HSDIV2_CLKO UT	PLL4_HSDIV2	
		CPTS_GENF2	NAVSS0_CPTS0	
		CPTS_GENF3	NAVSS0_CPTS0	
		CPTS0_GENF0	CPSW0	
		MAIN_PLL15_HSDIV2_CLK OUT	PLL15_HSDIV2	
TIMER16	TIMER16_ICLK	MAIN_SYSCCLK0/4	PLLCTRL0	TIMER16 Interface Clock

**Table 12-1585. TIMER Clocks and Resets (continued)**

TIMER16_FCLK		WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	TIMER16 Functional Clock. Output of multiplexor TIMERCLK16 MUX controlled by CTRLMMR_TIMER16_CLKSEL[3-0] CLK_SEL in Control Module (CTRL_MMR)
		HFOSC1_CLKOUT	HFOSC1	
		MAIN_PLL0_HSDIV1_CLKO UT	PLL0_HSDIV1	
		CLK_12M_RC	WKUP_RC_OSC_12M	
		MAIN_PLL3_HSDIV3_CLKO UT	PLL3_HSDIV3	
		MCU_EXT_REFCLK0	I/O pin	
		EXT_REFCLK1	I/O pin	
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CPTS_RFT_CLK	I/O pin	
		MAIN_PLL1_HSDIV3_CLKO UT	PLL1_HSDIV3	
		MAIN_PLL2_HSDIV6_CLKO UT	PLL4_HSDIV6	
		MAIN_PLL4_HSDIV2_CLKO UT	PLL4_HSDIV2	
		CPTS_GENF2	NAVSS0_CPTS0	
		CPTS_GENF3	NAVSS0_CPTS0	
		CPTS0_GENF0	CPSW0	
		MAIN_PLL15_HSDIV2_CLK OUT	PLL15_HSDIV2	
TIMER17	TIMER17_ICLK	MAIN_SYSCCLK0/4	PLLCTRL0	TIMER17 Interface Clock
	TIMER17_FCLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	TIMER17 Functional Clock. Output of multiplexor TIMERCLK17 MUX controlled by CTRLMMR_TIMER17_CLKSEL[3-0] CLK_SEL or TIMER16_POTIMERPWM in Control Module (CTRL_MMR)
		HFOSC1_CLKOUT	HFOSC1	
		MAIN_PLL0_HSDIV1_CLKO UT	PLL0_HSDIV1	
		CLK_12M_RC	WKUP_RC_OSC_12M	
		MAIN_PLL3_HSDIV3_CLKO UT	PLL3_HSDIV3	
		MCU_EXT_REFCLK0	I/O pin	
		EXT_REFCLK1	I/O pin	
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CPTS_RFT_CLK	I/O pin	
		MAIN_PLL1_HSDIV3_CLKO UT	PLL1_HSDIV3	
		MAIN_PLL2_HSDIV6_CLKO UT	PLL4_HSDIV6	
		MAIN_PLL4_HSDIV2_CLKO UT	PLL4_HSDIV2	
		CPTS_GENF2	NAVSS0_CPTS0	
		CPTS_GENF3	NAVSS0_CPTS0	
		CPTS0_GENF0	CPSW0	
		MAIN_PLL15_HSDIV2_CLK OUT	PLL15_HSDIV2	
TIMER18	TIMER18_ICLK	MAIN_SYSCCLK0/4	PLLCTRL0	TIMER18 Interface Clock

**Table 12-1585. TIMER Clocks and Resets (continued)**

TIMER18_FCLK		WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	TIMER18 Functional Clock. Output of multiplexor TIMERCLK18 MUX controlled by CTRLMMR_TIMER18_CLKSEL[3-0] CLK_SEL in <i>Control Module (CTRL_MMR)</i>
		HFOSC1_CLKOUT	HFOSC1	
		MAIN_PLL0_HSDIV1_CLKO UT	PLL0_HSDIV1	
		CLK_12M_RC	WKUP_RC_OSC_12M	
		MAIN_PLL3_HSDIV3_CLKO UT	PLL3_HSDIV3	
		MCU_EXT_REFCLK0	I/O pin	
		EXT_REFCLK1	I/O pin	
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CPTS_RFT_CLK	I/O pin	
		MAIN_PLL1_HSDIV3_CLKO UT	PLL1_HSDIV3	
		MAIN_PLL2_HSDIV6_CLKO UT	PLL4_HSDIV6	
		MAIN_PLL4_HSDIV2_CLKO UT	PLL4_HSDIV2	
		CPTS_GENF2	NAVSS0_CPTS0	
		CPTS_GENF3	NAVSS0_CPTS0	
		CPTS0_GENF0	CPSW0	
		MAIN_PLL15_HSDIV2_CLK OUT	PLL15_HSDIV2	
TIMER19	TIMER19_ICLK	MAIN_SYSCLK0/4	PLLCTRL0	TIMER19 Interface Clock
	TIMER19_FCLK	WKUP_HFOSC0_CLKOUT	WKUP_HFOSC0	TIMER19 Functional Clock. Output of multiplexor TIMERCLK19 MUX controlled by CTRLMMR_TIMER19_CLKSEL[3-0] CLK_SEL or TIMER18_POTIMERPWM in <i>Control Module (CTRL_MMR)</i>
		HFOSC1_CLKOUT	HFOSC1	
		MAIN_PLL0_HSDIV1_CLKO UT	PLL0_HSDIV1	
		CLK_12M_RC	WKUP_RC_OSC_12M	
		MAIN_PLL3_HSDIV3_CLKO UT	PLL3_HSDIV3	
		MCU_EXT_REFCLK0	I/O pin	
		EXT_REFCLK1	I/O pin	
		WKUP_LFXOSC0_CLKOUT	WKUP_LFXOSC0	
		CPTS_RFT_CLK	I/O pin	
		MAIN_PLL1_HSDIV3_CLKO UT	PLL1_HSDIV3	
		MAIN_PLL2_HSDIV6_CLKO UT	PLL4_HSDIV6	
		MAIN_PLL4_HSDIV2_CLKO UT	PLL4_HSDIV2	
		CPTS_GENF2	NAVSS0_CPTS0	
		CPTS_GENF3	NAVSS0_CPTS0	
		CPTS0_GENF0	CPSW0	
MAIN_PLL15_HSDIV2_CLK OUT	PLL15_HSDIV2			

Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
TIMER0	TIMER0_RST	MOD_G_RST	LPSC70	Asynchronous Reset to TIMER0
TIMER1	TIMER1_RST	MOD_G_RST	LPSC71	Asynchronous Reset to TIMER1
TIMER2	TIMER2_RST	MOD_G_RST	LPSC72	Asynchronous Reset to TIMER2

**Table 12-1585. TIMER Clocks and Resets (continued)**

TIMER3	TIMER3_RST	MOD_G_RST	LPSC73	Asynchronous Reset to TIMER3
TIMER4	TIMER4_RST	MOD_G_RST	LPSC0	Asynchronous Reset to TIMER4
TIMER5	TIMER5_RST	MOD_G_RST	LPSC0	Asynchronous Reset to TIMER5
TIMER6	TIMER6_RST	MOD_G_RST	LPSC0	Asynchronous Reset to TIMER6
TIMER7	TIMER7_RST	MOD_G_RST	LPSC0	Asynchronous Reset to TIMER7
TIMER8	TIMER8_RST	MOD_G_RST	LPSC0	Asynchronous Reset to TIMER8
TIMER9	TIMER9_RST	MOD_G_RST	LPSC0	Asynchronous Reset to TIMER9
TIMER10	TIMER10_RST	MOD_G_RST	LPSC0	Asynchronous Reset to TIMER10
TIMER11	TIMER11_RST	MOD_G_RST	LPSC0	Asynchronous Reset to TIMER11
TIMER12	TIMER12_RST	MOD_G_RST	LPSC0	Asynchronous Reset to TIMER12
TIMER13	TIMER13_RST	MOD_G_RST	LPSC0	Asynchronous Reset to TIMER13
TIMER14	TIMER14_RST	MOD_G_RST	LPSC0	Asynchronous Reset to TIMER14
TIMER15	TIMER15_RST	MOD_G_RST	LPSC0	Asynchronous Reset to TIMER15
TIMER16	TIMER16_RST	MOD_G_RST	LPSC0	Asynchronous Reset to TIMER16
TIMER17	TIMER17_RST	MOD_G_RST	LPSC0	Asynchronous Reset to TIMER17
TIMER18	TIMER18_RST	MOD_G_RST	LPSC0	Asynchronous Reset to TIMER18
TIMER19	TIMER19_RST	MOD_G_RST	LPSC0	Asynchronous Reset to TIMER19

**Table 12-1586. TIMER Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
TIMER0	TIMER0_INTR_PEND_0	C66SS0_INTRTR0_IN_0	C66SS0_INTRTR0	TIMER0 Interrupt Request	Level
		C66SS1_INTRTR0_IN_0	C66SS1_INTRTR0	TIMER0 Interrupt Request	Level
		GIC500_SPI_IN_256	COMPUTE_CLUSTER0	TIMER0 Interrupt Request	Level
		MAIN2MCU_LVL_INTRTR0_IN_108	MAIN2MCU_LVL_INTRTR0	TIMER0 Interrupt Request	Level
		R5FSS0_INTRTR0_IN_239	R5FSS0_INTRTR0	TIMER0 Interrupt Request	Level
		R5FSS1_INTRTR0_IN_239	R5FSS1_INTRTR0	TIMER0 Interrupt Request	Level
TIMER1	TIMER1_INTR_PEND_0	C66SS0_INTRTR0_IN_1	C66SS0_INTRTR0	TIMER1 Interrupt Request	Level
		C66SS1_INTRTR0_IN_1	C66SS1_INTRTR0	TIMER1 Interrupt Request	Level
		GIC500_SPI_IN_257	COMPUTE_CLUSTER0	TIMER1 Interrupt Request	Level
		MAIN2MCU_LVL_INTRTR0_IN_109	MAIN2MCU_LVL_INTRTR0	TIMER1 Interrupt Request	Level
		R5FSS0_INTRTR0_IN_240	R5FSS0_INTRTR0	TIMER1 Interrupt Request	Level
		R5FSS1_INTRTR0_IN_240	R5FSS1_INTRTR0	TIMER1 Interrupt Request	Level
TIMER2	TIMER2_INTR_PEND_0	C66SS0_INTRTR0_IN_2	C66SS0_INTRTR0	TIMER2 Interrupt Request	Level
		C66SS1_INTRTR0_IN_2	C66SS1_INTRTR0	TIMER2 Interrupt Request	Level
		GIC500_SPI_IN_258	COMPUTE_CLUSTER0	TIMER2 Interrupt Request	Level
		MAIN2MCU_LVL_INTRTR0_IN_110	MAIN2MCU_LVL_INTRTR0	TIMER2 Interrupt Request	Level
		R5FSS0_INTRTR0_IN_241	R5FSS0_INTRTR0	TIMER2 Interrupt Request	Level
		R5FSS1_INTRTR0_IN_241	R5FSS1_INTRTR0	TIMER2 Interrupt Request	Level
TIMER3	TIMER3_INTR_PEND_0	C66SS0_INTRTR0_IN_3	C66SS0_INTRTR0	TIMER3 Interrupt Request	Level
		C66SS1_INTRTR0_IN_3	C66SS1_INTRTR0	TIMER3 Interrupt Request	Level
		GIC500_SPI_IN_259	COMPUTE_CLUSTER0	TIMER3 Interrupt Request	Level
		MAIN2MCU_LVL_INTRTR0_IN_111	MAIN2MCU_LVL_INTRTR0	TIMER3 Interrupt Request	Level
		R5FSS0_INTRTR0_IN_242	R5FSS0_INTRTR0	TIMER3 Interrupt Request	Level
		R5FSS1_INTRTR0_IN_242	R5FSS1_INTRTR0	TIMER3 Interrupt Request	Level
TIMER4	TIMER4_INTR_PEND_0	C66SS0_INTRTR0_IN_4	C66SS0_INTRTR0	TIMER4 Interrupt Request	Level
		C66SS1_INTRTR0_IN_4	C66SS1_INTRTR0	TIMER4 Interrupt Request	Level



**Table 12-1586. TIMER Hardware Requests (continued)**

TIMER5	TIMER5_INTR_PEND_0	GIC500_SPI_IN_260	COMPUTE_CLUSTER0	TIMER4 Interrupt Request	Level
		MAIN2MCU_LVL_INTRTR0_I_N_112	MAIN2MCU_LVL_INTRTR0	TIMER4 Interrupt Request	Level
		R5FSS0_INTRTR0_IN_243	R5FSS0_INTRTR0	TIMER4 Interrupt Request	Level
		R5FSS1_INTRTR0_IN_243	R5FSS1_INTRTR0	TIMER4 Interrupt Request	Level
		C66SS0_INTRTR0_IN_5	C66SS0_INTRTR0	TIMER5 Interrupt Request	Level
		C66SS1_INTRTR0_IN_5	C66SS1_INTRTR0	TIMER5 Interrupt Request	Level
		GIC500_SPI_IN_261	COMPUTE_CLUSTER0	TIMER5 Interrupt Request	Level
		MAIN2MCU_LVL_INTRTR0_I_N_113	MAIN2MCU_LVL_INTRTR0	TIMER5 Interrupt Request	Level
		R5FSS0_INTRTR0_IN_244	R5FSS0_INTRTR0	TIMER5 Interrupt Request	Level
		R5FSS1_INTRTR0_IN_244	R5FSS1_INTRTR0	TIMER5 Interrupt Request	Level
		C66SS0_INTRTR0_IN_6	C66SS0_INTRTR0	TIMER6 Interrupt Request	Level
		C66SS1_INTRTR0_IN_6	C66SS1_INTRTR0	TIMER6 Interrupt Request	Level
TIMER6	TIMER6_INTR_PEND_0	GIC500_SPI_IN_262	COMPUTE_CLUSTER0	TIMER6 Interrupt Request	Level
		MAIN2MCU_LVL_INTRTR0_I_N_114	MAIN2MCU_LVL_INTRTR0	TIMER6 Interrupt Request	Level
		R5FSS0_INTRTR0_IN_245	R5FSS0_INTRTR0	TIMER6 Interrupt Request	Level
		R5FSS1_INTRTR0_IN_245	R5FSS1_INTRTR0	TIMER6 Interrupt Request	Level
		C66SS0_INTRTR0_IN_7	C66SS0_INTRTR0	TIMER7 Interrupt Request	Level
		C66SS1_INTRTR0_IN_7	C66SS1_INTRTR0	TIMER7 Interrupt Request	Level
TIMER7	TIMER7_INTR_PEND_0	GIC500_SPI_IN_263	COMPUTE_CLUSTER0	TIMER7 Interrupt Request	Level
		MAIN2MCU_LVL_INTRTR0_I_N_115	MAIN2MCU_LVL_INTRTR0	TIMER7 Interrupt Request	Level
		R5FSS0_INTRTR0_IN_246	R5FSS0_INTRTR0	TIMER7 Interrupt Request	Level
		R5FSS1_INTRTR0_IN_246	R5FSS1_INTRTR0	TIMER7 Interrupt Request	Level
		C66SS0_INTRTR0_IN_8	C66SS0_INTRTR0	TIMER8 Interrupt Request	Level
		C66SS1_INTRTR0_IN_8	C66SS1_INTRTR0	TIMER8 Interrupt Request	Level
TIMER8	TIMER8_INTR_PEND_0	GIC500_SPI_IN_264	COMPUTE_CLUSTER0	TIMER8 Interrupt Request	Level
		MAIN2MCU_LVL_INTRTR0_I_N_116	MAIN2MCU_LVL_INTRTR0	TIMER8 Interrupt Request	Level
		R5FSS0_INTRTR0_IN_247	R5FSS0_INTRTR0	TIMER8 Interrupt Request	Level

**Table 12-1586. TIMER Hardware Requests (continued)**

		R5FSS1_INTRTR0_IN_247	R5FSS1_INTRTR0	TIMER8 Interrupt Request	Level
TIMER9	TIMER9_INTR_PEND_0	C66SS0_INTRTR0_IN_9	C66SS0_INTRTR0	TIMER9 Interrupt Request	Level
		C66SS1_INTRTR0_IN_9	C66SS1_INTRTR0	TIMER9 Interrupt Request	Level
		GIC500_SPI_IN_265	COMPUTE_CLUSTER0	TIMER9 Interrupt Request	Level
		MAIN2MCU_LVL_INTRTR0_IN_117	MAIN2MCU_LVL_INTRTR0	TIMER9 Interrupt Request	Level
		R5FSS0_INTRTR0_IN_248	R5FSS0_INTRTR0	TIMER9 Interrupt Request	Level
		R5FSS1_INTRTR0_IN_248	R5FSS1_INTRTR0	TIMER9 Interrupt Request	Level
TIMER10	TIMER10_INTR_PEND_0	C66SS0_INTRTR0_IN_10	C66SS0_INTRTR0	TIMER10 Interrupt Request	Level
		C66SS1_INTRTR0_IN_10	C66SS1_INTRTR0	TIMER10 Interrupt Request	Level
		GIC500_SPI_IN_266	COMPUTE_CLUSTER0	TIMER10 Interrupt Request	Level
		MAIN2MCU_LVL_INTRTR0_IN_118	MAIN2MCU_LVL_INTRTR0	TIMER10 Interrupt Request	Level
		R5FSS0_INTRTR0_IN_249	R5FSS0_INTRTR0	TIMER10 Interrupt Request	Level
		R5FSS1_INTRTR0_IN_249	R5FSS1_INTRTR0	TIMER10 Interrupt Request	Level
TIMER11	TIMER11_INTR_PEND_0	C66SS0_INTRTR0_IN_11	C66SS0_INTRTR0	TIMER11 Interrupt Request	Level
		C66SS1_INTRTR0_IN_11	C66SS1_INTRTR0	TIMER11 Interrupt Request	Level
		GIC500_SPI_IN_267	COMPUTE_CLUSTER0	TIMER11 Interrupt Request	Level
		MAIN2MCU_LVL_INTRTR0_IN_119	MAIN2MCU_LVL_INTRTR0	TIMER11 Interrupt Request	Level
		R5FSS0_INTRTR0_IN_250	R5FSS0_INTRTR0	TIMER11 Interrupt Request	Level
		R5FSS1_INTRTR0_IN_250	R5FSS1_INTRTR0	TIMER11 Interrupt Request	Level
TIMER12	TIMER12_INTR_PEND_0	C66SS0_INTRTR0_IN_303	C66SS0_INTRTR0	TIMER12 Interrupt Request	Level
		C66SS1_INTRTR0_IN_303	C66SS1_INTRTR0	TIMER12 Interrupt Request	Level
		GIC500_SPI_IN_268	COMPUTE_CLUSTER0	TIMER12 Interrupt Request	Level
		MAIN2MCU_LVL_INTRTR0_IN_120	MAIN2MCU_LVL_INTRTR0	TIMER12 Interrupt Request	Level
		R5FSS0_CORE0_INTR_IN_168	R5FSS0_CORE0	TIMER12 Interrupt Request	Level
		R5FSS0_CORE1_INTR_IN_168	R5FSS0_CORE1	TIMER12 Interrupt Request	Level
		R5FSS1_CORE0_INTR_IN_168	R5FSS1_CORE0	TIMER12 Interrupt Request	Level
		R5FSS1_CORE1_INTR_IN_168	R5FSS1_CORE1	TIMER12 Interrupt Request	Level

**Table 12-1586. TIMER Hardware Requests (continued)**

TIMER13	TIMER13_INTR_PEN D_0	C66SS0_INTRTR0_IN_304	C66SS0_INTRTR0	TIMER13 Interrupt Request	Level
		C66SS1_INTRTR0_IN_304	C66SS1_INTRTR0	TIMER13 Interrupt Request	Level
		GIC500_SPI_IN_269	COMPUTE_CLUSTER0	TIMER13 Interrupt Request	Level
		MAIN2MCU_LVL_INTRTR0_I N_121	MAIN2MCU_LVL_INTRTR0	TIMER13 Interrupt Request	Level
		R5FSS0_CORE0_INTR_IN_1 69	R5FSS0_CORE0	TIMER13 Interrupt Request	Level
		R5FSS0_CORE1_INTR_IN_1 69	R5FSS0_CORE1	TIMER13 Interrupt Request	Level
		R5FSS1_CORE0_INTR_IN_1 69	R5FSS1_CORE0	TIMER13 Interrupt Request	Level
		R5FSS1_CORE1_INTR_IN_1 69	R5FSS1_CORE1	TIMER13 Interrupt Request	Level
TIMER14	TIMER14_INTR_PEN D_0	C66SS0_INTRTR0_IN_305	C66SS0_INTRTR0	TIMER14 Interrupt Request	Level
		C66SS1_INTRTR0_IN_305	C66SS1_INTRTR0	TIMER14 Interrupt Request	Level
		GIC500_SPI_IN_270	COMPUTE_CLUSTER0	TIMER14 Interrupt Request	Level
		MAIN2MCU_LVL_INTRTR0_I N_122	MAIN2MCU_LVL_INTRTR0	TIMER14 Interrupt Request	Level
		R5FSS0_CORE0_INTR_IN_1 70	R5FSS0_CORE0	TIMER14 Interrupt Request	Level
		R5FSS0_CORE1_INTR_IN_1 70	R5FSS0_CORE1	TIMER14 Interrupt Request	Level
		R5FSS1_CORE0_INTR_IN_1 70	R5FSS1_CORE0	TIMER14 Interrupt Request	Level
		R5FSS1_CORE1_INTR_IN_1 70	R5FSS1_CORE1	TIMER14 Interrupt Request	Level
	TIMER14_TIMER_PWM_0	TIMESYNC_INTRTR0_IN_2	TIMESYNC_INTRTR0	TIMER14 Timesync Event	Level
TIMER15	TIMER15_INTR_PEN D_0	C66SS0_INTRTR0_IN_306	C66SS0_INTRTR0	TIMER15 Interrupt Request	Level
		C66SS1_INTRTR0_IN_306	C66SS1_INTRTR0	TIMER15 Interrupt Request	Level
		GIC500_SPI_IN_271	COMPUTE_CLUSTER0	TIMER15 Interrupt Request	Level
		MAIN2MCU_LVL_INTRTR0_I N_123	MAIN2MCU_LVL_INTRTR0	TIMER15 Interrupt Request	Level
		R5FSS0_CORE0_INTR_IN_1 71	R5FSS0_CORE0	TIMER15 Interrupt Request	Level
		R5FSS0_CORE1_INTR_IN_1 71	R5FSS0_CORE1	TIMER15 Interrupt Request	Level
		R5FSS1_CORE0_INTR_IN_1 71	R5FSS1_CORE0	TIMER15 Interrupt Request	Level
		R5FSS1_CORE1_INTR_IN_1 71	R5FSS1_CORE1	TIMER15 Interrupt Request	Level
	TIMER15_TIMER_PWM_0	TIMESYNC_INTRTR0_IN_3	TIMESYNC_INTRTR0	TIMER15 Timesync Event	Level
TIMER16	TIMER16_INTR_PEN D_0	C66SS0_INTRTR0_IN_12	C66SS0_INTRTR0	TIMER16 Interrupt Request	Level

**Table 12-1586. TIMER Hardware Requests (continued)**

		C66SS1_INTRTR0_IN_12	C66SS1_INTRTR0	TIMER16 Interrupt Request	Level
		GIC500_SPI_IN_272	COMPUTE_CLUSTER0	TIMER16 Interrupt Request	Level
		MAIN2MCU_LVL_INTRTR0_I N_124	MAIN2MCU_LVL_INTRT R0	TIMER16 Interrupt Request	Level
		R5FSS0_CORE0_INTR_IN_1 72	R5FSS0_CORE0	TIMER16 Interrupt Request	Level
		R5FSS0_CORE1_INTR_IN_1 72	R5FSS0_CORE1	TIMER16 Interrupt Request	Level
		R5FSS1_CORE0_INTR_IN_1 72	R5FSS1_CORE0	TIMER16 Interrupt Request	Level
		R5FSS1_CORE1_INTR_IN_1 72	R5FSS1_CORE1	TIMER16 Interrupt Request	Level
TIMER17	TIMER16_TIMER_PW M_0	TIMESYNC_INTRTR0_IN_40	TIMESYNC_INTRTR0	TIMER16 Timesync Event	Level
	TIMER17_INTR_PEN D_0	C66SS0_INTRTR0_IN_13	C66SS0_INTRTR0	TIMER17 Interrupt Request	Level
		C66SS1_INTRTR0_IN_13	C66SS1_INTRTR0	TIMER17 Interrupt Request	Level
		GIC500_SPI_IN_273	COMPUTE_CLUSTER0	TIMER17 Interrupt Request	Level
		MAIN2MCU_LVL_INTRTR0_I N_125	MAIN2MCU_LVL_INTRT R0	TIMER17 Interrupt Request	Level
		R5FSS0_CORE0_INTR_IN_1 73	R5FSS0_CORE0	TIMER17 Interrupt Request	Level
		R5FSS0_CORE1_INTR_IN_1 73	R5FSS0_CORE1	TIMER17 Interrupt Request	Level
TIMER18		R5FSS1_CORE0_INTR_IN_1 73	R5FSS1_CORE0	TIMER17 Interrupt Request	Level
		R5FSS1_CORE1_INTR_IN_1 73	R5FSS1_CORE1	TIMER17 Interrupt Request	Level
	TIMER17_TIMER_PW M_0	TIMESYNC_INTRTR0_IN_41	TIMESYNC_INTRTR0	TIMER17 Timesync Event	Level
	TIMER18_INTR_PEN D_0	C66SS0_INTRTR0_IN_14	C66SS0_INTRTR0	TIMER18 Interrupt Request	Level
		C66SS1_INTRTR0_IN_14	C66SS1_INTRTR0	TIMER18 Interrupt Request	Level
		GIC500_SPI_IN_274	COMPUTE_CLUSTER0	TIMER18 Interrupt Request	Level
		MAIN2MCU_LVL_INTRTR0_I N_126	MAIN2MCU_LVL_INTRT R0	TIMER18 Interrupt Request	Level
		R5FSS0_CORE0_INTR_IN_1 74	R5FSS0_CORE0	TIMER18 Interrupt Request	Level
		R5FSS0_CORE1_INTR_IN_1 74	R5FSS0_CORE1	TIMER18 Interrupt Request	Level
		R5FSS1_CORE0_INTR_IN_1 74	R5FSS1_CORE0	TIMER18 Interrupt Request	Level
		R5FSS1_CORE1_INTR_IN_1 74	R5FSS1_CORE1	TIMER18 Interrupt Request	Level
	TIMER18_TIMER_PW M_0	TIMESYNC_INTRTR0_IN_42	TIMESYNC_INTRTR0	TIMER18 Timesync Event	Level
	TIMER19_INTR_PEN D_0	C66SS0_INTRTR0_IN_15	C66SS0_INTRTR0	TIMER19 Interrupt Request	Level

**Table 12-1586. TIMER Hardware Requests (continued)**

C66SS1_INTRTR0_IN_15	C66SS1_INTRTR0	TIMER19 Interrupt Request	Level
GIC500_SPI_IN_275	COMPUTE_CLUSTER0	TIMER19 Interrupt Request	Level
MAIN2MCU_LVL_INTRTR0_IN_127	MAIN2MCU_LVL_INTRTR0	TIMER19 Interrupt Request	Level
R5FSS0_CORE0_INTR_IN_175	R5FSS0_CORE0	TIMER19 Interrupt Request	Level
R5FSS0_CORE1_INTR_IN_175	R5FSS0_CORE1	TIMER19 Interrupt Request	Level
R5FSS1_CORE0_INTR_IN_175	R5FSS1_CORE0	TIMER19 Interrupt Request	Level
R5FSS1_CORE1_INTR_IN_175	R5FSS1_CORE1	TIMER19 Interrupt Request	Level
TIMER19_TIMER_PWM_0	TIMESYNC_INTRTR0_IN_43	TIMESYNC_INTRTR0	TIMER19 Timesync Event

#### 12.10.3.4 Timers Functional Description

Each timer contains a free-running upward counter with autoreload capability on overflow. The timer counter can be read and written on-the-fly (while counting). Each timer includes compare logic to allow an interrupt event on a programmable counter matching value. A dedicated output signal can be pulsed or toggled on either an overflow or a match event. This offers time-stamp trigger signaling or PWM signal sources. A dedicated input signal can be used to trigger an automatic timer counter capture or an interrupt event on a programmable input signal transition. A programmable clock divider (prescaler) allows reduction of the timer input clock frequency. All internal timer interrupt sources are merged into one module interrupt line and one wake-up line.

Each internal interrupt source can be independently enabled and disabled by a dedicated bit in the `TIMER_IRQSTATUS_SET` and `TIMER_IRQSTATUS_CLR` register for the interrupt features, and a dedicated bit of the `TIMER_IRQWAKEEN` register for a wake-up. In addition, timers have a mechanism implemented to generate an accurate tick interrupt.

For each timer implemented in the device, there are two possible clock sources:

- 32-kHz clock
- System clock

For more information of the selection of the input clock source, see *Clocking*.

Each timer supports three functional modes:

- Timer mode
- Capture mode
- Compare mode

The capture and compare modes are disabled by default after core reset.

##### 12.10.3.4.1 Timer Block Diagram

[Figure 12-1214](#) is a block diagram of the timers.

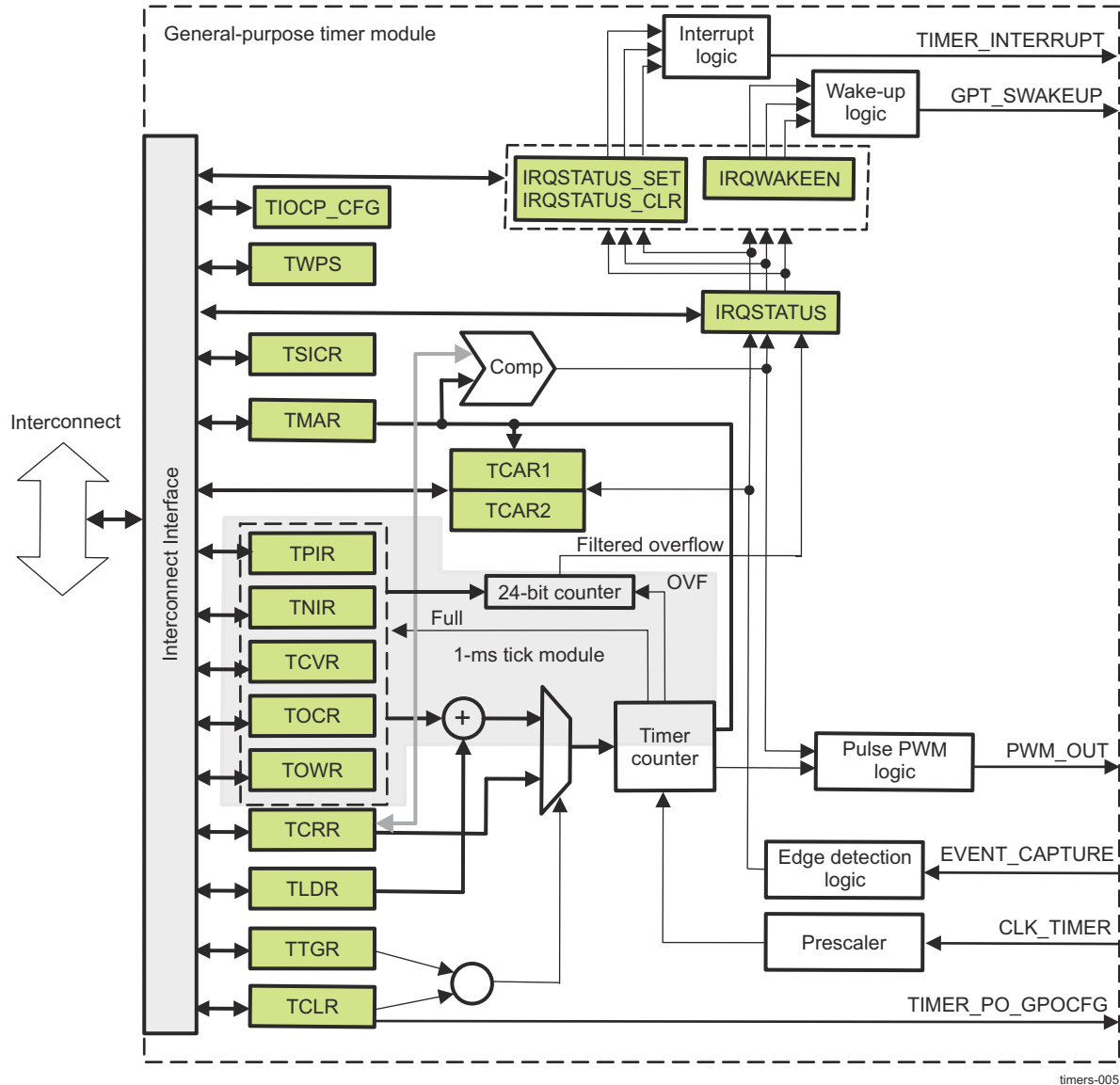


Figure 12-1214. Timer Block Diagram

#### 12.10.3.4.2 Timer Power Management

##### Note

Some of the timers features described in this section may not be supported on this family of devices. For more information, see *Timers Not Supported Features*.

The way the timer acknowledges the LPSC clock stop request is configurable through the `TIMER_TIOCP_CFG[3-2] IDLEMODE` bit field.

Table 12-1587 lists the IDLEMODE settings and the related acknowledgment modes.

Table 12-1587. IDLEMODE Settings

IDLEMODE Value	Selected Mode	Description
00	Force-idle	The clock stop request is unconditionally acknowledged from the timer, regardless of its internal operations. This mode must be used carefully, because it does not prevent the loss of data when the clock is switched off.

**Table 12-1587. IDLEMODE Settings (continued)**

IDLEMODE Value	Selected Mode	Description
01	No-idle	The clock stop request is never acknowledged from the timer. This mode is safe from a module point of view but is not efficient from a power-saving perspective because the clocks remain active.
10	Smart-idle	The timer acknowledges the clock stop request, basing its decision on its internal activity. The acknowledge signal is asserted only when all pending transactions and interrupt requests are treated. This is the best approach to efficient system power management.
11	Smart-idleWakeup	The module behaves like in Smart-idle mode, with the exception, that it can issue a wake-up request in sleep mode, if the functional clock is not cut off.

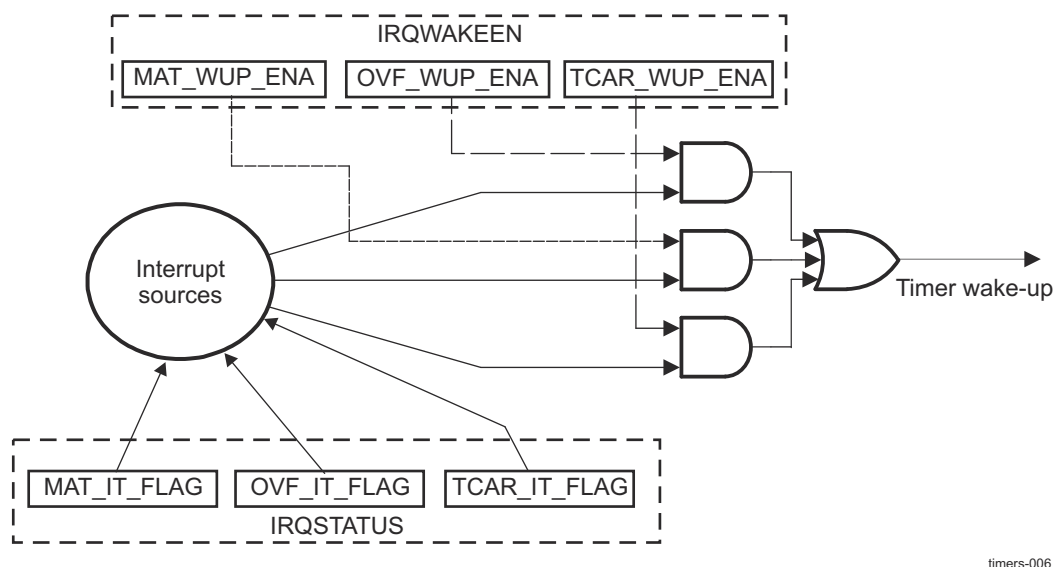
#### 12.10.3.4.2.1 Wake-Up Capability

##### Note

Some of the timers features described in this section may not be supported on this family of devices. For more information, see *Timers Not Supported Features*.

If the TIMER\_TIOCP\_CFG[3-2] IDLEMODE bit field sets the smart-idle mode or smart-idle with wake-up mode, the timer evaluates its internal capability to have the interface clock switched off. When there is no further internal activity (no pending interrupt sources: match, overflow, or timer capture events), the clock stop acknowledge signal is asserted and the timer enters sleep mode, ready to issue a wake-up request if configured in smart-idle with wake-up mode.

Figure 12-1215 shows the wake-up request generation.



timers-006

**Figure 12-1215. Wake-Up Request Generation**

The timer wake-up-enable register allows masking of the expected source of the wake-up event that generates a wake-up request. The register is synchronously programmed with the interface clock before the Clock management sends a clock stop request. The expected source of the wake-up event is an overflow (TIMER\_TCR), a timer match (the compare result of TIMER\_TCR and TIMER\_TMAR matches the counter value), and a timer capture (detection of an external pulse transition of the correct polarity on the TIMER\_PIEVENTCAPT).

When the wake-up event is issued, the associated interrupt status bit is set in the timer status register (TIMER\_IRQSTATUS). The pending wake-up event is reset when the set status bit is overwritten with 1.



### Note

The status bit must be reset to re-enter idle mode.

#### 12.10.3.4.3 Timer Software Reset

TIMER\_TIOCP\_CFG[0] SOFTRESET bit can initiate a software reset of the timer. This bit is autocleared to 0 when the reset is complete.

Before accessing or using the timer, the local host must ensure that internal reset is released by reading the TIMER\_TIOCP\_CFG[0] SOFTRESET bit. This bit monitors the internal reset status.

#### 12.10.3.4.4 Timer Interrupts

The timer can issue an overflow interrupt, a timer match interrupt, and a timer capture interrupt. Each internal interrupt source can be independently enabled and disabled in the interrupt-enable register (TIMER\_IRQSTATUS\_SET) and disabled in the interrupt-disable register (TIMER\_IRQSTATUS\_CLR). When the interrupt event is issued, the associated interrupt status bit is set in the timer status register (TIMER\_IRQSTATUS).

#### 12.10.3.4.5 Timer Mode Functionality

The timer is an upward counter that can be started and stopped at any time through the timer control register (the TIMER\_TCLR[0] ST bit). The timer counter register (TIMER\_TCRR) can be loaded when stopped or on-the-fly (while counting). TIMER\_TCRR can be loaded directly by a TIMER\_TCRR write access with a new timer value. TIMER\_TCRR can also be loaded with the value held in the timer load register (TIMER\_TLDR) by a trigger register (TIMER\_TTGR) write access. The loading of TIMER\_TCRR is done regardless of the written value of TIMER\_TTGR. The value of TIMER\_TCRR can be read when stopped or captured on-the-fly by a TIMER\_TCRR read access. The timer is stopped and the counter value is set to 0 when the module reset is asserted. The timer is maintained at stop after the reset is released.

In one-shot mode (the TIMER\_TCLR[1] AR bit is set to 0), the counter is stopped after counting overflow occurs (the counter value remains at 0).

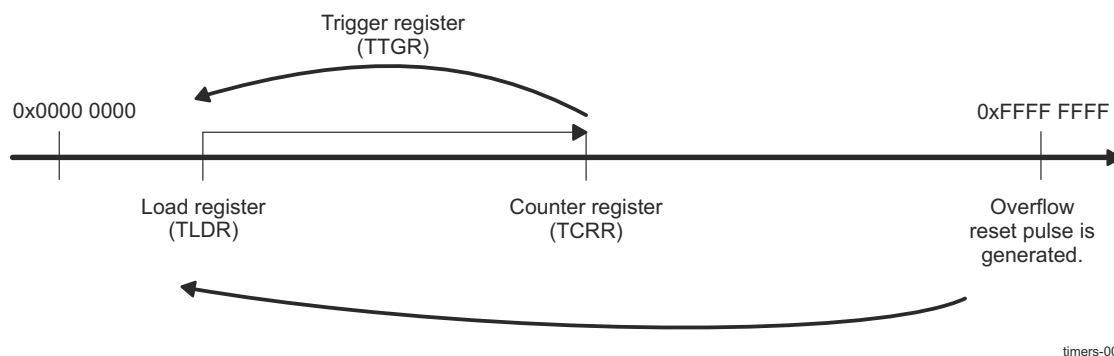
When the autoreload mode is enabled (the TIMER\_TCLR[1] AR bit is set to 1), TIMER\_TCRR is reloaded with the value of TIMER\_TLDR after a counting overflow occurs.

### CAUTION

Do not put the overflow value (0xFFFF FFFF) in the TIMER\_TLDR register because it can lead to undesirable results.

An interrupt can be issued on overflow if the overflow interrupt-enable bit is set in the timer interrupt-enable register (the TIMER\_IRQSTATUS\_SET[1] OVF\_EN\_FLAG bit is set to 1). A dedicated output pin (POTIMERPWM) can be programmed in the TIMER\_TCLR[12] PT bit through the TIMER\_TCLR[11-10] (PT and TRG bits) to generate one positive pulse (prescaler duration) or to invert the current value (toggle mode) when an overflow occurs. The TIMER\_TCLR[12] PT bit selects pulse/toggle modulation (the TIMER\_TCLR[11-10] TRG bit field selects trigger mode).

Figure 12-1216 shows the TIMER\_TCRR timing value.



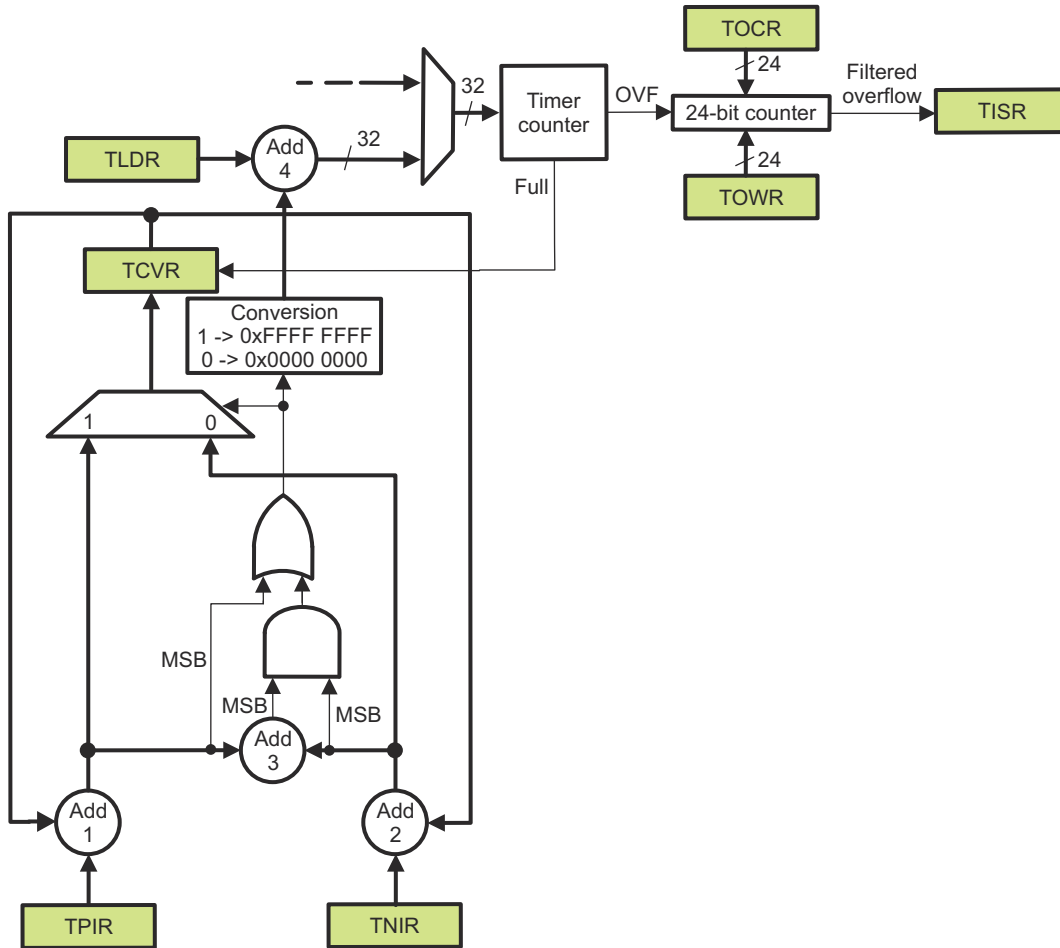
**Figure 12-1216. TIMER\_TCRR Timing Value**

#### 12.10.3.4.5.1 1-ms Tick Generation

The interrupt period is not exactly 1 ms, because the timer input clock is 32.768 kHz. If the clock counts up to 32, it obtains a 0.977-ms period; if it counts up to 33, it obtains a 1.007-ms period. For large granularity, the error is cumulative and can generate important deviations from the standard value.

To minimize the error between a true 1-ms tick and the tick generated by the 32.768-kHz timer, the sequencing of periods less than 1 ms and periods greater than 1 ms must be shuffled. An additional 1-ms block is used to correct this error. See [Figure 12-1217](#).

In this implementation, the increment sequencing is automatically managed by the timer to minimize the error. The user must define only the value of the timer positive increment register (the `TIMER_TPIR[31-0]` `POSITIVE_INC_VALUE` bit field) and the timer negative increment register (the `TIMER_TNIR[31-0]` `NEGATIVE_INC_VALUE` bit field). An automatic adaptation mechanism is used to simplify the programming model.



timers-009

**Figure 12-1217. Block Diagram of the 1-ms Tick Module**

The TIMER\_TPIR, TIMER\_TNIR, and TIMER\_TCVR registers and adders Add1, Add2, and Add3 are used to define whether the next value loaded in the timer counter register (the TIMER\_TCRR[31:0] TIMER\_COUNTER bit field) is the value of the TIMER\_TLDR[31:0] LOAD\_VALUE bit field (period less than 1 ms) or the value of TIMER\_TLDR[31:0] LOAD\_VALUE – 1 (period greater than 1 ms).

Table 12-1588 lists the value loaded in the TIMER\_TCRR according to the sign of the result of Add1, Add2, and Add3.

MSB = 0: Positive value; MSB = 1: Negative value

**Table 12-1588. Value Loaded in TIMER\_TCRR to Generate 1-ms Tick**

Add1 MSB	Add2 MSB	Add3 MSB	Value of TIMER_TCRR Register
0	0	0	TIMER_TLDR[31:0] LOAD_VALUE bit field
0	0	1	TIMER_TLDR[31:0] LOAD_VALUE bit field
0	1	0	TIMER_TLDR[31:0] LOAD_VALUE bit field
0	1	1	TIMER_TLDR[31:0] LOAD_VALUE – 1
1	0	0	N/A
1	0	1	N/A
1	1	0	TIMER_TLDR[31:0] LOAD_VALUE – 1
1	1	1	TIMER_TLDR[31:0] LOAD_VALUE – 1

The values of the TIMER\_TPIR and TIMER\_TNIR registers are calculated using the following formulas:

- Positive increment value =  $(\text{INTEGER}[F_{\text{clk}} \times T_{\text{tick}}] + 1) \times 1\text{e}6 - (F_{\text{clk}} \times T_{\text{tick}} \times 1\text{e}6)$
- Negative increment value =  $(\text{INTEGER}[F_{\text{clk}} \times T_{\text{tick}}] \times 1\text{e}6) - (F_{\text{clk}} \times T_{\text{tick}} \times 1\text{e}6)$

---

#### Note

$F_{\text{clk}}$  clock frequency (kHz)

$T_{\text{tick}}$  tick period (ms)

---

The timer overflow counter register (TIMER\_TOCR) and the timer overflow wrapping register (TIMER\_TOWR) are used to filter interrupts. When the timer overflows, it increments the 24-bit TIMER\_TOCR. When the values in the 24-bit TIMER\_TOCR match the values in the 24-bit TIMER\_TOWR and the timer overflow is asserted, the TIMER\_TOCR is reset and an interrupt is generated to the TIMER\_IRQSTATUS register.

---

#### Note

TIMER\_TOWR has to be set to requested value. For example, if no interrupt needs to be masked TIMER\_TOWR must be set to 0, if one interrupt needs to be masked TIMER\_TOWR must be set to 1, if two interrupts need to be masked TIMER\_TOWR must be set to 2 and so on.

It is important to have in mind that the case when FFFFFFF interrupts need to be masked is not possible.

---

With the conversion block in reset state (the positive increment register, negative increment register, and counter value register are zeroed), the programming model and the behavior of timers remain unchanged.

For 1-ms tick with a 32.768-kHz clock:

- TIMER\_TPIR[31-0] POSITIVE\_INC\_VALUE = 232,000
- TIMER\_TNIR[31-0] NEGATIVE\_INC\_VALUE = -768,000
- TIMER\_TLDR[31-0] LOAD\_VALUE = 0xFFFF FFE0

---

#### Note

Any value of the tick period can be generated with the appropriate value of the TIMER\_TPIR, TIMER\_TNIR, and TIMER\_TLDR.

By default, the TIMER\_TPIR, TIMER\_TNIR, TIMER\_TCVR, TIMER\_TOCR, and TIMER\_TOWR and the associated logic are in reset mode (all 0s) and have no effect on the programming model.

---

#### 12.10.3.4.6 Timer Capture Mode Functionality

When a transition is detected on the module input pin (PIEVENTCAPT), the timer value in the TIMER\_TCRR can be captured and saved in the TIMER\_TCAR1 or TIMER\_TCAR2 register function of the mode selected in the TIMER\_TCLR[13] CAPT\_MODE bit. The edge detection circuitry monitors transitions on the input pin (PIEVENTCAPT).

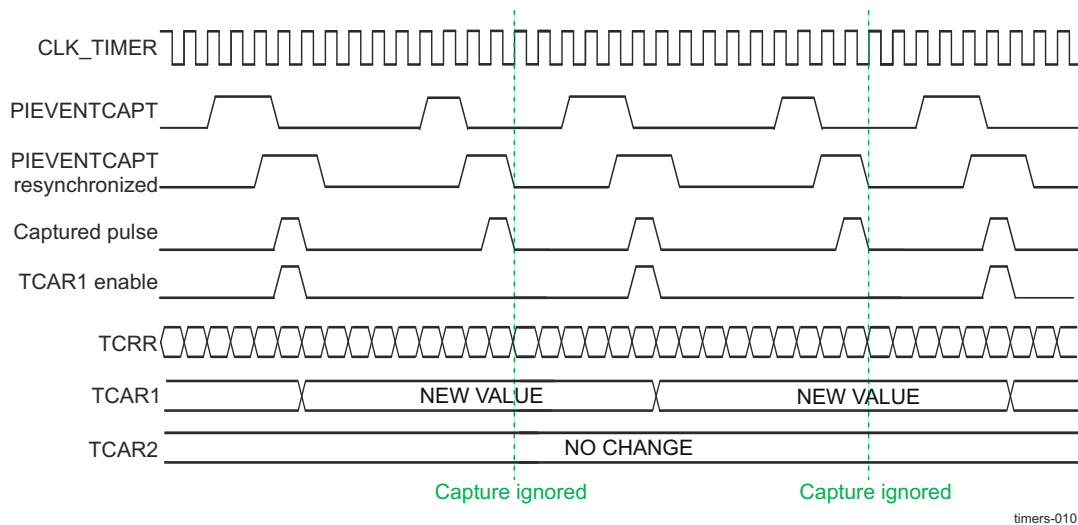
The rising edge, falling edge, or both, can be selected in the TIMER\_TCLR[9-8] TCM bit field to trigger the timer counter capture. The module sets the TIMER\_IRQSTATUS[2] TCAR\_IT\_FLAG bit when an active edge is detected, and at the same time, the counter value TIMER\_TCRR is stored in timer capture register TIMER\_TCAR1 or TIMER\_TCAR2, as follows:

- If the TIMER\_TCLR[13] CAPT\_MODE bit is 0, on the first enabled capture event, the value of the counter register is saved in the TIMER\_TCAR1 register, and the next events are ignored (no update on the TIMER\_TCAR1 register and no interrupt triggering) until the detection logic is reset or the TIMER\_IRQSTATUS[2] TCAR\_IT\_FLAG is cleared by writing 1 to it.
- If the TIMER\_TCLR[13] CAPT\_MODE bit is 1, on the first enabled capture event, the value of the counter register is saved in the TIMER\_TCAR1 register, and on the second enabled capture event, the value of the

counter register is saved in the `TIMER_TCAR2` register. If a capture interrupt is enabled, the interrupt triggers on the second event capture. All other events are ignored (no update on `TIMER_TCAR1/TIMER_TCAR2` and no interrupt triggering) until the detection logic is reset or the `TIMER_IRQSTATUS[2] TCAR_IT_FLAG` bit is cleared by writing 1 to it. This mechanism is useful for period calculation of a clock, if that clock is connected to the `PIEVENTCAPT` input pin.

The edge detection logic is reset (a new capture is enabled) when the active capture interrupt is served. The `TIMER_IRQSTATUS[2] TCAR_IT_FLAG` bit is cleared by writing 1 to it or when the edge detection mode bits (the `TIMER_TCLR[9-8] TCM` bit field) are changed from no-capture mode detection to any other mode. The timer functional clock (input to prescaler) is used to sample the input pin (`PIEVENTCAPT`). A negative or positive pulse input can be detected when the pulse time is greater than the functional clock period. An interrupt is issued on edge detection if the capture interrupt-enable bit is set in the `TIMER_IRQSTATUS_SET[2] TCAR_EN_FLAG` bit. See the examples in [Figure 12-1218](#) and [Figure 12-1219](#).

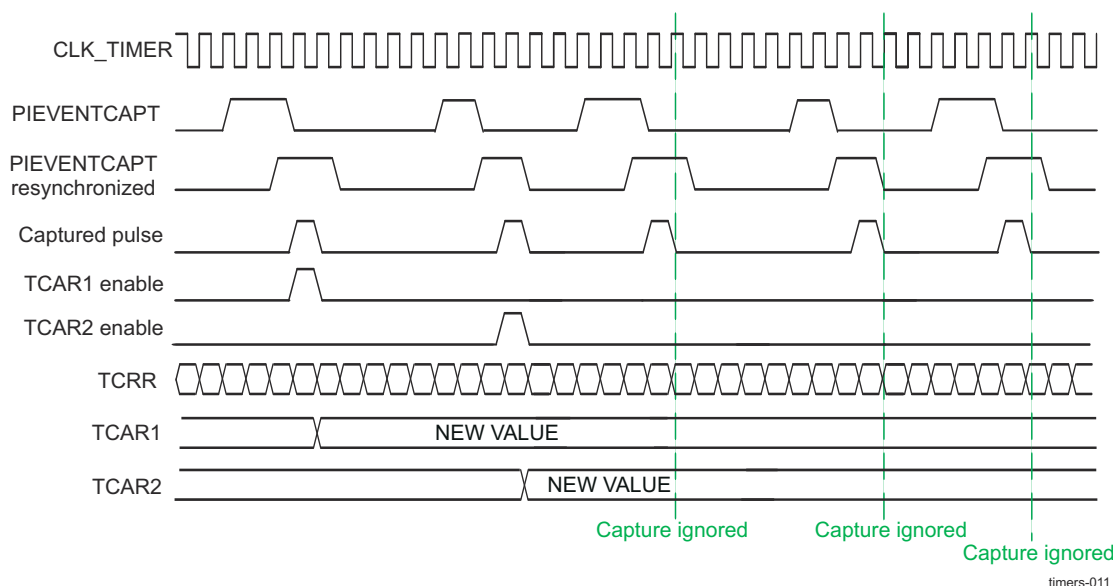
In [Figure 12-1218](#), the value of the `TIMER_TCLR[9-8] TCM` bit field is 1h, and the `TIMER_TCLR[13] CAPT_MODE` bit is 0. Only the rising edge of `PIEVENTCAPT` triggers a capture in the `TIMER_TCAR1` and `TIMER_TCAR2` registers, and only the `TIMER_TCAR1` register updates.



timers-010

**Figure 12-1218. Capture Wave Example for `TIMER_TCLR[13] CAPT_MODE = 0`**

In [Figure 12-1219](#), the value of the `TIMER_TCLR[9-8] TCM` bit field is 1h, and the `TIMER_TCLR[13] CAPT_MODE` bit is 1. Only the rising edge of `PIEVENTCAPT` triggers a capture in the `TIMER_TCAR1` register on the first enabled event, and the `TIMER_TCAR2` register updates on the second enabled event.



**Figure 12-1219. Capture Wave Example for TIMER\_TCLR[13] CAPT\_MODE = 1**

#### 12.10.3.4.7 Timer Compare Mode Functionality

When the compare-enable register `TIMER_TCLR[6]` CE bit is set to 1, the timer value (the `TIMER_TCRR[31-0]` `TIMER_COUNTER` bit field) is continuously compared to the value held in the timer match register (`TIMER_TMAR`). The value of the `TIMER_TMAR[31-0]` `COMPARE_VALUE` bit field can be loaded at any time (timer counting or stopped). When the `TIMER_TCRR` and the `TIMER_TMAR` values match, an interrupt is issued, if the `TIMER_IRQSTATUS_SET[0]` `MAT_EN_FLAG` bit is set.

To prevent any unwanted interrupts due to reset value matching effect, write a compare value to the `TIMER_TMAR` before setting the `TIMER_TCLR[6]` CE bit.

The dedicated output pin (`POTIMERPWM`) can be programmed in the `TIMER_TCLR[12]` PT bit through the `TIMER_TCLR[11-10]` TRG bit field to generate one positive pulse (timer clock duration) or to invert the current value (toggle mode) when an overflow or a match occurs.

#### 12.10.3.4.8 Timer Prescaler Functionality

A prescaler can be used to divide the timer counter input clock frequency. The prescaler is enabled when the `TIMER_TCLR[5]` PRE bit is set. The `TIMER_TCLR[4-2]` PTV bit field sets the  $2^n$  division ratio (prescaler value is  $2^{(PTV + 1)}$ ). The prescaler counter is reset when the timer counter is stopped or reloaded on-the-fly.

Table 12-1589 lists the prescaler/timer reload values versus contexts.

**Table 12-1589. Prescaler/Timer Reload Values Versus Contexts**

Context	Prescaler	Timer Counter
Overflow (when autoreload is on)	Reset	<code>TIMER_TLDR[31-0]</code>
<code>TIMER_TCRR</code> write	Reset	<code>TIMER_TCRR[31-0]</code>
<code>TIMER_TTGR</code> write	Reset	<code>TIMER_TLDR[31-0]</code>
Stop	Reset	Frozen

#### 12.10.3.4.9 Timer Pulse-Width Modulation

The timer can be configured to provide a programmable PWM output. The timer PWM (`POTIMERPWM`) output pin can be configured to toggle on an event. The `TIMER_TCLR[11-10]` TRG bit field determines on which register value the PWM pin toggles. Either overflow or both overflow and match can be selected to toggle the timer PWM pin when a compare condition occurs.

### Note

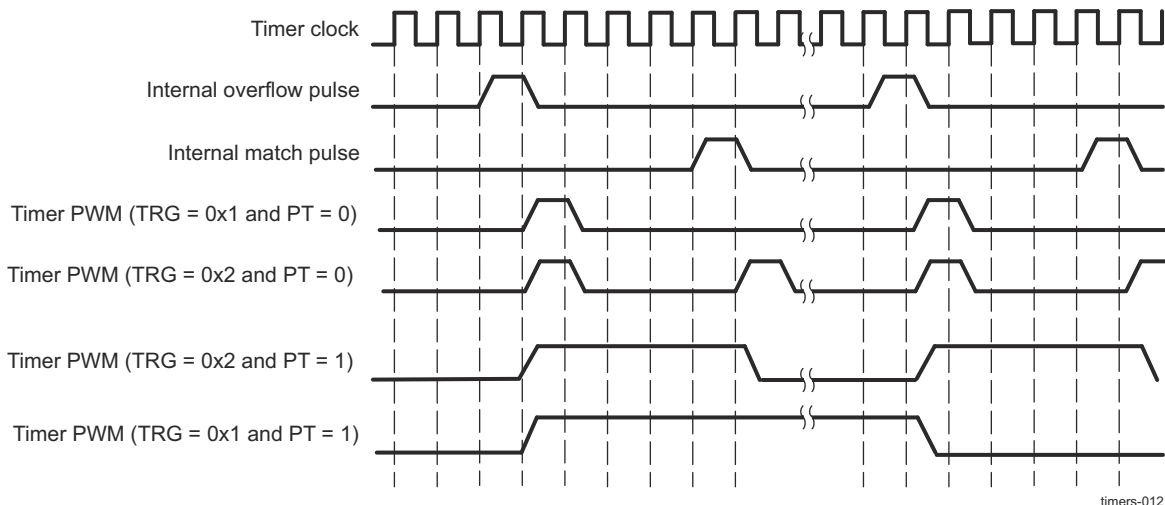
In toggle mode, when `TIMER_TCLR[11-10] TRG = 0x2` (overflow and match), the first event that toggles the PWM line is an overflow event.

The `TIMER_TCLR[7] SCPWM` bit can be programmed to set or clear the timer PWM output signal only while the counter is stopped or the trigger is off. This allows setting the output pin to a known state before modulation starts. Modulation synchronously stops when the `TIMER_TCLR[11-10] TRG` bit field is cleared and overflow occurs. This allows fixing a deterministic state of the output pin when modulation stops.

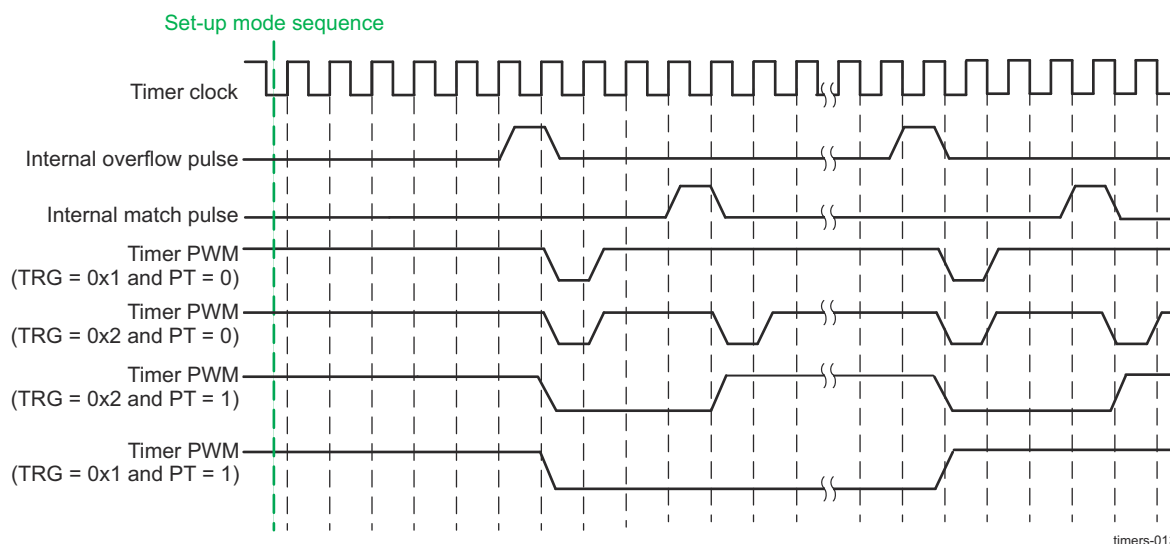
In [Figure 12-1220](#), the internal overflow pulse is set each time the  $(0xFFFF - \text{TIMER\_TLDR}[31-0] \text{ LOAD\_VALUE} + 1)$  value is reached, and the internal match pulse is set when the counter reaches the value of `TIMER_TMAR`. Depending on the value of the `TIMER_TCLR[12] PT` bit and `TIMER_TCLR[11-10] TRG` bit field, the timer provides pulse or PWM event on the output pin (`POTIMERPWM`).

The `TIMER_TLDR` and `TIMER_TMAR` must keep values below the overflow value (`0xFFFF FFFF`) by at least two units. If the PWM trigger events are both overflow and match, the difference between the values kept in the `TIMER_TMAR` and the value in the `TIMER_TLDR` must be at least two units. When match event is used, the compare mode `TIMER_TCLR[6] CE` bit must be set.

In [Figure 12-1220](#), the `TIMER_TCLR[7] SCPWM` bit is set to 0. In [Figure 12-1221](#), the `TIMER_TCLR[7] SCPWM` bit is set to 1. To obtain the desired wave form, start the counter at `0xFFFF FFFE` value (to ensure an overflow first) or adjust the line polarity (`TIMER_TCLR[7] SCPWM` bit).



**Figure 12-1220. Timing Diagram of PWM With `TIMER_TCLR[7] SCPWM` Bit = 0**



**Figure 12-1221. Timing Diagram of PWM With TIMER\_TCLR[7] SCPWM Bit = 1**

#### 12.10.3.4.10 Timer Counting Rate

The timer rate is defined by the following values:

- Value of the prescaler fields (the TIMER\_TCLR[5] PRE bit and TIMER\_TCLR[4-2] PTV bit field)
- Value loaded into the TIMER\_TLDR

Table 12-1590 lists the prescaler clock ratio values.

**Table 12-1590. Prescaler Clock Ratio Values**

TIMER_TCLR[5] PRE	TIMER_TCLR[4-2] PTV	Divisor (PS)
0	X	1
1	0	2
1	1	4
1	2	8
1	3	16
1	4	32
1	5	64
1	6	128
1	7	256

Thus, the timer overflow rate is expressed as:

$$\text{OVF\_Rate} = (0\text{xFFFF FFFF} - \text{TIMER\_TLDR} + 1) \times (\text{timer-functional clock period}) \times \text{PS}$$

With (timer-functional clock period) =  $1/(\text{timer-functional clock frequency})$  and PS =  $2^{(\text{PTV} + 1)}$  if prescaler is enabled, or PS = 1 if prescaler is disabled.

#### CAUTION

Internal resynchronization causes any write to the TIMER\_TCLR[1] ST bit to have some latency before the register is updated:

$2.5 \times$  functional clock cycles write\_TIMER\_TCLR\_latency  $3.5 \times$  functional clock cycles

Remember to consider this latency whenever the timer must be started or stopped by a software change to the TIMER\_TCLR[1] ST bit.



### CAUTION

- In non-PWM mode, TIMER\_TLDR must be maintained at less than or equal to 0xFFFF FFFE.
- In PWM mode, TIMER\_TLDR must be maintained at less than or equal to 0xFFFF FFDD.

For example, with a timer clock input of 32 kHz and the TIMER\_TCLR[5] PRE bit set to 0, the timer output period is as listed in [Table 12-1591](#).

**Table 12-1591. Value and Corresponding Interrupt Period**

TIMER_TLDR[31-0] LOAD_VALUE	Interrupt Period
0x0000 0000	37 h
0xFFFF 0000	2 s
0xFFFF FFF0	500 $\mu$ s
0xFFFF FFFE	62.5 $\mu$ s

#### 12.10.3.4.11 Timer Under Emulation

During emulation mode, the timer continues to run according to the value of the TIMER\_TIOCP\_CFG[1] EMUFREE bit.

If the TIMER\_TIOCP\_CFG[1] EMUFREE bit is set to 1, timer execution is not stopped in emulation mode and the interrupt is still generated when overflow or match is reached.

If the TIMER\_TIOCP\_CFG[1] EMUFREE bit is set to 0, the prescaler and timer are frozen and both resume on exit from emulation mode. The asynchronous external input pin (MCU\_TIMER\_IO[9-0] or TIMER\_IO[7-0]) is internally synchronized on two timer-clock rising edges.

#### 12.10.3.4.12 Accessing Timer Registers

All accesses are posted mode, under the assumption that  $\text{freq}(\text{timer clock}) < \text{freq}(\text{interface clock})/4$ , until software reconfiguration. In addition, it is not recommended to access the timer registers prior to the PLLs generating the timer and interface clocks have been configured and the clocks have stabilized. All registers are 32 bits wide, accessible through the configuration interface with 32-bit access (read/write).

Write operations to the following functional registers must be complete (the MSB must be written even if the MSB data is not used):

- TIMER\_TCLR
- TIMER\_TCR
- TIMER\_TLDR
- TIMER\_TTGR
- TIMER\_TMAR
- TIMER\_TPIR
- TIMER\_TNIR
- TIMER\_TCV
- TIMER\_TOCR
- TIMER\_TOWR

The following registers are not affected by the posted/nonposted mode selection; the write/read operation is effective and acknowledged (command accepted) after one interface clock cycle from command assertion:

- TIMER\_TIDR
- TIMER\_TIOCP\_CFG
- TIMER\_IRQSTATUS
- TIMER\_IRQSTATUS\_RAW
- TIMER\_IRQSTATUS\_SET
- TIMER\_IRQSTATUS\_CLR
- TIMER\_IRQWAKEEN
- TIMER\_TWPS

- TIMER\_TSICR

#### 12.10.3.4.12.1 Writing to Timer Registers

The host uses the configuration interface to write to the following registers synchronously with the timer interface clock:

- TIMER\_TLDR
- TIMER\_TCRR
- TIMER\_TCLR
- TIMER\_TIOCP\_CFG
- TIMER\_IRQSTATUS
- TIMER\_IRQSTATUS\_SET
- TIMER\_IRQSTATUS\_CLR
- TIMER\_IRQWAKEEN
- TIMER\_TTGR
- TIMER\_TSICR
- TIMER\_TMAR
- TIMER\_TPIR
- TIMER\_TNIR
- TIMER\_TCVR
- TIMER\_TOCR
- TIMER\_TOWR

#### 12.10.3.4.12.1.1 Write Posting Synchronization Mode

This mode is used if the TIMER\_TSICR[2] POSTED bit is set to 1 (default value).

This mode uses a posted write scheme to update any internal register (TIMER\_TCLR, TIMER\_TCRR, TIMER\_TLDR, TIMER\_TTGR, TIMER\_TMAR, TIMER\_TPIR, TIMER\_TNIR, TIMER\_TCVR, TIMER\_TOCR, and TIMER\_TOWR). Therefore, the write transaction is immediately acknowledged on the configuration interface, although the effective write operation occurs later because of a resynchronization in the timer clock domain. The advantage is that neither the interconnect, nor the device that requested the write transaction is stalled.

For each register, a status bit is provided in the timer write-posted status (TIMER\_TWPS) register. In this mode, it is mandatory that software check this status bit before any write access. If a write is attempted to a register with a previous access pending, the previous access is discarded without notice.

The timer module updates the value of the timer counter register synchronously with the interface clock. Consequently, any read access to TIMER\_TCRR does not add any resynchronization latency; the current value is always available.

---

#### Note

Because the overflow IRQ is generated when the value of TIMER\_TCRR reaches 0xFFFF FFFF, and not when it changes its value to the value after overflow, it is necessary to wait a delay of ( $1 \times PS \times$  timer functional clock period) before any read access to TIMER\_TCRR to ensure a correct reading of its content.

---



---

#### Note

If TIMER\_TTGR register is written during a posted write to TIMER\_TCRR, the value to be written to TIMER\_TCRR will be discarded.

If a posted write to TIMER\_TCVR is started, the user must not write to TIMER\_TPIR or TIMER\_TNIR before the TIMER\_TCVR write is finished, because the value of TIMER\_TCVR is re-evaluated, so both the value to be written, and the recalculated value will be discarded.

---

If a write access is pending for a register, reading from this register does not yield a correct result. Software synchronization must be used to avoid incorrect results.

Functional frequency range:  $\text{freq}(\text{timer clock}) < \text{freq}(\text{interface clock})/4$ .

#### **12.10.3.4.12.1.2 Write Nonposting Synchronization Mode**

This mode is used if the `TIMER_TSICR[2] POSTED` bit is set to 0. It uses a nonposted write scheme to update any internal register. Therefore, the write transaction is not acknowledged on the configuration interface until the effective write operation occurs after the resynchronization in the timer functional clock domain. The drawback is that the interconnect and the device that requested the write transaction are stalled during this period.

The same full resynchronization scheme is used for a read transaction, and the same stall period applies. A register read following a write to the same register is always coherent.

This mode is functional regardless of the ratio between the configuration interface frequency and the timer clock frequency.

#### **12.10.3.4.12.2 Reading From Timer Counter Registers**

##### **Note**

LSB/MSB accesses cannot be interleaved (that is, the sequence LSB register 1, LSB register 2, MSB register 1, MSB register 2 is not supported).

The `TIMER_TCRR` is a 32-bit “atomic datum” and its 16-bit capture is done on the 16-bit LSB first to allow atomic LSB16 + MSB16 capture. This capture scheme is also performed for the `TIMER_TCAR1` and `TIMER_TCAR2` registers as they can be changed due to internal processes too.

##### **12.10.3.4.12.2.1 Read Posted**

This mode is functional regardless of the ratio between the configuration interface frequency and the functional clock frequency. The recommended functional frequency range is  $\text{freq}(\text{timer}) < \text{freq}(\text{interface clock})/4$ .

Read posted mode is used if `TIMER_TSICR[2] POSTED = 0x1` or `TIMER_TSICR[3] READ_MODE` is set to 0. This mode uses a posted-read scheme for reading any internal timer register. The read transaction is immediately acknowledged on the configuration interface, prior to the value to be read has been resynchronized. With this method, neither the interconnect nor the device that requested the read transaction are stalled.

Read posted mode applies to `TIMER_TCRR`, `TIMER_TCAR1`, `TIMER_TCAR2`, `TIMER_TCVR`, and `TIMER_TOWR`, which needs resynchronization from functional to interface clock domains.

Note that in Posted mode, if the `TIMER_TCRR` is read immediately after wake-up and the interface clock is off during idle state, then it is possible to get an old value from just before going to idle state due to the fact the interface clock is needed for synchronization.

In order to avoid this situation, another synchronization mechanism is used for the first read operation after idle state. The `TIMER_TSICR[4] READ_AFTER_IDLE` bit is used to enable/disable the mechanism.

When the synchronization mechanism is disabled (`READ_AFTER_IDLE` bit is set to 1), first read transaction takes only 2 interface clock cycles, but the read value of `TIMER_TCRR` could be wrong.

When the synchronization mechanism is enabled (`READ_AFTER_IDLE` bit is set to 0), first read value of `TIMER_TCRR` is correct, but the read transaction takes more than 2 interface clock cycles.

##### **12.10.3.4.12.2.2 Read Non-Posted**

This mode is functional regardless of the ratio between the configuration interface frequency and the functional clock frequency. Recommended functional frequency range is  $\text{freq}(\text{timer}) \geq \text{freq}(\text{interface clock})/4$ .

Read non-posted mode is used if `TIMER_TSICR[2] POSTED = 0x0` and `TIMER_TSICR[3] READ_MODE = 0x1`. This mode uses a non-posted read scheme for reading internal timer registers. The read transaction is not acknowledged on the configuration interface until the effective read operation occurs, after the resynchronization

in the timer clock domain. The result is that both the interconnect and the device that requested the read transaction are stalled during this period.

This mode applies to `TIMER_TCRR`, `TIMER_TCAR1`, `TIMER_TCAR2`, `TIMER_TCVR`, and `TIMER_TOWR`, which need resynchronization from functional to interface clock domains.

#### **12.10.3.4.13 Timer Posted Mode Selection**

A choice between two synchronization modes is made taking into account the frequency ratio and the stall periods that can be supported by the system, without impacting the global performance.

The posted mode selection applies only to registers that require synchronization on or from the timer clock domain. For write operation, the registers affected by posted and non-posted selection are `TIMER_TCLR`, `TIMER_TLDR`, `TIMER_TCRR`, `TIMER_TTGR`, `TIMER_TMAR`, `TIMER_TPIR`, `TIMER_TNIR`, `TIMER_TCVR`, `TIMER_TOCR`, and `TIMER_TOWR`. For read operation, the registers affected by this selection are: `TIMER_TCRR`, `TIMER_TCAR1`, `TIMER_TCAR2`, `TIMER_TCVR`, and `TIMER_TOWR`.

The interface clock domain synchronous registers `TIMER_TIDR`, `TIMER_TIOCP_CFG`, `TIMER_IRQSTATUS`, `TIMER_IRQSTATUS_SET`, `TIMER_IRQWAKEEN`, `TIMER_TWPS`, and `TIMER_TSICR` are not affected by posted and non-posted mode selection. The operation (read or write) is effective and acknowledged after one interface clock cycle from the command assertion.

The configuration of posted or non-posted mode can be changed (overwritten) by software by writing in `TIMER_TSICR[2]` `POSTED` bit. The `TIMER_TSICR[3]` `READ_MODE` defines how the read operation is performed when the module is configured in non-posted mode (see `TIMER_TSICR`). The following cases are possible:

- `TIMER_TSICR[2]` `POSTED` = 0x1 and `TIMER_TSICR[3]` `READ_MODE` = x (don't care): read and write operations are expected in posted mode.
- `TIMER_TSICR[2]` `POSTED` = 0x0 and `TIMER_TSICR[3]` `READ_MODE` = 0x0: the write operation is executed in non-posted mode and read is executed in posted mode.
- `TIMER_TSICR[2]` `POSTED` = 0x0 and `TIMER_TSICR[3]` `READ_MODE` = 0x1: write is executed in non-posted mode and read is executed in non-posted mode.

### 12.10.3.5 Timers Low-Level Programming Models

This section describes the low-level hardware programming sequences for the configuration and use of the module.

#### 12.10.3.5.1 Timer Global Initialization

##### 12.10.3.5.1.1 Global Initialization of Surrounding Modules

This section identifies the requirements for initializing the surrounding modules when the timer module is to be used for the first time after a device reset. This initialization of surrounding modules is based on the integration and environment of the timer. For more information, see *Timers Integration* and *Timers Environment*. [Table 12-1592](#) summarizes the surrounding modules.

**Table 12-1592. Global Initialization of Surrounding Modules**

Surrounding Modules	Comments
C66SS0_INTRTR0 and C66SS1_INTRTR0	Device INTCs must be configured to enable the interrupt request generation. For information about enabling C66SS0_INTRTR0 and C66SS1_INTRTR0 interrupts, see <i>Interrupts</i> .
COMPUTE_CLUSTER0	Device INTCs must be configured to enable the interrupt request generation. For information about enabling COMPUTE_CLUSTER0 interrupts, see <i>Interrupts</i> .
MAIN2MCU_LVL_INTRTR0	Device INTCs must be configured to enable the interrupt request generation. For information about enabling MAIN2MCU_LVL_INTRTR0 interrupts, see <i>Interrupts</i> .
R5FSS0_INTRTR0 and R5FSS1_INTRTR0	Device INTCs must be configured to enable the interrupt request generation. For information about enabling R5FSS0_INTRTR0 and R5FSS1_INTRTR0 interrupts, see <i>Interrupts</i> .
R5FSS0_CORE0/1 and R5FSS1_CORE0/1	Device INTCs must be configured to enable the interrupt request generation. For information about enabling R5FSS0_CORE0/1 and R5FSS1_CORE0/1 interrupts, see <i>Interrupts</i> .
PLLCTRL0 and WKUP_PLLCTRL0	PLL controller's configuration must be done to enable the module clocks. For more information, see <i>Clocking</i> .
TIMERCLK0 MUX to TIMERCLK19 MUX	TIMERCLK0 MUX to TIMERCLK19 MUX configuration must be done to enable the module clocks. For more information, see <i>Control Module (CTRL_MMR)</i> .

#### 12.10.3.5.1.2 Timer Module Global Initialization

##### 12.10.3.5.1.2.1 Main Sequence – Timer Module Global Initialization

[Table 12-1593](#) identifies the main steps for initializing the timer module when the module is to be used for the first time.

**Table 12-1593. Timer Module Global Initialization**

Step	Register/Bit Field/Programming Model	Value
Execute software reset.	TIMER_TIOCP_CFG[0] SOFTRESET	0x1
Wait until reset release?	TIMER_TIOCP_CFG[0] SOFTRESET	0x0
Configure idle mode.	TIMER_TIOCP_CFG[3-2] IDLEMODE	0x-
Enable wake-up interrupt events.	TIMER_IRQWAKEEN[2-0]	0x-
Select posted mode.	TIMER_TSICR[2] POSTED	0x-

#### 12.10.3.5.2 Timer Operational Mode Configuration

##### 12.10.3.5.2.1 Timer Mode

##### 12.10.3.5.2.1.1 Main Sequence – Timer Mode Configuration

[Table 12-1594](#) lists the steps in the timer mode configuration.

**Table 12-1594. Timer Mode Configuration**

Step	Register/Bit Field/Programming Model	Value
Select autoreload mode.	TIMER_TCLR[1] AR	0x-
Set prescale timer value.	TIMER_TCLR[4-2] PTV	0x-
Enable prescaler.	TIMER_TCLR[5] PRE	0x1
Enable overflow interrupt.	TIMER_IRQSTATUS_SET[1] OVF_EN_FLAG	0x1

**Table 12-1594. Timer Mode Configuration (continued)**

Step	Register/Bit Field/Programming Model	Value
Load timer counter value.	TIMER_TCRR	0x-
Load timer load value.	TIMER_TLDR	0x-
Start the timer.	TIMER_TCLR[0] ST	0x1

### 12.10.3.5.2.2 Timer Compare Mode

#### 12.10.3.5.2.2.1 Main Sequence – Timer Compare Mode Configuration

Table 12-1595 lists the steps in the timer compare mode configuration.

**Table 12-1595. Timer Compare Mode Configuration**

Step	Register/Bit Field/Programming Model	Value
Select autoreload mode.	TIMER_TCLR[1] AR	0x-
Set prescale timer value.	TIMER_TCLR[4-2] PTV	0x-
Enable prescaler.	TIMER_TCLR[5] PRE	0x1
Enable match interrupt.	TIMER_IRQSTATUS_SET[0] MAT_EN_FLAG	0x1
Load timer counter value.	TIMER_TCRR	0x-
Load timer compare value.	TIMER_TMAR	0x-
Enable compare mode.	TIMER_TCLR[6] CE	0x1
Start the timer.	TIMER_TCLR[0] ST	0x1

### 12.10.3.5.2.3 Timer Capture Mode

#### 12.10.3.5.2.3.1 Main Sequence – Timer Capture Mode Configuration

Table 12-1596 lists the steps in the timer capture mode configuration.

**Table 12-1596. Timer Capture Mode Configuration**

Step	Register/Bit Field/Programming Model	Value
Initialize capture mode.	See <a href="#">Section 12.10.3.5.2.3.2</a> .	
Enable capture interrupt.	TIMER_IRQSTATUS_SET[2] TCAR_EN_FLAG	0x1
Start the timer.	TIMER_TCLR[0] ST	0x1
Detect event.	See <a href="#">Section 12.10.3.5.2.3.3</a> .	

#### 12.10.3.5.2.3.2 Subsequence – Initialize Capture Mode

Table 12-1597 lists the steps to initialize capture mode.

**Table 12-1597. Initialize Capture Mode**

Step	Register/Bit Field/Programming Model	Value
Select autoreload mode.	TIMER_TCLR[1] AR	0x-
Set prescale timer value.	TIMER_TCLR[4-2] PTV	0x-
Enable prescaler.	TIMER_TCLR[5] PRE	0x1
Select TIMER[19-0] or MCU_TIMER[9-0] Capture input at device pins TIMER_IO[7-0] for TIMER[19-0] or at pins MCU_TIMER_IO[9-0] for MCU_TIMER[9-0].	TIMER_TCLR[14] GPO_CFG	0x1
Select single or second event capture.	TIMER_TCLR[13] CAPT_MODE	0x-
Select transition capture mode.	TIMER_TCLR[9-8] TCM	0x-

#### 12.10.3.5.2.3.3 Subsequence – Detect Event

Table 12-1598 lists the steps in detecting an event.

**Table 12-1598. Detect Event**

Step	Register/Bit Field/Programming Model	Value
Wait until event detected?	TIMER_IRQSTATUS[2] TCAR_IT_FLAG	= 0x1
Read timer capture value.	TIMER_TCAR1 and/or TIMER_TCAR2	
Clear capture interrupt request.	TIMER_IRQSTATUS[2] TCAR_IT_FLAG	0x1

#### 12.10.3.5.2.4 Timer PWM Mode

##### 12.10.3.5.2.4.1 Main Sequence – Timer PWM Mode Configuration

[Table 12-1599](#) lists the steps in the timer PWM mode configuration.

**Table 12-1599. Timer PWM Mode Configuration**

Step	Register/Bit Field/Programming Model	Value
Select autoreload mode.	TIMER_TCLR[1] AR	0x-
Set prescale timer value.	TIMER_TCLR[4-2] PTV	0x-
Enable prescaler.	TIMER_TCLR[5] PRE	0x1
Select trigger output mode.	TIMER_TCLR[11-10] TRG	0x-
Select pulse or toggle modulation PWM mode.	TIMER_TCLR[12] PT	0x-
Select TIMER[19-0] or MCU_TIMER[9-0] PWM output at device pins TIMER_IO[7-0] for TIMER[19-0] or at pins MCU_TIMER_IO[9-0] for MCU_TIMER[9-0].	TIMER_TCLR[14] GPO_CFG	0x0
Configure PWM output pin default value.	TIMER_TCLR[7] SCPWM	0x-
Load timer load value.	TIMER_TLDR	0x-
Load timer compare value.	TIMER_TMAR	0x-
Enable compare.	TIMER_TCLR[6] CE	0x1
Start the timer.	TIMER_TCLR[0] ST	0x1

## 12.11 Internal Diagnostics Modules

This section describes the internal diagnostics modules in the device.

### 12.11.1 Dual Clock Comparator (DCC)

This section describes the Dual Clock Comparator (DCC) modules in the device.

#### 12.11.1.1 DCC Overview

The Dual Clock Comparator (DCC) is used to determine the accuracy of a clock signal during the time execution of an application. Specifically, the DCC is designed to detect drifts from the expected clock frequency. The desired accuracy can be programmed based on calculation for each application. The DCC measures the frequency of a selectable clock source using another input clock as a reference.

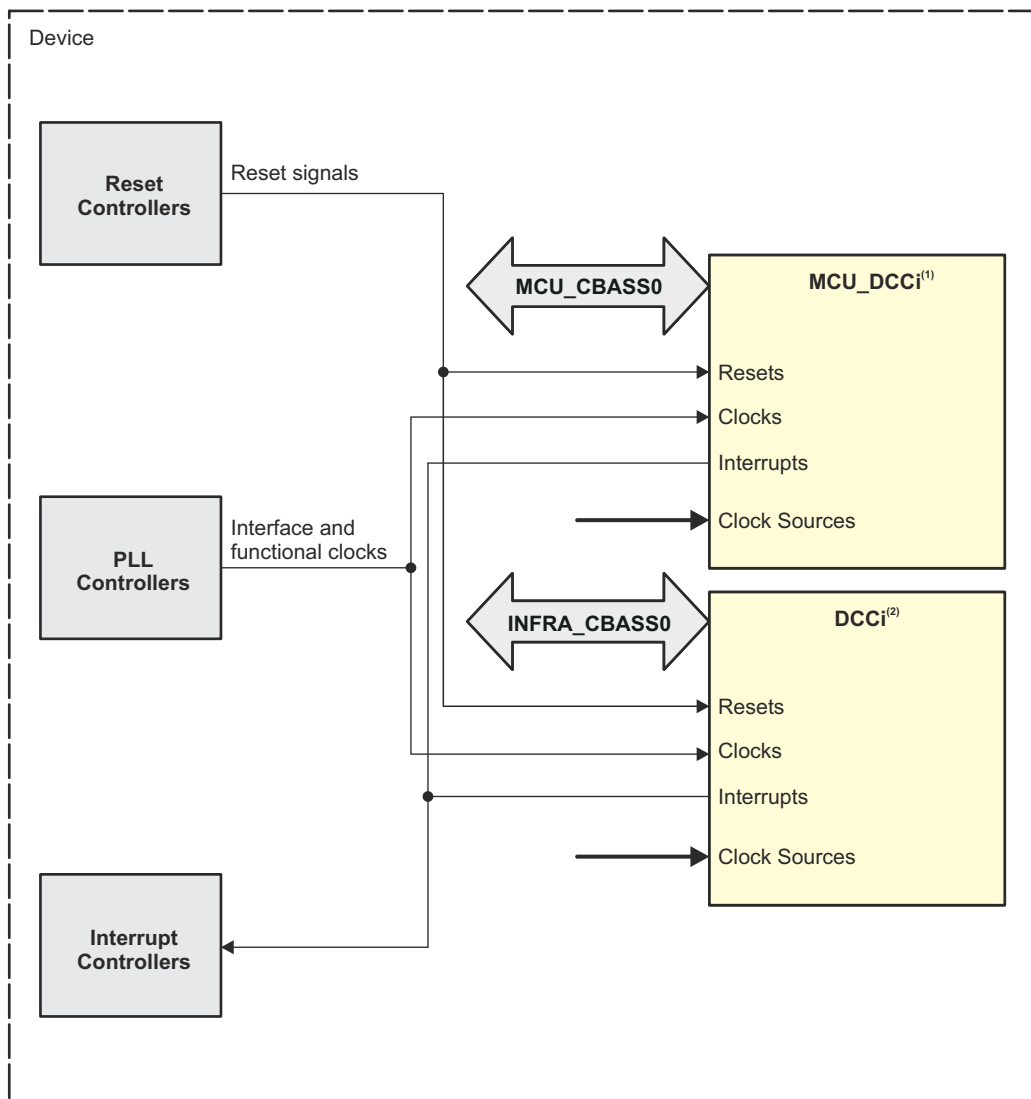
The device has sixteen instances of DCC modules. [Table 12-1600](#) shows DCC modules allocation across device domains.

**Table 12-1600. DCC Modules Allocation Across Device Domains**

Module Instance	Domain		
	WKUP	MCU	MAIN
MCU_DCC0	-	✓	-
MCU_DCC1	-	✓	-
MCU_DCC2	-	✓	-
DCC0	-	-	✓
DCC1	-	-	✓
DCC2	-	-	✓
DCC3	-	-	✓
DCC4	-	-	✓
DCC5	-	-	✓
DCC6	-	-	✓
DCC7	-	-	✓
DCC8	-	-	✓
DCC9	-	-	✓
DCC10	-	-	✓
DCC11	-	-	✓
DCC12	-	-	✓

[Figure 12-1222](#) shows the DCC modules overview.





dcc-001

- A.  $i = 0$  to  $2$
- B.  $i = 0$  to  $12$

**Figure 12-1222. DCC Modules Overview**

#### 12.11.1.1.1 DCC Features

The DCC uses two independent clock sources to detect when one is out of specification. Each DCC module implements the following features:

- Two independent counter blocks count clock pulses from each clock source
- Each counter block is programmable, however, for proper operation the counters must be programmed with seed values that respect the ratio of the two clock frequencies
- Configurable timebase for error signal
- Error signal generation when one of the clocks is out of specification
- Clock frequency measurement

#### 12.11.1.1.2 DCC Not Supported Features

The DCC does not support the following features:

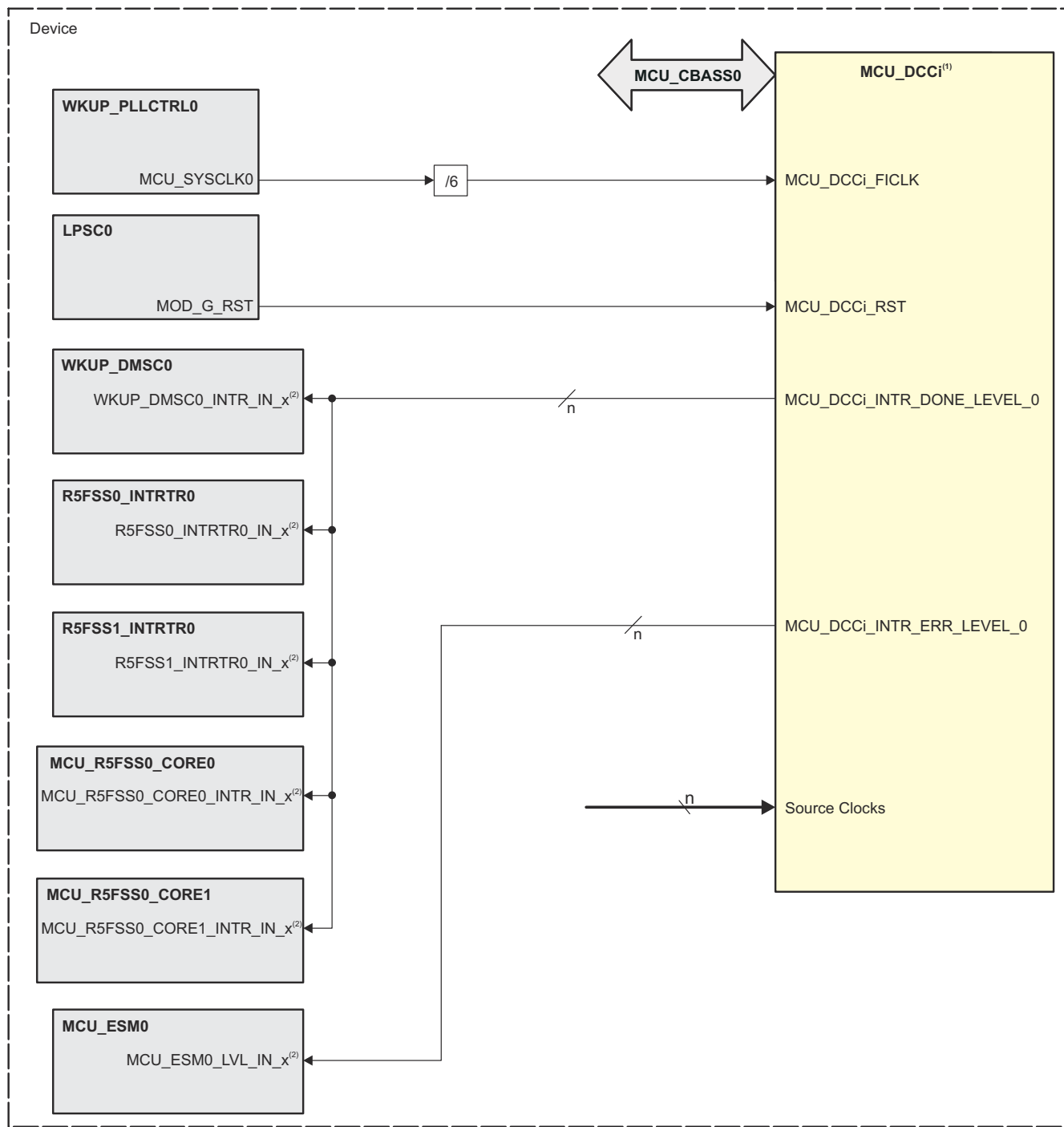
- Debug suspend functionality depreciated . Software will have to disable DCCs if it starts messing with clocks during debug.

### 12.11.1.2 DCC Integration

This section describes the DCC integration in the device, including information about clocks, resets, and hardware requests.

#### 12.11.1.2.1 DCC Integration in MCU Domain

There are three DCC modules integrated in the device MCU domain - MCU\_DCC0 through MCU\_DCC2. [Figure 12-1223](#) shows the integration of MCU\_DCC modules.



dcc-002

A.  $i = 0$  to 2

B. x = Interrupt port index, see [Table 12-1603](#)

**Figure 12-1223. MCU\_DCC Integration**

[Table 12-1601](#) through [Table 12-1603](#) summarize the integration of DCC in the device MCU domain.

**Table 12-1601. MCU\_DCC Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
MCU_DCC0	WKUP_PSC0	PD0	LPSC0	MCU_CBASS0
MCU_DCC1	WKUP_PSC0	PD0	LPSC0	MCU_CBASS0
MCU_DCC2	WKUP_PSC0	PD0	LPSC0	MCU_CBASS0

**Table 12-1602. MCU\_DCC Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
MCU_DCC0	MCU_DCC0_FICLK	MCU_SYSCLK0/6	WKUP_PLLCTRL0	MCU_DCC0 interface and functional clock
MCU_DCC1	MCU_DCC1_FICLK	MCU_SYSCLK0/6	WKUP_PLLCTRL0	MCU_DCC1 interface and functional clock
MCU_DCC2	MCU_DCC2_FICLK	MCU_SYSCLK0/6	WKUP_PLLCTRL0	MCU_DCC2 interface and functional clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
MCU_DCC0	MCU_DCC0_RST	MOD_G_RST	LPSC0	MCU_DCC0 asynchronous module reset
MCU_DCC1	MCU_DCC1_RST	MOD_G_RST	LPSC0	MCU_DCC1 asynchronous module reset
MCU_DCC2	MCU_DCC2_RST	MOD_G_RST	LPSC0	MCU_DCC2 asynchronous module reset

**Table 12-1603. MCU\_DCC Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCU_DCC0	MCU_DCC0_INTR_DONE_LVL_0	WKUP_DMSC0_INTR_IN_44	WKUP_DMSC0	MCU_DCC0 one-shot mode complete interrupt	Level
		R5FSS0_INTRTR0_IN_120	R5FSS0_INTRTR0	MCU_DCC0 one-shot mode complete interrupt	Level
		R5FSS1_INTRTR0_IN_120	R5FSS1_INTRTR0	MCU_DCC0 one-shot mode complete interrupt	Level
		MCU_R5FSS0_CORE0_INTR_IN_50	MCU_R5FSS0_CORE0	MCU_DCC0 one-shot mode complete interrupt	Level
		MCU_R5FSS0_CORE1_INTR_IN_50	MCU_R5FSS0_CORE1	MCU_DCC0 one-shot mode complete interrupt	Level
	MCU_DCC0_INTR_ERR_LEV_0	MCU_ESM0_LVL_IN_86	MCU_ESM0	MCU_DCC0 error interrupt	Level

MCU_DCC1	MCU_DCC1_INTR_DONE_LEVEL_0	WKUP_DMSC0_INTR_IN_45	WKUP_DMSC0	MCU_DCC1 one-shot mode complete interrupt	Level
		R5FSS0_INTRTR0_IN_121	R5FSS0_INTRTR0	MCU_DCC1 one-shot mode complete interrupt	Level
		R5FSS1_INTRTR0_IN_121	R5FSS1_INTRTR0	MCU_DCC1 one-shot mode complete interrupt	Level
		MCU_R5FSS0_CORE0_INTR_IN_51	MCU_R5FSS0_CORE0	MCU_DCC1 one-shot mode complete interrupt	Level
		MCU_R5FSS0_CORE1_INTR_IN_51	MCU_R5FSS0_CORE1	MCU_DCC1 one-shot mode complete interrupt	Level
	MCU_DCC1_INTR_ERR_LEVEL_0	MCU_ESM0_LVL_IN_87	MCU_ESM0	MCU_DCC1 error interrupt	Level
MCU_DCC2	MCU_DCC2_INTR_DONE_LEVEL_0	WKUP_DMSC0_INTR_IN_46	WKUP_DMSC0	MCU_DCC2 one-shot mode complete interrupt	Level
		R5FSS0_INTRTR0_IN_122	R5FSS0_INTRTR0	MCU_DCC2 one-shot mode complete interrupt	Level
		R5FSS1_INTRTR0_IN_122	R5FSS1_INTRTR0	MCU_DCC2 one-shot mode complete interrupt	Level
		MCU_R5FSS0_CORE0_INTR_IN_52	MCU_R5FSS0_CORE0	MCU_DCC2 one-shot mode complete interrupt	Level
		MCU_R5FSS0_CORE1_INTR_IN_52	MCU_R5FSS0_CORE1	MCU_DCC2 one-shot mode complete interrupt	Level
	MCU_DCC2_INTR_ERR_LEVEL_0	MCU_ESM0_LVL_IN_88	MCU_ESM0	MCU_DCC2 error interrupt	Level

### Table 12-1604. MCU\_DCC Input Source Clock Mapping

[illegible]

**Table 12-1604. MCU\_DCC Input Source Clock Mapping (continued)**

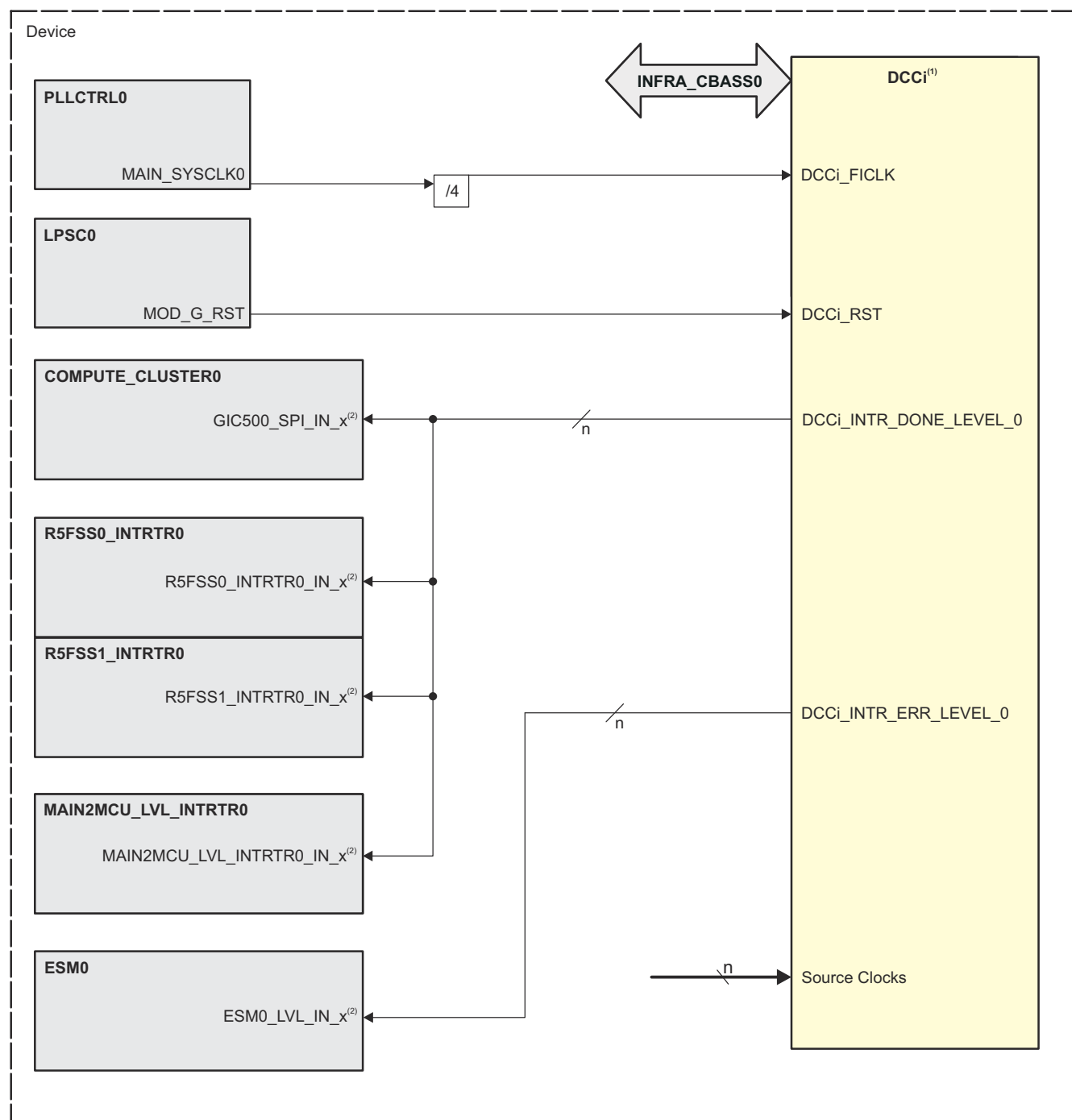
		MCU_DCC0												MCU_DCC1												MCU_DCC2													
		Input0			Input1									Input0			Input1									Input0			Input1										
		MUX0			MUX1									MUX0			MUX1									MUX0			MUX1										
		0-4, 6-9, B	A	5	0	1	2	3	4	5	6	7	8	9-F	0-4, 6-9, B	A	5	0	1	2	3	4	5	6	7	8	9-F	0-4, 6-9, B	A	5	0	1	2	3	4	5	6	7	8
DCC_CLKSRC0 / DCC_CLKSRC1 value:		-F											-F													-F													
Clock Source	Input:	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 0	C L O C K 1	C L O C K 2	C L O C K 3	C L O C K 4	C L O C K 5	C L O C K 6	C L O C K 7	F I C L K	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 0	C L O C K 1	C L O C K 2	C L O C K 3	C L O C K 4	C L O C K 5	C L O C K 6	C L O C K 7	F I C L K	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 0	C L O C K 1	C L O C K 2	C L O C K 3	C L O C K 4	C L O C K 5	C L O C K 6	C L O C K 7	F I C L K		
CLK_12M_RC	Internal 12.5MHz RC oscillator (Always ON)		✓											✓														✓									✓		
LFXOSC_CLKOUT	External 32768Hz Clock Input to Wkup/ MCU(Always ON)										✓			✓																									
MCU_EXT_REFCLK 0	External Misc clock input to Wkup/MCU											✓															✓												
MCU_PLLCTRL																																							
MCU_SYSCLK/3	CBASS Clock (CLK1/3), Pulsar Interface Clock (CLK1/3), DMSC Clock (CLK1/3)			✓																									✓										
MCU_SYSCLK/6	CBASS Clock (CLK1/6)												✓													✓												✓	
MCU_PLL0																																							
MCU_PLL0_HSDIV0_CLKOUT_DIV4	Input to MCU PLLCTRL																																						
MCU_PLL0_HSDIV1_CLKOUT	ADC (Optional)																																						
MCU_PLL1																																							
MCU_PLL1_HSDIV0_CLKOUT_DIV2	MCU SA2_UL PKA (400MHz)				✓																																		
MCU_PLL1_HSDIV1_CLKOUT	ADC					✓																																	
MCU_PLL1_HSDIV2_CLKOUT	MCAN						✓																																
MCU_PLL1_HSDIV3_CLKOUT	USART, I2C							✓																															
MCU_PLL1_HSDIV4_CLKOUT	QSPI (SDR) QSPI (SDR - Stretch Goal) QSPI (DDR) OSPI (SDR/DDR) OSPI (SDR/DDR - Stretch Goal) QSPI SDR Ref Clock (IO rate = 100)								✓																														
MCU_PLL2																																							
MCU_PLL2_HSDIV0_CLKOUT	CPSW2G (250, 125, 50, 5); , McSPI (50MHz)														✓																								

**Table 12-1604. MCU\_DCC Input Source Clock Mapping (continued)**

		MCU_DCC0												MCU_DCC1												MCU_DCC2																
		Input0		Input1										Input0		Input1										Input0		Input1														
		MUX0		MUX1										MUX0		MUX1										MUX0		MUX1														
		0-4, 6-9, B -F	A	5	0	1	2	3	4	5	6	7	8	9-F	0-4, 6-9, B -F	A	5	0	1	2	3	4	5	6	7	8	9-F	0-4, 6-9, B -F	A	5	0	1	2	3	4	5	6	7	8	9-F		
DCC_CLKSRC0 / DCC_CLKSRC1 value:																																										
Clock Source	Input:	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 0	C L O C K 1	C L O C K 2	C L O C K 3	C L O C K 4	C L O C K 5	C L O C K 6	C L O C K 7	F I C L K	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]		C L O C K 0	C L O C K 1	C L O C K 2	C L O C K 3	C L O C K 4	C L O C K 5	C L O C K 6	C L O C K 7	F I C L K	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]		C L O C K 0	C L O C K 1	C L O C K 2	C L O C K 3	C L O C K 4	C L O C K 5	C L O C K 6	C L O C K 7	F I C L K			
MCU_PLL2_HSDIV1_CLKOUT_DIV2	CPTS Clock Mux; Exported to Main Domain (Presently not used in MCU Domain)																✓																									
MCU_PLL2_HSDIV2_CLKOUT	DMTIMER																✓																									
MCU_PLL2_HSDIV3_CLKOUT	MCAN (Optional)																✓																									
MCU_PLL2_HSDIV4_CLKOUT	HYPERFLASH (166 DDR), OSPI optional ref clock to get 166MHz																	✓																								
OSPI0_LBCLKI	IO Loopback Clock input																								✓																	
OSPI1_LBCLKI	IO Loopback Clock input																																						✓			
MCU_CPTS_REFCLK	IO reference clock input																									✓																
MCU_RMII1_REFCLK	IO Reference clock input																																			✓						
MCU_RGMII1_RXC	IO Receive clock input																																				✓					
MAIN_PLL1_HSDIV5_CLKOUT (192MHz)	USART Clock (optional clock to support 12Mbaud rate)																																						✓			

### 12.11.1.2.2 DCC Integration in MAIN Domain

There are thirteen DCC modules integrated in the device MAIN domain - DCC0 through DCC12. [Figure 12-1224](#) shows the integration of DCC modules.



dcc-003

- A.  $i = 0$  to  $12$   
 B.  $x =$  Interrupt port index, see [Table 12-1607](#)

**Figure 12-1224. DCC Integration**

[Table 12-1605](#) through [Table 12-1607](#) summarize the integration of DCC in the device MAIN domain.

**Table 12-1605. DCC Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect



**Table 12-1605. DCC Integration Attributes (continued)**

DCC0	PSC0	PD0	LPSC0	INFRA_CBASS0
DCC1	PSC0	PD0	LPSC0	INFRA_CBASS0
DCC2	PSC0	PD0	LPSC0	INFRA_CBASS0
DCC3	PSC0	PD0	LPSC0	INFRA_CBASS0
DCC4	PSC0	PD0	LPSC0	INFRA_CBASS0
DCC5	PSC0	PD0	LPSC0	INFRA_CBASS0
DCC6	PSC0	PD0	LPSC0	INFRA_CBASS0
DCC7	PSC0	PD0	LPSC0	INFRA_CBASS0
DCC8	PSC0	PD0	LPSC0	INFRA_CBASS0
DCC9	PSC0	PD0	LPSC0	INFRA_CBASS0
DCC10	PSC0	PD0	LPSC0	INFRA_CBASS0
DCC11	PSC0	PD0	LPSC0	INFRA_CBASS0
DCC12	PSC0	PD0	LPSC0	INFRA_CBASS0

**Table 12-1606. DCC Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
DCC0	DCC0_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	DCC0 interface and functional clock
DCC1	DCC1_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	DCC1 interface and functional clock
DCC2	DCC2_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	DCC2 interface and functional clock
DCC3	DCC3_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	DCC3 interface and functional clock
DCC4	DCC4_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	DCC4 interface and functional clock
DCC5	DCC5_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	DCC5 interface and functional clock
DCC6	DCC6_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	DCC6 interface and functional clock
DCC7	DCC7_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	DCC7 interface and functional clock
DCC8	DCC7_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	DCC7 interface and functional clock
DCC9	DCC7_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	DCC7 interface and functional clock
DCC10	DCC7_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	DCC7 interface and functional clock
DCC11	DCC7_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	DCC7 interface and functional clock
DCC12	DCC7_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	DCC7 interface and functional clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
DCC0	DCC0_RST	MOD_G_RST	LPSC0	DCC0 asynchronous module reset
DCC1	DCC1_RST	MOD_G_RST	LPSC0	DCC1 asynchronous module reset

**Table 12-1606. DCC Clocks and Resets (continued)**

DCC2	DCC2_RST	MOD_G_RST	LPSC0	DCC2 asynchronous module reset
DCC3	DCC3_RST	MOD_G_RST	LPSC0	DCC3 asynchronous module reset
DCC4	DCC4_RST	MOD_G_RST	LPSC0	DCC4 asynchronous module reset
DCC5	DCC5_RST	MOD_G_RST	LPSC0	DCC5 asynchronous module reset
DCC6	DCC6_RST	MOD_G_RST	LPSC0	DCC6 asynchronous module reset
DCC7	DCC7_RST	MOD_G_RST	LPSC0	DCC7 asynchronous module reset
DCC8	DCC7_RST	MOD_G_RST	LPSC0	DCC7 asynchronous module reset
DCC9	DCC7_RST	MOD_G_RST	LPSC0	DCC7 asynchronous module reset
DCC10	DCC7_RST	MOD_G_RST	LPSC0	DCC7 asynchronous module reset
DCC11	DCC7_RST	MOD_G_RST	LPSC0	DCC7 asynchronous module reset
DCC12	DCC7_RST	MOD_G_RST	LPSC0	DCC7 asynchronous module reset

**Table 12-1607. DCC Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DCC0	DCC0_INTR_DONE_LEVEL_0	GIC500_SPI_IN_328	COMPUTE_CLUSTER0	DCC0 one-shot mode complete interrupt	Level
		R5FSS0_INTRTR0_IN_279	R5FSS0_INTRTR0	DCC0 one-shot mode complete interrupt	Level
		R5FSS1_INTRTR0_IN_279	R5FSS1_INTRTR0	DCC0 one-shot mode complete interrupt	Level
		MAIN2MCU_LVL_INTRTR0_IN_88	MAIN2MCU_LVL_INTRTR0	DCC0 one-shot mode complete interrupt	Level
	DCC0_INTR_ERR_LEVEL_0	ESM0_LVL_IN_104	ESM0	DCC0 error interrupt	Level
DCC1	DCC1_INTR_DONE_LEVEL_0	GIC500_SPI_IN_329	COMPUTE_CLUSTER0	DCC1 one-shot mode complete interrupt	Level
		R5FSS0_INTRTR0_IN_280	R5FSS0_INTRTR0	DCC1 one-shot mode complete interrupt	Level
		R5FSS1_INTRTR0_IN_280	R5FSS1_INTRTR0	DCC1 one-shot mode complete interrupt	Level
		MAIN2MCU_LVL_INTRTR0_IN_89	MAIN2MCU_LVL_INTRTR0	DCC1 one-shot mode complete interrupt	Level
	DCC1_INTR_ERR_LEVEL_0	ESM0_LVL_IN_105	ESM0	DCC1 error interrupt	Level
DCC2	DCC2_INTR_DONE_LEVEL_0	GIC500_SPI_IN_330	COMPUTE_CLUSTER0	DCC2 one-shot mode complete interrupt	Level

**Table 12-1607. DCC Hardware Requests (continued)**

		R5FSS0_INTRTR0_IN_281	R5FSS0_INTRTR0	DCC2 one-shot mode complete interrupt	Level
		R5FSS1_INTRTR0_IN_281	R5FSS1_INTRTR0	DCC2 one-shot mode complete interrupt	Level
		MAIN2MCU_LVL_INTRTR0_IN_90	MAIN2MCU_LVL_INTRTR0	DCC2 one-shot mode complete interrupt	Level
	DCC2_INTR_ERR_LEVEL_0	ESM0_LVL_IN_106	ESM0	DCC2 error interrupt	Level
DCC3	DCC3_INTR_DONE_LEVEL_0	GIC500_SPI_IN_331	COMPUTE_CLUSTER0	DCC3 one-shot mode complete interrupt	Level
		R5FSS0_INTRTR0_IN_282	R5FSS0_INTRTR0	DCC3 one-shot mode complete interrupt	Level
		R5FSS1_INTRTR0_IN_282	R5FSS1_INTRTR0	DCC3 one-shot mode complete interrupt	Level
		MAIN2MCU_LVL_INTRTR0_IN_91	MAIN2MCU_LVL_INTRTR0	DCC3 one-shot mode complete interrupt	Level
	DCC3_INTR_ERR_LEVEL_0	ESM0_LVL_IN_107	ESM0	DCC3 error interrupt	Level
DCC4	DCC4_INTR_DONE_LEVEL_0	GIC500_SPI_IN_332	COMPUTE_CLUSTER0	DCC4 one-shot mode complete interrupt	Level
		R5FSS0_INTRTR0_IN_283	R5FSS0_INTRTR0	DCC4 one-shot mode complete interrupt	Level
		R5FSS1_INTRTR0_IN_283	R5FSS1_INTRTR0	DCC4 one-shot mode complete interrupt	Level
		MAIN2MCU_LVL_INTRTR0_IN_92	MAIN2MCU_LVL_INTRTR0	DCC4 one-shot mode complete interrupt	Level
	DCC4_INTR_ERR_LEVEL_0	ESM0_LVL_IN_108	ESM0	DCC4 error interrupt	Level
DCC5	DCC5_INTR_DONE_LEVEL_0	GIC500_SPI_IN_333	COMPUTE_CLUSTER0	DCC5 one-shot mode complete interrupt	Level
		R5FSS0_INTRTR0_IN_284	R5FSS0_INTRTR0	DCC5 one-shot mode complete interrupt	Level
		R5FSS1_INTRTR0_IN_284	R5FSS1_INTRTR0	DCC5 one-shot mode complete interrupt	Level
		MAIN2MCU_LVL_INTRTR0_IN_93	MAIN2MCU_LVL_INTRTR0	DCC5 one-shot mode complete interrupt	Level
	DCC5_INTR_ERR_LEVEL_0	ESM0_LVL_IN_109	ESM0	DCC5 error interrupt	Level
DCC6	DCC6_INTR_DONE_LEVEL_0	GIC500_SPI_IN_334	COMPUTE_CLUSTER0	DCC6 one-shot mode complete interrupt	Level
		R5FSS0_INTRTR0_IN_285	R5FSS0_INTRTR0	DCC6 one-shot mode complete interrupt	Level
		R5FSS1_INTRTR0_IN_285	R5FSS1_INTRTR0	DCC6 one-shot mode complete interrupt	Level
		MAIN2MCU_LVL_INTRTR0_IN_94	MAIN2MCU_LVL_INTRTR0	DCC6 one-shot mode complete interrupt	Level

**Table 12-1607. DCC Hardware Requests (continued)**

	DCC6_INTR_ERR_LEVEL_0	ESM0_LVL_IN_110	ESM0	DCC6 error interrupt	Level
DCC7	DCC7_INTR_DONE_LEVEL_0	GIC500_SPI_IN_335	COMPUTE_CLUSTER0	DCC7 one-shot mode complete interrupt	Level
		R5FSS0_INTRTR0_IN_286 R5FSS0_INTRTR0_IN_BIT0_111	R5FSS0_INTRTR0	DCC7 one-shot mode complete interrupt	Level
		R5FSS1_INTRTR0_IN_286	R5FSS1_INTRTR0	DCC7 one-shot mode complete interrupt	Level
		MAIN2MCU_LVL_INTRTR0_IN_95	MAIN2MCU_LVL_INTRTR0	DCC7 one-shot mode complete interrupt	Level
	DCC7_INTR_ERR_LEVEL_0	ESM0_LVL_IN_111	ESM0	DCC7 error interrupt	Level
DCC8	DCC8_INTR_DONE_LEVEL_0	GIC500_SPI_IN_336	COMPUTE_CLUSTER0	DCC8 one-shot mode complete interrupt	Level
		R5FSS0_INTRTR0_IN_303	R5FSS0_INTRTR0	DCC8 one-shot mode complete interrupt	Level
		R5FSS1_INTRTR0_IN_303	R5FSS1_INTRTR0	DCC8 one-shot mode complete interrupt	Level
		MAIN2MCU_LVL_INTRTR0_IN_208	MAIN2MCU_LVL_INTRTR0	DCC8 one-shot mode complete interrupt	Level
	DCC8_INTR_ERR_LEVEL_0	ESM0_LVL_IN_112	ESM0	DCC8 error interrupt	Level
DCC9	DCC9_INTR_DONE_LEVEL_0	GIC500_SPI_IN_337	COMPUTE_CLUSTER0	DCC9 one-shot mode complete interrupt	Level
		R5FSS0_INTRTR0_IN_304	R5FSS0_INTRTR0	DCC9 one-shot mode complete interrupt	Level
		R5FSS1_INTRTR0_IN_304	R5FSS1_INTRTR0	DCC9 one-shot mode complete interrupt	Level
		MAIN2MCU_LVL_INTRTR0_IN_209	MAIN2MCU_LVL_INTRTR0	DCC9 one-shot mode complete interrupt	Level
	DCC9_INTR_ERR_LEVEL_0	ESM0_LVL_IN_113	ESM0	DCC9 error interrupt	Level
DCC10	DCC10_INTR_DONE_LEVEL_0	GIC500_SPI_IN_338	COMPUTE_CLUSTER0	DCC10 one-shot mode complete interrupt	Level
		R5FSS0_INTRTR0_IN_305	R5FSS0_INTRTR0	DCC10 one-shot mode complete interrupt	Level
		R5FSS1_INTRTR0_IN_305	R5FSS1_INTRTR0	DCC10 one-shot mode complete interrupt	Level
		MAIN2MCU_LVL_INTRTR0_IN_210	MAIN2MCU_LVL_INTRTR0	DCC10 one-shot mode complete interrupt	Level
	DCC10_INTR_ERR_LEVEL_0	ESM0_LVL_IN_114	ESM0	DCC10 error interrupt	Level
DCC11	DCC11_INTR_DONE_LEVEL_0	GIC500_SPI_IN_339	COMPUTE_CLUSTER0	DCC11 one-shot mode complete interrupt	Level
		R5FSS0_INTRTR0_IN_306	R5FSS0_INTRTR0	DCC11 one-shot mode complete interrupt	Level

**Table 12-1607. DCC Hardware Requests (continued)**

		R5FSS1_INTRTR0_IN_306	R5FSS1_INTRTR0	DCC11 one-shot mode complete interrupt	Level
		MAIN2MCU_LVL_INTRTR0_IN_211	MAIN2MCU_LVL_INTRTR0	DCC11 one-shot mode complete interrupt	Level
	DCC11_INTR_ERR_LEVEL_0	ESM0_LVL_IN_115	ESM0	DCC11 error interrupt	Level
DCC12	DCC12_INTR_DONE_LEV_EL_0	GIC500_SPI_IN_340	COMPUTE_CLUSTER0	DCC12 one-shot mode complete interrupt	Level
		R5FSS0_INTRTR0_IN_307	R5FSS0_INTRTR0	DCC12 one-shot mode complete interrupt	Level
		R5FSS1_INTRTR0_IN_307	R5FSS1_INTRTR0	DCC12 one-shot mode complete interrupt	Level
		MAIN2MCU_LVL_INTRTR0_IN_212	MAIN2MCU_LVL_INTRTR0	DCC12 one-shot mode complete interrupt	Level
	DCC12_INTR_ERR_LEVEL_0	ESM0_LVL_IN_116	ESM0	DCC12 error interrupt	Level

Table 12-1608 through Table 12-1612 summarize the DCC input source clocks in the device MAIN domain.

**Table 12-1608. DCC0 - DCC2 Input Source Clock Mapping**

		DCC0												DCC1												DCC2																		
		Input0			Input1									Input0			Input1									Input0			Input1															
		MUX0			MUX1									MUX0			MUX1									MUX0			MUX1															
		0-4, 6-9, B-F	A	5	0	1	2	3	4	5	6	7	8	9-F	0-4, 6-9, B-F	A	5	0	1	2	3	4	5	6	7	8	9-F	0-4, 6-9, B-F	A	5	0	1	2	3	4	5	6	7	8	9-F				
DCCCLKSRC0 / DCCCLKSRC1 value:																																												
Clock Source	Input:	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 3[3]	C L O C K 4[4]	C L O C K 5[5]	C L O C K 6[6]	C L O C K 7[7]	C L O C K 8[8]	C L O C K 9[9]	C L O C K A[A]	C L O C K B[B]	C L O C K C[C]	C L O C K D[D]	C L O C K E[E]	C L O C K F[F]	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 3[3]	C L O C K 4[4]	C L O C K 5[5]	C L O C K 6[6]	C L O C K 7[7]	C L O C K 8[8]	C L O C K 9[9]	C L O C K A[A]	C L O C K B[B]	C L O C K C[C]	C L O C K D[D]	C L O C K E[E]	C L O C K F[F]	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 3[3]	C L O C K 4[4]	C L O C K 5[5]	C L O C K 6[6]	C L O C K 7[7]	C L O C K 8[8]	C L O C K 9[9]	
HFOSC0_CLKOUT	External Reference Clock Input to WKUP/MCU	✓						✓					✓															✓																
HFOSC1_CLKOUT	External Reference Clock Input to MAIN		✓						✓				✓																✓															
CLK_12M_RC	Internal 12.5MHz RC oscillator (Always ON)			✓										✓															✓															
MAIN_PLLCTRL																																												
MAIN_SYSCLK0 (CLK1)	CLK1									✓																																		
CLK1/2	CLK1/2			✓																																								
CLK1/4	CLK1/4										✓				✓													✓		✓														✓
MAIN_PLL0_CLKOUT (MAIN PLL)																																												
MAIN_PLL0_HSDIV1_CLKOUT	DMTIMER Clock Mux				✓																																							
MAIN_PLL0_HSDIV2_CLKOUT	eMMCSD Clock Mux					✓																																						
MAIN_PLL0_HSDIV3_CLKOUT	GPWC Clock Mux						✓																																					

**Table 12-1608. DCC0 - DCC2 Input Source Clock Mapping (continued)**

DCCCLKSRC0 / DCCCLKSRC1 value:		DCC0												DCC1												DCC2															
		Input0		Input1										Input0		Input1										Input0		Input1													
		MUX0		MUX1										MUX0		MUX1										MUX0		MUX1													
		0-4, 6-9, B -F	A	5	0	1	2	3	4	5	6	7	8	9-F	0-4, 6-9, B -F	A	5	0	1	2	3	4	5	6	7	8	9-F	0-4, 6-9, B -F	A	5	0	1	2	3	4	5	6	7	8	9-F	
Clock Source	Input:	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 3[0]	C L O C K 4[1]	C L O C K 5[2]	C L O C K 6[3]	C L O C K 7[4]	C L O C K 8[5]	C L O C K 9[6]	C L O C K 10[7]	C L O C K 11[8]	C L O C K 12[9-F]	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 3[0]	C L O C K 4[1]	C L O C K 5[2]	C L O C K 6[3]	C L O C K 7[4]	C L O C K 8[5]	C L O C K 9[6]	C L O C K 10[7]	C L O C K 11[8]	C L O C K 12[9-F]	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 3[0]	C L O C K 4[1]	C L O C K 5[2]	C L O C K 6[3]	C L O C K 7[4]	C L O C K 8[5]	C L O C K 9[6]	C L O C K 10[7]	C L O C K 11[8]	C L O C K 12[9-F]	
MAIN_PLL0_HSDIV4_CLKOUT	MCAN Clock Mux							✓																																	
MAIN_PLL0_HSDIV5_CLKOUT	SPI Reference clock																✓																								
MAIN_PLL0_HSDIV6_CLKOUT	CPTS/IEP Clock Mux																	✓																							
MAIN_PLL0_HSDIV7_CLKOUT	ASRC SYS_CLK																		✓																						
MAIN_PLL1_CLKOUT (PER0 PLL)																																									
MAIN_PLL1_HSDIV0_CLKOUT	MAIN/ICSSG UARTs; I2C																			✓																					
MAIN_PLL1_HSDIV1_CLKOUT	CPSW9G CPPI CLK																				✓																				
MAIN_PLL1_HSDIV2_CLKOUT	eMMCSD Clock Mux																					✓																			
MAIN_PLL1_HSDIV3_CLKOUT	DMTIMER Clock Mux																						✓																		
MAIN_PLL1_HSDIV6_CLKOUT	UFS																							✓																	
MAIN_PLL1_HSDIV7_CLKOUT	USB - LPM/SOF CLK																																								
MAIN_PLL1_HSDIV8_CLKOUT	DPHY ESC Clock																																								
MAIN_PLL2_CLKOUT (PER1 PLL)																																									
MAIN_PLL2_HSDIV0_CLKOUT	ICSSG Core Clock Mux																																								
MAIN_PLL2_HSDIV1_CLKOUT_DIV2	GPMC Clock Mux; DSS Functional Clock																																								
MAIN_PLL2_HSDIV2_CLKOUT	eMMCSD Clock Mux; McASP/ATL clock mux																																								
MAIN_PLL2_HSDIV4_CLKOUT	SerDes Reference Clock																																								
MAIN_PLL2_HSDIV6_CLKOUT	DMTIMER Clock Mux																																								
MAIN_PLL3_CLKOUT (CPSW9G PLL)																																									
MAIN_PLL3_HSDIV0_CLKOUT	CPSW9G/ ICSSG RGMII/GMII Clock																																								
eDP0 REFCLK_OUT	SerDes Reference Clock Output												✓																												

**Table 12-1609. DCC3 - DCC5 Input Source Clock Mapping**

		DCC3												DCC4												DCC5																											
		Input0		Input1										Input0		Input1										Input0		Input1																									
		MUX0		MUX1										MUX0		MUX1										MUX0		MUX1																									
		0-4, 6-9, B-F	A	5	0	1	2	3	4	5	6	7	8	9-F	0-4, 6-9, B-F	A	5	0	1	2	3	4	5	6	7	8	9-F	0-4, 6-9, B-F	A	5	0	1	2	3	4	5	6	7	8	9-F													
DCCCLKSRC0 / DCCCLKSRC1 value:		C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 3[3]	C L O C K 4[4]	C L O C K 5[5]	C L O C K 6[6]	C L O C K 7[7]	C L O C K 8[8]	C L O C K 9[9]	C L O C K A[A]	C L O C K B[B]	C L O C K C[C]	C L O C K D[D]	C L O C K E[E]	C L O C K F[F]	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 3[3]	C L O C K 4[4]	C L O C K 5[5]	C L O C K 6[6]	C L O C K 7[7]	C L O C K 8[8]	C L O C K 9[9]	C L O C K A[A]	C L O C K B[B]	C L O C K C[C]	C L O C K D[D]	C L O C K E[E]	C L O C K F[F]	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 3[3]	C L O C K 4[4]	C L O C K 5[5]	C L O C K 6[6]	C L O C K 7[7]	C L O C K 8[8]	C L O C K 9[9]	C L O C K A[A]	C L O C K B[B]	C L O C K C[C]	C L O C K D[D]	C L O C K E[E]	C L O C K F[F]				
Clock Source	Input:	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 3[3]	C L O C K 4[4]	C L O C K 5[5]	C L O C K 6[6]	C L O C K 7[7]	C L O C K 8[8]	C L O C K 9[9]	C L O C K A[A]	C L O C K B[B]	C L O C K C[C]	C L O C K D[D]	C L O C K E[E]	C L O C K F[F]	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 3[3]	C L O C K 4[4]	C L O C K 5[5]	C L O C K 6[6]	C L O C K 7[7]	C L O C K 8[8]	C L O C K 9[9]	C L O C K A[A]	C L O C K B[B]	C L O C K C[C]	C L O C K D[D]	C L O C K E[E]	C L O C K F[F]	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 3[3]	C L O C K 4[4]	C L O C K 5[5]	C L O C K 6[6]	C L O C K 7[7]	C L O C K 8[8]	C L O C K 9[9]	C L O C K A[A]	C L O C K B[B]	C L O C K C[C]	C L O C K D[D]	C L O C K E[E]	C L O C K F[F]				
HFOSC0_CLKOUT	External Reference Clock Input to WKUP/MCU	✓											✓															✓																									
HFOSC1_CLKOUT	External Reference Clock Input to MAIN		✓										✓															✓																									
CLK_12M_RC	Internal 12.5MHz RC oscillator (Always ON)			✓										✓															✓																								
MAIN_PLLCTRL																																																					
MAIN_SYSCLK0 (CLK1)	CLK1																													✓																							
CLK1/2	CLK1/2														✓																																						
CLK1/4	CLK1/4				✓								✓															✓																									
MAIN_PLL0_CLKOUT T (MAIN PLL)																																																					
MAIN_PLL0_HSDIV 8_CLKOUT	VPFE Clock							✓																																													
MAIN_PLL1_CLKOUT T (PER0 PLL)																																																					
MAIN_PLL2_HSDIV 5_CLKOUT_DIV2	SA2_UL PKA Clock (450 MHz)															✓																																					
MAIN_PLL4_CLKOUT T (AUDIO0 PLL)																																																					
MAIN_PLL4_HSDIV 0_CLKOUT	McASP; some multiple of Audio Sampling rate				✓																																																
MAIN_PLL5_CLKOUT T (VIDEO0 PLL)																																																					
MAIN_PLL5_HSDIV 0_CLKOUT_DIV2	VXE384					✓																																															
MAIN_PLL5_HSDIV 1_CLKOUT_DIV2	D5520						✓																																														
MAIN_PLL6_CLKOUT T (GPU PLL)																																																					
MAIN_PLL6_HSDIV 0_CLKOUT_DIV4	GPU								✓																																												
MAIN_PLL7_CLKOUT T (C7x PLL)																																																					
MAIN_PLL7_HSDIV 0_CLKOUT_DIV4	C7x									✓																																											
MAIN_PLL8_CLKOUT T (ARM0 PLL)																																																					
MAIN_PLL8_HSDIV 0_CLKOUT_DIV8	ARM0 Core										✓																																										
MAIN_PLL12_CLKOUT (DDR PLL)																																																					

[illegible]



**Table 12-1609. DCC3 - DCC5 Input Source Clock Mapping (continued)**

		DCC3												DCC4												DCC5															
		Input0			Input1									Input0			Input1									Input0			Input1												
		MUX0			MUX1									MUX0			MUX1									MUX0			MUX1												
		0-4, 6-9, B-F	A	5	0	1	2	3	4	5	6	7	8	9-F	0-4, 6-9, B-F	A	5	0	1	2	3	4	5	6	7	8	9-F	0-4, 6-9, B-F	A	5	0	1	2	3	4	5	6	7	8	9-F	
DCCCLKSRC0 / DCCCLKSRC1 value:		C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 3[0]	C L O C K 4[1]	C L O C K 5[2]	C L O C K 6[3]	C L O C K 7[4]	C L O C K 8[5]	C L O C K 9[6]	C L O C K 10[7]	C L O C K 11[8]	C L O C K 12[9-F]	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 3[0]	C L O C K 4[1]	C L O C K 5[2]	C L O C K 6[3]	C L O C K 7[4]	C L O C K 8[5]	C L O C K 9[6]	C L O C K 10[7]	C L O C K 11[8]	C L O C K 12[9-F]	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 3[0]	C L O C K 4[1]	C L O C K 5[2]	C L O C K 6[3]	C L O C K 7[4]	C L O C K 8[5]	C L O C K 9[6]	C L O C K 10[7]	C L O C K 11[8]	C L O C K 12[9-F]	
Clock Source	Input:	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 3[0]	C L O C K 4[1]	C L O C K 5[2]	C L O C K 6[3]	C L O C K 7[4]	C L O C K 8[5]	C L O C K 9[6]	C L O C K 10[7]	C L O C K 11[8]	C L O C K 12[9-F]	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 3[0]	C L O C K 4[1]	C L O C K 5[2]	C L O C K 6[3]	C L O C K 7[4]	C L O C K 8[5]	C L O C K 9[6]	C L O C K 10[7]	C L O C K 11[8]	C L O C K 12[9-F]	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 3[0]	C L O C K 4[1]	C L O C K 5[2]	C L O C K 6[3]	C L O C K 7[4]	C L O C K 8[5]	C L O C K 9[6]	C L O C K 10[7]	C L O C K 11[8]	C L O C K 12[9-F]	
MAIN_PLL25_HSDI V1_CLKOUT_DIV2	VPAC																																								
GPMC_CLK	IO Clock input in Slave mode																																								
MCASP4_ACLKX	IO Clock input in Slave mode																																								
MCASP5_ACLKR	IO Clock input in Slave mode																																								

**Table 12-1610. DCC6 - DCC8 Input Source Clock Mapping**

DCCCLKSRC0 / DCCCLKSRC1 value:		DCC6												DCC7												DCC8															
		Input0			Input1									Input0			Input1									Input0			Input1												
		MUX0			MUX1									MUX0			MUX1									MUX0			MUX1												
		0-4, 6-9, B-F	A	5	0	1	2	3	4	5	6	7	8	9-F	0-4, 6-9, B-F	A	5	0	1	2	3	4	5	6	7	8	9-F	0-4, 6-9, B-F	A	5	0	1	2	3	4	5	6	7	8	9-F	
Clock Source	Input:	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 3[0]	C L O C K 4[1]	C L O C K 5[2]	C L O C K 6[3]	C L O C K 7[4]	C L O C K 8[5]	C L O C K 9[6]	C L O C K 10[7]	C L O C K 11[8]	C L O C K 12[9]	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 3[0]	C L O C K 4[1]	C L O C K 5[2]	C L O C K 6[3]	C L O C K 7[4]	C L O C K 8[5]	C L O C K 9[6]	C L O C K 10[7]	C L O C K 11[8]	C L O C K 12[9]	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 3[0]	C L O C K 4[1]	C L O C K 5[2]	C L O C K 6[3]	C L O C K 7[4]	C L O C K 8[5]	C L O C K 9[6]	C L O C K 10[7]	C L O C K 11[8]	C L O C K 12[9]	
HFOSC0_CLKOUT	External Reference Clock Input to WKUP/MCU	✓												✓													✓														
HFOSC1_CLKOUT	External Reference Clock Input to MAIN		✓												✓													✓													
CLK_12M_RC	Internal 12.5MHz RC oscillator (Always ON)			✓												✓													✓												
MAIN_PLLCTRL																																									
MAIN_SYSCLK0/2	CLK1/2																✓													✓											
MAIN_SYSCLK0/4	CLK1/4				✓									✓													✓														✓
MAIN_PLL3_CLKOUT (CPSW9G PLL)																																									
MAIN_PLL3_HSDIV1_CLKOUT	CPTS/IEP Clock Mux, ICSS Core clock Mux																																							✓	
MAIN_PLL3_HSDIV2_CLKOUT	eMMCSD Clock Mux																																							✓	
MAIN_PLL25_CLKOUT (Vision PLL)																																									

**Table 12-1610. DCC6 - DCC8 Input Source Clock Mapping (continued)**

DCCCLKSRC0 / DCCCLKSRC1 value:		DCC6												DCC7												DCC8														
		Input0		Input1										Input0		Input1										Input0		Input1												
		MUX0		MUX1										MUX0		MUX1										MUX0		MUX1												
		0-4, 6-9, B-F	A	5	0	1	2	3	4	5	6	7	8	9-F	0-4, 6-9, B-F	A	5	0	1	2	3	4	5	6	7	8	9-F	0-4, 6-9, B-F	A	5	0	1	2	3	4	5	6	7	8	9-F
Clock Source	Input:	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 0	C L O C K 1	C L O C K 2	C L O C K 3	C L O C K 4	C L O C K 5	C L O C K 6	C L O C K 7	C L O C K 8	C L O C K 9[0]	C L O C K 0[1]	C L O C K 1[2]	C L O C K 0	C L O C K 1	C L O C K 2	C L O C K 3	C L O C K 4	C L O C K 5	C L O C K 6	C L O C K 7	C L O C K 8	C L O C K 9-F	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 0	C L O C K 1	C L O C K 2	C L O C K 3	C L O C K 4	C L O C K 5	C L O C K 6	C L O C K 7	C L O C K 8	C L O C K 9-F	
MCASP0_ACLKX	IO Clock input in Slave mode				✓																																			
MCASP0_ACLKR	IO Clock input in Slave mode					✓																																		
MCASP1_ACLKX	IO Clock input in Slave mode						✓																																	
MCASP1_ACLKR	IO Clock input in Slave mode							✓																																
MCASP2_ACLKX	IO Clock input in Slave mode								✓																															
MCASP2_ACLKR	IO Clock input in Slave mode									✓																														
MCASP3_ACLKX	IO Clock input in Slave mode										✓																													
MCASP3_ACLKR	IO Clock input in Slave mode												✓																											
AUDIO_EXT_REFC_LK0	IO Clock input in Slave mode															✓																								
AUDIO_EXT_REFC_LK1	IO Clock input in Slave mode																✓																							
AUDIO_EXT_REFC_LK2	IO Clock input in Slave mode																	✓																						
AUDIO_EXT_REFC_LK3	IO Clock input in Slave mode																			✓																				
VPFE0_PCLK	IO Clock input for CCDC (Parallel Camera Port)																				✓																			
VOUT1_EXTPCLKIN	IO Reference clock input																					✓																		
VOUT2_EXTPCLKIN	IO Reference clock input																						✓																	
CPTS_RFT_CLK	IO Reference clock input																							✓																
CLK_32K	32KHz Clock (from WKUP domain)																													✓										
LFXOSC_CLKOUT	32.768KHz external clock (from WKUP domain)																															✓								
MCU_EXT_REFCLK_0	External Ref clock from WKUP domain																																✓							

**Table 12-1610. DCC6 - DCC8 Input Source Clock Mapping (continued)**

		DCC6												DCC7												DCC8													
		Input0		Input1										Input0		Input1										Input0		Input1											
		MUX0		MUX1										MUX0		MUX1										MUX0		MUX1											
		0-4, 6-9, B-F	A	5	0	1	2	3	4	5	6	7	8	9-F	0-4, 6-9, B-F	A	5	0	1	2	3	4	5	6	7	8	9-F	0-4, 6-9, B-F	A	5	0	1	2	3	4	5	6	7	8
DCCCLKSRC0 / DCCCLKSRC1 value:		C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 0	C L O C K 1	C L O C K 2	C L O C K 3	C L O C K 4	C L O C K 5	C L O C K 6	C L O C K 7	C L O C K 8	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 0	C L O C K 1	C L O C K 2	C L O C K 3	C L O C K 4	C L O C K 5	C L O C K 6	C L O C K 7	C L O C K 8	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 0	C L O C K 1	C L O C K 2	C L O C K 3	C L O C K 4	C L O C K 5	C L O C K 6	C L O C K 7	C L O C K 8	C L O C K 9-F	
Clock Source	Input:	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 0	C L O C K 1	C L O C K 2	C L O C K 3	C L O C K 4	C L O C K 5	C L O C K 6	C L O C K 7	C L O C K 8	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 0	C L O C K 1	C L O C K 2	C L O C K 3	C L O C K 4	C L O C K 5	C L O C K 6	C L O C K 7	C L O C K 8	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 0	C L O C K 1	C L O C K 2	C L O C K 3	C L O C K 4	C L O C K 5	C L O C K 6	C L O C K 7	C L O C K 8	C L O C K 9-F	
DPHY-RX0 byte clk RxByteClkHS_cl_I	IO Reference clock input (recovered byte clock from CSI0_RXCLK P/N)																																						
DPHY-RX1 byte clk RxByteClkHS_cl_I	IO Reference clock input (recovered byte clock from CSI1_RXCLK P/N)																																						

**Table 12-1611. DCC9 - DCC11 Input Source Clock Mapping**

DCCCLKSRC0 / DCCCLKSRC1 value:		DCC9												DCC10												DCC11													
		Input0			Input1									Input0			Input1									Input0			Input1										
		MUX0			MUX1									MUX0			MUX1									MUX0			MUX1										
		0-4, 6-9, B -F	A	5	0	1	2	3	4	5	6	7	8	9-F	0-4, 6-9, B -F	A	5	0	1	2	3	4	5	6	7	8	9-F	0-4, 6-9, B -F	A	5	0	1	2	3	4	5	6	7	8
Clock Source	Input:	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 0	C L O C K 1	C L O C K 2	C L O C K 3	C L O C K 4	C L O C K 5	C L O C K 6	C L O C K 7	F I C L K	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 0	C L O C K 1	C L O C K 2	C L O C K 3	C L O C K 4	C L O C K 5	C L O C K 6	C L O C K 7	F I C L K	C L O C K 0[0]	C L O C K 1[1]	C L O C K 2[2]	C L O C K 0	C L O C K 1	C L O C K 2	C L O C K 3	C L O C K 4	C L O C K 5	C L O C K 6	C L O C K 7	F I C L K		
HFOSC0_CLKOUT	External Reference Clock Input to WKUP/MCU	✓												✓													✓												
HFOSC1_CLKOUT	External Reference Clock Input to MAIN		✓												✓													✓											
CLK_12M_RC	Internal 12.5MHz RC oscillator (Always ON)			✓											✓													✓											
MAIN PLLCTRL																																							
MAIN_SYSCLK0/2	CLK1/2			✓												✓													✓										
MAIN_SYSCLK0/4	CLK1/4												✓													✓												✓	
MAIN_PLL3_CLKOUT (CPSW9G PLL)																																							
MAIN_PLL3_HSDIV3_CLKOUT	DMTIMER Clock Mux				✓																																		
MAIN_PLL3_HSDIV4_CLKOUT	SerDes Reference Clock (If 156.25 is needed VCO must be set for 2500MHz)					✓																																	

**Table 12-1611. DCC9 - DCC11 Input Source Clock Mapping (continued)**

DCCCLKSRC0 / DCCCLKSRC1 value:		DCC9												DCC10												DCC11													
		Input0		Input1										Input0		Input1										Input0		Input1											
		MUX0		MUX1										MUX0		MUX1										MUX0		MUX1											
		0-4, 6-9, B-F	A	5	0	1	2	3	4	5	6	7	8	9-F	0-4, 6-9, B-F	A	5	0	1	2	3	4	5	6	7	8	9-F	0-4, 6-9, B-F	A	5	0	1	2	3	4	5	6	7	8
Clock Source	Input:	C L O C K 0[0]	C L O C K 0[1]	C L O C K 0[2]	C L O C K 1	C L O C K 2	C L O C K 3	C L O C K 4	C L O C K 5	C L O C K 6	C L O C K 7	F I C L K	C L O C K 0[0]	C L O C K 0[1]	C L O C K 0[2]	C L O C K 1	C L O C K 2	C L O C K 3	C L O C K 4	C L O C K 5	C L O C K 6	C L O C K 7	F I C L K	C L O C K 0[0]	C L O C K 0[1]	C L O C K 0[2]	C L O C K 1	C L O C K 2	C L O C K 3	C L O C K 4	C L O C K 5	C L O C K 6	C L O C K 7	F I C L K					
MAIN_PLL25_CLKOUT (Vision PLL)																																							
MCASP4_ACLKR	IO Clock input in Slave mode							✓																															
MCASP5_ACLKX	IO Clock input in Slave mode								✓																														
MCASP6_ACLKX	IO Clock input in Slave mode											✓																											
MCASP6_ACLKR	IO Clock input in Slave mode																	✓																					
MCASP7_ACLKX	IO Clock input in Slave mode																				✓																		
MCASP7_ACLKR	IO Clock input in Slave mode																														✓								
MCASP8_ACLKX	IO Clock input in Slave mode																																		✓				
MCASP8_ACLKR	IO Clock input in Slave mode																																			✓			
RMII1_REFCLK	IO Reference clock input						✓																																
RGMII3_RXC	IO Receive clock input																	✓																					
RGMII4_RXC	IO Receive clock input																				✓																		
RGMII5_RXC	IO Receive clock input																						✓																
RGMII6_RXC	IO Receive clock input																								✓														
RGMII7_RXC	IO Receive clock input																																			✓			
RGMII8_RXC	IO Receive clock input										✓																												

**Table 12-1612. DCC12 Input Source Clock Mapping**

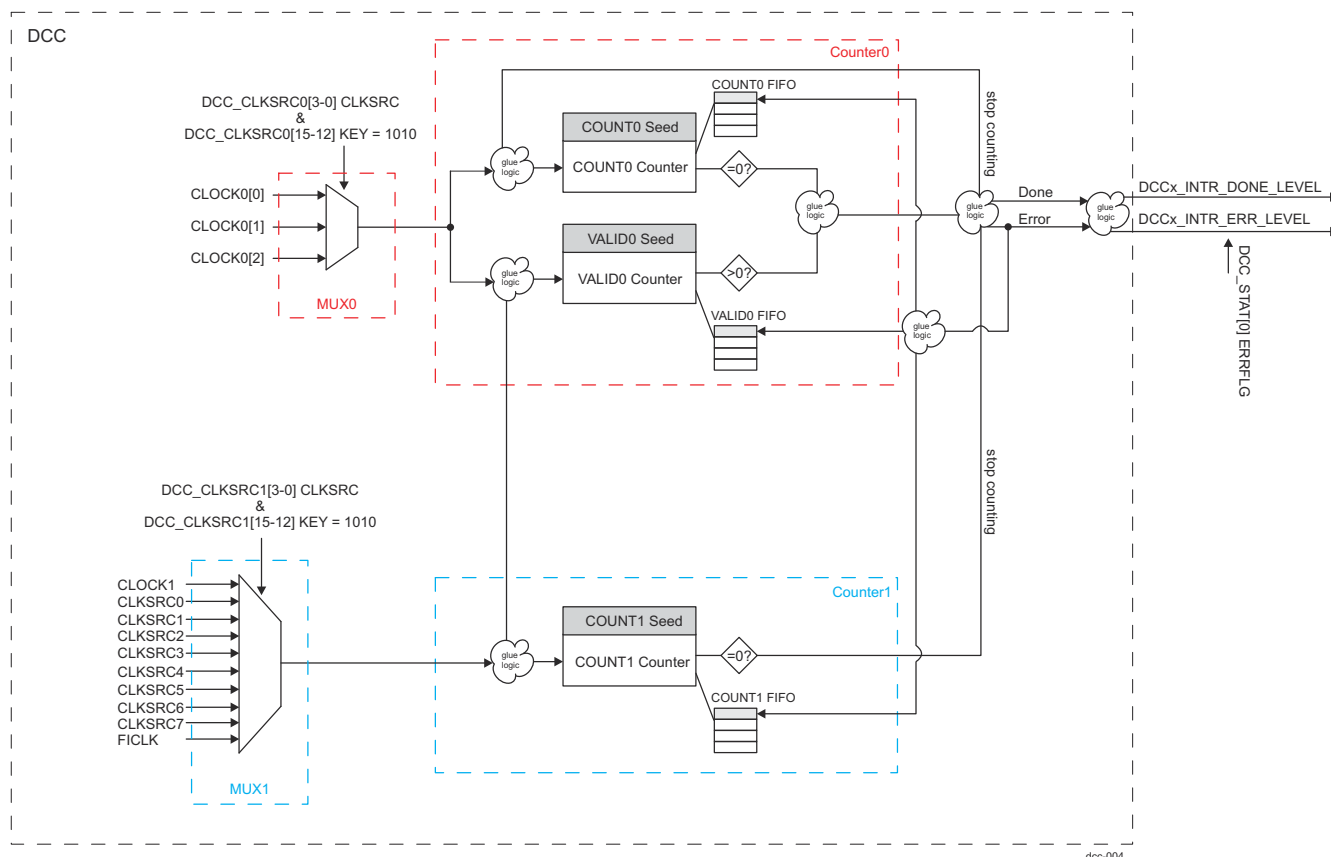
DCCCLKSRC0 / DCCCLKSRC1 value:		DCC12												
		Input0			Input1									
		MUX0			MUX1									
		0-4, 6-9, B-F	A	5	0	1	2	3	4	5	6	7	8	9-F
Clock Source	Input:	CLOC K0[0]	CLOC K0[1]	CLOC K0[2]	CLOC K1	CLKS RC0	CLKS RC1	CLKS RC2	CLKS RC3	CLKS RC4	CLKS RC5	CLKS RC6	CLKS RC7	FICLK
HFOSC0_CLKOUT	External Reference Clock Input to WKUP/MCU	✓												
HFOSC1_CLKOUT	External Reference Clock Input to MAIN		✓											

**Table 12-1612. DCC12 Input Source Clock Mapping (continued)**

DCCCLKSRC0 / DCCCLKSRC1 value:		DCC12												
		Input0			Input1									
		MUX0			MUX1									
		0-4, 6-9, B-F	A	5	0	1	2	3	4	5	6	7	8	9-F
Clock Source	Input:	CLOC K0[0]	CLOC K0[1]	CLOC K0[2]	CLOC K1	CLKS RC0	CLKS RC1	CLKS RC2	CLKS RC3	CLKS RC4	CLKS RC5	CLKS RC6	CLKS RC7	FICL K
CLK_12M_RC	Internal 12.5MHz RC oscillator (Always ON)			✓										
EXT_REFCLK1	External Misc clock input to Wkup/MCU													
MAIN_PLLCTRL														
MAIN_SYSCLK0/2	CLK1/2				✓									
MAIN_SYSCLK0/4	CLK1/4													✓
MAIN_PLL25_CLKOUT (Vision PLL)														
MCASP9_ACLKX	IO Clock input in Slave mode						✓							
MCASP9_ACLKR	IO Clock input in Slave mode								✓					
MCASP10_ACLKX	IO Clock input in Slave mode										✓			
MCASP10_ACLKR	IO Clock input in Slave mode												✓	
PRG0_RGMII1_RXC	IO Receive clock input					✓								
PRG0_RGMII2_RXC	IO Receive clock input							✓						
PRG1_RGMII1_RXC	IO Receive clock input									✓				
PRG1_RGMII2_RXC	IO Receive clock input											✓		

### 12.11.1.3 DCC Functional Description

The DCC module supports two selectable clock sources for Counter0 and Counter1. Counter0 receives CLOCK0[2] at reset. Counter1 receives CLOCK1 at reset. Figure 12-1225 shows the DCC functional block diagram.



**Figure 12-1225. DCC Functional Block Diagram**

#### 12.11.1.3.1 DCC Counter Operation

##### Note

For detailed DCC compute calculations, refer to [Continuous Monitor of the PLL Frequency With the DCC App Note](#).

The DCC has two parallel counters that count clock pulses for two independent clock sources:

- Counter0 generates a fixed-width counting window (VALID0) after a pre-programmed number of pulses (COUNT0). The values for VALID0 and COUNT0 can be programmed in DCCVALIDSEED0 and DCC\_CNTSEED0 registers respectively.
- Counter1 generates a fixed-width pulse (1 cycle) after a pre-programmed number of pulses (COUNT1). This pulse sets an error signal if Counter1 does not reach 0 during the time when VALID0 is running. The seed value for COUNT1 can be programmed in DCC\_CNTSEED1 register.

The error signal is generated by any one of the following conditions:

- Clock1 expires before the COUNT0 reaches 0.
- Clock1 expires after both COUNT0 and VALID0 reach 0.
- Clock1 not present.
- Clock0 not present.

Any of these errors causes the counters to stop counting. An application must then read out the counter values to determine what caused the error. Once the error is detected, the counters are stopped after 3 FICLK and 2 source clock cycles due to the cross clock domain synchronisation.

Reloads or restarts occur under two conditions:

- The module is reset or restarted through software (that is, software starts the module after reset, or software checks an error condition and decides to restart the module).
- COUNT0, COUNT1, and VALID0 all reach 0 without error.

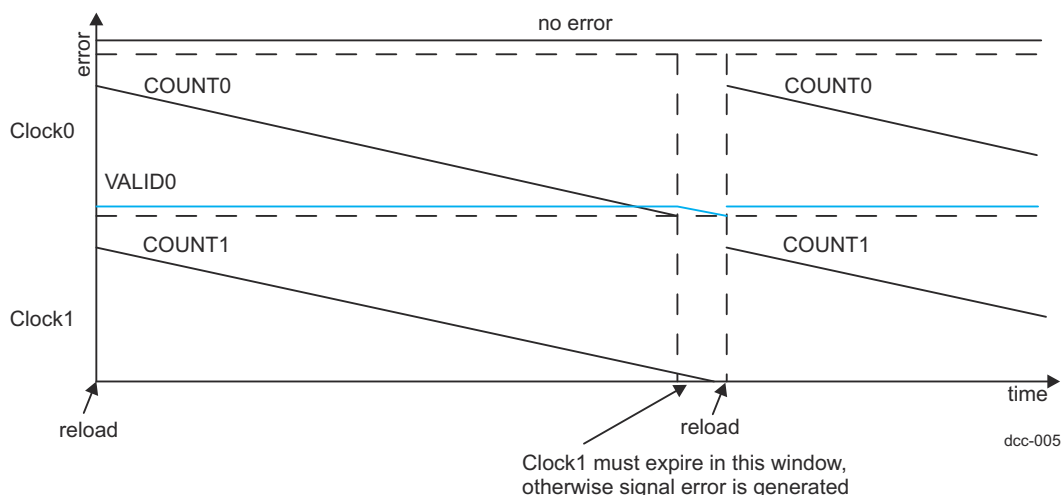
### CAUTION

The DCC module does not check jitter for Clock0 or Clock1.

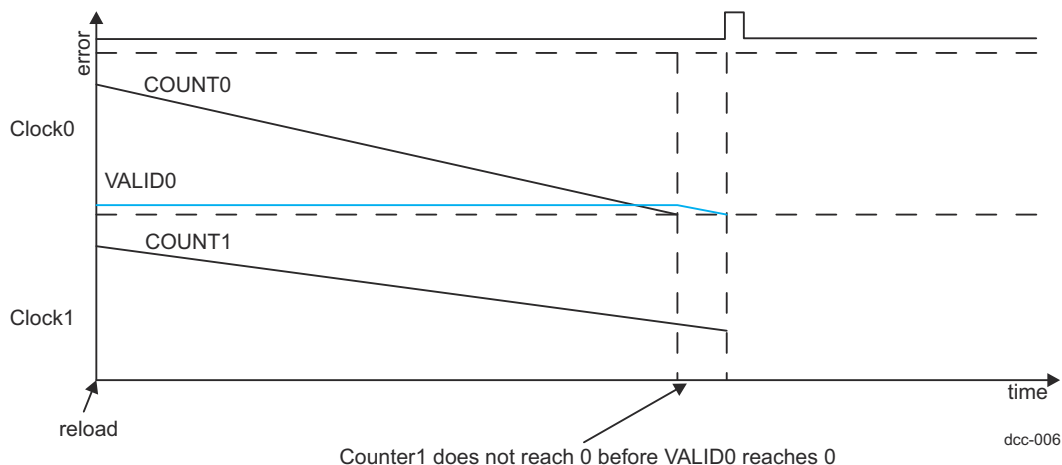
As the counter preset signal is synchronized to either of the source clock domains, the counters begin downcounting after two corresponding source clock cycles.

The error signal is captured to the FICLK domain. There is 1 FICLK period uncertainty on either side of the fixed width counting window (VALID0) in generating the error signal since the counters work in different clock domains. This should be accounted for when setting the count value for VALID0.

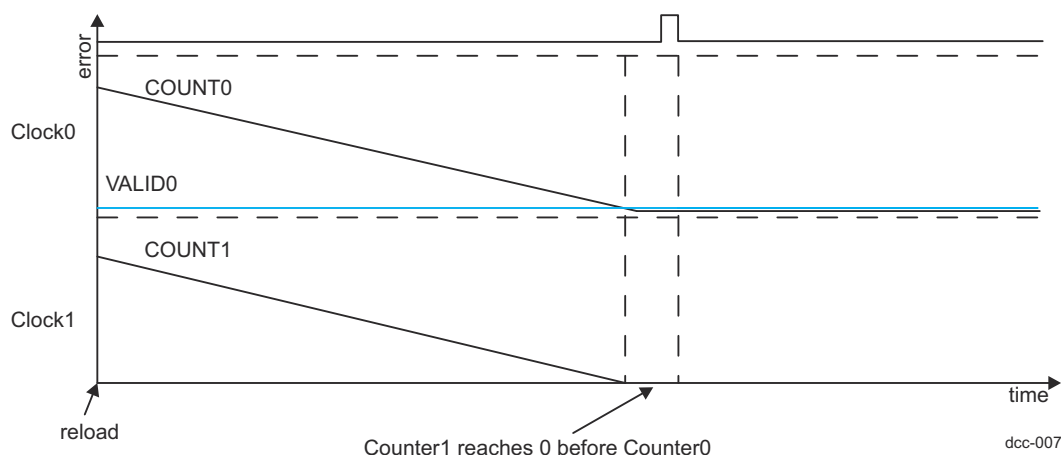
Figure 12-1226 through Figure 12-1230 shows examples of counters relationship and error generation.



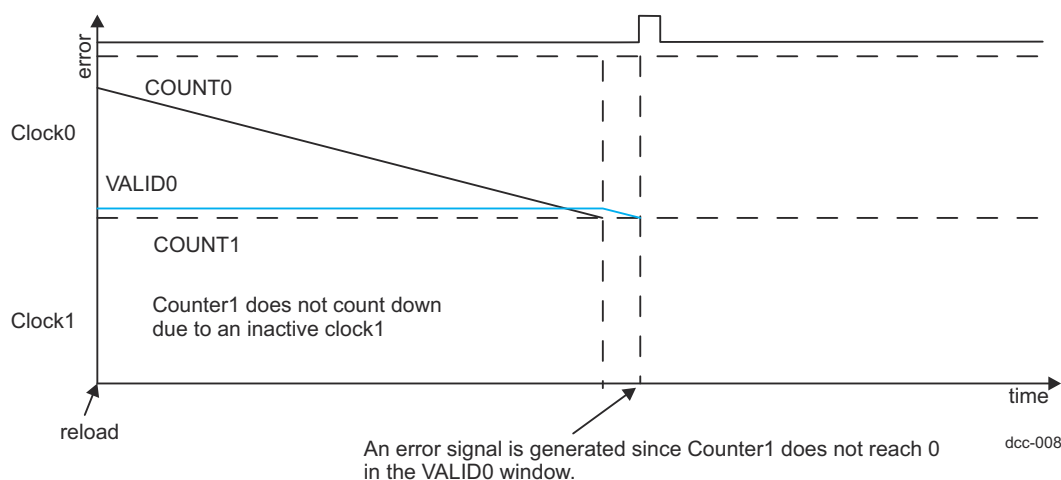
**Figure 12-1226. DCC Clock0 and Clock1 With no Error**



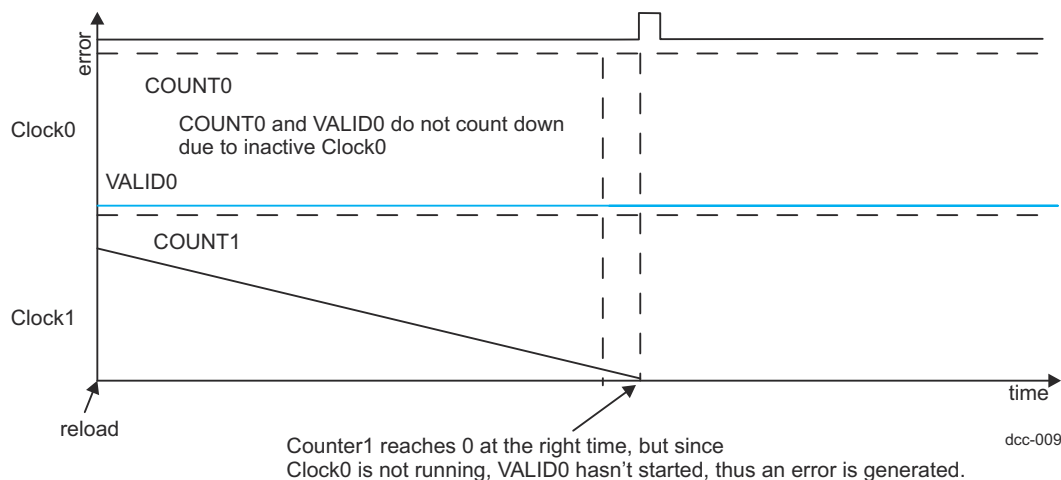
**Figure 12-1227. DCC Clock1 slower than Clock0 results in an error and stops counting**



**Figure 12-1228. DCC Clock1 faster than Clock0 results in an error and stops counting**



**Figure 12-1229. DCC Clock1 not present results in an error and stops counting**



**Figure 12-1230. DCC Clock0 not present results in an error and stops counting**

### 12.11.1.3.2 DCC Low Power Mode Operation

The DCC module does not function in Low Power Mode. It is the responsibility of software to stop the module before entering low power mode and to restart the module after exiting low power mode.



### 12.11.1.3.3 DCC Suspend Mode Behavior

The DCC module will continue running regardless of the state of emulation. All registers are readable via emulation reads.

### 12.11.1.3.4 DCC Single-Shot Mode

The DCC can be programmed to count down one time using single-shot mode. In this mode, the DCC stops operation when both COUNT0 and VALID0 reach 0.

At the end of one sequence in single-shot mode the DCC\_GCTRL[3-0] DCCENA bitfield is set to disabled, which stops further counting. Single-shot mode is enabled from the DCC\_GCTRL[11-8] SINGLESOT bitfield.

At the end of one sequence in single-shot mode, if there is no error which stops counting, then the done status bit is set in the DCC\_STAT[1] DONEFLG bitfield and a done interrupt DCCx\_INTR\_DONE\_LEVEL is generated. Software must clear the done bit before restarting the counting.

### 12.11.1.3.5 DCC Continuous mode

When DCC runs in continuous mode both the counts shall get reloaded with seed value upon completion of counts without error. If the counts end in error DCC stops the operation and counts are not reloaded.

#### 12.11.1.3.5.1 DCC Continue on Error

During debug, if there are events which are causing clocks to be anomalous over short period covering more than one evaluation window then it would be important to capture trajectory of error event and period around such event. To allow capturing the successive error events DCC can be programmed to continue after error. DCC\_GCTRL2[3-0] CONT\_ON\_ERR shall be set to value other than "0101" to enable this mode. It is recommended to write "1010" to avoid single soft errors.

#### 12.11.1.3.5.2 DCC Error Count

DCC also counts the number of error pulses generated since reset or since last time the error count is cleared. This is read/write register for CPU to clear when new trace of number of errors is required to be maintained.

### 12.11.1.3.6 DCC Control and count hand-off across clock domains

As the counters run in two different selectable clock domains and the register interface runs on the fixed bus clock domain, control signals and counter value hand-off have synchronizers implemented. These add to the margins of error while comparing the counts.

1. With all three counts synchronized to FICLK domain for comparison error detection there is delay in terms of FICLK domain.
2. Based on the error signal, the enable for the counters is synchronized back to the respective clock domains of the counters, which adds latency in terms of clock periods which are different, this would create a skew between start of count. Depending upon frequency ratios of the clocks used, the difference between two could vary.

Application needs to consider the worst case delay differences while measuring the clocks.

### 12.11.1.3.7 DCC Error Trajectory record

Once the clock errors out, the host can read the counter values to determine the extent of error to analyze type of failure. For short window comparisons this would become difficult, specially if there are back to back errors due to some transient event. Secondly, for random events which can cause an interrupt during the critical phase of application running, then event if not recorded may get overwritten and also not provide meaningful trace of error.

#### 12.11.1.3.7.1 DCC FIFO capturing for Errors

DCC provides the FIFO for capturing COUNT0, VALID0, and COUNT1 information which captures all three counts upon "Error" event. For "Done" event no results are captured by default.

#### **12.11.1.3.7.2 DCC FIFO in continuous capture mode**

To track the VALID0 counter values regardless of "Error" or not, FIFOs can be configured to capture the count for each compare window. This is useful in validation and characterization exercise. [DCC\\_GCTRL2\[11-7\]](#) FIFO\_NONERR control when set to value other than "0101" this mode is set; it is recommended to write "1010" to avoid single soft errors. Note, this capture is applicable only in continuous mode and not in single shot mode.

#### **12.11.1.3.7.3 DCC FIFO Details**

The FIFO is 4 deep for each count and updates new count information for all the non-full FIFOs. Information is updated on every configured trigger of error or cycle completion. If full, the next values are not written till at-least one entry is read. Application owns responsibility to read the FIFOs uniformly to keep synchronisation between three entries of the FIFO. Both empty and full indications for individual FIFOs is provided through the [DCC\\_STATUS2](#).

#### **12.11.1.3.7.4 DCC FIFO Debug mode behavior**

Upon debug access, the FIFO pointers should not advance hence not impacting the functional behavior. This requirement is same as other functional blocks in the device.

#### **12.11.1.3.8 DCC Count read registers**

DCC has provision to read the counts during operation. This is performed using DCC\_CNT0, DCC\_VALID0, and DCC\_CNT1 registers. Read from these registers in default mode allows reading the present value of count. This is useful when in single shot mode or mode where DCC stops upon error.

These registers can be used to read the FIFO through the DCC\_GCTRL2[7-4] FIFO\_READ configuration. Reads on the empty FIFO shall provide the contents of last pointed location. Application shall track the empty/full conditions of the FIFOs to track the count records consistently.

Regardless of FIFO\_READ configuration, the FIFO internally keeps updating records based on configured triggers till full.

#### 12.11.1.4 DCC Registers

Table 12-1614 lists the memory-mapped registers for the DCC registers. All register offset addresses not listed in Table 12-1614 through Table 12-1619 should be considered as reserved locations and the register contents should not be modified.

**Table 12-1613. DCC Instances**

Instance	Base Address
DCC0	0080 0000h
DCC1	0080 4000h
DCC2	0080 8000h
DCC3	0080 C000h
DCC4	0081 0000h
DCC5	0081 4000h
DCC6	0081 8000h
DCC7	0081 C000h
DCC8	0082 0000h
DCC9	0082 4000h
DCC10	0082 8000h
DCC11	0082 C000h
DCC12	0083 0000h
MCU_DCC0	4010 0000h
MCU_DCC1	4011 0000h
MCU_DCC2	4012 0000h

**Table 12-1614. DCC Registers**

Offset	Acronym	Register Name	DCC0 Physical Address	DCC1 Physical Address	DCC2 Physical Address
0h	<a href="#">DCC_GCTRL</a>	DCC Global Control Register	0080 0000h	0080 4000h	0080 8000h
4h	<a href="#">DCC_REV</a>	DCC Revision ID	0080 0004h	0080 4004h	0080 8004h
8h	<a href="#">DCC_CNTSEED0</a>	Count0 Seed Value Register	0080 0008h	0080 4008h	0080 8008h
Ch	<a href="#">DCC_VALIDSEED0</a>	Valid0 Seed Value Register	0080 000Ch	0080 400Ch	0080 800Ch
10h	<a href="#">DCC_CNTSEED1</a>	Count1 Seed Value Register	0080 0010h	0080 4010h	0080 8010h
14h	<a href="#">DCC_STATUS</a>	DCC Status Register	0080 0014h	0080 4014h	0080 8014h
18h	<a href="#">DCC_CNT0</a>	Count0 Value Register	0080 0018h	0080 4018h	0080 8018h
1Ch	<a href="#">DCC_VALID0</a>	Valid0 Value Register	0080 001Ch	0080 401Ch	0080 801Ch
20h	<a href="#">DCC_CNT1</a>	Count1 Value Register	0080 0020h	0080 4020h	0080 8020h
24h	<a href="#">DCC_CLKSRC1</a>	Clock Source Selection Register 1	0080 0024h	0080 4024h	0080 8024h
28h	<a href="#">DCC_CLKSRC0</a>	Clock Source Selection Register 0	0080 0028h	0080 4028h	0080 8028h
2Ch	<a href="#">DCC_GCTRL2</a>	DCC Global Control Register 2	0080 002Ch	0080 402Ch	0080 802Ch
30h	<a href="#">DCC_STATUS2</a>	DCC FIFO Status Register	0080 0030h	0080 4030h	0080 8030h
34h	<a href="#">DCC_ERRCNT</a>	Error Count Register	0080 0034h	0080 4034h	0080 8034h

**Table 12-1615. DCC Registers**

Offset	Acronym	Register Name	DCC3 Physical Address	DCC4 Physical Address	DCC5 Physical Address
0h	<a href="#">DCC_GCTRL</a>	DCC Global Control Register	0080 C000h	0081 0000h	0081 4000h
4h	<a href="#">DCC_REV</a>	DCC Revision ID	0080 C004h	0081 0004h	0081 4004h
8h	<a href="#">DCC_CNTSEED0</a>	Count0 Seed Value Register	0080 C008h	0081 0008h	0081 4008h
Ch	<a href="#">DCC_VALIDSEED0</a>	Valid0 Seed Value Register	0080 C00Ch	0081 000Ch	0081 400Ch
10h	<a href="#">DCC_CNTSEED1</a>	Count1 Seed Value Register	0080 C010h	0081 0010h	0081 4010h

**Table 12-1615. DCC Registers (continued)**

Offset	Acronym	Register Name	DCC3 Physical Address	DCC4 Physical Address	DCC5 Physical Address
14h	<a href="#">DCC_STATUS</a>	DCC Status Register	0080 C014h	0081 0014h	0081 4014h
18h	<a href="#">DCC_CNT0</a>	Count0 Value Register	0080 C018h	0081 0018h	0081 4018h
1Ch	<a href="#">DCC_VALID0</a>	Valid0 Value Register	0080 C01Ch	0081 001Ch	0081 401Ch
20h	<a href="#">DCC_CNT1</a>	Count1 Value Register	0080 C020h	0081 0020h	0081 4020h
24h	<a href="#">DCC_CLKSRC1</a>	Clock Source Selection Register 1	0080 C024h	0081 0024h	0081 4024h
28h	<a href="#">DCC_CLKSRC0</a>	Clock Source Selection Register 0	0080 C028h	0081 0028h	0081 4028h
2Ch	<a href="#">DCC_GCTRL2</a>	DCC Global Control Register 2	0080 C02Ch	0081 002Ch	0081 402Ch
30h	<a href="#">DCC_STATUS2</a>	DCC FIFO Status Register	0080 C030h	0081 0030h	0081 4030h
34h	<a href="#">DCC_ERRCNT</a>	Error Count Register	0080 C034h	0081 0034h	0081 4034h

**Table 12-1616. DCC Registers**

Offset	Acronym	Register Name	DCC6 Physical Address	DCC7 Physical Address	DCC8 Physical Address
0h	<a href="#">DCC_GCTRL</a>	DCC Global Control Register	0081 8000h	0081 C000h	0082 0000h
4h	<a href="#">DCC_REV</a>	DCC Revision ID	0081 8004h	0081 C004h	0082 0004h
8h	<a href="#">DCC_CNTSEED0</a>	Count0 Seed Value Register	0081 8008h	0081 C008h	0082 0008h
Ch	<a href="#">DCC_VALIDSEED0</a>	Valid0 Seed Value Register	0081 800Ch	0081 C00Ch	0082 000Ch
10h	<a href="#">DCC_CNTSEED1</a>	Count1 Seed Value Register	0081 8010h	0081 C010h	0082 0010h
14h	<a href="#">DCC_STATUS</a>	DCC Status Register	0081 8014h	0081 C014h	0082 0014h
18h	<a href="#">DCC_CNT0</a>	Count0 Value Register	0081 8018h	0081 C018h	0082 0018h
1Ch	<a href="#">DCC_VALID0</a>	Valid0 Value Register	0081 801Ch	0081 C01Ch	0082 001Ch
20h	<a href="#">DCC_CNT1</a>	Count1 Value Register	0081 8020h	0081 C020h	0082 0020h
24h	<a href="#">DCC_CLKSRC1</a>	Clock Source Selection Register 1	0081 8024h	0081 C024h	0082 0024h
28h	<a href="#">DCC_CLKSRC0</a>	Clock Source Selection Register 0	0081 8028h	0081 C028h	0082 0028h
2Ch	<a href="#">DCC_GCTRL2</a>	DCC Global Control Register 2	0081 802Ch	0081 C02Ch	0082 002Ch
30h	<a href="#">DCC_STATUS2</a>	DCC FIFO Status Register	0081 8030h	0081 C030h	0082 0030h
34h	<a href="#">DCC_ERRCNT</a>	Error Count Register	0081 8034h	0081 C034h	0082 0034h

**Table 12-1617. DCC Registers**

Offset	Acronym	Register Name	DCC9 Physical Address	DCC10 Physical Address	DCC11 Physical Address
0h	<a href="#">DCC_GCTRL</a>	DCC Global Control Register	0082 4000h	0082 8000h	0082 C000h
4h	<a href="#">DCC_REV</a>	DCC Revision ID	0082 4004h	0082 8004h	0082 C004h
8h	<a href="#">DCC_CNTSEED0</a>	Count0 Seed Value Register	0082 4008h	0082 8008h	0082 C008h
Ch	<a href="#">DCC_VALIDSEED0</a>	Valid0 Seed Value Register	0082 400Ch	0082 800Ch	0082 C00Ch
10h	<a href="#">DCC_CNTSEED1</a>	Count1 Seed Value Register	0082 4010h	0082 8010h	0082 C010h
14h	<a href="#">DCC_STATUS</a>	DCC Status Register	0082 4014h	0082 8014h	0082 C014h
18h	<a href="#">DCC_CNT0</a>	Count0 Value Register	0082 4018h	0082 8018h	0082 C018h
1Ch	<a href="#">DCC_VALID0</a>	Valid0 Value Register	0082 401Ch	0082 801Ch	0082 C01Ch
20h	<a href="#">DCC_CNT1</a>	Count1 Value Register	0082 4020h	0082 8020h	0082 C020h
24h	<a href="#">DCC_CLKSRC1</a>	Clock Source Selection Register 1	0082 4024h	0082 8024h	0082 C024h
28h	<a href="#">DCC_CLKSRC0</a>	Clock Source Selection Register 0	0082 4028h	0082 8028h	0082 C028h
2Ch	<a href="#">DCC_GCTRL2</a>	DCC Global Control Register 2	0082 402Ch	0082 802Ch	0082 C02Ch
30h	<a href="#">DCC_STATUS2</a>	DCC FIFO Status Register	0082 4030h	0082 8030h	0082 C030h
34h	<a href="#">DCC_ERRCNT</a>	Error Count Register	0082 4034h	0082 8034h	0082 C034h

**Table 12-1618. DCC Registers**

Offset	Acronym	Register Name	DCC12 Physical Address
0h	<a href="#">DCC_GCTRL</a>	DCC Global Control Register	0083 0000h
4h	<a href="#">DCC_REV</a>	DCC Revision ID	0083 0004h
8h	<a href="#">DCC_CNTSEED0</a>	Count0 Seed Value Register	0083 0008h
Ch	<a href="#">DCC_VALIDSEED0</a>	Valid0 Seed Value Register	0083 000Ch
10h	<a href="#">DCC_CNTSEED1</a>	Count1 Seed Value Register	0083 0010h
14h	<a href="#">DCC_STATUS</a>	DCC Status Register	0083 0014h
18h	<a href="#">DCC_CNT0</a>	Count0 Value Register	0083 0018h
1Ch	<a href="#">DCC_VALID0</a>	Valid0 Value Register	0083 001Ch
20h	<a href="#">DCC_CNT1</a>	Count1 Value Register	0083 0020h
24h	<a href="#">DCC_CLKSRC1</a>	Clock Source Selection Register 1	0083 0024h
28h	<a href="#">DCC_CLKSRC0</a>	Clock Source Selection Register 0	0083 0028h
2Ch	<a href="#">DCC_GCTRL2</a>	DCC Global Control Register 2	0083 002Ch
30h	<a href="#">DCC_STATUS2</a>	DCC FIFO Status Register	0083 0030h
34h	<a href="#">DCC_ERRCNT</a>	Error Count Register	0083 0034h

**Table 12-1619. DCC Registers**

Offset	Acronym	Register Name	MCU_DCC0 Physical Address	MCU_DCC1 Physical Address	MCU_DCC2 Physical Address
0h	<a href="#">DCC_GCTRL</a>	DCC Global Control Register	4010 0000h	4011 0000h	4012 0000h
4h	<a href="#">DCC_REV</a>	DCC Revision ID	4010 0004h	4011 0004h	4012 0004h
8h	<a href="#">DCC_CNTSEED0</a>	Count0 Seed Value Register	4010 0008h	4011 0008h	4012 0008h
Ch	<a href="#">DCC_VALIDSEED0</a>	Valid0 Seed Value Register	4010 000Ch	4011 000Ch	4012 000Ch
10h	<a href="#">DCC_CNTSEED1</a>	Count1 Seed Value Register	4010 0010h	4011 0010h	4012 0010h
14h	<a href="#">DCC_STATUS</a>	DCC Status Register	4010 0014h	4011 0014h	4012 0014h
18h	<a href="#">DCC_CNT0</a>	Count0 Value Register	4010 0018h	4011 0018h	4012 0018h
1Ch	<a href="#">DCC_VALID0</a>	Valid0 Value Register	4010 001Ch	4011 001Ch	4012 001Ch
20h	<a href="#">DCC_CNT1</a>	Count1 Value Register	4010 0020h	4011 0020h	4012 0020h
24h	<a href="#">DCC_CLKSRC1</a>	Clock Source Selection Register 1	4010 0024h	4011 0024h	4012 0024h
28h	<a href="#">DCC_CLKSRC0</a>	Clock Source Selection Register 0	4010 0028h	4011 0028h	4012 0028h
2Ch	<a href="#">DCC_GCTRL2</a>	DCC Global Control Register 2	4010 002Ch	4011 002Ch	4012 002Ch
30h	<a href="#">DCC_STATUS2</a>	DCC FIFO Status Register	4010 0030h	4011 0030h	4012 0030h
34h	<a href="#">DCC_ERRCNT</a>	Error Count Register	4010 0034h	4011 0034h	4012 0034h

### 12.11.1.4.1 DCC\_GCTRL Register (Offset = 0h) [reset = X]

DCC\_GCTRL is shown in [Figure 12-1231](#) and described in [Table 12-1621](#).

Return to [Summary Table](#).

Starts / stops the counters. Clears the error signal.

**Table 12-1620. DCC\_GCTRL Instances**

Instance	Physical Address
DCC0	0080 0000h
DCC1	0080 4000h
DCC2	0080 8000h
DCC3	0080 C000h
DCC4	0081 0000h
DCC5	0081 4000h
DCC6	0081 8000h
DCC7	0081 C000h
DCC8	0082 0000h
DCC9	0082 4000h
DCC10	0082 8000h
DCC11	0082 C000h
DCC12	0083 0000h
MCU_DCC0	4010 0000h
MCU_DCC1	4011 0000h
MCU_DCC2	4012 0000h

**Figure 12-1231. DCC\_GCTRL Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R/W-X															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DONEENA				SINGLESOT				ERRENA				DCCENA			
R/W-5h				R/W-5h				R/W-5h				R/W-5h			

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1621. DCC\_GCTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R/W	X	
15-12	DONEENA	R/W	5h	<p>The DONEENA bit enables/disables the done interrupt signal, but has no effect on the done status flag in DCC_STAT register.</p> <p>User, privilege, and debug mode (read):</p> <p>0101 = the done signal is disabled</p> <p>others = the done signal is enabled</p> <p>Privilege and debug mode (write):</p> <p>0101 = disable done signal generation</p> <p>others = enable done signal generation</p>

**Table 12-1621. DCC\_GCTRL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
11-8	SINGLESOT	R/W	5h	<p>The SINGLESOT bit enables/disables repetitive operation of the DCC.</p> <p>User, privilege, and debug mode (read):</p> <p>1010 = stop counting when counter0 and valid0 both reach zero</p> <p>1011 = stop counting when counter1 reaches zero</p> <p>others = continuously repeat (until error)</p> <p>Privilege and debug mode (write):</p> <p>1010 = stop counting when counter0 and valid0 both reach zero</p> <p>1011 = stop counting when counter1 reaches zero</p> <p>others = continuously repeat (until error)</p>
7-4	ERRENA	R/W	5h	<p>The ERRENA bit enables/disables the error signal.</p> <p>User, privilege, and debug mode (read):</p> <p>0101 = the error signal is disabled</p> <p>others = the error signal is enabled</p> <p>Privilege and debug mode (write):</p> <p>0101 = disable error signal generation</p> <p>others = enable error signal generation</p>
3-0	DCCENA	R/W	5h	<p>The DCCENA bit starts and stops the operation of the dcc.</p> <p>User, privilege, and debug mode (read):</p> <p>0101 = counters are stopped</p> <p>others = counters are running</p> <p>Privilege and debug mode (write):</p> <p>0101 = stop counters and error-checking</p> <p>others = load the counters with their seed values and begin counting</p>

### 12.11.1.4.2 DCC\_REV Register (Offset = 4h) [reset = X]

DCC\_REV is shown in [Figure 12-1232](#) and described in [Table 12-1623](#).

Return to [Summary Table](#).

Specifies the module version.

**Table 12-1622. DCC\_REV Instances**

Instance	Physical Address
DCC0	0080 0004h
DCC1	0080 4004h
DCC2	0080 8004h
DCC3	0080 C004h
DCC4	0081 0004h
DCC5	0081 4004h
DCC6	0081 8004h
DCC7	0081 C004h
DCC8	0082 0004h
DCC9	0082 4004h
DCC10	0082 8004h
DCC11	0082 C004h
DCC12	0083 0004h
MCU_DCC0	4010 0004h
MCU_DCC1	4011 0004h
MCU_DCC2	4012 0004h

**Figure 12-1232. DCC\_REV Register**

31	30	29	28	27	26	25	24
SCHEME		RESERVED		FUNC			
R-1h		R-X		R-0h			
23	22	21	20	19	18	17	16
FUNC							
R-0h							
15	14	13	12	11	10	9	8
RTL				MAJOR			
R-0h				R-2h			
7	6	5	4	3	2	1	0
CUSTOM		MINOR					
R-0h		R-4h					

LEGEND: R = Read Only; -n = value after reset

**Table 12-1623. DCC\_REV Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	SCHEME	R	1h	User, privilege, and debug mode (read): Returns 01. Privilege and debug mode (write): Writes have no effect.
29-28	RESERVED	R	X	



**Table 12-1623. DCC\_REV Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
27-16	FUNC	R	0h	Reflects software-compatibility. If there is no level of software compatibility, a unique func number is assigned for compatible modules, the same number is maintained. User, privilege, and debug mode (read): 0x0 Privilege and debug mode (write): Writes have no effect.
15-11	RTL	R	0h	Incremented for releases due to spec changes or post-release design changes. Reset to 0 when either MAJOR or MINOR is incremented. User, privilege, and debug mode (read): 0x0 Privilege and debug mode (write): Writes have no effect.
10-8	MAJOR	R	2h	Represents major changes to the module (e.g. entirely new features are added/changed). The major revision number for this module. User, privilege, and debug mode (read): 0x2 Privilege and debug mode (write): Writes have no effect.
7-6	CUSTOM	R	0h	Indicates a special version of the module. May not be supported by standard software. User, privilege, and debug mode (read): 0x0 Privilege and debug mode (write): Writes have no effect.
5-0	MINOR	R	4h	Represents minor changes to the module (e.g. enhancements to existing features). The minor revision number for this module. User, privilege, and debug mode (read): 0x4 Privilege and debug mode (write): Writes have no effect.

### 12.11.1.4.3 DCC\_CNTSEED0 Register (Offset = 8h) [reset = X]

DCC\_CNTSEED0 is shown in [Figure 12-1233](#) and described in [Table 12-1625](#).

Return to [Summary Table](#).

Seed value for the counter attached to clock source 0

**Table 12-1624. DCC\_CNTSEED0 Instances**

Instance	Physical Address
DCC0	0080 0008h
DCC1	0080 4008h
DCC2	0080 8008h
DCC3	0080 C008h
DCC4	0081 0008h
DCC5	0081 4008h
DCC6	0081 8008h
DCC7	0081 C008h
DCC8	0082 0008h
DCC9	0082 4008h
DCC10	0082 8008h
DCC11	0082 C008h
DCC12	0083 0008h
MCU_DCC0	4010 0008h
MCU_DCC1	4011 0008h
MCU_DCC2	4012 0008h

**Figure 12-1233. DCC\_CNTSEED0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												COUNTSEED0																			
R/W-X												R/W-0h																			

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1625. DCC\_CNTSEED0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	RESERVED	R/W	X	
19-0	COUNTSEED0	R/W	0h	<p>This field contains the seed value that gets loaded into counter 0 (clock source 0).</p> <p>User, privilege, and debug mode (read): Returns the current seed value for counter 0.</p> <p>Privilege and debug mode (write): Sets the current seed value for counter 0.</p>

#### 12.11.1.4.4 DCC\_VALIDSEED0 Register (Offset = Ch) [reset = X]

DCC\_VALIDSEED0 is shown in [Figure 12-1234](#) and described in [Table 12-1627](#).

Return to [Summary Table](#).

Seed value for the timeout counter attached to clock source 0.

**Table 12-1626. DCC\_VALIDSEED0 Instances**

Instance	Physical Address
DCC0	0080 000Ch
DCC1	0080 400Ch
DCC2	0080 800Ch
DCC3	0080 C00Ch
DCC4	0081 000Ch
DCC5	0081 400Ch
DCC6	0081 800Ch
DCC7	0081 C00Ch
DCC8	0082 000Ch
DCC9	0082 400Ch
DCC10	0082 800Ch
DCC11	0082 C00Ch
DCC12	0083 000Ch
MCU_DCC0	4010 000Ch
MCU_DCC1	4011 000Ch
MCU_DCC2	4012 000Ch

**Figure 12-1234. DCC\_VALIDSEED0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																VALIDSEED0															
R/W-X																R/W-0h															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1627. DCC\_VALIDSEED0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R/W	X	
15-0	VALIDSEED0	R/W	0h	<p>This field contains the seed value that gets loaded into the valid duration counter for clock source 0.</p> <p>User, privilege, and debug mode (read): Returns the current seed value for VALID0.</p> <p>Privilege and debug mode (write): Sets the current seed value for VALID0.</p>

#### 12.11.1.4.5 DCC\_CNTSEED1 Register (Offset = 10h) [reset = X]

DCC\_CNTSEED1 is shown in [Figure 12-1235](#) and described in [Table 12-1629](#).

Return to [Summary Table](#).

Seed value for the counter attached to clock source 1.

**Table 12-1628. DCC\_CNTSEED1 Instances**

Instance	Physical Address
DCC0	0080 0010h
DCC1	0080 4010h
DCC2	0080 8010h
DCC3	0080 C010h
DCC4	0081 0010h
DCC5	0081 4010h
DCC6	0081 8010h
DCC7	0081 C010h
DCC8	0082 0010h
DCC9	0082 4010h
DCC10	0082 8010h
DCC11	0082 C010h
DCC12	0083 0010h
MCU_DCC0	4010 0010h
MCU_DCC1	4011 0010h
MCU_DCC2	4012 0010h

**Figure 12-1235. DCC\_CNTSEED1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												COUNTSEED1																			
R/W-X												R/W-0h																			

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1629. DCC\_CNTSEED1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	RESERVED	R/W	X	
19-0	COUNTSEED1	R/W	0h	<p>This field contains the seed value that gets loaded into counter 1 (clock source 1).</p> <p>User, privilege, and debug mode (read): Returns the current seed value for counter 1.</p> <p>Privilege and debug mode (write): Sets the current seed value for counter 1.</p>

#### 12.11.1.4.6 DCC\_STATUS Register (Offset = 14h) [reset = X]

DCCSTATUS is shown in [Figure 12-1236](#) and described in [Table 12-1631](#).

Return to [Summary Table](#).

Specifies the status of the DCC Module.

**Table 12-1630. DCC\_STATUS Instances**

Instance	Physical Address
DCC0	0080 0014h
DCC1	0080 4014h
DCC2	0080 8014h
DCC3	0080 C014h
DCC4	0081 0014h
DCC5	0081 4014h
DCC6	0081 8014h
DCC7	0081 C014h
DCC8	0082 0014h
DCC9	0082 4014h
DCC10	0082 8014h
DCC11	0082 C014h
DCC12	0083 0014h
MCU_DCC0	4010 0014h
MCU_DCC1	4011 0014h
MCU_DCC2	4012 0014h

**Figure 12-1236. DCC\_STATUS Register**

31	30	29	28	27	26	25	24
RESERVED							
R/W-X							
23	22	21	20	19	18	17	16
RESERVED							
R/W-X							
15	14	13	12	11	10	9	8
RESERVED							
R/W-X							
7	6	5	4	3	2	1	0
RESERVED						DONE	ERR
R/W-X						R/W-0h	R/W-0h

LEGEND: R/W = Read/Write ; -n = value after reset

**Table 12-1631. DCC\_STATUS Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-2	RESERVED	R/W	X	

**Table 12-1631. DCC\_STATUS Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
1	DONE	R/W	0h	<p>Indicates when single-shot mode is complete without error. Writing a 1 to this bit clears the flag.</p> <p>User, privilege, and debug mode (read): 0 = single-shot mode is not done 1 = single-shot mode is done</p> <p>Privilege and debug mode (write): 0 = no effect 1 = clear the done flag</p>
0	ERR	R/W	0h	<p>Indicates whether or not an error has occurred. Writing a 1 to this bit clears the flag.</p> <p>User, privilege, and debug mode (read): 0 = an error has not occurred 1 = an error has occurred</p> <p>Privilege and debug mode (write): 0 = no effect 1 = clear the error flag</p>

#### 12.11.1.4.7 DCC\_CNT0 Register (Offset = 18h) [reset = X]

DCC\_CNT0 is shown in [Figure 12-1237](#) and described in [Table 12-1633](#).

Return to [Summary Table](#).

Value of the counter attached to clock source 0.

**Table 12-1632. DCC\_CNT0 Instances**

Instance	Physical Address
DCC0	0080 0018h
DCC1	0080 4018h
DCC2	0080 8018h
DCC3	0080 C018h
DCC4	0081 0018h
DCC5	0081 4018h
DCC6	0081 8018h
DCC7	0081 C018h
DCC8	0082 0018h
DCC9	0082 4018h
DCC10	0082 8018h
DCC11	0082 C018h
DCC12	0083 0018h
MCU_DCC0	4010 0018h
MCU_DCC1	4011 0018h
MCU_DCC2	4012 0018h

**Figure 12-1237. DCC\_CNT0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												COUNT0																			
R-X												R-0h																			

LEGEND: R = Read Only; -n = value after reset

**Table 12-1633. DCC\_CNT0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	X	
19-0	COUNT0	R	0h	This field contains the current value of counter 0. User, privilege, and debug mode (read): Returns the current value for counter 0. Privilege and debug mode (write): Writes have no effect.

### 12.11.1.4.8 DCC\_VALID0 Register (Offset = 1Ch) [reset = X]

DCC\_VALID0 is shown in [Figure 12-1238](#) and described in [Table 12-1635](#).

Return to [Summary Table](#).

Value of the valid counter attached to clock source 0.

**Table 12-1634. DCC\_VALID0 Instances**

Instance	Physical Address
DCC0	0080 001Ch
DCC1	0080 401Ch
DCC2	0080 801Ch
DCC3	0080 C01Ch
DCC4	0081 001Ch
DCC5	0081 401Ch
DCC6	0081 801Ch
DCC7	0081 C01Ch
DCC8	0082 001Ch
DCC9	0082 401Ch
DCC10	0082 801Ch
DCC11	0082 C01Ch
DCC12	0083 001Ch
MCU_DCC0	4010 001Ch
MCU_DCC1	4011 001Ch
MCU_DCC2	4012 001Ch

**Figure 12-1238. DCC\_VALID0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																VALID0															
R-X																R-0h															

LEGEND: R = Read Only; -n = value after reset

**Table 12-1635. DCC\_VALID0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	X	
15-0	VALID0	R	0h	This field contains the current value of valid counter 0. User, privilege, and debug mode (read): Returns the current value for valid counter 0. Privilege and debug mode (write): writes have no effect.



#### 12.11.1.4.9 DCC\_CNT1 Register (Offset = 20h) [reset = X]

DCC\_CNT1 is shown in [Figure 12-1239](#) and described in [Table 12-1637](#).

Return to [Summary Table](#).

Value of the counter attached to clock source 1.

**Table 12-1636. DCC\_CNT1 Instances**

Instance	Physical Address
DCC0	0080 0020h
DCC1	0080 4020h
DCC2	0080 8020h
DCC3	0080 C020h
DCC4	0081 0020h
DCC5	0081 4020h
DCC6	0081 8020h
DCC7	0081 C020h
DCC8	0082 0020h
DCC9	0082 4020h
DCC10	0082 8020h
DCC11	0082 C020h
DCC12	0083 0020h
MCU_DCC0	4010 0020h
MCU_DCC1	4011 0020h
MCU_DCC2	4012 0020h

**Figure 12-1239. DCC\_CNT1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												COUNT1																			
R-X												R-0h																			

LEGEND: R = Read Only; -n = value after reset

**Table 12-1637. DCC\_CNT1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	X	
19-0	COUNT1	R	0h	This field contains the current value of counter 1. User, privilege, and debug mode (read): Returns the current value for counter 1. Privilege and debug mode (write): writes have no effect.

### 12.11.1.4.10 DCC\_CLKSRC1 Register (Offset = 24h) [reset = X]

DCC\_CLKSRC1 is shown in [Figure 12-1240](#) and described in [Table 12-1639](#).

Return to [Summary Table](#).

Selects the clock source for counter 1.

**Table 12-1638. DCC\_CLKSRC1 Instances**

Instance	Physical Address
DCC0	0080 0024h
DCC1	0080 4024h
DCC2	0080 8024h
DCC3	0080 C024h
DCC4	0081 0024h
DCC5	0081 4024h
DCC6	0081 8024h
DCC7	0081 C024h
DCC8	0082 0024h
DCC9	0082 4024h
DCC10	0082 8024h
DCC11	0082 C024h
DCC12	0083 0024h
MCU_DCC0	4010 0024h
MCU_DCC1	4011 0024h
MCU_DCC2	4012 0024h

**Figure 12-1240. DCC\_CLKSRC1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R/W-X															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY				RESERVED								CLKSRC1			
R/W-5h				R/W-X								R/W-0h			

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1639. DCC\_CLKSRC1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R/W	X	
15-12	KEY	R/W	5h	<p>This field enables or disables clock source selection for counter 1.</p> <p>User, privilege, and debug mode (read): Returns the current value of the key.</p> <p>Privilege and debug mode (write): Sets the key value.</p> <p>Key values: 1010: The CLKSRC field selects the clock source for counter 1. others: Clock source selection is disabled.</p> <p>The secondary oscillator (clock source 1) is selected for counter 1.</p>
11-5	RESERVED	R/W	X	

**Table 12-1639. DCC\_CLKSRC1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4-0	CLKSRC1	R/W	0h	<p>This field specifies the clock source for counter 1, when the KEY field enables this feature.</p> <p>User, privilege, and debug mode (read): Returns the current value of CLKSRC.</p> <p>Privilege and debug mode (write): Sets the value of CLKSRC.</p>

### 12.11.1.4.11 DCC\_CLKSRC0 Register (Offset = 28h) [reset = X]

DCC\_CLKSRC0 is shown in [Figure 12-1241](#) and described in [Table 12-1641](#).

Return to [Summary Table](#).

Selects the clock source for counter 0.

**Table 12-1640. DCC\_CLKSRC0 Instances**

Instance	Physical Address
DCC0	0080 0028h
DCC1	0080 4028h
DCC2	0080 8028h
DCC3	0080 C028h
DCC4	0081 0028h
DCC5	0081 4028h
DCC6	0081 8028h
DCC7	0081 C028h
DCC8	0082 0028h
DCC9	0082 4028h
DCC10	0082 8028h
DCC11	0082 C028h
DCC12	0083 0028h
MCU_DCC0	4010 0028h
MCU_DCC1	4011 0028h
MCU_DCC2	4012 0028h

**Figure 12-1241. DCC\_CLKSRC0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R/W-X															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY				RESERVED								CLKSRC0			
R/W-0h				R/W-X								R/W-5h			

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1641. DCC\_CLKSRC0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	RESERVED	R/W	X	
15-12	KEY	R/W	0h	<p>This field enables or disables clock source selection for counter 0.</p> <p>User, privilege, and debug mode (read): Returns the current value of the key.</p> <p>Privilege and debug mode (write): Sets the key value.</p> <p>Key values: 1010: The CLKSRC field selects the clock source for counter 0. others: Clock source selection is disabled. The external oscillator (XTAL) is selected for counter 0.</p>
11-4	RESERVED	R/W	X	

**Table 12-1641. DCC\_CLKSRC0 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
3-0	CLKSRC0	R/W	5h	<p>This field specifies the clock source for counter 0.</p> <p>User, privilege, and debug mode (read): Returns the current value of CLKSRC0.</p> <p>Privilege and debug mode (write): Sets the value of CLKSRC0.</p>

### 12.11.1.4.12 DCC\_GCTRL2 Register (Offset = 2Ch) [reset = X]

DCC\_GCTRL2 is shown in [Figure 12-1242](#) and described in [Table 12-1643](#).

Return to [Summary Table](#).

Allows configuring different modes of operation for DCC.

**Table 12-1642. DCC\_GCTRL2 Instances**

Instance	Physical Address
DCC0	0080 002Ch
DCC1	0080 402Ch
DCC2	0080 802Ch
DCC3	0080 C02Ch
DCC4	0081 002Ch
DCC5	0081 402Ch
DCC6	0081 802Ch
DCC7	0081 C02Ch
DCC8	0082 002Ch
DCC9	0082 402Ch
DCC10	0082 802Ch
DCC11	0082 C02Ch
DCC12	0083 002Ch
MCU_DCC0	4010 002Ch
MCU_DCC1	4011 002Ch
MCU_DCC2	4012 002Ch

**Figure 12-1242. DCC\_GCTRL2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R/W-X															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				FIFO_NONERR				FIFO_READ				CONT_ON_ERR			
R/W-X				R/W-5h				R/W-5h				R/W-5h			

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1643. DCC\_GCTRL2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-12	RESERVED	R/W	X	
11-8	FIFO_NONERR	R/W	5h	<p>Enables/disables FIFO writes without the error event on completion of comparison window.</p> <p>User, privilege, and debug mode (read): Returns the current field value.</p> <p>Privilege and debug mode (write): Sets the value of field value.</p> <p>Source values: 0101: Counter values are captured to non-full FIFO only upon Error event. Others: Write counter values to non-full FIFO upon completion of comparison window regardless of error or not.</p>

**Table 12-1643. DCC\_GCTRL2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
7-4	FIFO_READ	R/W	5h	<p>Enables the counter read registers reflect FIFO output instead of the live counter value.</p> <p>User, privilege, and debug mode (read): Returns the current field value.</p> <p>Privilege and debug mode (write): Sets the value of field value.</p> <p>Source values: 0101: Counter value is read directly. Others: Counters FIFO output is read.</p>
3-0	CONT_ON_ERR	R/W	5h	<p>Continues to next window of comparison despite the error condition.</p> <p>User, privilege, and debug mode (read): Returns the current field value.</p> <p>Privilege and debug mode (write): Sets the value of field value.</p> <p>Enable values: 0101: Comparison and counter reload is stopped from advancing if error is detected. Others: Counters get reloaded with seed and continue counting despite the error condition.</p>

### 12.11.1.4.13 DCC\_STATUS2 Register (Offset = 30h) [reset = X]

DCC\_STATUS2 is shown in [Figure 12-1243](#) and described in [Table 12-1645](#).

Return to [Summary Table](#).

Specifies the status of the DCC FIFOs.

**Table 12-1644. DCC\_STATUS2 Instances**

Instance	Physical Address
DCC0	0080 0030h
DCC1	0080 4030h
DCC2	0080 8030h
DCC3	0080 C030h
DCC4	0081 0030h
DCC5	0081 4030h
DCC6	0081 8030h
DCC7	0081 C030h
DCC8	0082 0030h
DCC9	0082 4030h
DCC10	0082 8030h
DCC11	0082 C030h
DCC12	0083 0030h
MCU_DCC0	4010 0030h
MCU_DCC1	4011 0030h
MCU_DCC2	4012 0030h

**Figure 12-1243. DCC\_STATUS2 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-X							
23	22	21	20	19	18	17	16
RESERVED							
R-X							
15	14	13	12	11	10	9	8
RESERVED							
R-X							
7	6	5	4	3	2	1	0
RESERVED		COUNT1_FIFO_FULL	VALID0_FIFO_FULL	COUNT0_FIFO_FULL	COUNT1_FIFO_EMPTY	VALID0_FIFO_EMPTY	COUNT0_FIFO_EMPTY
R-X		R-0h	R-0h	R-0h	R-1h	R-1h	R-1h

LEGEND: R = Read Only; -n = value after reset

**Table 12-1645. DCC\_STATUS2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	X	



**Table 12-1645. DCC\_STATUS2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	COUNT1_FIFO_FULL	R	0h	Count1 FIFO Full. Indicates whether Count1 FIFO is full. User, privilege, and debug mode (read): 0: Count1 FIFO is not full 1: Count1 FIFO is full. Privilege and debug mode (write): Writes have no effect.
4	VALID0_FIFO_FULL	R	0h	Valid0 FIFO Full. Indicates whether Valid0 FIFO is full. User, privilege, and debug mode (read): 0: Valid0 FIFO is not full 1: Valid0 FIFO is full. Privilege and debug mode (write): Writes have no effect.
3	COUNT0_FIFO_FULL	R	0h	Count0 FIFO Full. Indicates whether Count0 FIFO is full. User, privilege, and debug mode (read): 0: Count0 FIFO is not full 1: Count0 FIFO is full. Privilege and debug mode (write): Writes have no effect.
2	COUNT1_FIFO_EMPTY	R	1h	Count1 FIFO Empty. Indicates whether Count1 FIFO is empty. User, privilege, and debug mode (read): 0: Count1 FIFO is not empty 1: Count1 FIFO is empty. Privilege and debug mode (write): Writes have no effect.
1	VALID0_FIFO_EMPTY	R	1h	Valid0 FIFO Empty. Indicates whether Valid0 FIFO is empty. User, privilege, and debug mode (read): 0: Valid0 FIFO is not empty 1: Valid0 FIFO is empty. Privilege and debug mode (write): Writes have no effect.
0	COUNT0_FIFO_EMPTY	R	1h	Count0 FIFO Empty. Indicates whether Count0 FIFO is empty. User, privilege, and debug mode (read): 0: Count0 FIFO is not empty 1: Count0 FIFO is empty. Privilege and debug mode (write): Writes have no effect.

#### 12.11.1.4.14 DCC\_ERRCNT Register (Offset = 34h) [reset = X]

DCC\_ERRCNT is shown in [Figure 12-1244](#) and described in [Table 12-1647](#).

Return to [Summary Table](#).

Counts number of errors since last clear.

**Table 12-1646. DCC\_ERRCNT Instances**

Instance	Physical Address
DCC0	0080 0034h
DCC1	0080 4034h
DCC2	0080 8034h
DCC3	0080 C034h
DCC4	0081 0034h
DCC5	0081 4034h
DCC6	0081 8034h
DCC7	0081 C034h
DCC8	0082 0034h
DCC9	0082 4034h
DCC10	0082 8034h
DCC11	0082 C034h
DCC12	0083 0034h
MCU_DCC0	4010 0034h
MCU_DCC1	4011 0034h
MCU_DCC2	4012 0034h

**Figure 12-1244. DCC\_ERRCNT Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
RESERVED																						ERRCNT															
R/W-X																						R/W-0h															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 12-1647. DCC\_ERRCNT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-10	RESERVED	R/W	X	
9-0	ERRCNT	R/W	0h	Counts the number of errors after the last write to this register or reset. If reached terminal count the count freezes. User needs to clear it.

## 12.11.2 Error Signaling Module (ESM)

This section describes the Error Signaling Module (ESM) in the device.

### 12.11.2.1 ESM Overview

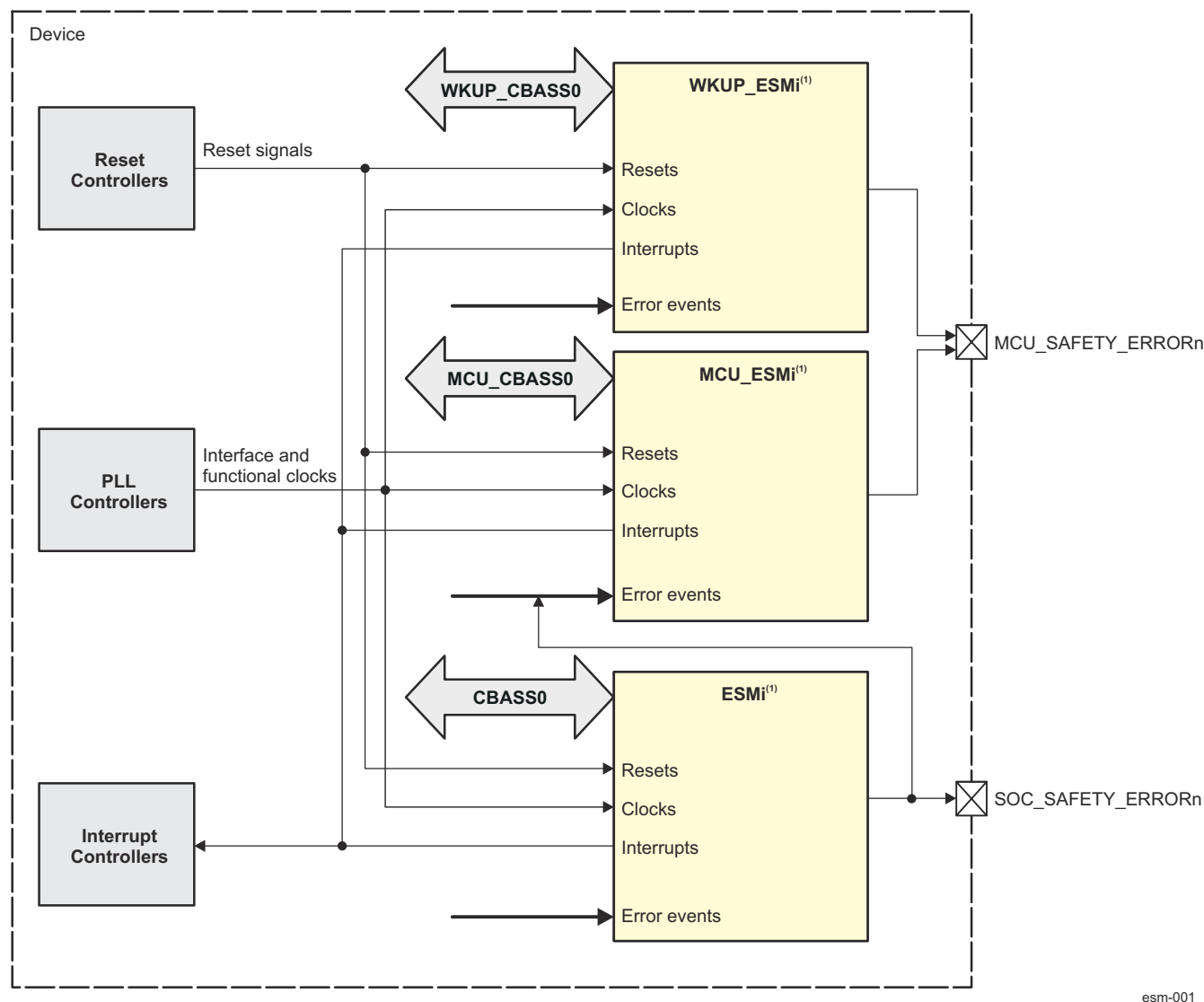
The Error Signaling Module (ESM) aggregates events and/or errors from throughout the device into one location. It can signal both low and high priority interrupts to a processor to deal with an event and/or manipulate an I/O error pin to signal an external hardware that an error has occurred. Therefore an external controller is able to reset the device or keep the system in a safe, known state.

The device has three instances of ESM modules. [Table 12-1648](#) shows ESM modules allocation within device domains.

**Table 12-1648. ESM Modules Allocation within Device Domains**

Module Instance	Domain		
	WKUP	MCU	MAIN
WKUP_ESM0	✓	-	-
MCU_ESM0	-	✓	-
ESM0	-	-	✓

[Figure 12-1245](#) shows the ESM modules overview.



esm-001

A.  $i = 0$ 
**Figure 12-1245. ESM Modules Overview**

### 12.11.2.1.1 ESM Features

Each ESM module implements the following features:

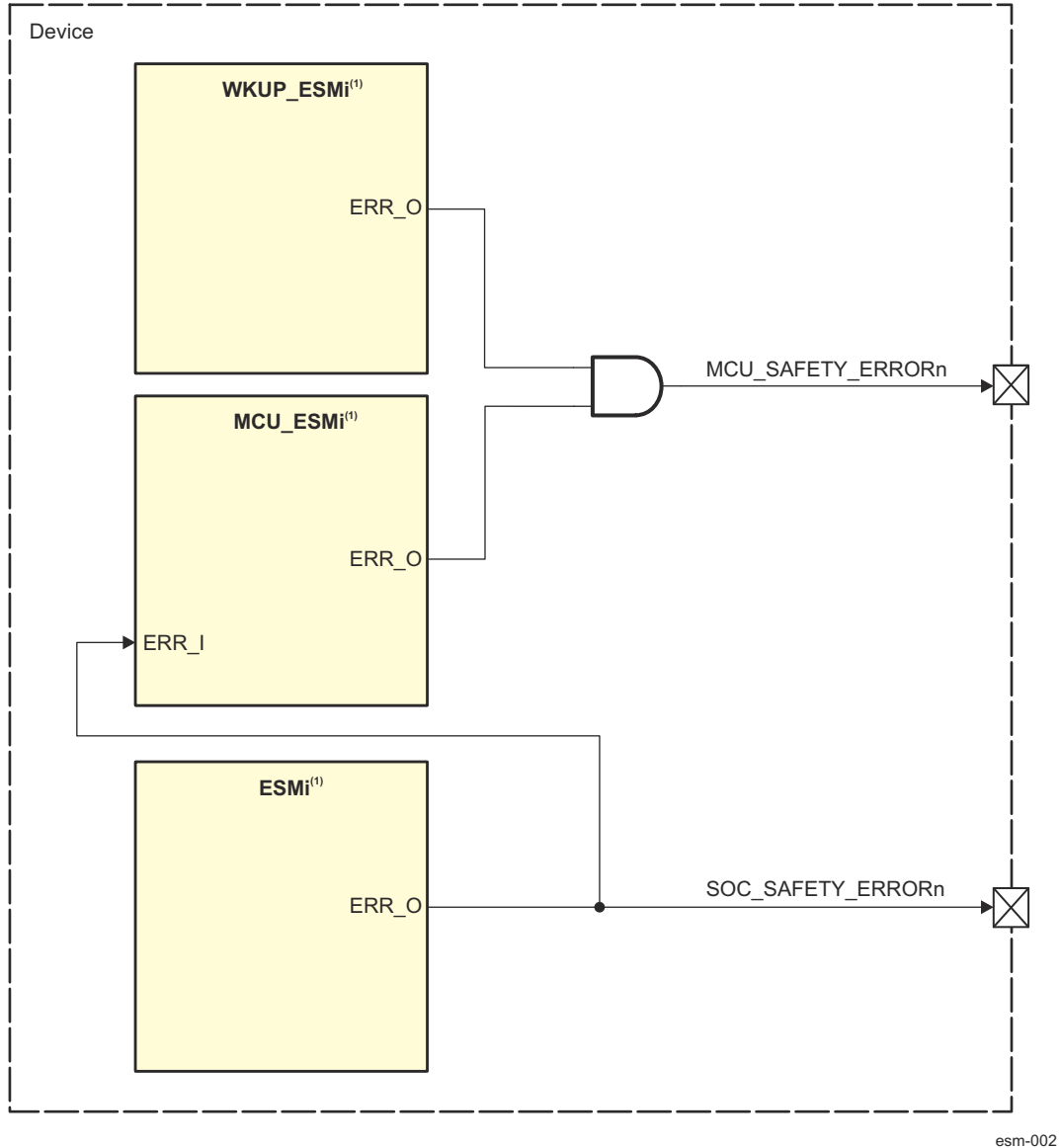
- Up to 1024 error event inputs
  - Implemented in groups of 32 events
  - Level or Pulse inputs (Pulse inputs are triple redundant)
- Selectable low and high priority interrupt error pin prioritization of each error event
- Error pin to signal severe device failure
- Configurable timebase for error signal
- Error forcing capability
- Internal redundant flops on critical fields

### 12.11.2.2 ESM Environment

The WKUP\_ESM0, MCU\_ESM0, and ESM0 modules are hereinafter referred to as ESM module.

This section describes the ESM external connections (environment).

Figure 12-1246 shows the ESM pins used for typical connections with external devices.



A.  $i = 0$

**Figure 12-1246. ESM Modules Environment**

Table 12-1649 describes the ESM I/O signals.

**Table 12-1649. ESM I/O Signals**

Module Pin	Device Level Signal	I/O <sup>(1)</sup>	Description	Module Pin Reset Value <sup>(1)</sup>
<b>WKUP_ESM0</b>				
ERR_O	MCU_SAFETY_ERRORn	O	Active low error output signal.	0 <sup>(2)</sup>
<b>MCU_ESM0</b>				

**Table 12-1649. ESM I/O Signals (continued)**

Module Pin	Device Level Signal	I/O <sup>(1)</sup>	Description	Module Pin Reset Value <sup>(1)</sup>
ERR_O	MCU_SAFETY_ERRORn	O	Active low error output signal.	0 <sup>(2)</sup>
<b>ESM0</b>				
ERR_O	SOC_SAFETY_ERRORn	O	Active low error output signal.	0 <sup>(2)</sup>

(1) O = Output;

(2) During POR, the pin state is weak pull-down and the I/O is under the control of the device. Once PORz are de-asserted, SAFETY\_ERRORn will be driven by the ESM module

### Note

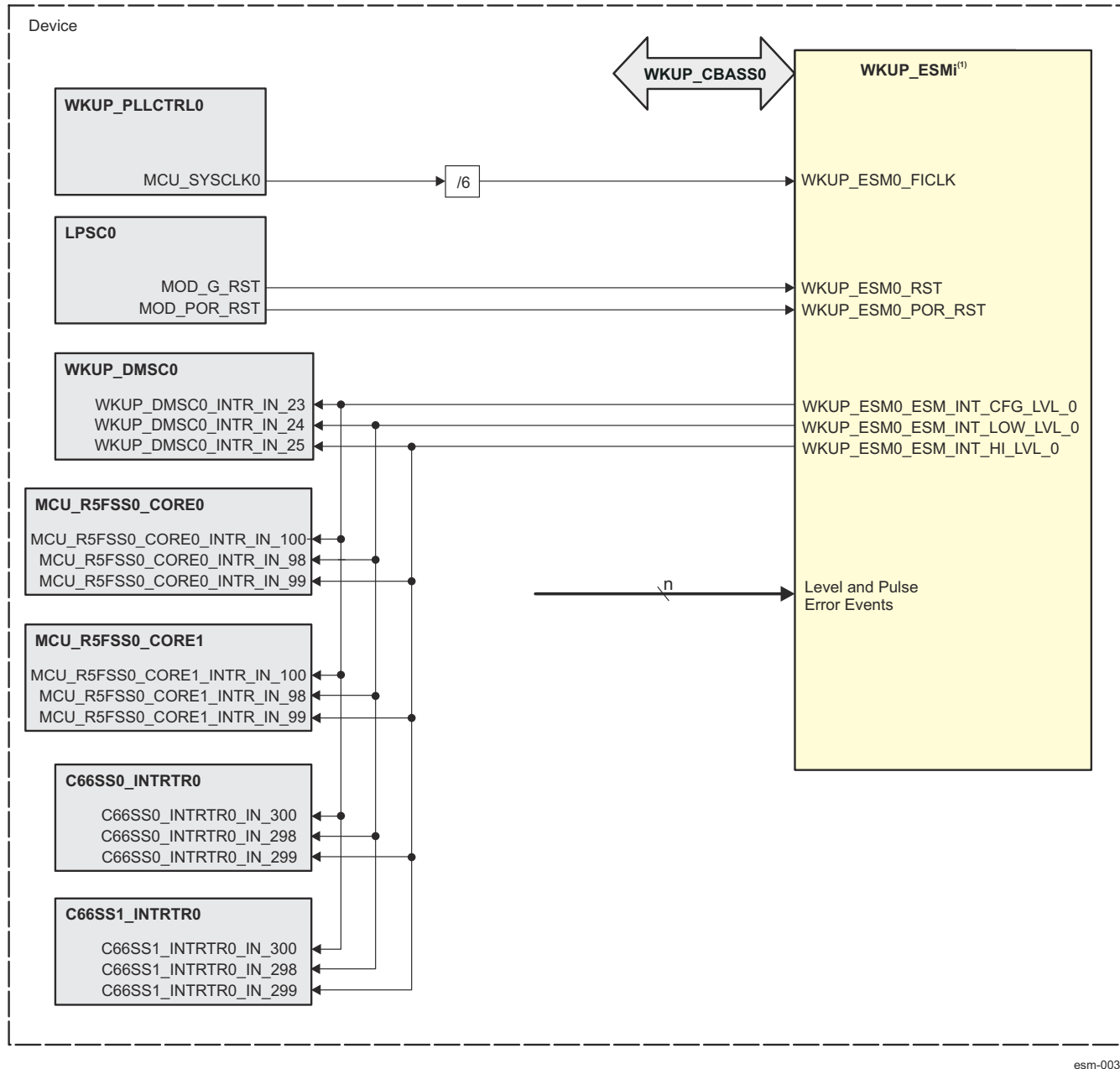
For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

### 12.11.2.3 ESM Integration

This section describes the ESM integration in the device, including information about clocks, resets, and hardware requests.

#### 12.11.2.3.1 ESM Integration in WKUP Domain

There is one ESM module integrated in the device WKUP domain - WKUP\_ESM0. [Figure 12-1247](#) shows the integration of WKUP\_ESM0.



esm-003

A.  $i = 0$

**Figure 12-1247. WKUP\_ESM Integration**

[Table 12-1650](#) through [Table 12-1652](#) summarize the integration of ESM in the device WKUP domain.

**Table 12-1650. WKUP\_ESM Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
WKUP_ESM0	WKUP_PSC0	PD0	LPSC0	WKUP_CBASS0

**Table 12-1651. WKUP\_ESM Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
WKUP_ESM0	WKUP_ESM0_FICLK	MCU_SYSCLK0/6	WKUP_PLLCTRL0	WKUP_ESM0 Interface and Functional clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
WKUP_ESM0	WKUP_ESM0_RST	MOD_G_RST	LPSC0	WKUP_ESM0 Asynchronous module reset
	WKUP_ESM0_POR_RST	MOD_POR_RST	LPSC0	WKUP_ESM0 Power-on module reset

**Table 12-1652. WKUP\_ESM Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
WKUP_ESM0	WKUP_ESM0_ESM_INT_C_FG_LVL_0	WKUP_DMSC0_INTR_IN_23	WKUP_DMSC0	WKUP_ESM0 configuration error interrupt	Level
	WKUP_ESM0_ESM_INT_L_OW_LVL_0	WKUP_DMSC0_INTR_IN_24	WKUP_DMSC0	WKUP_ESM0 low priority interrupt	Level
	WKUP_ESM0_ESM_INT_HI_LVL_0	WKUP_DMSC0_INTR_IN_25	WKUP_DMSC0	WKUP_ESM0 high priority interrupt	Level
	WKUP_ESM0_ESM_INT_C_FG_LVL_0	MCU_R5FSS0_CORE0_INTR_IN_100	MCU_R5FSS0_CORE0	WKUP_ESM0 configuration error interrupt	Level
	WKUP_ESM0_ESM_INT_L_OW_LVL_0	MCU_R5FSS0_CORE0_INTR_IN_98	MCU_R5FSS0_CORE0	WKUP_ESM0 low priority interrupt	Level
	WKUP_ESM0_ESM_INT_HI_LVL_0	MCU_R5FSS0_CORE0_INTR_IN_99	MCU_R5FSS0_CORE0	WKUP_ESM0 high priority interrupt	Level
	WKUP_ESM0_ESM_INT_C_FG_LVL_0	MCU_R5FSS0_CORE1_INTR_IN_100	MCU_R5FSS0_CORE1	WKUP_ESM0 configuration error interrupt	Level
	WKUP_ESM0_ESM_INT_L_OW_LVL_0	MCU_R5FSS0_CORE1_INTR_IN_98	MCU_R5FSS0_CORE1	WKUP_ESM0 low priority interrupt	Level
	WKUP_ESM0_ESM_INT_HI_LVL_0	MCU_R5FSS0_CORE1_INTR_IN_99	MCU_R5FSS0_CORE1	WKUP_ESM0 high priority interrupt	Level
	WKUP_ESM0_ESM_INT_C_FG_LVL_0	C66SS0_INTRTR0_IN_300	C66SS0_INTRTR0	WKUP_ESM0 configuration error interrupt	Level
	WKUP_ESM0_ESM_INT_L_OW_LVL_0	C66SS0_INTRTR0_IN_298	C66SS0_INTRTR0	WKUP_ESM0 low priority interrupt	Level
	WKUP_ESM0_ESM_INT_HI_LVL_0	C66SS0_INTRTR0_IN_299	C66SS0_INTRTR0	WKUP_ESM0 high priority interrupt	Level
WKUP_ESM0	WKUP_ESM0_ESM_INT_C_FG_LVL_0	C66SS1_INTRTR0_IN_300	C66SS1_INTRTR0	WKUP_ESM0 configuration error interrupt	Level
	WKUP_ESM0_ESM_INT_L_OW_LVL_0	C66SS1_INTRTR0_IN_298	C66SS1_INTRTR0	WKUP_ESM0 low priority interrupt	Level
	WKUP_ESM0_ESM_INT_HI_LVL_0	C66SS1_INTRTR0_IN_299	C66SS1_INTRTR0	WKUP_ESM0 high priority interrupt	Level



---

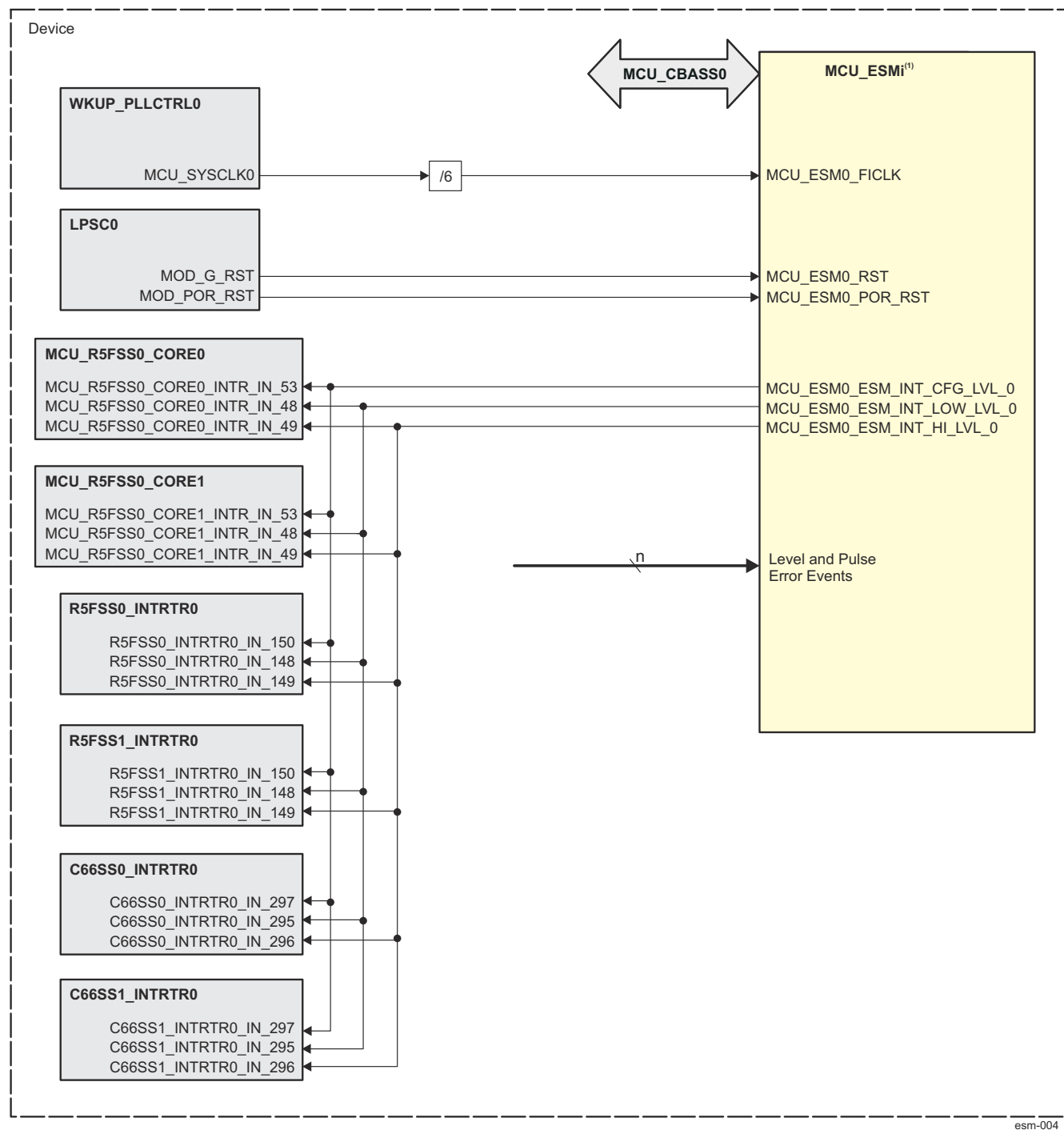
**Note**

[Table 12-1652](#) lists only the WKUP\_ESM0 interrupt outputs. For the mapping of system interrupt error events to WKUP\_ESM0 interrupt inputs, see *Interrupt Sources*.

---

**12.11.2.3.2 ESM Integration in MCU Domain**

There is one ESM module integrated in the device MCU domain - MCU\_ESM0. [Figure 12-1247](#) shows the integration of MCU\_ESM0.



A.  $i = 0$

**Figure 12-1248. MCU\_ESM Integration**

Table 12-1650 through Table 12-1652 summarize the integration of ESM in the device MCU domain.

**Table 12-1653. MCU\_ESM Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect

**Table 12-1653. MCU\_ESM Integration Attributes (continued)**

MCU_ESM0	WKUP_PSC0	PD0	LPSC0	MCU_CBASS0
----------	-----------	-----	-------	------------

**Table 12-1654. MCU\_ESM Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
MCU_ESM0	MCU_ESM0_FICLK	MCU_SYSCLK0/6	WKUP_PLLCTRL0	MCU_ESM0 Interface and Functional clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
MCU_ESM0	MCU_ESM0_RST	MOD_G_RST	LPSC0	MCU_ESM0 Asynchronous module reset
	MCU_ESM0_POR_RST	MOD_POR_RST	LPSC0	MCU_ESM0 Power-on module reset

**Table 12-1655. MCU\_ESM Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCU_ESM0	MCU_ESM0_ESM_INT_LOW_LVL_0	MCU_R5FSS0_CORE0_IN_TR_IN_48	MCU_R5FSS0_CORE0	MCU_ESM0 low priority interrupt	Level
	MCU_ESM0_ESM_INT_HI_LVL_0	MCU_R5FSS0_CORE0_IN_TR_IN_49	MCU_R5FSS0_CORE0	MCU_ESM0 high priority interrupt	Level
	MCU_ESM0_ESM_INT_CFG_LVL_0	MCU_R5FSS0_CORE0_IN_TR_IN_53	MCU_R5FSS0_CORE0	MCU_ESM0 configuration error interrupt	Level
	MCU_ESM0_ESM_INT_LOW_LVL_0	MCU_R5FSS0_CORE1_IN_TR_IN_48	MCU_R5FSS0_CORE1	MCU_ESM0 low priority interrupt	Level
	MCU_ESM0_ESM_INT_HI_LVL_0	MCU_R5FSS0_CORE1_IN_TR_IN_49	MCU_R5FSS0_CORE1	MCU_ESM0 high priority interrupt	Level
	MCU_ESM0_ESM_INT_CFG_LVL_0	MCU_R5FSS0_CORE1_IN_TR_IN_53	MCU_R5FSS0_CORE1	MCU_ESM0 configuration error interrupt	Level
	MCU_ESM0_ESM_INT_LOW_LVL_0	R5FSS0_INTRTR0_IN_148	R5FSS0_INTRTR0	MCU_ESM0 low priority interrupt	Level
	MCU_ESM0_ESM_INT_HI_LVL_0	R5FSS0_INTRTR0_IN_149	R5FSS0_INTRTR0	MCU_ESM0 high priority interrupt	Level
	MCU_ESM0_ESM_INT_CFG_LVL_0	R5FSS0_INTRTR0_IN_150	R5FSS0_INTRTR0	MCU_ESM0 configuration error interrupt	Level
	MCU_ESM0_ESM_INT_LOW_LVL_0	R5FSS1_INTRTR0_IN_148	R5FSS1_INTRTR0	MCU_ESM0 low priority interrupt	Level
	MCU_ESM0_ESM_INT_HI_LVL_0	R5FSS1_INTRTR0_IN_149	R5FSS1_INTRTR0	MCU_ESM0 high priority interrupt	Level
	MCU_ESM0_ESM_INT_CFG_LVL_0	R5FSS1_INTRTR0_IN_150	R5FSS1_INTRTR0	MCU_ESM0 configuration error interrupt	Level
	MCU_ESM0_ESM_INT_LOW_LVL_0	C66SS0_INTRTR0_IN_295	C66SS0_INTRTR0	MCU_ESM0 low priority interrupt	Level
	MCU_ESM0_ESM_INT_HI_LVL_0	C66SS0_INTRTR0_IN_296	C66SS0_INTRTR0	MCU_ESM0 high priority interrupt	Level
	MCU_ESM0_ESM_INT_CFG_LVL_0	C66SS0_INTRTR0_IN_297	C66SS0_INTRTR0	MCU_ESM0 configuration error interrupt	Level
	MCU_ESM0_ESM_INT_LOW_LVL_0	C66SS1_INTRTR0_IN_295	C66SS1_INTRTR0	MCU_ESM0 low priority interrupt	Level
	MCU_ESM0_ESM_INT_HI_LVL_0	C66SS1_INTRTR0_IN_296	C66SS1_INTRTR0	MCU_ESM0 high priority interrupt	Level

**Table 12-1655. MCU\_ESM Hardware Requests (continued)**

MCU_ESM0_ESM_INT_C FG_LVL_0	C66SS1_INTRTR0_IN_297	C66SS1_INTRTR0	MCU_ESM0 configuration error interrupt	Level
--------------------------------	-----------------------	----------------	---	-------

---

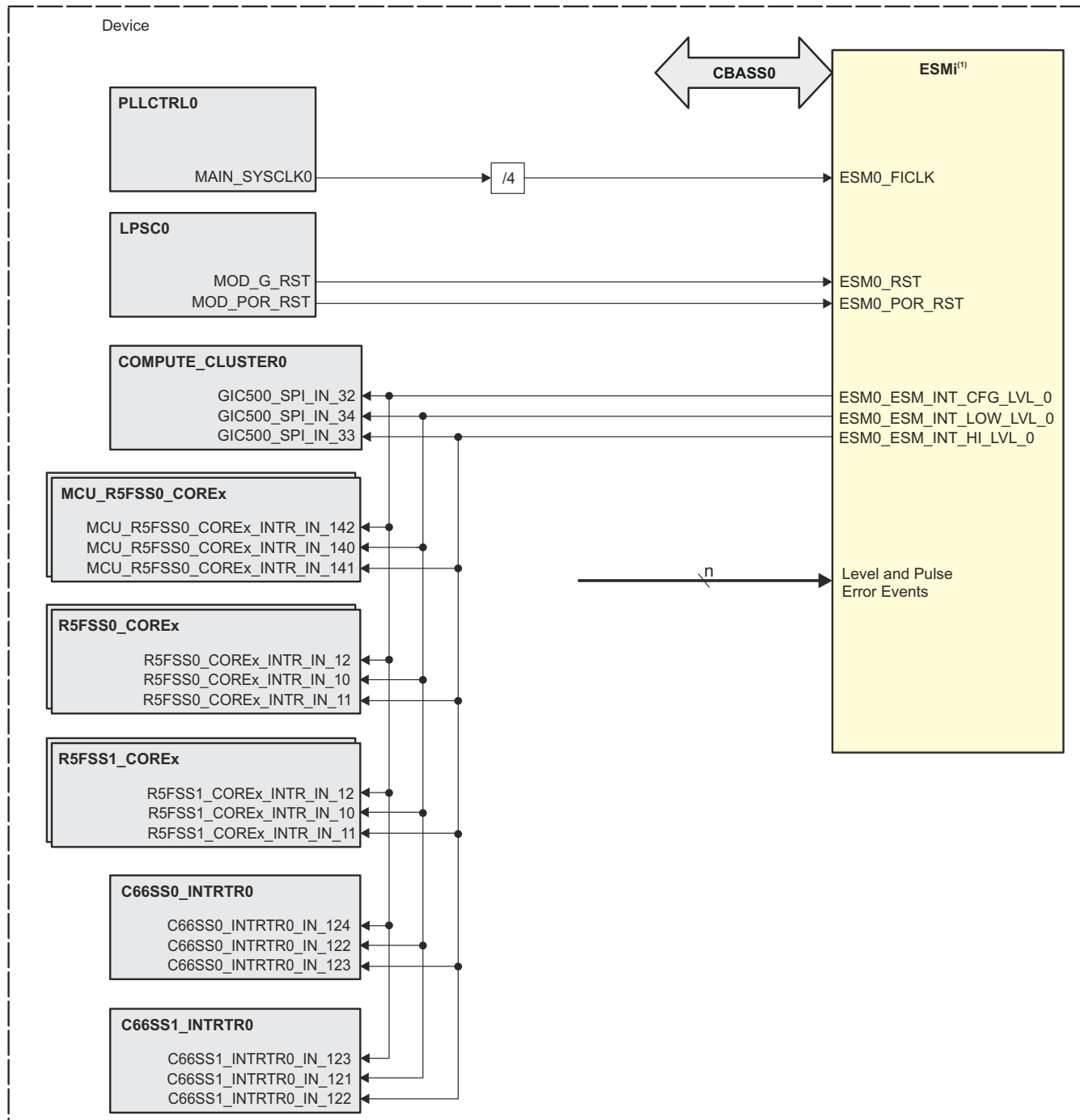
**Note**

[Table 12-1655](#) lists only the MCU\_ESM0 interrupt outputs. For the mapping of system interrupt error events to MCU\_ESM0 interrupt inputs, see *Interrupt Sources*.

---

**12.11.2.3.3 ESM Integration in MAIN Domain**

There is one ESM module integrated in the device MAIN domain - ESM0. [Figure 12-1247](#) shows the integration of ESM0.



esm-005

- A.  $i = 0$   
B.  $x = 0$  to  $1$

**Figure 12-1249. ESM Integration**

Table 12-1650 through Table 12-1652 summarize the integration of ESM in the device MAIN domain.

**Table 12-1656. ESM Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect

**Table 12-1656. ESM Integration Attributes (continued)**

ESM0	PSC0	PD0	LPSC0	CBASS0
------	------	-----	-------	--------

**Table 12-1657. ESM Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
ESM0	ESM0_FICLK	MAIN_SYSCLK0/4	PLLCTRL0	ESM0 Interface and Functional clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
ESM0	ESM0_RST	MOD_G_RST	LPSC0	ESM0 Asynchronous module reset
	ESM0_POR_RST	MOD_POR_RST	LPSC0	ESM0 Power-on module reset

**Table 12-1658. ESM Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
ESM0	ESM0_ESM_INT_LOW_LVL_0	GIC500_SPI_IN_34	COMPUTE_CLUSTER0	ESM0 low priority interrupt	Level
	ESM0_ESM_INT_HI_LVL_0	GIC500_SPI_IN_33	COMPUTE_CLUSTER0	ESM0 high priority interrupt	Level
	ESM0_ESM_INT_CFG_LVL_0	GIC500_SPI_IN_32	COMPUTE_CLUSTER0	ESM0 configuration error interrupt	Level
	ESM0_ESM_INT_LOW_LVL_0	MCU_R5FSS0_CORE0_INTR_IN_140	MCU_R5FSS0_CORE0	ESM0 low priority interrupt	Level
	ESM0_ESM_INT_HI_LVL_0	MCU_R5FSS0_CORE0_INTR_IN_141	MCU_R5FSS0_CORE0	ESM0 high priority interrupt	Level
	ESM0_ESM_INT_CFG_LVL_0	MCU_R5FSS0_CORE0_INTR_IN_142	MCU_R5FSS0_CORE0	ESM0 configuration error interrupt	Level
	ESM0_ESM_INT_LOW_LVL_0	MCU_R5FSS0_CORE1_INTR_IN_140	MCU_R5FSS0_CORE1	ESM0 low priority interrupt	Level
	ESM0_ESM_INT_HI_LVL_0	MCU_R5FSS0_CORE1_INTR_IN_141	MCU_R5FSS0_CORE1	ESM0 high priority interrupt	Level
	ESM0_ESM_INT_CFG_LVL_0	MCU_R5FSS0_CORE1_INTR_IN_142	MCU_R5FSS0_CORE1	ESM0 configuration error interrupt	Level
	ESM0_ESM_INT_LOW_LVL_0	R5FSS0_CORE0_INTR_IN_10	R5FSS0_CORE0	ESM0 low priority interrupt	Level
	ESM0_ESM_INT_HI_LVL_0	R5FSS0_CORE0_INTR_IN_11	R5FSS0_CORE0	ESM0 high priority interrupt	Level
	ESM0_ESM_INT_CFG_LVL_0	R5FSS0_CORE0_INTR_IN_12	R5FSS0_CORE0	ESM0 configuration error interrupt	Level
	ESM0_ESM_INT_LOW_LVL_0	R5FSS0_CORE1_INTR_IN_10	R5FSS0_CORE1	ESM0 low priority interrupt	Level
	ESM0_ESM_INT_HI_LVL_0	R5FSS0_CORE1_INTR_IN_11	R5FSS0_CORE1	ESM0 high priority interrupt	Level
	ESM0_ESM_INT_CFG_LVL_0	R5FSS0_CORE1_INTR_IN_12	R5FSS0_CORE1	ESM0 configuration error interrupt	Level
	ESM0_ESM_INT_LOW_LVL_0	R5FSS1_CORE0_INTR_IN_10	R5FSS1_CORE0	ESM0 low priority interrupt	Level
	ESM0_ESM_INT_HI_LVL_0	R5FSS1_CORE0_INTR_IN_11	R5FSS1_CORE0	ESM0 high priority interrupt	Level

**Table 12-1658. ESM Hardware Requests (continued)**

ESM0_ESM_INT_CFG_LV_L_0	R5FSS1_CORE0_INTR_IN_12	R5FSS1_CORE0	ESM0 configuration error interrupt	Level
ESM0_ESM_INT_LOW_LV_L_0	R5FSS1_CORE1_INTR_IN_10	R5FSS1_CORE1	ESM0 low priority interrupt	Level
ESM0_ESM_INT_HI_LVL_0	R5FSS1_CORE1_INTR_IN_11	R5FSS1_CORE1	ESM0 high priority interrupt	Level
ESM0_ESM_INT_CFG_LV_L_0	R5FSS1_CORE1_INTR_IN_12	R5FSS1_CORE1	ESM0 configuration error interrupt	Level
ESM0_ESM_INT_LOW_LV_L_0	C66SS0_INTRTR0_IN_122	C66SS0_INTRTR0	ESM0 low priority interrupt	Level
ESM0_ESM_INT_HI_LVL_0	C66SS0_INTRTR0_IN_123	C66SS0_INTRTR0	ESM0 high priority interrupt	Level
ESM0_ESM_INT_CFG_LV_L_0	C66SS0_INTRTR0_IN_124	C66SS0_INTRTR0	ESM0 configuration error interrupt	Level
ESM0_ESM_INT_LOW_LV_L_0	C66SS1_INTRTR0_IN_121	C66SS1_INTRTR0	ESM0 low priority interrupt	Level
ESM0_ESM_INT_HI_LVL_0	C66SS1_INTRTR0_IN_122	C66SS1_INTRTR0	ESM0 high priority interrupt	Level
ESM0_ESM_INT_CFG_LV_L_0	C66SS1_INTRTR0_IN_123	C66SS1_INTRTR0	ESM0 configuration error interrupt	Level

### Note

[Table 12-1658](#) lists only the ESM0 interrupt outputs. For the mapping of system interrupt error events to ESM0 interrupt inputs, see *Interrupt Sources*.

#### 12.11.2.4 ESM Functional Description

The Error Signaling Module (ESM) centralizes fault reports. It provides mechanisms to classify errors by severity and to provide programmable error response. The error classification in the ESM is determined by programmed configuration for each individual error input. For each individual error input the configuration can be set to assert an output error pin, or generate an interrupt to a CPU, or both. When an individual error input is configured to generate an interrupt, the configuration will also select whether the interrupt that is generated will be one of high priority or low priority.

By reporting the faults in a central location, the system may determine what caused the fault and what action can be taken. In general, the faults can be split into two categories:

- Corrected faults
- Non-corrected faults

The ESM reports errors in two ways:

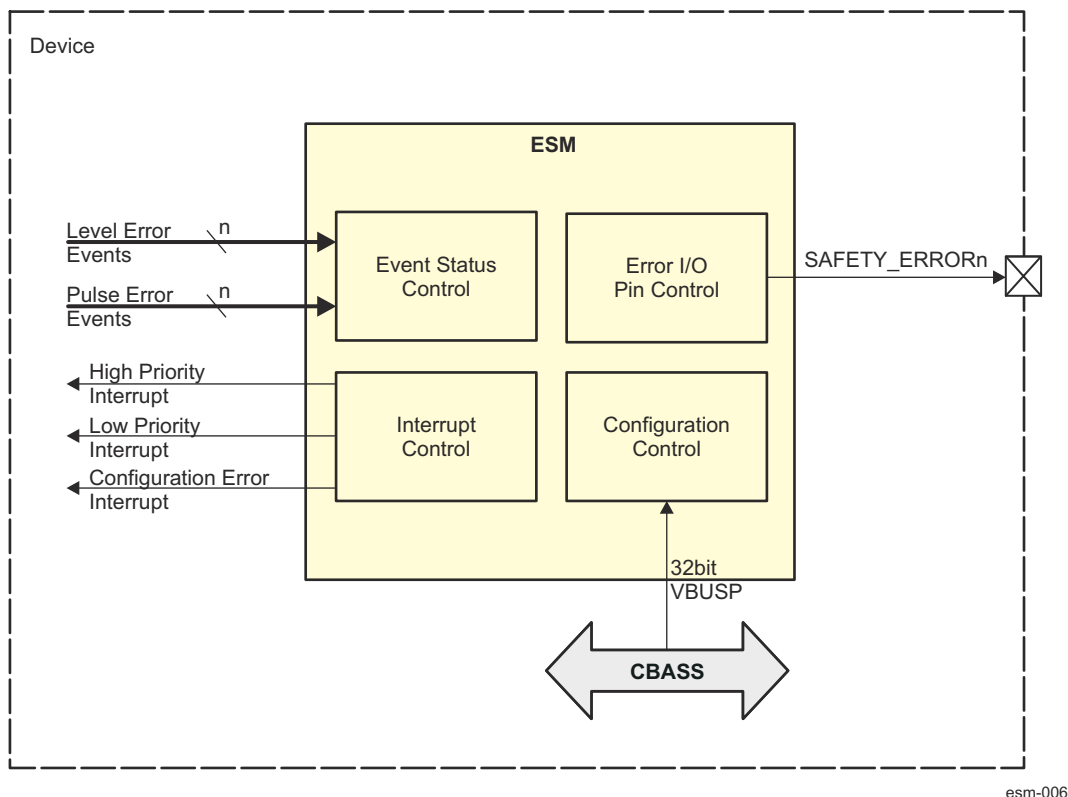
- An interrupt to a processor in the device. This allows the device to analyze and try to recover from an error.
- An external ERROR pin. This allows the system outside of the SoC to monitor for potentially fatal errors(errors that the device cannot self-recover from). Moreover, the external I/O (ERROR pin) will remain asserted (active low) for a minimum period of time. After that period of time, if the error has been cleared by an internal processor, the pin will go inactive (high). If it does not go inactive in that time, then an external agent should intervene, as there may be an unrecoverable error.

Both mechanisms can be used at the same time for the same fault, signaling both an interrupt and the external ERROR pin. This allows the device to attempt to recover, but if it fails, then the external system is still alerted. If it succeeds, then it can remove the ERROR pin assertion so that the external system knows that a potentially unsafe condition was avoided.

Lastly, the ESM does not specify any methods of intervention, only the process of alerting internal CPUs and external monitor(s) of an existing error event.

[Figure 12-1250](#) shows the ESM module block diagram.





**Figure 12-1250. ESM Block Diagram**

#### 12.11.2.4.1 ESM Interrupt Requests

The ESM module generates three output interrupts to the device interrupt controllers:

- Configuration error interrupt (see [Section 12.11.2.4.1.1](#))
- High priority error interrupt (see [Section 12.11.2.4.1.2](#))
- Low priority error interrupt (see [Section 12.11.2.4.1.3](#))

The error interrupt outputs are provided so that a processor in the device can be signaled to intervene when an error event occurs. Each error event input can be enabled, via software, to cause an error interrupt to occur (via the ESM\_INTR\_EN\_SET\_j register). Additionally, each error event input can be programmed to influence either the low priority (default) interrupt or the high priority interrupt (via the ESM\_INT\_PRIO\_j). The low priority interrupt is intended for events that are of interest, but do not require immediate intervention. For example, an indication that there was a single bit error that was corrected may signal a low priority interrupt, so that information can be collected for statistical purposes. A high priority interrupt is intended for events that need immediate attention. For example, an indication that there was an uncorrected two-bit error may be signaled as a high priority interrupt.

##### 12.11.2.4.1.1 ESM Configuration Error Interrupt

The configuration error interrupt (ESM\_INT\_CFG\_LVL\_0) indicates that there is an inconsistency in the configuration of one (or more) error group *j* registers (MMRs).

In such inconsistency in the internal copies of any of the MMRs caused by fault associated with error group *j*, the corresponding raw status will be set in the ESM\_ERR\_RAW register. If the corresponding bit is enabled in the ESM\_ERR\_EN\_SET register, a configuration error interrupt will be triggered.

When a configuration error interrupt is received, the acting processor must perform the following steps:

1. Read the ESM\_ERR\_STS register to determine which group has a configuration error
2. Write the correct values to the following group registers:

---

**Note**

If there has been a configuration error, the values in the below registers cannot be guaranteed to be correct anymore. Therefore, software should maintain a copy of the correct values prior to an error to ensure that they can be re-programmed with the correct configuration.

---

- a. ESM\_INTR\_EN\_SET\_j
  - b. ESM\_INTR\_EN\_CLR\_j
  - c. ESM\_INT\_PRIO\_j
  - d. ESM\_PIN\_EN\_SET\_j
  - e. ESM\_PIN\_EN\_CLR\_j
3. Service any pending interrupts in steps 4, 5, and 6 below
- 

**Note**

The raw status of any pending interrupts may be inconsistent. Servicing the interrupt will return it to consistency via the error group ESM\_RAW\_j register.

---

4. Write 0x1 to the appropriate bits in the ESM\_ERR\_STS register. This step will clear the raw status
  - a. If the error event is still asserted (or re-asserted) the raw status will be set back to 0x1
  - b. If there are no additional errors, the level interrupt will go low
5. Write the end of interrupt vector to the ESM\_EOI register
  - a. If there are additional configuration error interrupts pending, then a new pulse will be generated and the level interrupt will remain asserted
  - b. If there are no additional low priority error interrupts pending, there will be no new pulse.

**12.11.2.4.1.2 ESM Low Priority Error Interrupt**

Events mapped to the low priority error interrupt (ESM\_INT\_LOW\_LVL\_0) are intended to be events of interest that should be addressed eventually, not events that require immediate attention. An example would be an event indicating a corrected error. The system may want to track this for statistical purposes, but it does not require immediate attention.

Any error event can be mapped to the low priority error interrupt. A low priority error interrupt will be generated when an event is enabled to cause an interrupt (via the ESM\_INTR\_EN\_SET\_j register) and mapped to the low priority error interrupt (via ESM\_INT\_PRIO\_j register) and the raw status is set (via the ESM\_RAW\_j register).

When a low priority error interrupt is received, the acting processor must perform the following steps:

1. Read the ESM\_LOW\_PRI register
    - a. If both [31-16]PLS and [15-0]LVL bit fields are equal to 0xFFFF, then interrupt is no longer asserted and the interrupt routine has ended.
    - b. If either [31-16]PLS or [15-0]LVL bit fields are not equal to 0xFFFF, software has two options for determining what event to service:
      - i. First option: Record the value in [31-16]PLS and [15-0]LVL bit fields. Determine which is higher priority. This is based on the global event map number of the highest priority low priority error event
- 

**Note**

The global event map is neither defined by nor maintain in ESM. The global event map must be defined and maintained by software using software determined memory resources. It is conceivable that the global event map will be predefined and loaded during the execution of a secondary boot loader.

---

- ii. Second option:
  1. Read the ESM\_LOW register to determine which event group(s) have pending low priority error interrupts

2. Read the desired error group  $j$  ESM\_STS\_ $j$  register
  3. Identify which low priority interrupt to service
2. Determine, based on the global event map for the device, where the error event came from
3. Service the error event based on the IP's specification:
  - a. The system may take several actions including (but not limited to):
    - i. Fixing the error
    - ii. Resetting the errored IP peripheral
    - iii. Resetting the device
    - iv. Communicating outside the device via the error pin for outside intervention

The rest of the steps assume that the error has been handled and the system wants to clear the error event. Clearing the error event depends on whether the event is a level (see [Section 12.11.2.4.1.2.1](#)) or pulse (see [Section 12.11.2.4.1.2.2](#)) event.

#### 12.11.2.4.1.2.1 ESM Low Priority Error Level Event

When a low priority error level event has to be cleared, the acting processor must perform the following steps:

1. Clear the low priority error level event at the source
2. Write 0x1 to the appropriate bit in the error group  $j$  of the ESM\_STS\_ $j$  register. This step will clear the raw status
  - a. If the error event is still asserted (or re-asserted) the raw status will be set back to 0x1
  - b. If there are no error events, the level will de-assert

#### Note

There is a possible software race condition if software manages to write to the Clear register before the de-asserted level from the source has been synchronized to the ESM clock. If this is an issue, software may perform a read-back at the source IP before writing the clear register to insure order.

3. Write the end of interrupt vector to the ESM\_EOI interrupt register
  - a. If there are low priority error level events enabled and pending, then a new pulse will be generated
  - b. If there are no additional low priority error level events enabled and pending, there will be no new pulse
4. Write a CLEAR (0x5) to the ESM\_PIN\_CTRL register. This step is optional if the event is not enabled to influence the error pin (error group  $j$  ESM\_PIN\_EN\_SET\_ $j$  register), but may be done regardless as an extra CLEAR is not harmful.

#### 12.11.2.4.1.2.2 ESM Low Priority Error Pulse Event

When a low priority error pulse event has to be cleared, the acting processor must perform the following steps:

1. Write 0x1 to the appropriate bit in the error group  $j$  of the ESM\_STS\_ $j$  register. This will clear the raw status and will de-assert the level interrupt.
2. Write the end of interrupt vector to the ESM\_EOI interrupt register
  - a. If there are additional low priority error pulse events enabled and pending, then a new pulse will be generated and the level interrupt will remain asserted
  - b. If there are no additional low priority error pulse events enabled and pending, there will be no new pulse
3. Clear the error event at the source. The source may generate a new pulse which will show up as a new error event at the ESM
4. Write a CLEAR (0x5) to the ESM\_PIN\_CTRL register. This step is optional if the event is not enabled to influence the error pin (error group  $j$  ESM\_PIN\_EN\_SET\_ $j$  register), but may be done regardless as an extra CLEAR is not harmful.

#### 12.11.2.4.1.3 ESM High Priority Error Interrupt

Events mapped to the high priority error interrupt are intended to be events that require immediate intervention from the system because a potentially dangerous error has occurred. An example would be an event indicating an uncorrected error. The system will want to diagnose the issue and intervene to ensure there are no violations.

Any error event can be mapped to the high priority error interrupt. A high priority error interrupt will be generated when an event is enabled to cause an interrupt (via the ESM\_INTR\_EN\_SET\_j) register and mapped to the high priority interrupt (via the ESM\_INT\_PRIO\_j) register and the raw status is set (via the ESM\_RAW\_j).

When a high priority error interrupt is received, the acting processor must perform the following steps:

1. Read the ESM\_HI\_PRI register
  - a. If both [31-16] PLS and [15-0] LVL bit fields are equal to 0xFFFF, then interrupt is no longer asserted and the interrupt routine has ended.
  - b. If either [31-16] PLS or [15-0] LVL bit fields are not equal to 0xFFFF, software has two options for determining what event to service:
    - i. First option: Record the value in [31-16] PLS and [15-0] LVL bit fields. Determine which is higher priority. This is based on the global event map number of the highest priority high priority error event.

---

#### Note

The global event map is neither defined by nor maintain in ESM. The global event map must be defined and maintained by software using software determined memory resources. It is conceivable that the global event map will be predefined and loaded during the execution of a secondary boot loader.

- ii. Second option:
    1. Read the ESM\_HI register to determine which event group(s) have pending high priority error interrupts
    2. Read the desired error group *j* ESM\_STS\_j register
    3. Identify which high priority error interrupt to service
2. Determine, based on the global event map for the device, where the error event came from
3. Service the error event based on the IP's specification:
  - a. The system may take several actions including (but not limited to):
    - i. Fixing the error
    - ii. Resetting the errored IP peripheral
    - iii. Resetting the device
    - iv. Communicating outside the device via the error pin for outside intervention

The rest of the steps assume that the error has been handled and the system wants to clear the error event. Clearing the error event depends on whether the event is a level (see [Section 12.11.2.4.1.3.1](#)) or pulse (see [Section 12.11.2.4.1.3.2](#)) event.

#### 12.11.2.4.1.3.1 ESM High Priority Error Level Event

When a high priority error level event has to be cleared, the acting processor must perform the following steps:

1. Clear the error event at the source
2. Write 0x1 to the appropriate bit in the error group *j* of the ESM\_STS\_j register. This step will clear the raw status
  - a. If the error event is still asserted (or re-asserted) the raw status will be set back to 0x1
  - b. If there are no error events, the level will de-assert

---

#### Note

There is a possible software race condition if software manages to write to the Clear register before the de-asserted level from the source has been synchronized to the ESM clock. If this is an issue, software may perform a read-back at the source IP before writing the clear register to insure order.

3. Write the end of interrupt vector to the ESM\_EOI interrupt register
  - a. If there are high priority error level events enabled and pending, then a new pulse will be generated
  - b. If there are no additional high priority error level events enabled and pending, there will be no new pulse

4. Write a CLEAR (0x5) to the ESM\_PIN\_CTRL register. This step is optional if the event is not enabled to influence the error pin (error group  $j$  ESM\_PIN\_EN\_SET\_ $j$  register, but may be done regardless as an extra CLEAR is not harmful.

#### 12.11.2.4.1.3.2 ESM High Priority Error Pulse Event

When a high priority error pulse event has to be cleared, the acting processor must perform the following steps:

1. Write 0x1 to the appropriate bit in the error group  $j$  of the ESM\_STS\_ $j$  register. This will clear the raw status and will de-assert the level interrupt.
2. Write the end of interrupt vector to the ESM\_EOI interrupt register
  - a. If there are high priority error pulse events enabled and pending, then a new pulse will be generated and the level interrupt will remain asserted
  - b. If there are no additional high priority error pulse events enabled and pending, there will be no new pulse
3. Clear the error event at the source. The source may generate a new pulse which will show up as a new error event at the ESM
4. Write a CLEAR (0x5) to the ESM\_PIN\_CTRL register. This step is optional if the event is not enabled to influence the error pin (error group  $j$  ESM\_PIN\_EN\_SET\_ $j$  register), but may be done regardless as an extra CLEAR is not harmful.

#### 12.11.2.4.2 ESM Error Event Inputs

The ESM can have up to 1024 error event inputs, configurable by groups of 32. For the mapping of system interrupt error events to ESM interrupt inputs, see *Interrupt Sources*.

Level error events (active high) are synchronized to the ESM clock. This synchronized value is captured in to a flop.

Pulse error events use rising edge detection. Each pulse error event has 3 redundant inputs. Each input has its own edge detection logic. Multiple transmission protects against Single Event Upsets (SEUs, transient errors) causing a pulse to be lost during transmission and against failure of the edge detection logic. Once an edge has been detected on any of the three inputs, the ESM\_RAW\_ $j$  status is set. Subsequent pulses are likely to come concurrently or quickly enough that software will not have reacted yet. This logic is intentionally biased against false negatives and towards false positives. An SEU that causes an event where none actually occurred will cause software to be called in to action. Software must observe that there is no real error and clear the false status.

#### 12.11.2.4.3 ESM Error Pin Output

The error pin output (ERR\_O) is used to signal an external agent that it needs to (or may need to) intervene because of an error. Each error event input can be programmed, via software, to influence the error pin output (via the ESM\_PIN\_EN\_SET\_ $j$  register). The error pin output is active low.

During Power-On Reset (POR), the error pin is active (asserted low) and the device drives this via a weak internal pull-down. The I/O is under the control of the device. When POR is removed from the ESM, it will be driving the error pin inactive (high) so the device can hand over control to the ESM. The user may also add an external pull-down that is only active when the device is in reset.

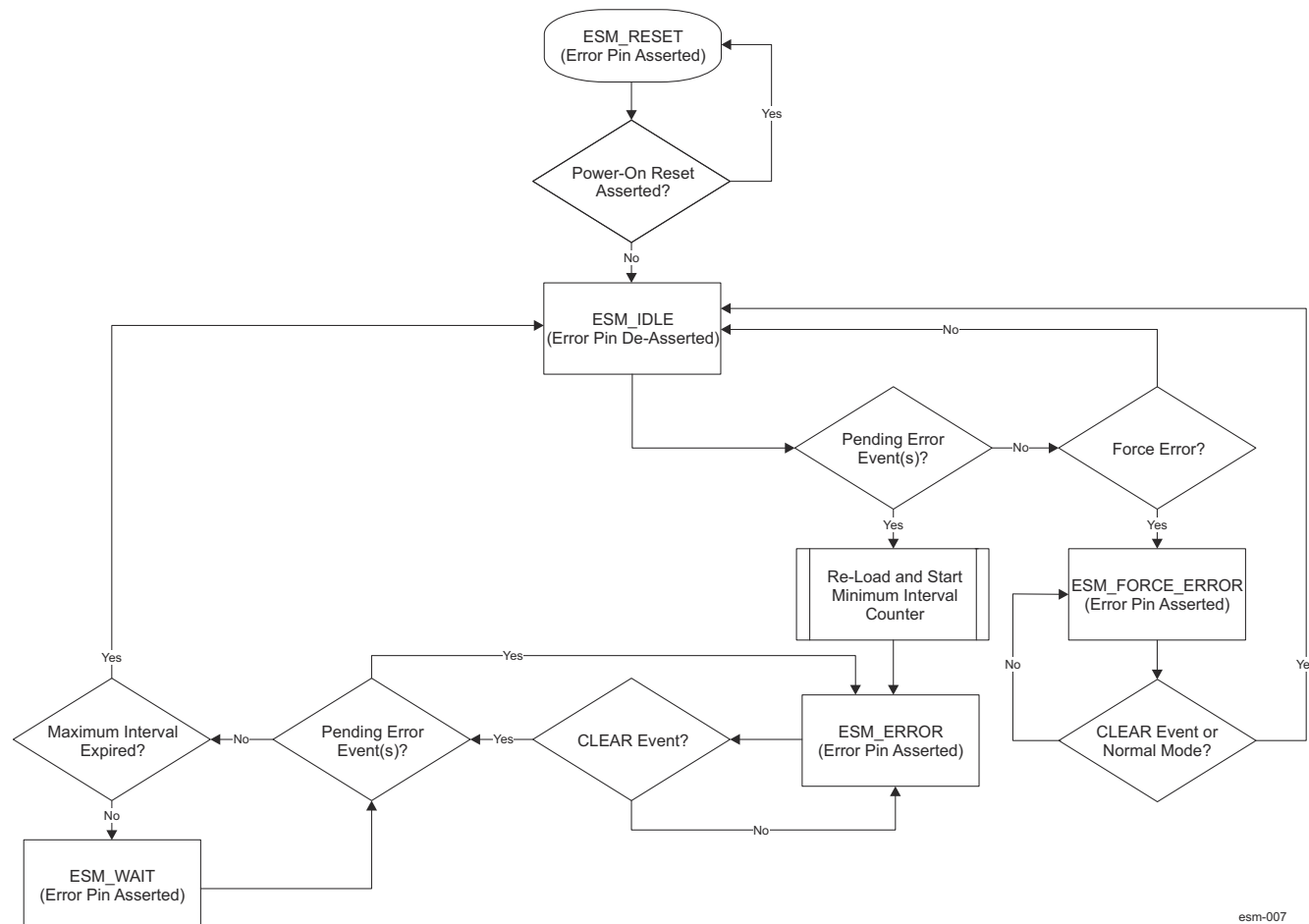
During a warm reset the state of the error pin is unchanged, that is the error pin logic is only reset by a POR. The device leaves the I/O active during a warm reset.

The ESM has also a software error forcing capability on the error pin. That is, a force error can be set via the ESM\_EN[3-0] KEY bit field.

### CAUTION

The isolation value for the ERR\_O output of ESM is active (0). This is intended and supposed to protect against an accidental transition to a low power state. If the actual low power mode transition is intended, the PMIC should be made aware by software to ignore the ESM error signal. Otherwise device resets will be asserted from companion chip.

Figure 12-1251 describes the behavior of the error pin. Not shown is that a reset (Power-On-Reset only) will immediately transition the error pin to the ESM\_RESET state and a Global Soft Reset will immediately transition the error pin to the ESM\_IDLE state. A pending error interrupt event is any error event with the raw state set and the error pin Influence enabled. There are two types of "clear" events associated with servicing the error pin. The first is to clear the status of the pending event (see Section 12.11.2.4.1 for how to clear level and pulse pending events). The second is the CLEAR event meant to de-assert the error pin.



**Figure 12-1251. ESM Error Pin State Flowchart**

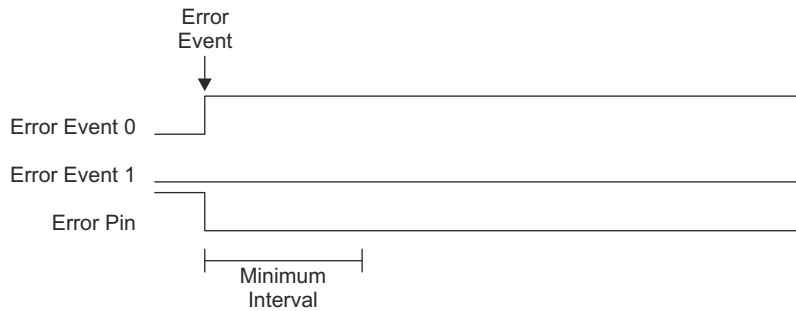
If an error event happens that has been programmed to influence the error pin, the error pin will assert (active low) for a minimum time (as programmed by the ESM\_PIN\_CNTR\_PRE register). In order for the error pin to de-assert, the following 3 things must happen:

1. The minimum time interval must expire
2. The event that caused the error pin to assert must be cleared (see Section 12.11.2.4.1)
3. A CLEAR (0x5) must be written to the ESM\_PIN\_CTRL register.

#### Note

Step 3 should happen after step 2, but either (or both) of these steps may happen before or after step 1.

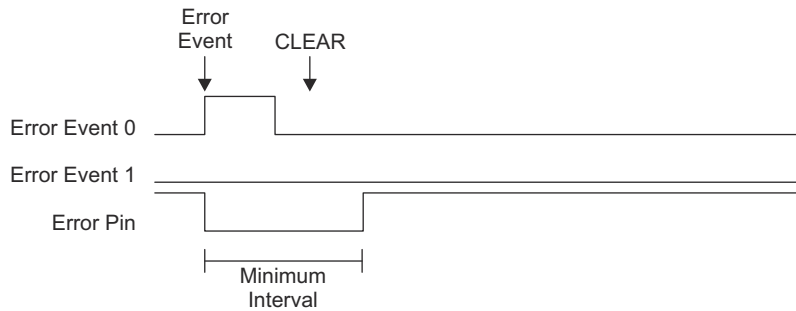
Figure 12-1252 shows a typical error pin assertion.



esm-008

**Figure 12-1252. ESM Error Pin Assertion**

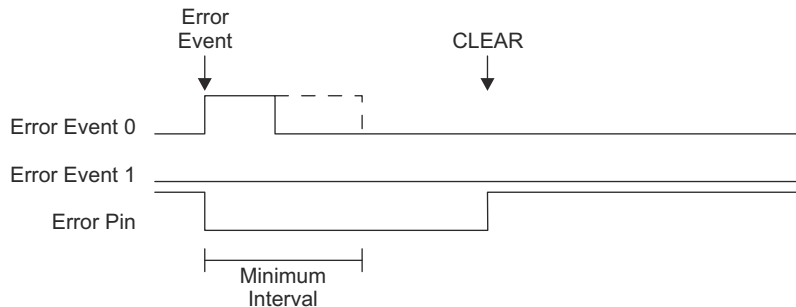
If, during the minimum time, CLEAR is written to the error key, then the error pin will de-assert after the minimum time interval, as shown in [Figure 12-1253](#).



esm-009

**Figure 12-1253. ESM Error Pin Assertion with CLEAR during Minimum Interval**

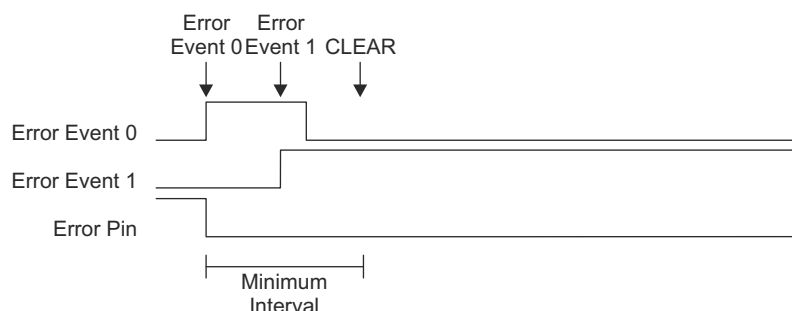
If CLEAR is not written till after the minimum time interval, the error pin will de-assert when CLEAR is written. This is regardless of whether the error interrupt event itself is removed before or after the minimum time interval, as shown by the dotted line in [Figure 12-1254](#).



esm-010

**Figure 12-1254. ESM Error Pin Asserting with CLEAR after Minimum Interval**

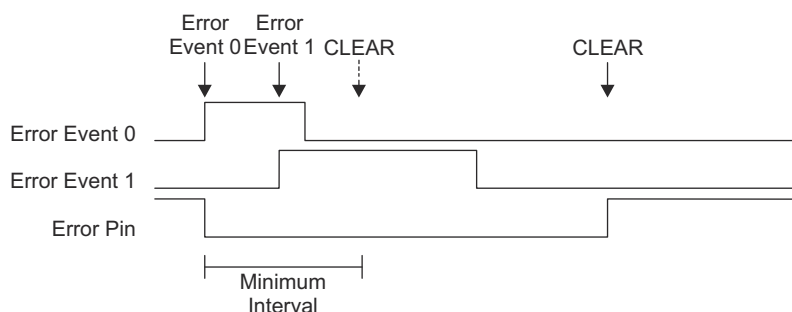
When in the ESM\_ERROR state and a CLEAR event happen, if there are still pending error events, the ESM stays in the ESM\_ERROR state with the error pin asserted. Multiple error events when in the ESM\_ERROR state do not reset the minimum time interval counter as shown in [Figure 12-1255](#).



esm-011

**Figure 12-1255. ESM Error Pin Asserting with Interval Reset by Additional Error Event(s)**

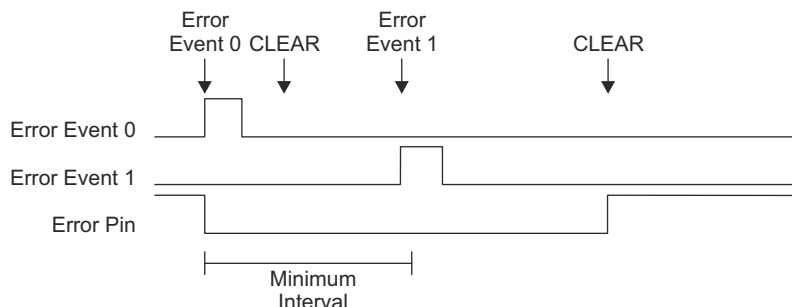
A CLEAR event causes a re-evaluation of whether there are any pending error events. As such, a single CLEAR can be used to clear the error pin after multiple error events. Multiple CLEAR events can occur (such as the one with the dotted arrow shown in Figure 12-1256), but are not necessary. No matter how many error events occur nor when (or how many) CLEAR events occur, the error pin will always be asserted for at least the minimum time interval



esm-012

**Figure 12-1256. ESM Error Pin Asserting with Single CLEAR for Multiple Events**

If all error events are cleared and the ESM is in the ESM\_WAIT state, waiting for the minimum time interval to expire, and a new error interrupt event occurs, the ESM will go back to the ESM\_ERROR state. The minimum time interval will not reset, but a new CLEAR event will be required as shown in Figure 12-1257.



esm-013

**Figure 12-1257. ESM Error Pin Asserting with New Error During Minimum Time Interval**

Table 12-1659 shows some common scenarios of how the error pin as well as the two associated registers (ESM\_PIN\_CTRL and ESM\_PIN\_STS) will be set.



**Table 12-1659. ESM Error Pin Scenarios**

Scenario	Error Pin State Value	ESM_PIN_CTRL[3-0] KEY	ESM_PIN_STS[0] VAL status value	Additional Notes
POR Asserted	0	N/A	N/A	Registers are inaccessible. Device disables the I/O and pulls down internally.
After de-assertion of POR	1	0x0 (Normal Mode)	0x0	-
After de-assertion of Warm Reset (error was not asserted when reset asserted)	1	0x0 (Normal Mode)	0x0	-
After de-assertion of Warm Reset (error was asserted when reset asserted)	0	0x0 (Normal Mode)	0x1	-
Force error pin	0	0xA (Force Error Mode)	0x0	Forcing error on the pin via software.

#### 12.11.2.4.4 ESM Minimum Time Interval

The minimum time interval is the minimum amount of time that the error pin will be asserted (active low) when an enabled error interrupt event happens. This value is system dependent, but should be enough time so that the external monitoring agent can always see the error pin asserted, but short enough so that if all of the error events are cleared, then the error pin can be de-asserted before the external agent decides to intervene. This is highly dependent on the application and the Fault Tolerant Time Interval.

The Minimum Time Interval counter is clock cycle based, therefore the time of the interval is a combination of the value in the ESM\_PIN\_CNTR\_PRE register and the clock frequency of the ESM. Software must calculate the value accordingly. The Minimum Time Interval should be set according to the needs of the application.

#### 12.11.2.4.5 ESM Protection for Registers

The configuration for each error group *j* of registers are backed up by 3 flops in order to protect against single or double-bit errors. When written, all 3 bits are set to the same value. When read (and for functioning of the internal state machines) the value is the OR of all 3 bits. Whenever any of the bits disagree, the Configuration Error interrupt is asserted (if enabled). The registers covered by this mechanism are:

- ESM\_ERR\_RAW
- ESM\_ERR\_STS
- ESM\_ERR\_EN\_SET
- ESM\_ERR\_EN\_CLR
- ESM\_RAW<sub>j</sub>
- ESM\_STS<sub>j</sub>
- ESM\_INTR\_EN\_SET<sub>j</sub>
- ESM\_INTR\_EN\_CLR<sub>j</sub>
- ESM\_INT\_PRIO<sub>j</sub>
- ESM\_PIN\_EN\_SET<sub>j</sub>
- ESM\_PIN\_EN\_CLR<sub>j</sub>

The error pin control register ESM\_PIN\_CTRL contains a multi-bit field KEY. The key value ensures normal operation on the error pin and that an error even will be generated if one occurs. Software should periodically read check the KEY bit field value and make sure it is 0x0. If the value is not 0x0, software must re-write it to this key value (unless in test mode forcing an error on the pin) to ensure the normal operation. [Table 12-1660](#) lists the KEY values and their respective meaning.

**Table 12-1660. ESM Error Pin Control Values**

ESM_PIN_CTRL[3-0] KEY	Description
N/A	Registers are inaccessible. Device disables the I/O and pulls down internally.
0x0 (Normal)	Normal operation mode - Error pin will activate when an enabled error event occurs.

**Table 12-1660. ESM Error Pin Control Values (continued)**

ESM_PIN_CTRL[3-0] KEY	Description
0xA (Force Error)	Force error mode - Forces the error pin active. To clear the error pin (return to the ESM_IDLE state) write this field back to normal mode (writing a CLEAR event will also work). Force error mode must be set only while in IDLE. Attempting force error while in another state will have no effect.
0x5 (CLEAR)	CLEAR Event - generates a CLEAR event to the ESM state machine. KEY will return to normal mode (0x0) on the next cycle.
Other Values	All other values - Normal mode. Writing any of these values will have no effect. When reading any of these values indicates that one or more bits have experienced a single event upset, software should write the field back to 0x0. The ESM will continue to operate in normal mode.

The error pin counter pre-load register ESM\_PIN\_CNTR\_PRE should also be read and checked periodically by software.

#### 12.11.2.4.6 ESM Clock Stop

The ESM can only be put in to clock stop if all of the internal state machines are idle and the [3-0] KEY bit field for the global enable command is cleared in the ESM\_EN register.

### 12.11.3 Memory Cyclic Redundancy Check (MCRC) Controller

This chapter describes the Memory Cyclic Redundancy Check (MCRC) controller in the device.

#### 12.11.3.1 MCRC Overview

VBUSM CRC controller is a module which is used to perform CRC (Cyclic Redundancy Check) to verify the integrity of a memory system. A signature representing the contents of the memory is obtained when the contents of the memory are read into MCRC Controller. The responsibility of MCRC controller is to calculate the signature for a set of data and then compare the calculated signature value against a pre-determined good signature value. MCRC controller provides four channels to perform CRC calculation on multiple memories in parallel and can be used on any memory system.

**Table 12-1661. MCRC Allocation Across Device Domains**

Instance	Domain		
	WKUP	MCU	MAIN
NAVSS0_MCRC0	-	-	✓ (NAVSS)
MCU_NAVSS0_MCRC0	-	✓ (MCU NAVSS)	-

#### 12.11.3.1.1 MCRC Features

MCRC has the following features:

- Four channels to perform background signature verification on any memory subsystem
- Data compression on 8-, 16-, 32-, and 64-bit data size
- Maximum-length PSA (Parallel Signature Analysis) register constructed based on 64-bit primitive polynomial
- Each channel has a CRC Value Register which contains the pre-determined CRC value
- Use timed base event trigger from timer to initiate DMA data transfer
- Programmable 20-bit pattern counter per channel to count the number of data patterns for compression
- Three modes of operation:
  - Auto
  - Semi-CPU
  - Full-CPU
- For each channel, CRC can be performed either by MCRC Controller or by CPU
- Automatically performs signature verification without CPU intervention in AUTO mode
- Generates interrupt to CPU in Semi-CPU mode to allow CPU to perform signature verification itself
- Generates CRC fail interrupt in AUTO mode if signature verification fails
- Generates Timeout interrupt if CRC is not performed within the time limit
- Generates DMA request per channel to initiate CRC value transfer
- An 128-byte block burst address for the PSA register to DMA without constant mode bus attribute

#### 12.11.3.1.2 MCRC Not Supported Features

The following features are not supported by the module:

- Data trace capability on VBUSM, ITCM and DTCM data buses.



**Table 12-1664. MCRC Hardware Requests (continued)**

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
NAVSS0_MCRC0	INT_MCRC_INTR	IN_INTR[388]	INTR_ROUTER0	MCRC Interrupt signal	Level
DMA Events					
Module Instance	Module DMA Event	Destination DMA Event Input	Destination	Description	Type
NAVSS0_MCRC0	DMA_EVENT_INTR0	IN_INTR[384]	INTR_ROUTER0	MCRC DMA event for channel 1	Pulse
		INTAGGR_LEVI_PEND[0]	UDMASS_INTR_AGGR0		
	DMA_EVENT_INTR1	IN_INTR[385]	INTR_ROUTER0	MCRC DMA event for channel 2	Pulse
		INTAGGR_LEVI_PEND[1]	UDMASS_INTR_AGGR0		
	DMA_EVENT_INTR2	IN_INTR[386]	INTR_ROUTER0	MCRC DMA event for channel 3	Pulse
		INTAGGR_LEVI_PEND[2]	UDMASS_INTR_AGGR0		
	DMA_EVENT_INTR3	IN_INTR[387]	INTR_ROUTER0	MCRC DMA event for channel 4	Pulse
		INTAGGR_LEVI_PEND[3]	UDMASS_INTR_AGGR0		

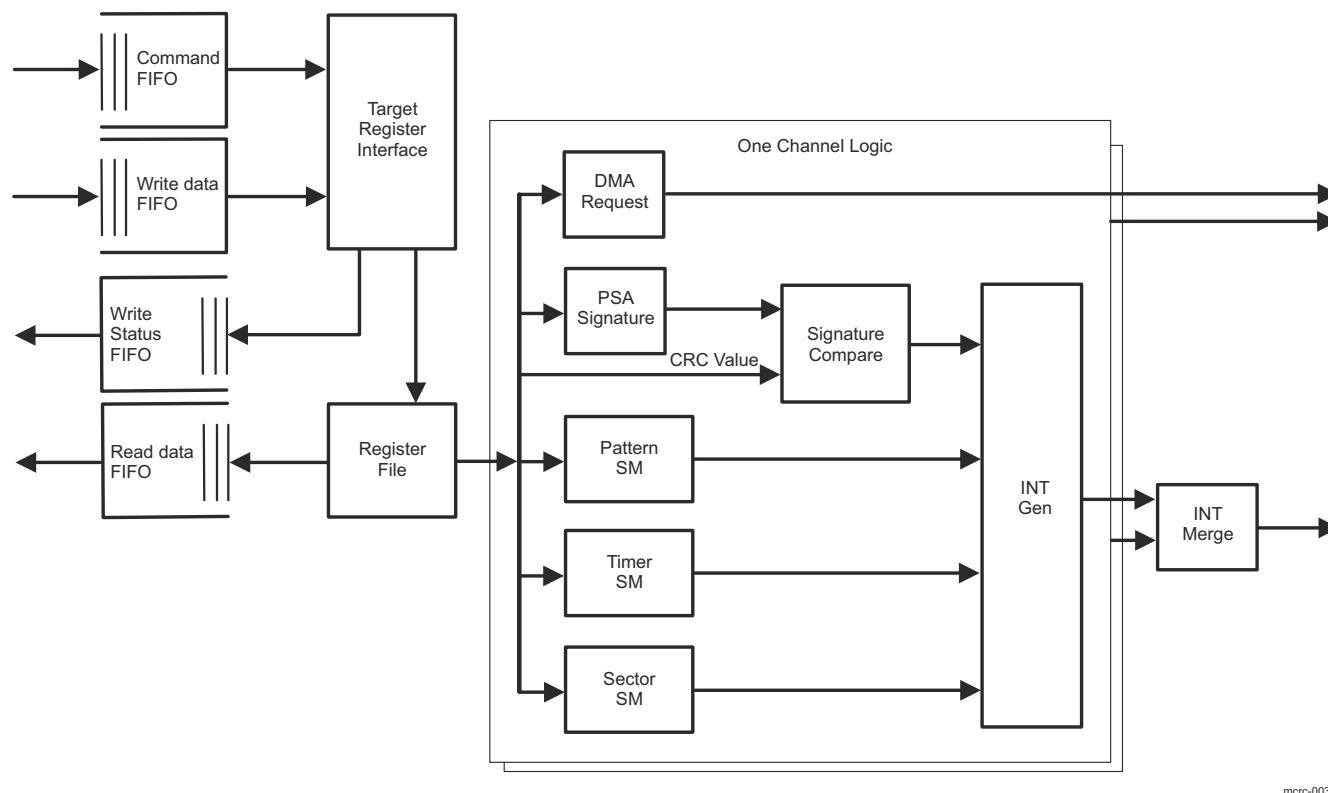
### Note

If DMA\_EVENT\_INTR[0-3] needs to be converted to level interrupts, then UDMASS\_INTR\_AGGR0 can be used to generate events to the PSILSS. PSILSS would route events back to the interrupt aggregator to be turned into level output interrupts.

### 12.11.3.3 MCRC Functional Description

#### 12.11.3.3.1 MCRC Block Diagram

Figure 12-1259 shows the MCRC internal blocks.



**Figure 12-1259. MCRC Block Diagram**

- **Command FIFO**: The Command FIFO pipelines the commands to the target register interface. The Command and Write FIFOs allow the data to be coincident for processing. If there is no space for writes in the Write Status FIFO or no space in the Read Data FIFO for reads, the command processing will be halted until there is space in the appropriate FIFO. This FIFO is 4 elements deep.
- **Write FIFO**: The Write FIFO pipelines the write data to the target register interface. The Command and Write FIFOs allow the data to be coincident for processing. If there is no space for writes in the Write Status FIFO or no space in the Read Data FIFO for reads, the command processing will be halted until there is space in the appropriate FIFO. This FIFO is 2 elements deep.
- **Write Status FIFO**: The Write Status FIFO pipelines the write status back to the VBUSM. A write status will be issued on the final data phase of a write command. This FIFO is 2 elements deep.
- **Read Data FIFO**: The Read Data FIFO pipelines the read data back to the VBUSM. This FIFO is 3 elements deep.
- **Target Register Interface**: The Target Register Interface directs the written data to the register file.
- **PSA Signature**: The PSA Signature creates the signature of the data written. This data will then be compared to the CRC Value or read by software to determine goodness.
- **Pattern State Machine**: The Pattern State Machine determines when a block of data has been serviced.
- **Timer State Machine**: The Timer State Machine determines when overrun and under-run events are detected.
- **Sector State Machine**: The Sector State Machine determines when a sector error should be captured so the software can determine the errant block of data.
- **Signature Compare**: The Signature Compare block compares the current signature to the CRC Value register and sends the result to the Interrupt Generation block.

### 12.11.3.3.2 MCRC General Operation

There are four channels in MCRC controller and for each channel there is a PSA (Parallel Signature Analysis) Signature Register (MCRC\_PSA\_SIGREGL1-4) and a CRC (Cyclic Redundancy Check) Value Register (MCRC\_CRC\_REGL1-4).

A memory can be organized into multiple sectors with each sector consisting of multiple data patterns. A data pattern can be a 8-, 16-, 32-, or 64-bit data. MCRC module performs the signature calculation and compares the signature to a pre-determined value. The PSA Signature Register compresses an incoming data pattern into a signature when it is written. When one sector of data patterns are written into PSA Signature Register, a final signature corresponding to the sector is obtained. CRC Value Register stores the pre-determined signature corresponding to one sector of data patterns. The calculated signature and the pre-determined signature are then compared to each other for signature verification. To minimize CPU's involvement, data patterns transfer can be carried out at the background of CPU using DMA controller. DMA is setup to transfer data from memory of which the contents to be verified to the memory mapped PSA Signature Register. When DMA transfers data to the memory mapped PSA Signature Register, a signature is generated.

A programmable 20-bit data pattern counter is used for each channel to define the number of data patterns to calculate for each sector. Signature verification can be performed automatically by MCRC controller in AUTO mode or by CPU itself in Semi-CPU or Full-CPU mode. In AUTO mode, a self-sustained CRC signature calculation can be achieved without any CPU intervention.

### 12.11.3.3.3 MCRC Modes of Operation

MCRC Controller can operate in AUTO, Semi-CPU, and Full-CPU modes.

#### 12.11.3.3.3.1 AUTO Mode

In AUTO mode, MCRC Controller in conjunction with DMA controller can perform CRC without CPU intervention. A sustained transfer of data to both the PSA Signature Register (MCRC\_PSA\_SIGREGL1-4) and a CRC (Cyclic Redundancy Check) Value Register (MCRC\_CRC\_REGL1-4) are performed in the background of CPU. When a mismatch is detected, an interrupt is generated to the CPU. A 16-bit, current sector ID register (MCRC\_CRC\_CURSEC\_REG1-4) is provided to identify which sector causes a CRC failure.

#### 12.11.3.3.3.2 Semi-CPU Mode

In Semi-CPU mode, DMA controller is also utilized to perform data patterns transfer to PSA Signature Register (MCRC\_PSA\_SIGREGL1-4). Instead of performing signature verification automatically, the CRC controller generates an compression complete interrupt to CPU after each sector is compressed. Upon responding to the interrupt the CPU performs the signature verification by reading the calculated signature stored at the PSA Sector Signature Register (MCRC\_PSA\_SECSIGREGL1-4) and compare it to a pre-determined CRC value.

#### 12.11.3.3.3.3 Full-CPU Mode

In Full-CPU mode, the CPU does the data patterns transfer and signature verification all by itself. When CPU has enough throughput, it can perform data patterns transfer by reading data from the memory system to the PSA Signature Register (MCRC\_PSA\_SIGREGL1). After certain number of data patterns are compressed, the CPU can read from the PSA Signature Register and compare the calculated signature to the pre-determined CRC signature value. In Full-CPU mode, neither interrupt nor DMA request is generated. All counters are also disabled.

### 12.11.3.3.4 PSA Signature Register

The 64-bit PSA Signature Register (MCRC\_PSA\_SIGREGL1-4 and MCRC\_PSA\_SIGREGLH1-4) is based on one of the selected CRC polynomials to produce the maximum length LFSR (Linear Feedback Shift Register).

$$\text{CRC16: } f(x) = x^{16} + x^{12} + x^5 + 1 \quad (29)$$

$$\text{CRC32: } f(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1 \quad (30)$$

$$\text{CRC64: } f(x) = x^{64} + x^4 + x^3 + x + 1 \quad (31)$$

$$\text{SAE J1850 CRC8: } f(x) = x^8 + x^4 + x^3 + x^2 + 1 \quad (32)$$

$$\text{H2F Autosar 4.0: } f(x) = x^5 + x^3 + x^2 + x + 1 \quad (33)$$

$$\text{CASTAGNOLI, iSCSI: } f(x) = x^{28} + x^{27} + x^{26} + x^{25} + x^{23} + x^{22} + x^{20} + x^{19} + x^{18} + x^{14} + x^{13} + x^{11} + x^{10} + x^9 + x^8 + x^6 + 1 \quad (34)$$

$$\text{E2E Profile 4: } f(x) = x^{31} + x^{30} + x^{29} + x^{28} + x^{26} + x^{23} + x^{21} + x^{19} + x^{18} + x^{15} + x^{14} + x^{13} + x^{12} + x^{11} + x^9 + x^8 + x^4 + x + 1 \quad (35)$$

- A. More details of the 64-bit primitive polynomial can be found at W. Stahnke, Primitive binary polynomials, Math. Comp. 27 (1973), 977–980.

There is one PSA Signature Register per CRC channel. PSA Signature Register can be both read and written. When it is written, it can either compress the data or just capture the data depending on the state of CHI\_MODE bits (where i = 1 to 4). If CHI\_MODE=0x0 (Data Capture), a seed value can be planted in the PSA Signature Register without compression. Other modes other than Data Capture will result with the data compressed by PSA Signature Register when it is written. Each channel can be planted with different seed value before compression starts. When PSA Signature Register is read, it gives the calculated signature.

MCRC Controller should be used in conjunction with the on-chip DMA controller to produce optimal system performance. The incoming data pattern to PSA Signature Register is typically initiated by the DMA controller. When DMA is properly setup, it would read data from the pre-determined memory system and write them to the memory mapped PSA Signature Register. Each time PSA Signature Register is written a signature is generated. CPU itself can also perform data transfer by reading from the memory system and perform write operation to PSA Signature Register if CPU has enough throughput to handle data patterns transfer.

After a system reset and when AUTO mode is enabled, MCRC Controller automatically generates a DMA request to request the pre-determined CRC value corresponding to the first sector of memory to be checked.

In AUTO mode, when one sector of data patterns is compressed, the signature stored at the PSA Signature Register is first copied to the PSA Sector Signature Register (MCRC\_PSA\_SECSIGREGL1-4) and PSA Signature Register is then cleared out to all zeros. An automatic signature verification is then performed by comparing the signature stored at the PSA Sector Signature Register to the CRC Value Register (MCRC\_CRC\_REGL1-4). After the comparison the MCRC Controller can generate a DMA request. Upon receiving the DMA request the DMA controller will update the CRC Value Register by transferring the next pre-determined signature value associated with the next sector of memory system. If the signature verification fails then MCRC Controller can generate a CRC fail interrupt.

In Full-CPU mode, no DMA request and interrupt are generated at all. The number of data patterns to be compressed is determined by CPU itself. Full-CPU mode is useful when DMA controller is not available to perform background data patterns transfer. Software can periodically generate a software interrupt to CPU and use CPU to accomplish data transfer and signature verification.

MCRC Controller supports double word, word, half word and byte access to the PSA Signature Register. During a non-doubleword write access, all unwritten byte lanes are padded with zeros before compression. Note that comparison between PSA Sector Signature Register and CRC Value Register is always in 64 bits because a compressed value is always expressed in 64 bits.

There is a software reset per channel for PSA Signature Register. When set, the PSA Signature Register is reset to all zeros.

PSA Signature Register is reset to zero under the following conditions:

- System reset
- PSA Software reset
- One sector of data patterns is compressed



### 12.11.3.3.5 PSA Sector Signature Register

After one sector of data is compressed, the final resulting signature calculated by PSA Signature Register is transferred to the 64-bit PSA Sector Signature Register (MCRC\_PSA\_SECSIGREGL1-4 and MCRC\_PSA\_SECSIGREGH1-4). PSA Signature Register is a read-only register. During Semi-CPU mode, the host CPU must read from the PSA Sector Signature Register instead of reading from PSA Signature Register for signature verification to avoid data coherency issue. The PSA Signature Register can be updated with new signature before the host CPU is able to retrieve it.

In Semi-CPU mode, no DMA request is generated. When one sector of data patterns is compressed, CRC controller first generates a compression complete interrupt. Responding to the interrupt, CPU will read the PSA Sector Signature Register and compare it to the known good signature or write the signature value to another memory location to build a signature file. In Semi-CPU mode, CPU must perform the signature verification in a manner to prevent any overrun condition. The overrun condition occurs when the compression complete interrupt is generated after one sector of data patterns is compressed and CPU has not read from the PSA Sector Signature Register to perform necessary signature verification before PSA Sector Signature Register is overridden with a new value. An overrun interrupt can be enabled to generate when overrun condition occurs.

### 12.11.3.3.6 CRC Value Register

Associated with each channel there is a 64-bit CRC Value Register (MCRC\_CRC\_REGL1-4 and MCRC\_CRC\_REGH1-4).

The CRC Value Register stores the pre-determined CRC value. After one sector of data patterns is compressed by PSA Signature Register, MCRC Controller can automatically compare the resulting signature stored at the PSA Sector Signature Register with the pre-determined value stored at the CRC Value Register if AUTO mode is enabled. If the signature verification fails, MCRC Controller can be enabled to generate an CRC fail interrupt. When the channel is set up for Semi-CPU mode, CRC controller first generates a compression complete interrupt to CPU. Upon servicing the interrupt, CPU will then read the PSA Sector Signature Register and then read the corresponding CRC value stored at another location and compare them. CPU must not read from the CRC Value Register during Semi-CPU or Full-CPU mode because the CRC Value Register is not updated during these two modes.

In AUTO mode, for first sector's signature, DMA request is generated when mode is programmed to AUTO. For subsequent sectors, DMA request is generated after each sector is compressed. Responding to the DMA request, DMA controller reloads the CRC Value Register for the next sector of memory system to be checked. The user software needs to configure the DMA to ensure that the DMA first writes to the MCRC\_CRC\_REGL1 followed by the MCRC\_CRC\_REGH1 register in during the AUTO Mode.

When CRC Value Register is updated with a new CRC value, an internal flag is set to indicate that CRC Value Register contains the most current value. This flag is cleared when CRC comparison is performed. Each time at the end of the final data pattern compression of a sector, MCRC Controller first checks to see if the corresponding CRC Value Register has the most current CRC value stored in it by polling the flag. If the flag is set then the CRC comparison can be performed. If the flag is not set then it means the CRC Value Register contains stale information. A CRC underrun interrupt is generated. When an underrun condition is detected, signature verification is not performed.

MCRC Controller supports double word, word, half word and byte access to the CRC Value Register. As noted before comparison between PSA Sector Signature Register and CRC Value Register during AUTO mode is carried out in 64 bits.

### 12.11.3.3.7 Raw Data Register

The raw or un-compressed data written to the PSA Signature Register is also saved in the 64-bit Raw Data Register (MCRC\_RAW\_DATAAREGL1-4 and MCRC\_RAW\_DATAAREGH1-4). This register is read only.

### 12.11.3.3.8 Example DMA Controller Setup

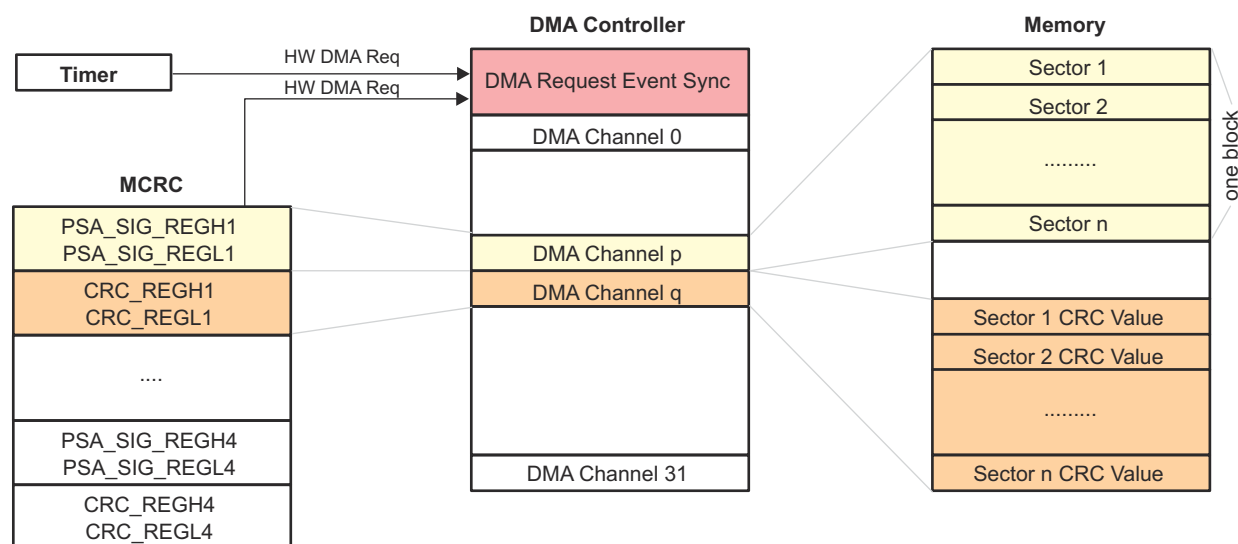
DMA controller needs to be setup properly in either AUTO or Semi-CPU mode as DMA controller is used to transfer data patterns. Hardware or a combination of hardware and software DMA triggering are supported.

### 12.11.3.3.8.1 AUTO Mode Using Hardware Timer Trigger

There are two DMA channels associated with each CRC channel when in AUTO mode. One DMA channel is setup to transfer data patterns from the source memory to the PSA Signature Register (MCRC\_PSA\_SIGREGL1-4). The second DMA channel is setup to transfer the pre-determined signature to the CRC Value Register (MCRC\_CRC\_REGL1-4). The trigger source for the first DMA channel can be either by hardware or by software. As illustrated in Figure 12-1260, a timer can be used to trigger a DMA request to initiate transfer from the source memory system to PSA Signature Register. In AUTO mode, MCRC Controller also generates DMA request after one sector of data patterns is compressed to initiate transfer of the next CRC value corresponding to the next sector of memory. Thus a new CRC value is always updated in the CRC Value Register (MCRC\_CRC\_REGL1-4) by DMA synchronized to each sector of memory.

A block of memory system is usually divided into many sectors. All sectors are the same size. The sector size is programmed in the MCRC\_CRC\_PCOUNT\_REG1-4 and the number of sectors in one block is programmed in the MCRC\_CRC\_SCOUNT\_REG1-4 of the respective channel. MCRC\_CRC\_PCOUNT\_REG1-4 multiplies MCRC\_CRC\_SCOUNT\_REG1-4 and multiplies transfer size of each data pattern should give the total block size in number of bytes.

The total size of the memory system to be examined is also programmed in the respective transfer count register inside DMA module. The DMA transfer count register is divided into two parts. They are element count and frame count. Note that a hardware DMA request can be programmed to trigger either one frame or one entire block transfer. In Figure 12-1260, a hardware DMA request from a timer is used as a trigger source to initiate DMA transfer. If all four CRC channels are active in AUTO mode then a total of four DMA requests would be generated by MCRC Controller.

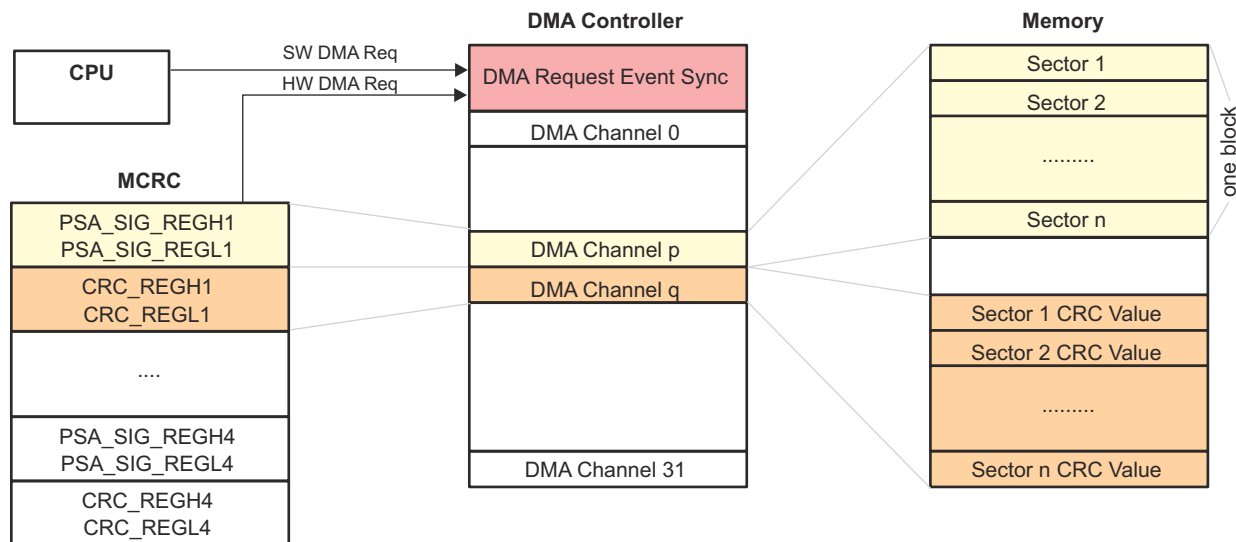


mrcr-004

**Figure 12-1260. AUTO Mode Using Hardware Timer Trigger**

### 12.11.3.3.8.2 AUTO Mode Using Software Trigger

The data patterns transfer can also be initiated by software. CPU can generate a software DMA request to activate the DMA channel to transfer data patterns from source memory system to the PSA Signature Register. To generate a software DMA request CPU needs to set the corresponding DMA channel in the DMA software trigger register. Note that just one software DMA request from CPU is enough to complete the entire data patterns transfer for all sectors. Please see [AUTO Mode With Software CPU Trigger](#) for illustration.

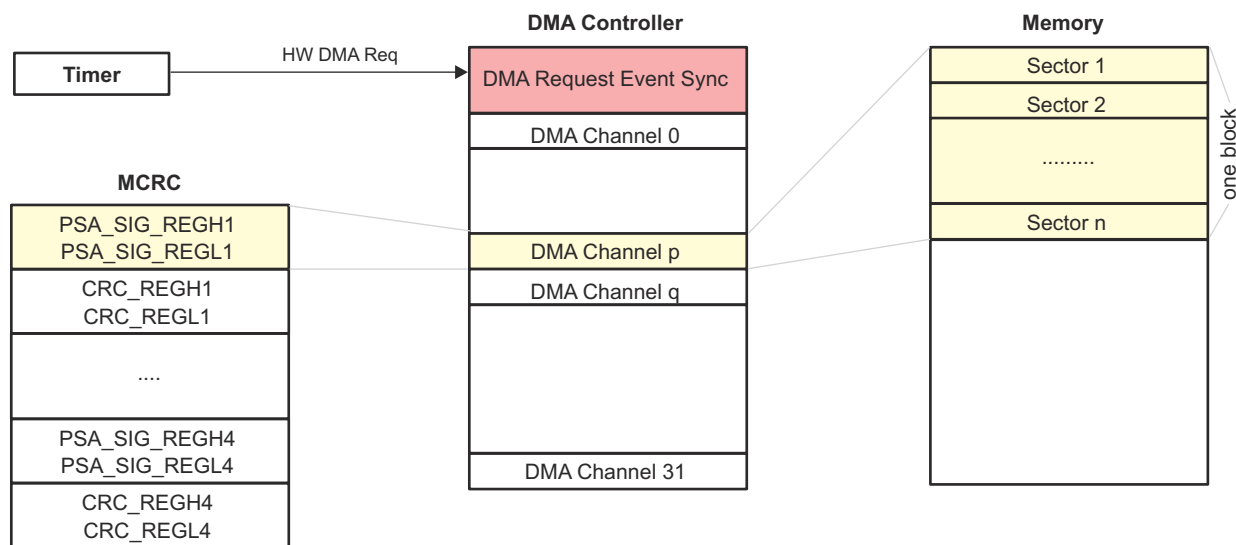


mcr-005

**Figure 12-1261. AUTO Mode With Software CPU Trigger**

#### 12.11.3.3.8.3 Semi-CPU Mode Using Hardware Timer Trigger

During Semi-CPU mode, no DMA request is generated by CRC controller. Therefore, no DMA channel is allocated to update CRC Value Register. CPU should not read from CRC Value Register in semi-CPU mode as it contains stale value. Note that no signature verification is performed at all during this mode. Similar to AUTO mode, either by hardware or by software DMA request can be used as a trigger for data patterns transfer. [Figure 12-1262](#) illustrates the DMA setup using semi-CPU mode with hardware timer trigger.



mcr-006

**Figure 12-1262. Semi-CPU Mode With Hardware Timer Trigger**

**Table 12-1665. DMA Request and Counter Logic Operation According to CRC Mode**

CRC Mode	DMA Request	Pattern Counter	Sector Counter	Timeout Counter
AUTO	Active	Active	Active	Active
Semi-CPU	Inactive	Active	Active	Active
Full CPU	Inactive	Inactive	Inactive	Inactive

### 12.11.3.3.9 Pattern Count Register

There is a 20-bit data pattern counter for every CRC channel. The data pattern counter is a down counter and can be pre-loaded with a programmable value stored in the Pattern Count Register (MCRC\_CRC\_PCOUNT\_REG1-4). When the data pattern counter reaches zero, a compression complete interrupt is generated in Semi-CPU mode and an automatic signature verification is performed in AUTO mode. In AUTO only, DMA request is generated to trigger the DMA controller to update the CRC Value Register.

#### Note

The data pattern count must be divisible by the total transfer count as programmed in DMA controller. The total transfer count is the product of element count and frame count.

### 12.11.3.3.10 Sector Count Register/Current Sector Register

Each channel contains a 16-bit sector counter. The sector count register stores the number of sectors. Sector counter is a free running counter and is incremented by one each time when one sector of data patterns is compressed. When the signature verification fails, the current value stored in the sector counter is saved into current sector register (MCRC\_CRC\_CURSEC\_REG1-4). If signature verification fails, CPU can read from the current sector register to identify the sector which causes the CRC mismatch. To aid and facilitate the CPU in determining the cause of a CRC failure it is advisable to use the following equation during CRC and DMA setup.

$$\text{CRC Pattern Count} \times \text{CRC Sector Count} = \text{DMA Element Count} \times \text{DMA Frame Count} \quad (36)$$

The current sector register is frozen from being updated until both the current sector register is read and CRC fail (CHi\_CRC\_FAIL) status bit is cleared by CPU. If CPU does not respond to the CRC failure in a timely manner before another sector produces a signature verification failure, the current sector register is not updated with the new sector number. An overrun interrupt is generate instead. If current sector register is already frozen with an erroneous sector and emulation is entered with SUSPEND signal goes to high then the register still remains frozen even it is read.

In Semi-CPU mode, the current sector register is used to indicate the sector for which the compression complete has last happened.

Current sector register is reset when the PSA software reset is enabled.

#### Note

Both data pattern count and sector count registers must be greater than or equal to one for the counters to count. After reset, pattern count and sector count registers default to zero and the associated counters are inactive.

### 12.11.3.3.11 Interrupts

CRC generate several types of interrupts per channel. Associated with each interrupt there is a interrupt enable bit (see MCRC\_CRC\_INTS). No interrupt is generated in Full-CPU mode.

- Compression complete interrupt
- CRC fail interrupt
- Overrun interrupt
- Underrun interrupt
- Timeout interrupt

**Table 12-1666. Interrupt Conditions Per CRC Mode**

CRC Mode	Compression Complete	CRC Fail	Overrun	Underrun	Timeout
AUTO	No	Yes	Yes	Yes	Yes
Semi-CPU	Yes	No	Yes	No	Yes

**Table 12-1666. Interrupt Conditions Per CRC Mode (continued)**

CRC Mode	Compression Complete	CRC Fail	Overrun	Underrun	Timeout
Full-CPU	No	No	No	No	No

#### 12.11.3.3.11.1 Compression Complete Interrupt

Compression complete interrupt is generated in Semi-CPU mode only. When the data pattern counter reaches zero, the compression complete flag is set and the interrupt is generated

#### 12.11.3.3.11.2 CRC Fail Interrupt

CRC fail interrupt is generated in AUTO mode only. When the signature verification fails, the CRC fail flag is set, and CPU must take action to address the fail condition and clear the CRC fail flag after it resolves the CRC mismatch.

#### 12.11.3.3.11.3 Overrun Interrupt

Overrun Interrupt is generated in either AUTO or Semi-CPU mode. During AUTO mode, if a CRC fail is detected then the current sector number is recorded in the current sector register (MCRC\_CRC\_CURSEC\_REG1-4). If CRC fail status bit is not cleared and current sector register is not read by the host CPU before another CRC fail is detected for another sector then an overrun interrupt is generated. During Semi-CPU mode, when the data pattern counter finishes counting, it generates a compression complete interrupt. At the same time the signature is copied into the PSA Sector Signature Register (MCRC\_PSA\_SECSIGREGL1-4). If the host CPU does not read the signature from PSA Sector Signature Register before it is updated again with a new signature value then an overrun interrupt is generated.

#### 12.11.3.3.11.4 Underrun Interrupt

Underrun interrupt only occurs in AUTO mode. The interrupt is generated when the CRC Value Register is not updated with the corresponding signature when the data pattern counter finishes counting. During AUTO mode, MCRC Controller generates DMA request to update CRC Value Register in synchronization to the corresponding sector of the memory. Signature verification is also performed if underrun condition is detected. And CRC fail interrupt is generated at the same time as the underrun interrupt.

#### 12.11.3.3.11.5 Timeout Interrupt

To ensure that the memory system is examined within a pre-defined time frame and no loss of incoming data there is a 24-bit timeout counter per CRC channel. The timeout counter can be pre-loaded with two different pre-load values, watchdog timeout pre-load value (MCRC\_CRC\_WDTPOLD1-4) and block complete timeout pre-load value (MCRC\_CRC\_BCTOPLD1-4). The timeout counter is clocked by a prescaler clock which is permanently running at division 64 of FICLK clock.

Watchdog timeout pre-load register (MCRC\_CRC\_WDTPOLD1-4) is used to check if DMA does supply a block of data responding to a request in a given time frame. Block complete timeout pre-load register (MCRC\_CRC\_BCTOPLD1-4) is used to check if one complete block of data patterns are compressed within a specific time frame. The timeout counter is first pre-loaded with MCRC\_CRC\_WDTPOLD1-4 after either AUTO or Semi-CPU mode is selected and starts to down count. If the timeout counter expires before DMA transfers any data pattern to PSA Signature Register then a timeout interrupt is generated. An incoming data pattern before the timeout counter expires will automatically pre-load the timeout counter with MCRC\_CRC\_BCTOPLD1-4 the block complete timeout pre-load value.

Block complete timeout pre-load value is used to check if one block of data patterns are compressed within a given time limit. If the timeout counter pre-loaded with MCRC\_CRC\_BCTOPLD1-4 value expires before one block of data patterns are compressed a timeout interrupt is generated. When one block (pattern count × sector count) of data patterns are compressed before the counter has expired, the counter is pre-loaded with MCRC\_CRC\_WDTPOLD1-4 value again. If the timeout counter is pre-loaded with zero then the counter is disable and no timeout interrupt is generated.

### 12.11.3.3.11.6 Interrupt Offset Register

MCRC Controller only generates one interrupt request to interrupt manager. An interrupt offset register (MCRC\_CRC\_INT\_OFFSET\_REG) is provided to indicate the source of the pending interrupt with highest priority. [Table 12-1667](#) shows the offset interrupt vector address of each interrupt in an ascending order of priority.

**Table 12-1667. Interrupt Offset Mapping**

Interrupt Condition	Offset Value
Phantom	0x0
Ch1 CRC Fail	0x1
Ch2 CRC Fail	0x2
Ch3 CRC Fail	0x3
Ch4 CRC Fail	0x4
reserved	0x5-0x8
Ch1 Compression Complete	0x9
Ch2 Compression Complete	0xA
Ch3 Compression Complete	0xB
Ch4 Compression Complete	0xC
reserved	0xD-0x10
Ch1 Overrun	0x11
Ch2 Overrun	0x12
Ch3 Overrun	0x13
Ch4 Overrun	0x14
reserved	0x15-0x18
Ch1 Underrun	0x19
Ch2 Underrun	0x1A
Ch3 Underrun	0x1B
Ch4 Underrun	0x1C
reserved	0x1D-0x20
Ch1 Timeout	0x21
Ch2 Timeout	0x22
Ch3 Timeout	0x23
Ch4 Timeout	0x24

### 12.11.3.3.11.7 Error Handling

When an interrupt is generated, host CPU must take appropriate actions to identify the source of error and restart the respective channel in DMA and MCRC module. To restart a CRC channel user must perform the following steps in the ISR:

1. Write to software reset bit in MCRC\_CRC\_CTRL0 register to reset the respective PSA Signature Register
2. Reset the CHi\_MODE bits to 0 in MCRC\_CRC\_CTRL2 register as Data capture mode
3. Set the CHi\_MODE bits in MCRC\_CRC\_CTRL2 register to desired new mode again
4. Release software reset

The host CPU must use byte write to restart each individual channel.

### 12.11.3.3.12 Power Down Mode

MCRC module can be put into power down mode when the power down control bit MCRC\_CRC\_CTRL1[0] PWDN is set. The module wakes up when the PWDN bit is cleared.

#### **12.11.3.3.13 Emulation**

A read access from a register in functional mode can sometimes trigger a certain internal event to follow. For example, reading interrupt offset register triggers an event to clear the corresponding interrupt status flag. During emulation when SUSPEND signal is high, a read access from any register should only return the register contents to the bus and should not trigger or mask any event as it would have in functional mode. This is to prevent debugger from reading the interrupt offset register, that is, during refreshing screen and cause the corresponding interrupt status flag to get cleared. Timeout counters are stopped to generate timeout interrupts in emulation mode. No VBUSM bus error will be generated if reading from the unimplemented locations. .

CEMUDBG is the VBUSM suspend signal which need not explicitly indicate that whether CPU is in suspend mode or not. In data trace mode, a separate suspend signal CPU\_EMUSP, is used to indicate MCRC controller that the CPU is in suspend mode or not.



### 12.11.3.4 MCRC Programming Examples

#### 12.11.3.4.1 Example: Auto Mode Using Time Based Event Triggering

A large memory area with 2 Mbyte (256 k × 32-bit [doubleword]) is to be checked in the background of CPU. CRC is to be performed every 1 Kbyte (128 doubleword). Therefore there will be 2048 pre-recorded CRC values. For illustration purpose, we map MCRC\_CRC\_REGL1 register to DMA channel 1 and MCRC\_PSA\_SIGREGL1 register to DMA channel 2. Let's assume all DMA transfers are carried out in 64-bit transfer size.

##### 12.11.3.4.1.1 DMA Setup

- Setup DMA channel 1 with the starting address from which the pre-determined CRC values are stored. Setup the destination address to the MCRC\_CRC\_REGL1. Put the source address at post increment addressing mode and put the destination address at constant addressing mode. Use hardware DMA request for channel 1 to trigger a frame transfer.
- Setup DMA channel 2 with the source address from which the contents of memory to be verified. Setup the destination address to the MCRC\_PSA\_SIGREGL1. Program the element transfer count to 128 and the frame transfer count to 2048. Program the read and write element size to 64 bits. Put the source address at post increment addressing mode and put the destination address at constant address mode. Use hardware DMA request for channel 2 to trigger an entire block transfer.

##### 12.11.3.4.1.2 Timer Setup

The timer can be any general purpose timer which is capable of generating a time based DMA request.

- Setup Timer to generate DMA request associated with DMA channel 2. For example, software can setup the timer to generate a DMA request every 10 ms.

##### 12.11.3.4.1.3 CRC Setup

- Program the pattern count MCRC\_CRC\_PCOUNT\_REG1 to 128
- Program the sector count MCRC\_CRC\_SCOUNT\_REG1 to 2048
- For example, we want the entire 2 Mbytes to be compressed within 5 ms. We can program the block complete timeout pre-load (MCRC\_CRC\_BCTOPLD1-4) value to 15625 (5 ms / (1 FICLK period × 64)) if CRC is operating at 200 MHz.
- Enable AUTO mode and all interrupts.

After AUTO mode is selected MCRC Controller automatically generates a DMA request on channel 1. Around the same time the timer module also generates a DMA request on DMA channel 2. When the first incoming data pattern arrives at the MCRC\_PSA\_SIGREGL1/H1, the MCRC Controller will compress it. After some time, the DMA controller would update the MCRC\_CRC\_REGL1/H1 with a pre-determined value matching the calculated signature for the first sector of 128 64-bit data patterns. After one sector of data patterns are compressed, the MCRC Controller generate a CRC fail interrupt if signature stored at the MCRC\_PSA\_SECSIGREGL1/H1 does not match the MCRC\_CRC\_REGL1/H1 Register. MCRC Controller generates a DMA request on DMA channel 1 when one sector of data patterns are compressed. This routine will continue until the entire 2 Mbytes are consumed. If the timeout counter reached zero before the entire 2 Mbytes are compressed, a timeout interrupt is generated. After 2Mbytes are transferred, the DMA can generate an interrupt to CPU. The entire operation will continue again when DMA responds to the DMA request from both the timer and MCRC Controller. The CRC is performed totally without any CPU intervention.

#### 12.11.3.4.2 Example: Auto Mode Without Using Time Based Triggering

A small but highly secured memory area with 1 Kbytes is to be checked in the background of CPU. CRC is to be performed every 1 Kbytes. Therefore, there is only one pre-recorded CRC value. For illustration purpose, we map channel 1 MCRC\_CRC\_REGL1/H1 to DMA channel 1 and channel 1 MCRC\_PSA\_SIGREGL1/H1 to DMA channel 2. Assume all transfers carried out by DMA are in 64-bit transfer size.

##### 12.11.3.4.2.1 DMA Setup

- Setup DMA channel 1 with the source address from which the pre-determined CRC value is stored. Setup the destination address to the MCRC\_CRC\_REGL1 Register. Put the source address at constant addressing



mode and put the destination address at constant addressing mode. Use hardware DMA request for channel 1.

- Setup DMA channel 2 with the source address from which the memory area to be verified. Setup the destination address to the MCRC\_PSA\_SIGREGL1/H1. Program the element transfer count to 128 and the frame transfer count to 1. Put the source address at post increment addressing mode and put the destination address at constant address mode. Generate a software DMA request on channel 2 after CRC has completed its setup. Enable autoinitiation for DMA channel 2.

#### 12.11.3.4.2.2 CRC Setup

- Program the MCRC\_CRC\_PCOUNT\_REG1 to 128
- Program the MCRC\_CRC\_SCOUNT\_REG1 to 1
- Leaving the timeout count MCRC\_CRC\_BCTOPLD1 register with the reset value of zero means no timeout interrupt is generated
- Enable AUTO mode and all interrupts

After AUTO mode is selected the MCRC Controller automatically generates a DMA request on channel 1. At the same time the CPU generates a software DMA request on DMA channel 2. When the first incoming data pattern arrives at the MCRC\_PSA\_SIGREGL1/H1, the MCRC Controller will compress it. After some time, the DMA controller would update the MCRC\_CRC\_REGL1/H1 with a pre-determined value matching the calculated signature for the first sector of 128 64-bit data patterns. After one sector of data patterns are compressed, the MCRC Controller generate a CRC fail interrupt if signature stored at the MCRC\_PSA\_SECSIGREGL1/H1 does not match the MCRC\_CRC\_REGL1/H1. MCRC Controller generate a DMA request on DMA channel 1 again after one sector is compressed. After 1 Kbytes are transferred, the DMA can generate an interrupt to CPU. Responding to the DMA interrupt CPU can restart the CRC routine by generating a software DMA request onto channel 2 again.

#### 12.11.3.4.3 Example: Semi-CPU Mode

If DMA controller is available in a system the CRC module can also operate in semi-CPU mode. This means that CPU can still make use of the DMA to perform data patterns transfer to CRC controller in the background. The difference between semi-CPU mode and AUTO mode is that CRC controller does not automatically perform the signature verification. CRC controllers generates a compression complete interrupt to CPU when the one sector of data patterns are compressed. CPU needs to perform the signature verification itself.

A memory area with 2 Mbytes is to be verified with the help of the CPU. CRC operation is to be performed every 1 Kbyte. Since there are 2 Mbytes (256 K doublewords) of memory to be check and we want to perform a CRC every 1 Kbytes (128 doublewords) and therefore there must be 2048 pre-recorded CRC values. In Semi-CPU mode, the MCRC\_CRC\_REGL1/H1 is not updated and contains indeterminate data.

##### 12.11.3.4.3.1 DMA Setup

- Setup DMA channel 1 with the source address from which the memory area to be verified are mapped. Setup the destination address to the MCRC\_PSA\_SIGREGL1/H1. Put the starting address at post increment addressing mode and put the destination address at constant address mode. Use hardware DMA request to trigger an entire block transfer for channel 1. Disable autoinitiation for DMA channel 1.

##### 12.11.3.4.3.2 Timer Setup

The timer can be any general purpose timer which is capable of generating a time-based DMA request.

- Setup Timer to generate DMA request associated with DMA channel 1. For example, software can setup the timer to generate a DMA request every 10 ms.

##### 12.11.3.4.3.3 CRC Setup

- Program the MCRC\_CRC\_PCOUNT\_REG1 to 128
- Program the MCRC\_CRC\_SCOUNT\_REG1 to 2048
- For example, we want the entire 2 Mbytes to be compressed within 5 ms. We can program the block complete timeout pre-load value MCRC\_CRC\_BCTOPLD1 to 15625 (5 ms / (1 FICLK period × 64)) if CRC is operating at 200 MHz
- Enable AUTO mode and all interrupts.

The timer module first generates a DMA request on DMA channel 1 when it is enabled. When the first incoming data pattern arrives at the MCRC\_PSA\_SIGREGL1/H1, the CRC controller will compress it. After one sector of data patterns are compressed, the CRC controller generate a compression complete interrupt. Upon responding to the interrupt the CPU would read from the MCRC\_PSA\_SECSIGREGL1/H1. It is up to the CPU on how to deal with the PSA value just read. It can compare it to a known signature value or it can write it to another memory location to build a signature file or even transfer the signature out of the device via SCI or SPI. This routine will continue until the entire 2 Mbytes are consumed. The latency of the interrupt response from CPU can cause overrun condition. If CPU does not read from MCRC\_PSA\_SECSIGREGL1 before the PSA value is overridden with the signature of the next sector of memory, an overrun interrupt will be generated by CRC controller.

#### **12.11.3.4.4 Example: Full-CPU Mode**

In a system without the availability of DMA controller, the CRC routine can be operated by CPU provided the CPU has enough throughput. CPU needs to read from the memory area from which CRC is to be performed.

A memory area with 2 Mbytes is to be checked with the help of the CPU. CRC operation is to be performed every 1 Kbyte. In CPU mode, the MCRC\_CRC\_REGL1/H1 is not updated and contains indeterminate data.

##### **12.11.3.4.4.1 CRC Setup**

- All control registers can be left in their reset state. Only enable Full-CPU mode.

CPU itself reads from the memory and write the data to the MCRC\_PSA\_SIGREGL1/MCRC\_PSA\_SIGREGH1 inside MCRC Controller. When the first incoming data pattern arrives at the MCRC\_PSA\_SIGREGL1/MCRC\_PSA\_SIGREGH1, the MCRC Controller will compress it. After n data patterns are compressed, CPU can read from the MCRC\_PSA\_SIGREGL1/MCRC\_PSA\_SIGREGH1. It is up to the CPU on how to deal with the PSA signature value just read. It can compare it to a known signature value stored at another memory location.

## 12.11.4 ECC Aggregator

This section describes the common ECC aggregator functionality.

### 12.11.4.1 ECC Aggregator Overview

To increase functional and system reliability the memories (for example, FIFOs, queues, SRAMs and others) in many device modules and subsystems are protected by error correcting code (ECC). This is accomplished through an ECC aggregator and ECC wrapper. The ECC aggregator is connected to these memories (hereinafter ECC RAMs) and involved in the ECC process. Each memory is surrounded by an ECC wrapper which performs the ECC detection and correction. The wrapper communicates via serial interface with the aggregator which has memory mapped configuration interface.

The ECC aggregator is also connected to interconnect ECC components that protect the command, address and data buses of the system interconnect. ECC is calculated for the data bus and parity and redundancy for the command and address buses. Each interconnect ECC component has the same serial interface for communication with the aggregator as the ECC wrapper. An ECC aggregator may be connected to both endpoints - the ECC wrapper and interconnect ECC component.

The ECC aggregator, ECC wrapper and interconnect ECC component are considered as single entity and are hereinafter referred to as ECC aggregator unless otherwise explicitly specified.

[Table 12-1668](#) lists the device modules and subsystems which have ECC aggregator.

**Table 12-1668. Device Modules and Subsystems with ECC Aggregator**

Module Instance	Domain		
	WKUP	MCU	MAIN
WKUP_CBASS0	✓	-	-
WKUP_VTM0	✓	-	-
MCU_ADC0	-	✓	-
MCU_ADC1	-	✓	-
MCU_CBASS0	-	✓	-
MCU_CPSW0	-	✓	-
MCU_FSS0_HPBO	-	✓	-
MCU_FSS0_OSPI0	-	✓	-
MCU_FSS0_OSPI1	-	✓	-
MCU_I3C0	-	✓	-
MCU_I3C1	-	✓	-
MCU_MCAN0	-	✓	-
MCU_MCAN1	-	✓	-
MCU_MSRAM_1MB0	-	✓	-
MCU_NAVSS0	-	✓	-
A72SS0	-	-	✓
C71SS0	-	-	✓
CBASS0	-	-	✓
COMPUTE_CLUSTER0	-	-	✓
CPSW0	-	-	✓
CSI_RX_IF0	-	-	✓
CSI_RX_IF1	-	-	✓
CSI_TX_IF0	-	-	✓
DDRSS0	-	-	✓
DMPAC0	-	-	✓
DSS0	-	-	✓

**Table 12-1668. Device Modules and Subsystems with ECC Aggregator (continued)**

Module Instance	Domain		
	WKUP	MCU	MAIN
GIC0	-	-	✓
I3C0	-	-	✓
MCAN0 to MCAN13	-	-	✓
MLBSS0	-	-	✓
MMCSD0	-	-	✓
MMCSD1	-	-	✓
MMCSD2	-	-	✓
MSRAM16KX256	-	-	✓
NAVSS0	-	-	✓
PCIE0	-	-	✓
PCIE1	-	-	✓
PCIE2	-	-	✓
PCIE3	-	-	✓
PRU_ICSSG0	-	-	✓
PRU_ICSSG1	-	-	✓
PSRAM2KECC0	-	-	✓
PSRAMECC0	-	-	✓
UFS0	-	-	✓
USB3SS0	-	-	✓
USB3SS1	-	-	✓
VPAC0	-	-	✓

#### 12.11.4.1.1 ECC Aggregator Features

The ECC aggregator has the following features:

- Reduces memory software errors via single error correction (SEC) and double error detection (DED)
- Provides a mechanism to control and monitor the ECC protected memories in a module or subsystem
- SEC and DED over the system interconnect data bus and parity and redundancy for the system interconnect command and address buses
- Generates an interrupt for correctable error
- Generates an interrupt for non-correctable error
- Supports inject only mode for diagnostic purposes
- Supports software readable status for single and double-bit ECC errors and associated information such as row address where error has occurred and data bits that have been flipped
- Supports up to 256 ECC endpoints. An ECC endpoint can be either ECC RAM or interconnect ECC component.
- Detects single bit error via parity checking on:
  - Memory mapped configuration interface FIFO
  - Serial interface FIFO
  - Serial interface transaction
- Single bit error detection via parity checking results in a non-correctable error interrupt
- Supports timeout mechanism on transactions over the ECC serial interface. Timeout occurrence results in a non-correctable error interrupt.
- Certain control bits have redundancy and if a bit flips an interrupt is generated

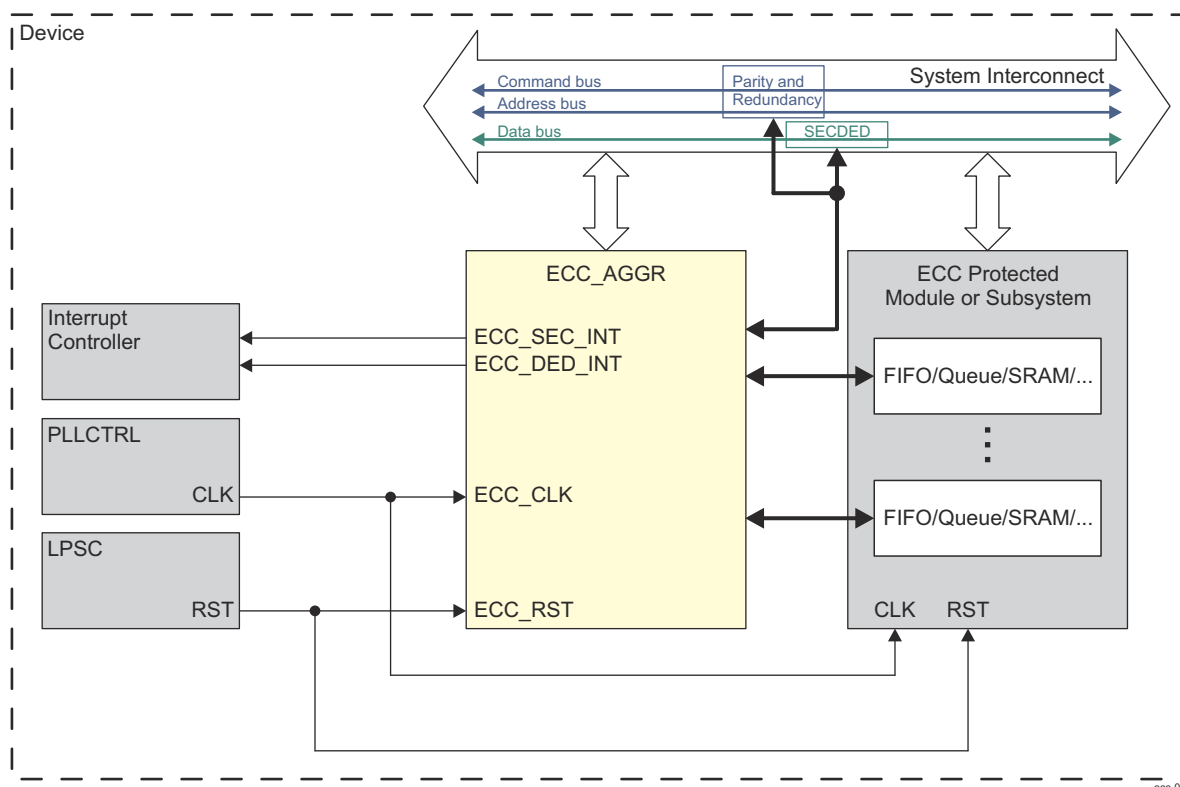
### 12.11.4.2 ECC Aggregator Integration

This section describes an ECC aggregator integration in the device, including information about clocks, resets, and hardware requests.

#### Note

For a list of the device modules and subsystems which have ECC aggregator, see .

Table 12-1669 shows the integration of an ECC aggregator module.



**Figure 12-1263. ECC Aggregator Integration**

Table 12-1669 through Table 12-1671 summarize the integration of an ECC aggregator module.

**Table 12-1669. ECC Aggregator Integration Attributes**

Module Instance	Attributes			
	Power Sleep Controller	Power Domain	Module Domain	Interconnect
ECC_AGGR	Same PSC as the corresponding module or subsystem	Same PD as the corresponding module or subsystem	Same LPSC as the corresponding module or subsystem	Same CBASS as the corresponding module or subsystem

**Table 12-1670. ECC Aggregator Clocks and Resets**

Clocks				
Module Instance	Module Clock Input	Source Clock Signal	Source	Description
ECC_AGGR	ECC_CLK	Same as the corresponding module or subsystem <sup>(1)</sup>	Same as the corresponding module or subsystem <sup>(1)</sup>	ECC aggregator clock
Resets				
Module Instance	Module Reset Input	Source Reset Signal	Source	Description
ECC_AGGR	ECC_RST	Same as the corresponding module or subsystem <sup>(1)</sup>	Same as the corresponding module or subsystem <sup>(1)</sup>	ECC aggregator reset

(1) For information about clock and reset sources, see the respective *Clocks and Resets* table of the module or subsystem with ECC aggregator.

**Table 12-1671. ECC Aggregator Hardware Requests**

Interrupt Requests					
Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
ECC_AGGR	ECC_SEC_INT	See <sup>(1)</sup>	See <sup>(1)</sup>	Interrupt for correctable error (SEC)	Level
	ECC_DED_INT	See <sup>(1)</sup>	See <sup>(1)</sup>	Interrupt for non-correctable error (DED, parity, redundancy, timeout)	Level
DMA Events					
Module Instance	Module DMA Event	Destination DMA Event Input	Destination	Description	Type
ECC_AGGR	-	-	-	-	-

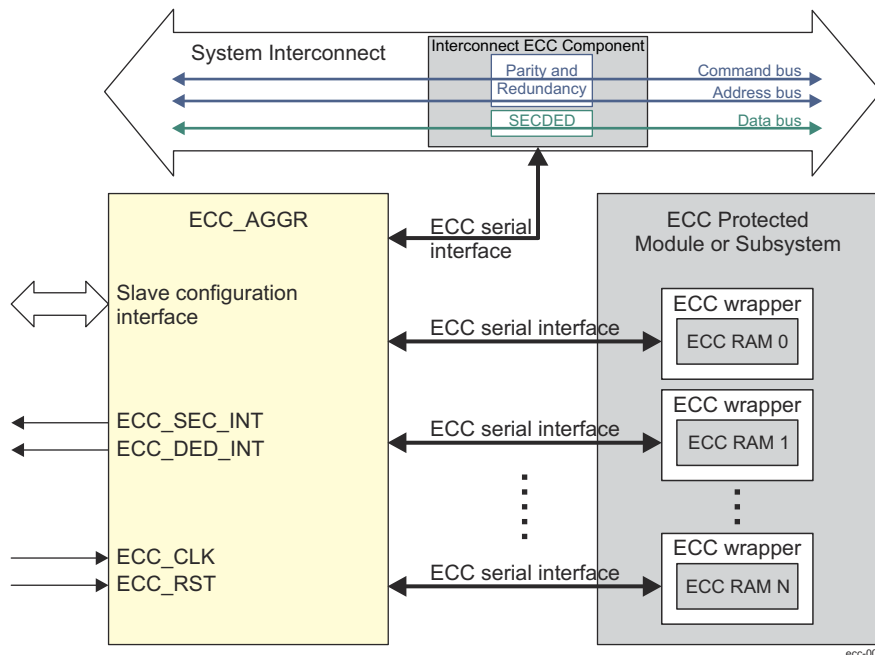
(1) See the respective *Hardware Requests* table of the module or subsystem with ECC aggregator.

### 12.11.4.3 ECC Aggregator Functional Description

This section describes the architecture and functional details of the ECC aggregator.

#### 12.11.4.3.1 ECC Aggregator Block Diagram

Figure 12-1264 shows the ECC aggregator block diagram.



**Figure 12-1264. ECC Aggregator Block Diagram**

The ECC aggregator is connected to one or more ECC endpoints each of which has assigned a unique ID used when the endpoint is accessed for status information or configuration. The ECC aggregator provides software access to all ECC related registers through its memory mapped slave configuration interface while the serial interface is used to communicate with the ECC endpoints. Upon detection of single or double-bit error the corresponding interrupt line is asserted.

#### 12.11.4.3.2 ECC Aggregator Register Groups

The ECC aggregator has ECC control, status and interrupt registers for each ECC endpoint in a module or subsystem. These registers are memory mapped and occupy 1 KB address space although part of it may contain reserved locations. The registers are split in the following types:

- **Global registers.** They are common to all ECC endpoints associated with the ECC aggregator and include the ECC\_VECTOR and ECC\_REV registers. Each ECC endpoint has assigned a unique ID. When this ID is written to the ECC\_VECTOR[10-0] ECC\_VECTOR field the corresponding endpoint is selected either for control or for status reading.
- **ECC control and status registers.** These registers are specific to each ECC endpoint and reside in the range from address offset 0x10 to 0x28, if the endpoint is ECC RAM or from 0x10 to 0x24, if the endpoint is interconnect ECC component. They are memory mapped but are accessed through the ECC serial interface. They are also selected by the ECC endpoint ID written to the ECC\_VECTOR[10-0] ECC\_VECTOR field. Because of latency on the serial interface the ECC control and status registers are read by performing special sequence as described in [Section 12.11.4.3.3](#). These registers have also different functionality for both types of endpoints - ECC RAM and interconnect ECC component.
- **Interrupt registers.** They include interrupt status, interrupt enable, interrupt disable, and EOI registers. For more information, see *Interrupts*.



#### 12.11.4.3.3 Read Access to the ECC Control and Status Registers

Read accesses to the ECC control and status registers for each ECC endpoint represent read operations over the ECC serial interface and are triggered by performing the following sequence:

- Software writes the following in the ECC\_VECTOR register:
  - The ECC endpoint ID in the ECC\_VECTOR[10-0] ECC\_VECTOR field to select particular ECC endpoint.
  - The register read address in the ECC\_VECTOR[23-16] RD\_SVBUS\_ADDRESS field to select which register has to be read through the ECC serial interface.
  - A value of 0x1 in the ECC\_VECTOR[15] RD\_SVBUS bit to trigger read operation through the ECC serial interface.
- Software polls the ECC\_VECTOR[24] RD\_SVBUS\_DONE bit to check if it is 0x1. This indicates that the read operation on the ECC serial interface has completed.
- Software reads the data from the register previously selected by the ECC\_VECTOR[23-16] RD\_SVBUS\_ADDRESS field.

The following is an example for serial read operation:

- Write 0x0010 8005 to the ECC\_VECTOR register. This sends read request to the ECC\_WRAP\_REV or ECC\_CBASS\_REV register (address = 0x10) associated with ECC endpoint with ID = 5.
- Poll the ECC\_VECTOR[24] RD\_SVBUS\_DONE bit until value of 0x1 is read.
- Read the ECC\_WRAP\_REV or ECC\_CBASS\_REV register to get its value.

#### 12.11.4.3.4 Serial Write Operation

Write operations over the ECC serial interface are performed as follows:

- Software specifies the ECC endpoint ID in the ECC\_VECTOR[10-0] ECC\_VECTOR field. The ECC\_VECTOR[23-16] RD\_SVBUS\_ADDRESS field is a don't care but the ECC\_VECTOR[15] RD\_SVBUS bit must be set to 0x0.
- Software performs regular write operation to the desired address. If the ECC endpoint ID has already been specified, step 1 can be skipped. Unlike serial read operations it is not necessary to always specify the endpoint ID before performing serial write operation.

The following is an example for serial write operation:

- Write 0x0000 0008 to the ECC\_VECTOR register.
- Write 0x0000 000F to the ECC\_CTRL register. This sends write request with data 0x0000 000F to the ECC\_CTRL register associated with ECC RAM with ID = 8.

#### 12.11.4.3.5 Interrupts

The ECC aggregator generates the following interrupts:

- Correctable interrupt (ECC\_SEC\_INT) where hardware can correct the error but notifies the system in case of SEC.
- Non-correctable interrupt (ECC\_DED\_INT) where hardware cannot correct the error in cases of DED, parity check, redundancy check or timeout occurrence.

The following is the sequence for servicing interrupts:

- Software enables the interrupts for an ECC endpoint by writing 0x1 to the corresponding bit of the following interrupt enable registers:
  - ECC\_SEC\_ENABLE\_SET\_REG0 through ECC\_SEC\_ENABLE\_SET\_REG7 for the correctable interrupt
  - ECC\_DED\_ENABLE\_SET\_REG0 through ECC\_DED\_ENABLE\_SET\_REG7 for the non-correctable interrupt
- On receiving an interrupt, software checks which ECC endpoint has caused the error by reading the following interrupt status registers:
  - ECC\_SEC\_STATUS\_REG0 through ECC\_SEC\_STATUS\_REG7 for the correctable interrupt
  - ECC\_DED\_STATUS\_REG0 through ECC\_DED\_STATUS\_REG7 for the non-correctable interrupt
- Software performs serial read operations as described in [Section 12.11.4.3.3](#) to read the following status registers that contain details about the error:

- If the endpoint is ECC RAM:
  - ECC\_ERR\_STAT1
  - ECC\_ERR\_STAT2
  - ECC\_ERR\_STAT3
- If the endpoint is interconnect ECC component:
  - ECC\_CBASS\_ERR\_STAT1
  - ECC\_CBASS\_ERR\_STAT2
- After the interrupt has been serviced, depending on the error type, software should clear the corresponding status bits in the ECC\_ERR\_STAT1 and ECC\_ERR\_STAT3 registers or in the ECC\_CBASS\_ERR\_STAT1 register. Software has to poll these registers to guarantee that status bits are cleared as there is no other indication for write completion over the ECC serial interface.  
The value of the \*\_PEND\_CLR fields in the ECC\_CBASS\_ERR\_STAT1 register must be read and then written back to decrement the count of each field back to 0x0. A further error capture into the ECC\_CBASS\_ERR\_STAT1 register does not occur unless all its fields are 0x0. The decrement value should not be larger than the read value. If a field in the ECC\_CBASS\_ERR\_STAT1 register should not be modified, write a value of 0x0 to that field.
- Software writes 0x1 to the corresponding end of interrupt register to clear the interrupt:
  - ECC\_SEC\_EOI\_REG for the correctable interrupt
  - ECC\_DED\_EOI\_REG for the non-correctable interrupt

#### 12.11.4.3.6 Inject Only Mode

There are modules that already perform the ECC generation and checking as part of their data path. In this case, the ECC wrapper may be configured in inject only mode, if needed. In this mode the ECC wrapper does not perform ECC detection and correction. The inject only mode allows users to inject single or double-bit errors so that the module logic can be tested for diagnostic purposes.

---

#### Note

There is no software control to enable inject only mode. It is configured via tie-off value. Inject only and ECC modes are mutually exclusive.

---

The interconnect ECC component also supports error injection mode. There is error injection logic for testing of the error checking logic (checkers). The injection logic can be configured to inject either single or double bit error and what data pattern to be used for injection (ECC\_CBASS\_CTRL[11-8] ECC\_PATTERN). The ECC\_CBASS\_ERR\_CTRL1 and ECC\_CBASS\_ERR\_CTRL2 registers should be written first to setup the injection. Then, either the ECC\_CBASS\_CTRL[3] FORCE\_SE or the ECC\_CBASS\_CTRL[4] FORCE\_DE bit must be set to 0x1 to start the injection. Both bits must not be set at the same time. If the injection should continue in incrementing mode, then the ECC\_CBASS\_CTRL[5] FORCE\_N\_BIT bit should be set to 0x1. Once the FORCE\_N\_BIT is set, then each successive injection can simply write the ECC\_CBASS\_CTRL register to set the FORCE\_SE or FORCE\_DE again. Reading 0x0 from either the FORCE\_SE or the FORCE\_DE bit indicates that the injection has completed, as these bits automatically clear when the checker indicates that it has performed the injection. The time for an injection to complete is not guaranteed, so some delay is needed between successive injections.



This chapter describes the on-chip debug support.

<b>13.1 Introduction to SoC Debug Framework.....</b>	<b>3530</b>
--	-------------

## 13.1 Introduction to SoC Debug Framework

The SoC debug capabilities are centered around the Debug Subsystem (DebugSS). The DebugSS works in conjunction with the debug capabilities integrated in the various processing units (such as CPUs and hardware accelerators) to provide a comprehensive hardware platform for a rich debug and development experience.

The SoC debug framework provides support for the following features:

- Top-level debug support
  - IEEE 1149.1 (JTAG + boundary scan) and IEEE 1149.6 (boundary scan extensions)
  - Centralized debug access gateway (DebugSS) that provides support for debug access and trace in multiple CPU core subsystems
    - JTAG can access all debug and system resources
    - Other system masters can also access all debug resources (self-hosted debug)
  - Direct debug master to system memory that bypasses any CPU datapath used for debug
  - Global execution and system level debug triggering (simultaneous run/halt)
    - ARM CoreSight cross trigger architecture
      - Channel Interface (CI)
      - Cross Trigger Interface (CTI), Cross Trigger Matrix (CTM)
    - TI legacy cross trigger architecture (for C66x only)
    - ARM/TI trigger merge at the DebugSS level
  - Trace architecture that includes various trace sources, sinks and supporting modules
    - Processor trace (A72, R5F, C66x, C71x)
    - Discrete event trace
      - Counter, Timer, and System Event Trace (CTSET) module
    - CBA4 compliant bus probing
      - CPTracer2 (CPT2) aggregators and probes
      - Three levels of bus probing (SoC, MCU domain, MSMC)
    - Software message trace
      - ARM CoreSight STM-500 module
    - Universal debug time distribution for correlating various trace sources
      - Wide Timestamp Interface (WTI)
    - Trace distribution via ARM Advanced Trace Bus (ATB) network
    - Trace storing into dedicated on-chip Trace Buffer Router (TBR) modules
    - Trace streaming off-chip
      - Export via ARM CoreSight Trace Port Interface Unit (TPIU)
      - Export via high-speed functional interfaces (Ethernet, PCIe)
        - Requires software assistance
  - Advanced debug power, clock and reset management
  - Ownership aware peripheral debug events (suspend mapping)
  - Authenticating a debug connection on device configured for high-security (HS) operations
- Compute Cluster debug support
  - A72SS
    - Invasive debug (halt mode, monitor mode)
    - Non-invasive debug (processor trace, performance monitoring)
    - CoreSight Embedded Trace Macrocell (ETM)
    - Cross triggering via CoreSight CTI and CTM
  - C71SS
    - Core debug
    - Advanced Event Triggering (AET)
    - Real-time trace
    - Physical address watchpoints

- Event analysis
- MSMC
  - Bus probing
  - Discrete event trace
  - Cross triggering via CoreSight CTI and CTM
- R5FSS debug support (applies to all three R5FSS instances)
  - Invasive debug (halt mode, monitor mode)
  - Non-invasive debug (processor trace, performance monitoring)
  - CoreSight ETM-R5
  - Cross triggering via CoreSight CTI and CTM
- C66SS debug support (applies to both C66SS instances)
  - Invasive debug (halt mode, monitor mode, real-time debug)
  - Non-invasive debug (processor trace)
  - Cross triggering
    - AET
    - TI legacy trigger channels (EMU0, EMU1)
  - Dedicated AMBA trace bus DSP Trace Formatter (ADTF) per C66x core
    - Provides the required C66x trace formatting for on-chip store or off-chip export
    - Resides outside the C66x CorePac
- DMPAC debug support
  - HWA basic debug (system visibility, run control)
  - Cross triggering
  - Discrete event trace
- VPAC debug support
  - HWA basic debug (system visibility, run control)
  - Cross triggering
  - Discrete event trace

---

**Note**

For DMSC debug support, refer to the DMSC Addendum.

---

## Revision History



### Changes from March 8, 2022 to December 31, 2024 (from Revision C (March 2022) to Revision D (December 2024))

	Page
• Updated Note in Introduction section.....	79
• Updated Device JTAG ID table.....	117
• Updated Quality of Service (QoS) section.....	221
• Updated ISC Config Registers per Region diagram.....	228
• Added xSPI Boot Configuration Fields table.....	285
• Added eMMC Flash section.....	303
• Updated X.509 Certificate information from 'Optional.'.....	315
• Updated PRU_ICSSG Control Registers table.....	356
• Power: Added Power Management Unit section with overview of the contents of the Power Management Unit.....	361
• Power: Added filtering for AM263/Px.....	361
• Power: Added further detail surrounding connections with the 1.8V LDO.....	361
• Add POK Types.....	361
• Updated POKs in POR Module Programming image.....	372
• Removed Note.....	381
• Updated VTM Features.....	381
• Updated Internal RC Oscillator section.....	480
• Updated MAIN Domain PLLs Integration - Part 1 diagram.....	484
• Updated PLLs in MAIN Domain section.....	487
• Updated Master Interfaces section.....	547
• Added R5FSS Interrupts section.....	562
• Updated Spinlock Software Reset section.....	827
• removed unnecessary text from diagram.....	827
• Added detail to main spinlock operations.....	829
• Removed unnecessary text in diagram.....	829
• Updated DDRSS Not Supported Features section.....	850
• Updated Inline ECC for SDRAM Data section.....	860
• Added Note to DDR Controller Functional Description section.....	867
• Added Note to BIST Engine section.....	869
• Updated PI Functional Description section.....	901
• Added RAT Clocks and Resets and RAT Hardware Requests tables.....	930
• Updated WKUP_DMSC0 Interrupt Map.....	966
• Updated MCU_R5FSS0_CORE0 Interrupt Map.....	973
• Updated MCU_R5FSS0_CORE1 Interrupt Map.....	981
• Updated GIC500 SPI Interrupt Map.....	992
• Updated R5FSS0_INTRTR0 Interrupt Map.....	1049
• Updated R5FSS1_INTRTR0 Interrupt Map.....	1057
• Update CPTS_TS_COMP_LEN_REG from 24 to 32 bits.....	1328
• Update CPTS_TS_COMP_LEN_REG from 24 to 32 bits.....	1328
• [Trigger Configuration (per Bit)] Clarify configuration of GPIO interrupt generation.....	1402

• [Trigger Configuration (per Bit)] updated method to return the value of the FAL_TRIG register. User can read SET_FAL_TRIG <b>or</b> CLR_FAL_TRIG registers to obtain FAL_TRIG value (rather than SET_FAL_TRIG <b>and</b> CLR_FAL_TRIG).	1402
• Added HS Mode entry to Features section.	1409
• Removed HS Mode footnote from I2C Register Values for Maximum I2C Bit Rates in I2C F/S, I2C HS Modes table.	1426
• Updated Figure MCSPI Overview. Moved and Renamed Figures MCSPI3/4 Connectivity Details to New Subsection MCSPI Internal Connectivity.	1472
• [MCSPI Protocol and Data Format] Added CLKG bit field information to Programmable MCSPI Clock bullet point.	1480
• [Peripheral Receive-Only Mode] Added clarification to definition of full-duplex mode (requires 2 serial data lines).	1509
• Renamed master words with "controller" and slave words with "peripheral" in complete UART chapter.	1534
• Added note to UART Frame Data Format diagram.	1538
• [SIR Free-Format Mode] Added additional information per design feedback.	1545
• [SIP Generation] add additional information for SIP_MODE registers bit setting.	1547
• [UART Interrupts] added additional register bit information for 001100 row in UART Mode Interrupts table.	1570
• [Wake-Up Interrupt] modified topic to include conditions for wake-up interrupt.	1571
• [Transmit FIFO Trigger] changed register naming to align with RA. Updated note to correct register bits.	1574
• [Receive FIFO Trigger] changed register naming to align with RA. Updated note to correct register bits.	1574
• [FIFO DMA Mode Operation] Updated register naming to match RA.	1577
• [Multi-drop Parity Mode with Address Match] added line at end of topic detailing supported and unsupported modes.	1599
• [Time-guard] added details related to other modes.	1601
• Make corrections to VLAN_Unaware.	1639
• Update bit from CTL_EN to EXT_EN.	1662
• Update bit from CTL_EN to EXT_EN.	1662
• Update Encoder/Decoder 3 Data to 103:78.	1664
• Update Event FIFO depth from 10 to 32.	1691
• Updated LTSSM State Encoding table.	1873
• Updated PMA bullets in 2-L SerDes Features section.	1894
• Updated PMA bullets in 4-L SerDes Features section.	1911
• Updated OSPI DMA bullet in FSS Not Supported Features.	1922
• Changed OTFA references to OTFE.	1927
• Remove misleading statement: Supports dual Quad-SPI mode for fast boot applications.	1934
• Added the requirement of RESET_OUT[1:0] signals when OSPI flash memory is used for Boot.	1936
• Maximum bytes supported by STIG is 16.	1952
• Added eMMC PHY BIST section.	2119
• Updated UFS Features section.	2190
• Removed UFS Encryption/Decryption Support section.	2198
• (RTI Digital Watchdog): Added note that this feature is only available for the WWDT defined modules.	3373
• (RTI Digital Windowed Watchdog): Fixed error in RTI Digital Windowed Watchdog Operation Block Diagram.	3374
• [PSA Signature Register] added CRC polynomial equations for all supported CRC polynomials.	3509

This page intentionally left blank.



## IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI's products are provided subject to [TI's Terms of Sale](#) or other applicable terms available either on [ti.com](https://www.ti.com) or provided in conjunction with such TI products. TI's provision of these resources does not expand or otherwise alter TI's applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265  
Copyright © 2024, Texas Instruments Incorporated